MAINDEC 2

MEMORY CHECKERBOARD

Abstract: Memory Checkerboard is a program which, in the presence of the worst possible noise, tests the PDP-1 memory for read-out errors. The program has five options:

1) Setting SS1 causes the program to ignore all errors.

2) SS2 selects the end of core which is to contain the program.

3) SS3-SS6 select the noise pattern or patterns to be used.

4) The TEST WORD switches determine which planes, if any, are excluded from the test.

5) Limits can be specified for the area of core to be tested.

CHAPTER 1

CONSOLE OPERATING PROCEDURE

The five tables listed below describe the console operating procedure to be used when running the Memory Checkerboard program.

TABLE 1-1 TAPES REQUIRED FOR TEST

The MAINDEC 2 program tape contains both low and high core versions of MAINDEC 2.

TABLE 1-2 SWITCHES

Switch	Setting	Function
SENSE SWITCH 1	0	Program halts on error.
	1	Program does not halt on error.
SENSE SWITCH 2	0	Core locations 0161-7777 checked.
	1	Core locations 0000-7616 checked.
SENSE SWITCH 3	1	Causes pattern . . . . 0 1 1 0 . . . 1 0 0 1 . . . 1 0 0 1 . . . 0 1 1 0 . . . to be tested.

TABLE 1-2 SWITCHES (continued)

Switch	Setting	Function
SENSE SWITCH 4	1	Causes pattern . . . . 1 1 0 0 . . . 1 1 0 0 . . . 0 0 1 1 . . . 0 0 1 1 . . . to be tested.
SENSE SWITCH 5	1	Causes pattern . . . . 0 1 1 0 . . . 1 0 0 1 . . . 1 0 0 1 . . . 0 1 1 0 . . . to be tested.
SENSE SWITCH 6	1	Causes pattern . . . . 1 0 0 1 . . . 1 0 0 1 . . . 0 1 1 0 . . . 0 1 1 0 . . . to be tested.
TEST WORD		Switches up cause core planes to be excluded from test.
ADDRESS	0000	Starting address of MAINDEC 2 when loaded in low end of core.
	7617	Starting address of MAINDEC 2 when loaded in high end of core.

TABLE 1-3 LOAD SEQUENCE

a) Turn off SENSE SWITCHES 1 and 2.

b) Set SENSE SWITCHES 3-6 and TEST WORD switches.

c) Load MAINDEC 2 tape into memory. Half of the tape should be read, inserting the program at the low end of core. (Leave reader on)

d) After running tests, repeat step (b) and turn SENSE SWITCH 2 ON. The remainder of tape in the reader will be read into the high end of core. Note: if for any reason the low-loading version did not operate, turn SENSE SWITCH 2 ON and depress read-in! This will cause the high-loading version to be read in.

e) To set limits to checked area:

   1) Stop program.

   2) Write lower limit of area to be checked into 0150 if program is in low end, or 7767 if program is in high end.

   3) Write upper limit of area to be checked into 0147 if program is in low end, or 7766 if program is in high end.

   4) Set ADDRESS switches to 0000 if program is in low end, or 7617 if program is in high end.

   5) Depress START.

f) The test can be repeated by reloading MAINDEC 2 tape in the reader, turning reader on, and turning SENSE SWITCH 2 off. The program will be read into the low end of core again.

Note:

1) Manually-set limits must be within the following ranges:
   Program in low end -- $0161 < \text{lower limit} < \text{upper limit} < 7777$
   Program in high end -- $0000 \leq \text{lower limit} \leq \text{upper limit} \leq 7676$

2) The manually set limits must be reset each time the program is reread from tape.

3) A single location may be checked by setting both the low and high limits equal to the address of location desired.

TABLE 1-4 ERROR HALTS

After a programmed error halt, the IO contains the address of the memory location which caused the halt and the AC displays the bits in error as 1 bits within a field of correct 0 bits, or as 0 bits within a field of correct 1 bits.

Error No.	Contents of MA	Cause of Error Halt
Errhlt 1	0142	Bits dropped or picked up shown in AC. Address of location which caused error shown in IO.
Other	any other	Not a programmed error halt. Noise conditions probably caused program malfunction.

TABLE 1-5 POST-ERROR RESTART PROCEDURE

The operator should record the memory location and the bit position of the error. He may wish to change the settings of various SENSE SWITCHES and TEST WORD switches before restarting.

Error No.	Procedure
Errhlt 1	To resume checking from point where error occured, depress CONTINUE; or: To reread and restart the program, depress READ IN; or: To restart the program from the beginning, set ADDRESS switches to 0000 if the program is in the low end (to 7617 if the program is in the high end), then depress START.
Other	This type of halt was probably caused by program malfunction. Consequently there is no set procedure for determining the cause of the halt. However, the contents of the program counter and the other console indicators may often provide clues as to the cause. The operator may wish to change the setting of SS2 to test the area of core previously occupied by the program. He must then restart by depressing READ IN to restore the program.

CHAPTER 2

SUGGESTED APPLICATIONS OF THE MEMORY CHECKERBOARD PROGRAM

The procedures described below provide useful methods of testing the PDP-1 memory. Since a memory stack may contain planes which are not all wired alike, it is often necessary to select several of the four available noise patterns. If the operator is not familiar with the wiring of the particular machine he should select all four noise patterns to ensure a worst pattern for every core plane.

a. DAILY TEST PROCEDURE - Before beginning normal operation of the computer, test the memory, using each pattern at least once. Turn on SS3 through SS6 and run the Memory Checkerboard until each pattern has been used. Then change SENSE SWITCH 2 and repeat the same procedure to test the remainder of core. Make sure that SS1 and all the TEST WORD switches are down throughout the test.

b. FULL TEST PROCEDURE - For a full test, each noise pattern should be tested a number of times using both the high and the low form of the program. This should be done by testing one pattern at a time until all four patterns have been used.

   If an error is found, exclude correct areas of core from the check and continue to run to duplicate the error. If SS1 is on, the program does not halt on an error, but continues checking. This feature allows the operator to get an oscilloscope trace of a recurring error.

c. MARGIN CHECKS - Perform margin checks using the full test procedure (b above).

   Detailed methods for checking margins are presented in paragraph 11-7b of the PDP-1 Maintenance Manual.

CHAPTER 3

PROGRAM DESCRIPTION

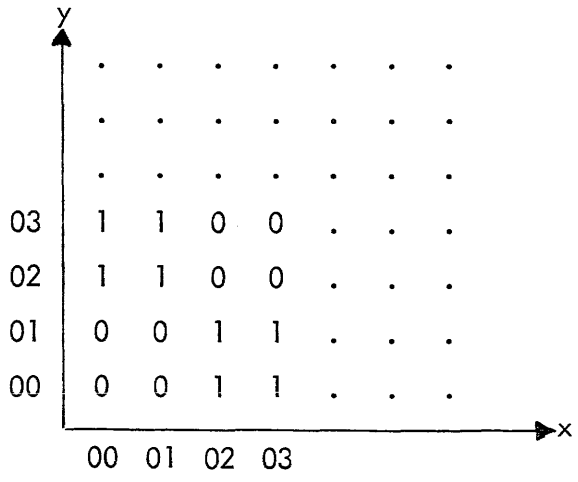
3-1 GENERAL

Memory Checkerboard is a program which, in the presence of the worst possible noise, tests the PDP-1 memory for read-out errors. The worst noise condition exists on the sense winding when the cores it threads are set in a "checkerboard" pattern. Because all sense wires are not threaded in the same manner, four worst patterns (and their complements) are needed to ensure that memory is checked completely.

Various options are available to the operator while using Memory Checkerboard. First, he may select, from among the four available, the pattern or patterns that are to be used. Second, he may locate the program in either the high or the low end of memory. Third, he may limit the range of memory locations to be checked. Fourth, he may exclude any bit or bits from the check; i.e. he may exclude any desired selection of core planes. Finally, he may set the program to ignore all errors.

3-2 NOISE PATTERNS

Each memory plane is a square 64 x 64 core array (100_8 by 100_8). A given core is addressed by specifying an x and a y coordinate. The x coordinate is specified by the low order six bits of the address (bits 12-17), and the y coordinate by the high order six bits (bits 6-11). The basic pattern for creating noise consists of alternate blocks of four zeros and four ones arranged throughout core in a checkerboard array as shown below:



The remaining three patterns are constructed by shifting the basic pattern. One variation is obtained by shifting the basic pattern one bit position in the negative x direction; another is obtained by shifting the basic pattern one bit position in the negative y direction, and the third by shifting the basic pattern one bit position in both the negative x and y directions. All four patterns and their complements are available for use within the program.

3-3 PROGRAM LOCATION

Memory Checkerboard requires 160_8 memory locations; these locations are not checked. In order that all memory locations may be checked, the program exists in two forms. When located in lower core the program loads and checks locations 0161 through 7777; when located in high core the program checks locations 0000 through 7615.

The desired location of the program is indicated by SS2. If SS2 is up, the program is loaded in high core; if down, in low core. Sense switch 2 is tested before each program cycle. A program cycle comprises the testing of each of the sense switches 3 through 6 and, for sense switches which are on, the checking of the corresponding patterns.

3-4 LOCATIONS CHECKED

Normally, the program checks all locations except those used to store the program itself. However, the operator can limit the area to be checked by writing the upper and lower limits of the desired area into the specified memory locations; refer to Table 1-3 (c). Even if such limits are set, the noise pattern is still written into all core except for the program area. A single location can be checked by setting the upper and lower limits of the area equal.

3-5 BITS CHECKED

Core planes can be excluded from the test by turning on the TEST WORD switch for the corresponding bit position. Bits thus excluded are always loaded with zeros when the pattern is written. If any errors occur in these positions, they are ignored.

3-6 IGNORE ERRORS

Normally, the program will halt when an error is found. However, if SS1 is up it continues checking and ignores the error. This feature is valuable whenever the operator wishes to use a scope and therefore wants the computer to repeat the error.

3-7 PROGRAM FUNCTION

Memory Checkerboard loads all memory locations not occupied by program with a noise pattern, and then reads out each memory location in turn and checks it for errors. After the original contents of a location have been checked, the location is loaded with the complement of its initial contents and rechecked.

In this way Memory Checkerboard tests for both dropped bits and picked-up bits. When this double check has been completed the noise pattern contents of the location are restored and the next location is checked. The program has five major sections: a subroutine which constructs the pattern for noise, a control routine, a write routine, a check routine, and an error routine.

a. PATTERN SUBROUTINE - The heart of the program is the subroutine which constructs the pattern for noise. The contents of a location are determined by comparing the second-lowest order bits (bits 16 and 10) in the X and Y coordinates (bits 12-17 and 6-11 respectively) of the location's address. If bits 10 and 16 are the same the location is loaded with zeros; if different, it is loaded with the complement of the TEST WORD switches. These bits change each time either the X or the Y coordinate is increased by two; thus resulting in the basic checkerboard pattern previously described.

   The basic pattern can be shifted by adding a number, called the offset, to the location's address before its contents are selected. If an offset of 001_8 is used, the basic pattern is shifted one bit position in the negative X direction; if the offset is 100_8 the basic pattern is shifted one bit position in the negative Y direction. In this manner the three additional patterns are constructed from the basic pattern. The patterns, the offsets used to construct them, and the sense switches which select them are tabulated below:

Pattern	Offset	Sense Switch
. . . . 0 1 1 0 . . . 1 0 0 1 . . . 1 0 0 1 . . . 0 1 1 0 . . .	101	3
. . . . 1 1 0 0 . . . 1 1 0 0 . . . 0 0 1 1 . . . 0 0 1 1 . . .	000 (basic)	4
. . . . 0 0 1 1 . . . 1 1 0 0 . . . 1 1 0 0 . . . 0 0 1 1 . . .	100	5
. . . . 1 0 0 1 . . . 1 0 0 1 . . . 0 1 1 0 . . . 0 1 1 0 . . .	001	6

b CONTROL ROUTINE - The setting of SS2 designates the end of core in which the program should be stored (SS2 up for loading program in high core; down for low core). The control routine tests SS2 to ensure that the program is in the desired location. If the program is in the wrong end of core, the RIM loader stores the other form of the program in the opposite end of core (the end designated by SS2) and transfers control to it.

After ensuring that the program is stored in the desired location, the control routine stores the complement of the TEST WORD switches for later use. Next the routine tests the sense switches in order, SS3 through SS6. If a given switch

is up, the control routine tests the memory by means of the checkerboard pattern corresponding to that switch. If more than one of the four switches SS3-SS6 are up, then the corresponding tests are executed in sequence.

Should all four switches be down, the control routine continues to test these switches until one of them is raised, or the operator intervenes. For each selected pattern, the control routine constructs an "offset" (described in a above).

After all checks with a pattern have been completed, the control routine repeats the entire writing and checking cycle with the complement of the pattern. Flag 2 is on when the complement cycle is being run. Upon completion of a complement cycle, the control routine tests the next sense switch.

c WRITE ROUTINE - The write routine loads the selected noise pattern in all core locations not occupied by the program. It does this by using the pattern subroutine (described in a above) to obtain the contents for each location. Flag 1 is on while the write routine is operating.

d CHECK ROUTINE - When a write cycle has been completed, the check routine tests each location in the area to be checked. The contents of a location first are masked to exclude the bits specified by the TEST WORD switches, and then are compared with the correct contents (as determined by the pattern subroutine). If the contents of the location are incorrect, the program enters the error routine. If correct, the check routine loads the location with the complement of its initial contents, then masks and checks again in the same manner. This double check (contents and complement) tests for both dropped bits and picked-up bits. After the second check, the initial contents are restored in the location and the check routine proceeds to the next location. Flag 1 is off when the check routine is in operation.

e ERROR ROUTINE - The error routine is entered whenever an error occurs in a non-excluded bit. This routine first tests SS1; if up, the error is ignored. If SS1 is down, the computer halts; the IO displays the address of the location that was in error; the AC displays the correct contents of the location (except for the bit or bits in error, which are complemented).

f RESTART AFTER ERROR - After an error halt the operator may restart in three ways: 1) depress CONTINUE to continue checking from the point at which the error occurred; or 2) set the ADDRESS SWITCHES and depress START to restart the program from the beginning; or 3) depress READ IN and re-read the program from tape.

3-7 RIM LOADER

A RIM loader is used to read the programs into core. When the setting of SS2 is changed, the program transfers to the loader and reads the program into the opposite end of core.

A tape in read-in mode format contains a series of instructions; the even numbered instructions make up the program to be stored in core; the odd numbered instructions are dio instructions, each having an address that determines the storage location for the next (even numbered) instruction. The last even numbered instruction of the tape may be followed by a jmp command. This jmp specifies the starting address for the stored program and, after read-in is completed, causes program execution to begin.

The RIM loader reads in an odd numbered instruction and, by sensing the sign of the IO, checks to determine if the instruction is a dio or a jmp. A +IO indicates that the instruction is dio (since the dio op code = 32); conversely, -IO indicates jmp (op code = 60). If the instruction is a dio, the loader reads in the next instruction and executes the dio. On the other hand, if the instruction is a jmp, the computer executes the jmp thereby leaving the RIM loader and beginning operations at the address specified by the jmp. In this way, the RIM loader simulates the computer read-in mode.