By Walter M. White, 343 s. Madison Ave., 18, Pasedena, CA 91101 

MEMORY TEST PROGRAM FOR MARK-8

Purpose of the program is to thoroughly check the semiconductor memory used with the 8008 for ICs that will not accept data correctly or have problems in their internal address decoding circuitry.

The program starts at location 000 and halts at location 101 upon completion. Pseudo random bit patterns are written into the entire memory (except that portion where the program resides) and then are read back and checked against the pattern written. This process is repeated with 207 different starting points in the random bit pattern. Execution time is approximately 45 minutes for 16K of memory.

No peripheral devices are required, only the means of jamming a NO-O? instruction (LAA =3008) into the interrupt instruction port is necessary.

Upon detection of a read error, the program will halt. The LED register connected to output port Ø will indicated which bit of the 8 bit byte was incorrectly read, thus establishing which column the defective memory IC is in. Jamming a NO-OP into the interrupt instuctlon port will cause the program to halt with the high order address bits of the defective IC. The two displays enables the user to pinpoint the defective IC. The lowest order 8 bits of the address are not displayed as it is immaterial which bit inside the Ie is defective, the entire IC must be replaced. Jamming a second NO-OP will cause the program to continue until completion or until another read error is detected. Note: if the program halts with all eight LEDs lit, it indicates a short on the address lines or a wiring error in the memory address circuitry.

| LOCATION | INSTRUCTION | | COMMENT |
|---|---|---|---|
| This section clears display and initializes registers | | | |
| 000 | 046 | LEI | Load register E with in!tial "random number" |
| 001 | 001 | 001 | |
| 002 | 026 | LOI | Load highest 6 bits of last address in |
| 003 | xxx | xxx | memory plus 1 ($100_8$ for 16K, $004_8$ for lK, |
| 004 | 056 | LH! | $008_8$ for 2K, etc.) |
| 005 | 000 | 000 | |
| 006 | 066 | LLI | Load starting memory test address |
| 007 | 120 | 120 | |
| 010 | 250 | XRA | Olear LED display |
| 011 | 121 | OUT | |
| This section writes random pattern into memory | | | |
| 012 | 334 | LDE | store starting random number for each pass |
| 013 | 106 | CAL | Call random number |
| 014 | 105 | 105 | |
| 015 | 000 | 000 | |
| 016 | 230 | SEA | Set all bits of accum. equal to carry bit |
| 017 | 310 | LMA | Store |
| 020 | 060 | INL | Increment memory address location and |
| 021 | 110 | JFZ | continue writing |
| 022 | 013 | 013 | |
| 023 | 000 | 000 | |
| 024 | 050 | INH | |
| 025 | 305 | LAH | |
| 026 | 272 | cpe | Continue writing until maximum memory |
| 027 | 110 | ,]FZ | address 1s reached |
| 030 | 013 | 013 | |
| 031 | 000 | 000 | |

This section reads data and checks it against pattern written

| | | | |
|---|---|---|---|
| 032 | 066 | LLI | Load starting memory address |
| 033 | 120 | 120 | |
| 034 | 056 | LHI | |
| 035 | 000 | 000 | |
| 036 | 343 | LED | Recover starting random number used for |
| 037 | 106 | CAL | last memory write pass |
| 040 | 105 | 105 | Call random number |
| 041 | 000 | 000 | |
| 042 | 230 | SBA | Set all bits of accum: equal to carry bit |
| 043 | 257 | XRM | Compare with memory data |
| 044 | 150 | JTZ | Jump around error routine if data compares OK |
| 045 | 056 | 056 | |
| 046 | 000 | 000 | |
| 047 | 121 | OUT | Display bits in error |
| 050 | 001 | HLT | User jams a NO-OP to display address of bad Ie |
| 051 | 305 | LiR | Display high order bits of memory that failed |
| 052 | 121 | OUT | |
| 053 | 001 | HLT | User jams a NO-OP to continue with program |
| *054* | 250 | IRA | Clear display |
| 055 | 121 | OUT | |
| 056 | 060 | INL | Increment memory address location |
| 057 | 110 | JFZ | |
| 060 | 037 | 037 | |
| 061 | 000 | 000 | |
| 062 | 050 | INH | |
| 063 | 305 | LAH | |
| 064 | 272 | cpe | Continue reading and comparing until maximum |
| 065 | 110 | JFZ | memory address is reached |
| 066 | 037 | *037* | |
| *067* | *000* | 000 | |

This section initializes the random number subroutine with a different number for the next write pass through memory.

| | | | |
|---|---|---|---|
| 070 | 343 | LED | Recover starting random number used for |
| 071 | 106 | CAL | last memory pass |
| 072 | 105 | 105 | Call random number |
| 073 | 000 | 000 | |
| *074* | 074 | CPI | Check if all random bit patters have been used |
| *075* | 001 | 001 | |
| 076 | 110 | JFZ | Jump to memory write routine |
| 077 | 004 | 004 | |
| *100* | 000 | 000 | |
| 101 | 001 | HLT | Program stops here when complete |
| 102 | 104 | JMP | |
| 103 | 101 | 101 | |
| 104 | 000 | 000 | |

This section is a pseudo random number generating subroutine. It generates 207 of the possible 256 combination of 8 bits and can be used as the basis for a number of computer games. The main program above uses only the bit that is shifted into the carry position, not the actual random number generated.

| | | | |
|---|---|---|---|
| 105 | 304 | LAE | Load accum. with previous random number |
| 106 | 032 | RAR | Rotate 3 bit positions |
| *107* | *032* | RAR | |
| *110* | 032 | RAR | |
| 111 | 254 | XRE | Exclusive OR with previous random number |
| 112 | 032 | RAR | Rotate new bit into carry |
| 113 | 304 | LAE | Load accum with previous random number |
| 114 | 032 | RAR | Rotate carry into 17 creating new random number |
| 115 | 340 | LEA | Save number in register E |
| 116 | 007 | RET | |