Cook's Memory Test For the 6502

Version 3 of this test follows two others: one for the KIM, one for the 8080.

Fred Monsour
309 Camellia Drive
Charlottesville VA 22903

Whether you built your system chip by chip or bought it fully assembled, you've probably had, or at least worried about, memory problems. There was a time—before I stabilized my KIM-1 expansion—when I collected memory tests as some people collect stamps.

Unfortunately, that didn't deter the memory bugs, which crept in just as I was about to save a 200-line program or show my latest software acquisition to some innocent bystander. Worst of all was the feeling that I had wasted my time and money on a computer that couldn't be trusted.

About the time I was considering computercide, Charles Cook published his "walking bit" test routine ("Memory Troubleshooting Techniques," *Kilobaud*, Oct. 1977, p. 58). That looked like KIM's salvation, but since my machine-language programming expertise extended to adding two hexadecimal numbers, I waited for someone else to put it into code. By the time Rod Hallen's 8080 version appeared a few months later ("It's Here: Cook's Memory Test," *Kilobaud*, July 1978, p. 70), I had progressed to adding *three* hexadecimal numbers together.

So after seeing how well the routine had worked for Rod, I decided to try a 6502 version. Cook's algorithm unearthed several rambunctious 2102s, which had been given A-1 ratings by other memory tests minutes earlier. Since I replaced those, my system's been almost 100 percent reliable. At least now I'm confident that if bad memory is the problem, I'll know about it.

The program listing given here is slightly modified from my original code in order to make the program more portable. At a small sacrifice in size, I have used only one subroutine from the KIM monitor—OUTCH, which prints the character in the accumulator. If you use a different output routine, no registers need be preserved.

It tests 8K of RAM in less than 3 seconds, so I let it cycle several times to be sure all is well. At the end of each cycle it prints an asterisk to let you know it's making progress.

The flowchart in Fig. 1 is similar to those presented in the articles by Cook and Hallen. I won't go into the logic of the algorithm any further; refer to the previous articles for a complete discussion. Generally, though, an error at branch A indicates a byte that can't have all 0s stored in it. Error B usually is due to shorted or open address lines, while error C is due to shorted or open input data lines. Error D signifies a byte that can't store all 1s.

No memory test is perfect, but this is one of the best. Add it to your arsenal and be prepared when memory bugs strike. ■

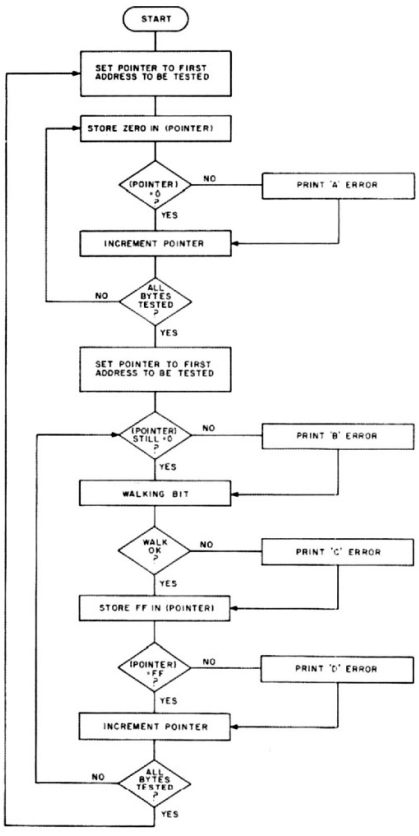

Fig. 1. Program flowchart.

Program listing. To use the program, put the first address to test in locations 0000/0001, the last address to test + 1 in 0002/0003. Start the program running at address 0200.

```
0010: 0200                ORG    $0200
0020:
0030:                     ************************************
0040:                     *                                  *
0050:                     *  COOK*S MEMORY TEST FOR THE 6502 *
0060:                     *           BY FRED MONSOUR        *
0070:                     *                                  *
0080:                     ************************************
0090:
0100:
0110: 0200           POINTL *     $00FA
0120: 0200           POINTH *     $00FB
0130: 0200           START  *     $0000
0140: 0200           END    *     $0002
0150: 0200           OUTCH  *     $1EA0   KIM SUB TO PRINT CHAR IN ACCUMULATOR
0160: 0200           TEMP   *     $00FC
0170:
0180: 0200 20 5D 02  BEGINA JSR   INIT    SET POINTERS TO FIRST TEST ADDRESS
0190:
0200:                 ** TEST FOR STORING ALL ZEROS **
0210: 0203 A9 00     NEXTA  LDAIM $00     TEST PROBE = 00
0220: 0205 A8               TAY           ZERO Y-INDEX
0230: 0206 91 FA            STAIY POINTL  STORE ZERO IN TEST ADDRESS
0240: 0208 B1 FA            LDAIY POINTL  GET IT BACK
0250: 020A F0 05            BEQ   INCA    IF STILL ZERO MOVE ON
0260: 020C A2 41            LDXIM *A      ELSE ERROR CODE = A
0270: 020E 20 66 02         JSR   ERR     PRINT THE BAD NEWS
0280: 0211 20 89 02  INCA   JSR   INC     POINT TO NEXT ADDRESS
0290: 0214 20 80 02         JSR   COMPR   SEE IF DONE WITH FIRST PASS
0300: 0217 90 EA            BCC   NEXTA   IF NOT, TEST NEXT ADDRESS
0310:
0320:                 ** TEST FOR RETAINING ZEROS **
0330: 0219 20 5D 02         JSR   INIT    RESET POINTER FOR SECOND PASS
0340: 021C B1 FA     BEGINB LDAIY POINTL  SEE IF ZERO STILL THERE
0350: 021E F0 05            BEQ   WALK    YES: DO NEXT TEST
0360: 0220 A2 42            LDXIM *B      NO! ERROR CODE = B
0370: 0222 20 66 02         JSR   ERR     PRINT THE BAD NEWS
0380:
0390:                 ** WALKING BIT TEST **
0400: 0225 A2 08     WALK   LDXIM $08     BIT COUNTER = 8
0410: 0227 A9 01            LDAIM $01     PROBE FIRST BIT
0420: 0229 91 FA     NEXTB  STAIY POINTL  STORE BIT IN TEST ADDRESS
0430: 022B D1 FA            CMPIY POINTL  STORED OK?
0440: 022D F0 0D            BEQ   SHIFT   YES: DO NEXT BIT
0450: 022F 9A               TXS           NO! SAVE BIT COUNTER...
0460: 0230 48               PHA           AND TEST PROBE...
0470: 0231 8A               TXA           GET BAD BIT # (COUNTING FROM LEFT)
0480: 0232 20 94 02         JSR   PRBYT   PRINT IT...
0490: 0235 68               PLA           RESTORE TEST PROBE
0500: 0236 A2 43            LDXIM *C      ERROR CODE = C
0510: 0238 20 66 02         JSR   ERR     PRINT THE BAD NEWS
0520: 023B BA               TSX           RESTORE BIT COUNTER
0530: 023C 0A        SHIFT  ASLA          NOW SHIFT PROBE TO NEXT BIT...
0540: 023D CA               DEX           ONE LESS BIT LEFT TO TEST
0550: 023E D0 E9            BNE   NEXTB   FINISH ALL 8 BITS
0560:
0570:                 ** TEST FOR STORING ALL ONES **
0580: 0240 A9 FF            LDAIM $FF     TEST PROBE = FF
0590: 0242 91 FA            STAIY POINTL  STORE FF IN TEST ADDRESS
0600: 0244 D1 FA            CMPIY POINTL  STORE OK?
0610: 0246 F0 05            BEQ   INCB    YES: MOVE ON
0620: 0248 A2 44            LDXIM *D      NO! ERROR CODE = D
0630: 024A 20 66 02         JSR   ERR     PRINT THE BAD NEWS
0640: 024D 20 89 02  INCB   JSR   INC     POINT TO NEXT ADDRESS
0650: 0250 20 80 02         JSR   COMPR   SEE IF DONE WITH SECOND PASS
0660: 0253 90 C7            BCC   BEGINB  IF NOT, TEST NEXT ADDRESS
0670: 0255 A9 2A            LDAIM **      IF SO, PRINT '**'...
0680: 0257 20 90 02         JSR   PRNT    TO SAY SO
0690: 025A 4C 00 02         JMP   BEGINA  AND DO WHOLE THING AGAIN
0700:
0710:                       ** SUBROUTINES **
0720: 025D A5 00     INIT   LDAZ  START   PUT LSB OF START ADDRESS...
0730: 025F 85 FA            STA   POINTL  IN LSB OF POINTER
0740: 0261 A5 01            LDAZ  START+01 PUT MSB OF START ADDRESS...
0750: 0263 85 FB            STA   POINTH  IN MSB OF POINTER
0760: 0265 60               RTS           RETURN
0770:
0780: 0266 48        ERR    PHA           SAVE TEST PROBE
0790: 0267 98               TYA           SAVE Y-INDEX
0800: 0268 48               PHA
0810: 0269 8A               TXA           GET ERROR CODE
0820: 026A 20 90 02         JSR   PRNT    PRINT IT
0830: 026D A5 FB            LDA   POINTH  GET MSB OF BAD ADDRESS
0840: 026F 20 94 02         JSR   PRBYT   PRINT IT
0850: 0272 A5 FA            LDA   POINTL  GET LSB OF BAD ADDRESS
0860: 0274 20 94 02         JSR   PRBYT   PRINT IT
0870: 0277 A9 20            LDAIM $20     GET ASCII SPACE
0880: 0279 20 90 02         JSR   PRNT    PRINT IT
0890: 027C 68               PLA           RESTORE...
0900: 027D A8               TAY           Y-INDEX
0910: 027E 68               PLA           RESTORE TEST PROBE
0920: 027F 60               RTS           RETURN
0930:
0940: 0280 A5 FA     COMPR  LDA   POINTL  THIS ROUTINE CLEARS CARRY FLAG...
0950: 0282 C5 02            CMP   END     IF POINTER < END ADDRESS
0960: 0284 A5 FB            LDA   POINTH  ELSE SETS IT
0970: 0286 E5 03            SBC   END
0980: 0288 60               RTS
0990:
1000: 0289 E6 FA     INC    INC   POINTL  THIS ROUTINE...
1010: 028B D0 02            BNE   RET     INCREMENTS POINTER
1020: 028D E6 FB            INC   POINTH  (EQUIV. OF KIM INCPT)
1030: 028F 60        RET    RTS
1040:
1050: 0290 20 A0 1E  PRNT   JSR   OUTCH   KIM SUB TO PRINT CHAR IN ACCUM.
1060: 0293 60               RTS
1070:
1080: 0294 85 FC     PRBYT  STA   TEMP    THIS ROUTINE BREAKS...
1090: 0296 4A               LSRA          BYTE IN HALF AND
1100: 0297 4A               LSRA          PRINTS EACH HALF AS ASCII CHAR
1110: 0298 4A               LSRA          (EQUIV. OF KIM PRTBYT)
1120: 0299 4A               LSRA
1130: 029A 20 A5 02         JSR   HEXASC
1140: 029D A5 FC            LDA   TEMP
1150: 029F 20 A5 02         JSR   HEXASC
1160: 02A2 A5 FC            LDA   TEMP
1170: 02A4 60               RTS
1180:
1190: 02A5 29 0F     HEXASC ANDIM $0F     THIS ROUTINE CONVERTS...
1200: 02A7 C9 0A            CMPIM $0A     HEX DIGIT TO ASCII
1210: 02A9 18               CLC           (EQUIV. OF KIM HEXTA)
1220: 02AA 30 02            BMI   HEXASD
1230: 02AC 69 07            ADCIM $07
1240: 02AE 69 30            ADCIM $30
1250: 02B0 4C 90 02         JMP   PRNT

SYMBOL TABLE
BEGINA 0200    BEGINB 021C    COMPR  0280    END    0002
ERR    0266    HEXASA 02AE    HEXASC 0285    INCA   0211
INCB   0248    INC    0289    INIT   025D    NEXTA  0203
NEXTB  0229    OUTCH  1EA0    POINTH 00FB    POINTL 00FA
PRBYT  0294    PRNT   0290    RET    028F    SHIFT  023C
START  0000    TEMP   00FC    WALK   0225
```