

IDENTIFICATION

Product Code: MAINDEC-15-D0CA-D (D)
Product Name: Memory Address Test
Date: January 5, 1970
Maintainer: Diagnostics Group
Author: Edward P. Steinberger

14

1. ABSTRACT

The Memory Address Test checks the memory system of the PDP-15 to ensure that all memory locations not occupied by the program in a given 4K memory stack can be uniquely addressed. It does this by writing the address of a memory location into itself and checking to see that it is there. The complement of the address is also written to ensure that all bits of a word can be accessed. Checks are also made to ensure that only one memory location is written into whenever memory is addressed, and that cores of different memory locations are not shorted together inside the memory stack. Errors are indicated to the operator via the teleprinter.

2. REQUIREMENTS

Equipment

Standard PDP-15 Computer

Storage

The program uses all of 4K memory for the program or as a test are. The program occupies memory from location 07200 to 07770 and tests all locations below 07200.

3. LOADING PROCEDURE

Method

Put HRI tape of program in reader (High speed, if available).

Set ADDRESS SWITCHES to 07200 ; the BANK MODE switch to a 1.

Depress I/O RESET.

Depress and release READ IN key.

4. STARTING PROCEDURE

4.1 Control Switch Settings

The following is a table of accumulator switch settings and their action on the program.

<u>AC Switch</u>	<u>Set As</u>	<u>Action</u>
0	1	Halt on error
	0	Don't halt on error
1	1	Don't print errors
	0	Print errors
2	1	Ring bell on error
	0	Ring bell after N passes

<u>AC Switch</u>	<u>Set As</u>	<u>Action</u>
3	1	Loop on current number (address or complement)
	0	
4	1	Loop on current location
	0	
5	1	Loop on current test
	0	
6	1	Skip 1's in 0's test (Test 4)
	0	
7	1	Halt after completing all tests

Switch 3 operates only with the first test, switch 4 with first and fourth, 3 has presence over 4 (in first test), and 4 over 5 (in first and fourth tests). N is an arbitrary number (initially 200000₈) for the first and fourth tests, and 4₈ for the second and third. It may be changed at the operator's discretion by modifying the contents of locations 07202 and 07200 respectively to the appropriate LAW-N instruction.

4.2 Starting Address

The starting address of the program is 07200. The restart addresses are 07200, 07257, 07310, and 07353 (see listing).

4.3 Program and/or Operator Action

Set ADDRESS SWITCHES to 07200.

Set ACCUMULATOR SWITCHES to desired positions (see Section 4.1). Normal setting is 500000.

Depress I/O RESET.

Depress START.

5. OPERATING PROCEDURE

5.1 Operational Switch Settings

See Section 4.1.

5.2 Program and/or Operation Action

- a. To put the program in the 'SCOPE mode, the ACCUMULATOR SWITCHES should be set to 260000 (don't halt, don't print, bell after N passes, loop on current number, loop on current location).

- b. To test an individual location (first test), store the address to be tested, in location 07756 (POINT) and restart the computer at location 07211 (FIRST+7) to store the address in itself, or location 07225 (SECOND) to store the complement (1's) in the address. ACCUMULATOR SWITCHES should be set to 260000 (see a).
- c. To run the individual location test (whole first test), restart the computer at location 07200 (FIRST-2) with AC Switches 3 and 4 a 0 and 5 a 1.
- d. To narrow the first test to less than all of memory not occupied by the program, place the new upper limit in location 07762 (UPLIM), then start the computer at location 07200 (FIRST-2).
- e. To test all of memory in the forward direction only, restart the computer at location 07257 (THIRD) with AC switch 5=1.
- f. To test all of memory in the reverse direction only, restart the computer at location 07310 (FIFTH) with AC switch 5 = 1.
- g. To narrow the second and third tests to less than all of memory not occupied by the program, place the new upper limit in location 07763 (UPLIM1), then restart at the appropriate address for the particular test. Checks will still be made starting from location 00000 up to the limit for the forward test, or from the limit down to location 000000 for the reverse test.
- h. To write 1's into a particular memory location (fourth test) store the address to contain all 1's in location 07755 (PNTR1) and restart the computer at location 07360 (SEVENH+5) with AC switch 4 = 1.
- i. To narrow the fourth test to less than all of memory not occupied by the program, place the new upper limit in location 07762 (UPLIM), then start the computer at the appropriate starting address. Checks will still be made starting from location 00000 up to the limit.
- j. To run the Write 1's in a Field of 0's test (fourth test), restart the computer at location 07353 (SEVENH) with AC switch 5 = 1.

6. ERRORS

Unless AC switch 1 is a 1, all errors will be printed on the teleprinter.

6.1 Error Halts and Description

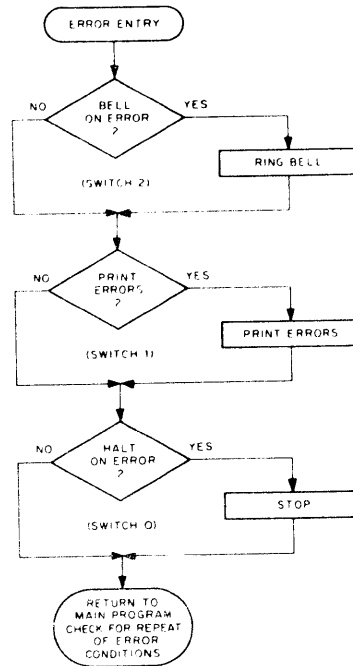
There is only one error halt in the program at location 07503. This halt will occur anytime there is an error; there is no useful information in the AC. The computer will halt at location 07451 if all tests are not repeated.

6.2 Error Recovery

If AC switch 0 = 1, the computer will halt on an error. To recover and repeat the failure, reset AC switches 0 to 5 as necessary (see Section 4.1), and then depress CONTINUE.

To test a particular location, see Section 5.2.

6.3 Error Switch Hierarchy



6.4 Error Typeout Examples

6.4.1 Individual Location Test

ADDRESS	GOOD	BAD
001234	001234	001230

The above example shows that location 1234 dropped bit 15 when its ADDRESS was written in it.

6.4.2 Forward Write-Read and Reverse Complement Write-Read Tests (both typeouts are the same)

Forward Write-Read Test

ADDRESS	GOOD	BAD
001234	001234	003234

The above example shows that location 1234 contained 3234 when it should have contained 001234. This may have been due to the "picking up" bit 7 in the memory buffer register when addressing location 0001234, or it may have been due to "double addressing" (addressing location 001234 when the memory address register contains 001234 and 003234).

6.4.3 Write 1's in a Field of 0's Test

WRITE ONES IN A FIELD OF ZEROES

ADDRESS	GOOD	BAD
001234	000000	000004
001235	000000	000004

The above examples shows that location 01234 contained 000004 when it should have contained 000000. This may have been due to bit 15 of location 01234 being shorted to bit 15 of location 01234, 01235. This same error may have caused location 01235 to be in error. If location 07764 (PNTR1) contained 01235 at the first error and 01234 at the second error, this is probably true.

7. EXECUTION TIME

Tests 1, 2 and 3 - insignificant

Test 4 - 7 minutes

8. PROGRAM DESCRIPTION

There are four basic parts to the program. The first tests each memory location, not occupied by the program, to assure that all bits of each may be accessed. It does this by first writing the address of a location in itself, and then checking to see if it was properly written. The 1's complement of the address is then written into the same location and checked, to assure that all bits of a memory location may be written and read. The second portion writes sequential addresses into their respective memory locations in the forward direction for all of memory not occupied by the program, and then each location is checked in the forward direction to assure that it contains its own address. The third portion writes the complement of each address not occupied by the program into sequential memory locations respectively in the reverse direction (from the highest location down

to the lowest) and then checks each of the locations in the reverse direction to ensure that it contains its respective address complemented.

The fourth portion of the test writes 0 in all memory locations not occupied by the program, and then writes all 1's in one memory location. All of the memory locations are then checked to assure that they contain 0, except the one location containing all 1's.

8.1 Individual Location Test

- a. The first function performed is that of initialization. Several loop counters and message header locations are initialized.
- b. An internal pointer for the program is set to 0 to indicate the first location to be tested is location 0.
- c. The number which is in the internal pointer is then stored in the memory location specified by the pointer and then the memory location is checked to see the correct number has been stored. An error causes the timeout subroutine to be called.
- d. A check is then made to see if this portion of the test (current number) should be repeated (switch 3). If so, the program goes to (c). If not, the program goes to (e).
- e. The number in the internal pointer is obtained, complemented and stored in the memory location specified by the pointer and then the location is checked to see the correct number has been stored. An error causes the timeout subroutine to be called.
- f. A check is made to see if this portion (number complemented) of the test (current number) should be repeated (switch 3). If so, the program goes to (e). If not, the program goes to (g).
- g. A check is made to see if the current location should be tested again (switch 4) and if so, the program goes to (c). If not, the number in the internal pointer is incremented and the program goes to (c) until all locations have been tested.
- h. A check is then performed to see if the whole test should be repeated (switch 5). If so, the program goes to (b). If not, the program goes on to the next portion.

8.2 Forward Write-Read Test

- a. The first function performed is that of initialization. Several message header locations are initialized.
- b. An internal pointer for the program is initialized to 0 to indicate the first location to be tested is location 0.

- c. The number which is in the internal pointer is then stored in the memory location specified by the pointer.
- d. The pointer is then incremented and (c) is repeated until all memory locations have had their addresses stored in them.
- e. The internal pointer is then set back to 0.
- f. The number which has been stored in the location specified by the pointer is obtained and checked to see that it is correct.
- g. The pointer is then incremented and (f) is repeated until all memory locations have been checked. Any error causes typeout on the teleprinter.
- h. A check is then made to see if this test should be repeated (switch 5). If so, the program goes to (b). If not, the program goes on to the next portion.

8.3 Reverse - Complement Write-Read Test

- a. The first function performed is that of initialization. Several message header locations are initialized.
- b. An internal pointer for the program is initialized to the first location under the program (07177).
- c. The number which is in the internal pointer is obtained, complemented (1's) and then stored in the memory location specified by the pointer.
- d. The pointer is then decremented (1 is subtracted from it) and (c) is repeated until all memory locations have had the complement addresses stored in them.
- e. The internal pointer is then set back to 07177.
- f. The number which has been stored in the location specified by the pointer is obtained and checked to see that it is correct.
- g. The pointer is then decremented and (f) is repeated until all memory locations have been checked. Any error causes typeout on the teleprinter.
- h. A check is then made to see if this test should be repeated (switch 5). If so, the program goes to (b). If not, the program goes to (i).
- i. A check is then made to see if the next test should be skipped (switch 6). If so, the program goes to 8.1 (a). If not, the computer goes to 8.4 (a).

8.4 Write 1's in a Field of 0's Test

- a. The first function performed is that of initialization. Several message header locations are initialized.
- b. The all 1's pointer is set to 0.
- c. All of memory tested is then cleared.
- d. All 1's is then stored in the location pointed to by the all 1's pointer
- e. A memory pointer is then set to 00000.
- f. The contents of the memory pointer is then checked against the all 1's pointer. If they are the same, the AC is set to all 1's, otherwise it is set to 0. This is then stored in a temporary storage location.
- g. The contents of the location indicated by the memory pointer are then obtained and compared against the contents of the temporary storage location. If they are the same, the program goes on to (h). If they are different, the error typeout routine is called after which the program goes on to (h).
- h. The memory location pointer is then incremented and checked to see if it is at the upper limit. If it is not, the program goes to (f). If the pointer is at the upper limit, the program goes to (i).
- i. A check is then made to see if the bell should be rung (switch 1).
- j. A check is then made to see if the current location should be looped upon. If not, the program goes to (l). If the current location should be looped upon, the program goes to (k).
- k. If there was no 1's in 0's error, the location containing 1's is cleared, and control goes to (d). If there was a 1's in 0's error, the program goes to (c).
- l. If there was a 1's in 0's error, the all 1's pointer is incremented, and if it is not the upper limit, the program goes to (c). If it is the upper limit, the program goes to (m). If there was no 1's in 0's error, the location containing all 1's is cleared, the all 1's pointer is incremented, and if not the upper limit, control goes to (d). If it is the upper limit, control goes to (m).
- m. A check is then made to see if the test should be repeated (switch 5). If so, control goes to (b). If the test is not repeated, control goes to (n).
- n. A check is made to see if all tests should be repeated (switch 7). If so, control goes to 8.1 (a). If all tests are not to be repeated, the computer halts.

```

          .TITLE M.A.T.
/
/MEMORY ADDRESS TEST
      .FULL
      .LOC 7200
07200
/INDIVIDUAL LOCATION TEST
07200 777774 LAW 17774 /INITIALIZE LOOP
07201 047745 DAC COUNT1
07202 760000 FIRST LAW
07203 047744 DAC COUNT /COUNTER
07204 207746 LAC JMPT
07205 047465 DAC CHANGE /CHANGE
07206 207764 LAC ZZZ1
07207 047754 DAC PNTR /AND MESSAGE POINTER
07210 147756 DZM POINT /ZERO POINTER
07211 207756 LAC POINT /GET ADDRESS TO BE STORED
07212 067756 DAC* POINT /AND STORE IT IN ITSELF
07213 047761 DAC TEMP /AND TEMP
07214 227756 LAC* POINT /GET THE NUMBER WRITTEN IN MEMORY
07215 547756 SAD POINT /AND CHECK IT AGAINST ITSELF
07216 607221 JMP .+3 /ALL OK.
07217 740001 CMA /ERROR
07220 107453 JMS ERROR
07221 750004 LAS
07222 507747 AND MASK1
07223 740200 SZA /LOOP ON CURRENT NUMBER?
07224 607211 JMP FIRST+7 /YES, LOOP
07225 207756 SECOND LAC POINT /NO, GET ADDRESS TO BE TESTED
07226 740001 CMA
07227 067756 DAC* POINT /STORE COMPLEMENT
07230 047761 DAC TEMP
07231 227756 LAC* POINT /GET NUMBER STORED
07232 740001 CMA
07233 547756 SAD POINT /AND CHECK IT
07234 607237 JMP .+3 /ALL OK
07235 740001 CMA /ERROR
07236 107453 JMS ERROR
07237 750004 LAS
07240 507747 AND MASK1
07241 740200 SZA /LOOP ON CURRENT NUMBER?
07242 607225 JMP SECOND /YES, LOOP
07243 750004 LAS
07244 507750 AND MASK2
07245 740200 SZA /LOOP ON CURRENT LOCATION?
07246 607211 JMP FIRST+7 /YES, LOOP
07247 107556 JMS BELL /BELL AFTER N PASSES?
07250 447756 ISZ POINT /INCREMENT POINT FOR
07251 207756 LAC POINT /NEXT LOCATION
07252 547762 SAD UPLIM /IS IT THE UPPER LIMIT?
07253 741000 SKP /YES, SKIP
07254 607212 JMP FIRST+10 /NO, TEST NEXT LOCATION
07255 107622 JMS LOOP3 /REPEAT TEST?
07256 607210 JMP FIRST+6 /YES
.EJECT

```

```

/FORWARD WRITE- READ TEST
/
07257 207746 THIRD LAC JMPT /INITIALIZE
07260 047465 DAC CHANGE /CHANGE
07261 207765 LAC ZZZ2
07262 047754 DAC PNTR /AND MESSAGE POINTER
07263 147756 DZM POINT /ZERO POINTER
07264 207756 LAC POINT /GET NUMBER TO BE STORED
07265 067756 DAC* POINT /STORE IT
07266 547763 SAD UPLIM1 /WAS IT THE UPPER LIMIT?
07267 607272 JMP .+3 /YES
07270 447756 ISZ POINT /NO, INCREMENT POINT
07271 607264 JMP THIRD+5 /GO BACK AND STORE
07272 147756 FOURTH DZM POINT /CLEAR POINT
07273 207756 LAC POINT
07274 047761 DAC TEMP
07275 227756 LAC* POINT /GET WORD STORED
07276 547756 SAD POINT /CHECK IT
07277 741000 SKP /CORRECT, SKIP
07300 107453 JMS ERROR /ERROR
07301 547763 SAD UPLIM1 /UPPER LIMIT
07302 607305 JMP .+3 /YES
07303 447756 ISZ POINT /NO, INCREMENT POINT
07304 607273 JMP FOURTH+1 /GO BACK AND CHECK NEXT LOCATION
07305 107572 JMS BELL1 /RING BELL?
07306 107622 JMS LOOP3
07307 607263 JMP THIRD+4 /YES
.EJECT

```

```

/REVERSE - COMPLEMENT WRITE-READ TEST
/
FIFTH 07310 207746 LAC JMPT /INITIALIZE
      07311 047465 DAC CHANGE /CHANGE
      07312 207766 LAC ZZZ3
      07313 047754 DAC PNTR /AND MESSAGE POINTER
      07314 207763 LAC UPLIM1
      07315 047756 DAC POINT /AND POINT (TO HIGHEST LOCATION)
      07316 740001 CMA /COMPLEMENT ADDRESS IN POINT
      07317 067756 DAC* POINT /AND STORE IT
      07320 740001 CMA /RECOMPLEMENT IT
      07321 741200 SNA /IS IT 0?
      07322 607326 JMP SIXTH /YES, DONE DEPOSITING
      07323 750001 CLA:CMA /NO, SET AC=-1
      07324 347756 TAD POINT /SUBTRACT 1 FROM POINT
      07325 607315 JMP FIFTH+5 /GO BACK TO STORE

/SIXTH 07326 207763 LAC UPLIM1 /RE-INITIALIZE
      07327 047756 DAC POINT /POINT
      07330 740001 CMA
      07331 047761 DAC TEMP
      07332 227756 LAC* POINT /GET CONTENT OF ADDRESS
      07333 740001 CMA /COMPLEMENT IT
      07334 547756 SAD POINT /CHECK IT
      07335 607340 JMP .+3 /ALL OK.
      07336 740001 CMA /ERROR
      07337 107453 JMS ERROR
      07340 751201 SNA:CLA:CMA /CHECKED ALL MEMORY LOCATIONS
      07341 607344 JMP .+3 /YES
      07342 347756 TAD POINT /NO, SUBTRACT 1 FROM POINT
      07343 607327 JMP SIXTH+1 /GO BACK AND CHECK NEXT
      07344 107572 JMS BELL1 /BELL?
      07345 107622 JMS LOOP3 /REPEAT TEST?
      07346 607314 JMP FIFTH+4 /YES
      07347 750004 LAS
      07350 507752 AND MASK4
      07351 740200 SZA /SKIP NEXT TEST?
      07352 607204 JMP FIRST+2 /YES
      .EJECT

```

```

/WRITE 1'S IN A FIELD OF 0'S
/
07353 207746 SEVENH LAC JMPT /INITIALIZE
07354 047465 DAC CHANGE /CHANGE
07355 207767 LAC ZZZ4 /AND MESSAGE
07356 047754 DAC PNTR /POINTER
07357 147755 DZM PNTR1 /ZERO ALL ONES POINTER
07360 147756 DZM POINT /ZERO POINT
07361 167756 DZM* POINT /CLEAR MEMORY
07362 447756 ISZ POINT
07363 207756 LAC POINT
07364 547762 SAD UPLIM
07365 751001 SKP:CLA:CMA
07366 607361 JMP .-5
07367 067755 DAC* PNTR1 /STORE 1'S IN 1 MEMORY LOCATION
07370 147756 EIGHTH DZM POINT /ZERO POINT
07371 207756 LAC POINT /GET POINT
07372 547755 SAD PNTR1 /SAME AS PNTR1
07373 751000 SKP:CLA /YES, CLEAR AC AND SKIP
07374 750001 CLA:CMA /NO, SET AC TO -1
07375 744001 CMA:CLL /COMPLEMENT AC, IF POINT = PNTR1
07376 047761 DAC TEMP /AC = 1'S, OTHERWISE 0
07377 227756 LAC* POINT /PICK UP CONTENTS OF MEMORY
07400 547761 SAD TEMP /GOOD DATA?
07401 741000 SKP /YES
07402 107453 JMS ERROR /NO
07403 447756 ISZ POINT /INCREMENT POINT
07404 207756 LAC POINT /TO NEXT LOCATION
07405 547762 SAD UPLIM /UPPER LIMIT?
07406 741000 SKP /YES
07407 607372 JMP EIGHTH+2 /NO, GO BACK TO TEST NEXT LOCATION
07410 107556 JMS BELL /RING BELL?
07411 750004 LAS
07412 507750 AND MASK2
07413 741200 SNA /LOOP ON CURRENT LOCATION?
07414 607424 JMP .+10 /NO
07415 207453 LAC ERROR /YES, GET C(ERROR)
07416 547743 SAD CONST /WAS THERE A 1'S IN 0'S ERROR
07417 607422 JMP .+3 /YES
07420 167755 DZM* PNTR1 /NO, ZERO 1'S LOCATION
07421 607365 JMP EIGHTH-3 /GO BACK TO STORE 1'S
07422 147453 DZM ERROR /ZERO ERROR
07423 607360 JMP SEVENH+5 /GO BACK TO ZERO MEMORY FIRST
07424 207453 LAC ERROR
07425 547743 SAD CONST /WAS THERE A 1'S IN 0'S ERROR?
07426 741000 SKP /YES, SKIP
07427 607436 JMP .+7 /NO
07430 147453 DZM ERROR /ZERO ERROR
.EJECT

```

07431	447755	ISZ PNTR1 /INCREMENT ALL 1'S
07432	207755	LAC PNTR1 /POINTER TO NEXT LOCATION
07433	547762	SAD UPLIM /UPPER LIMIT?
07434	607444	JMP .+10 /YES
07435	607360	JMP SEVENH+5 /NO
07436	167755	DZM* PNTR1 /ZERO LOCATION CONTAINING ALL 1'S
07437	447755	ISZ PNTR1 /INCREMENT ALL 1'S
07440	207755	LAC PNTR1 /POINTER TO NEXT LOCATION
07441	547762	SAD UPLIM /UPPER LIMIT?
07442	741000	SKP /YES
07443	607365	JMP EIGHTH-3 /NO
07444	107622	JMS LOOP3 /REPEAT TEST?
07445	607357	JMP SEVENH+4 /YES
07446	750004	LAS
07447	507753	AND MASK5
07450	740200	SZA /HALT?
07451	740040	XX /YES
07452	607204	JMP FIRST+2 /REPEAT ALL TESTS
		.EJECT

```

/ERROR PRINT ROUTINE
/
07453 000000 ERROR 0
07454 047622 DAC ANSWER /STORE BAD NUMBER
07455 750004 LAS
07456 742010 RTL
07457 740100 SMA /RING BELL?
07460 607463 JMP .+3 /NO
07461 760207 LAW 207
07462 107606 JMS TYPE
07463 741400 SZL /PRINT ERRORS
07464 607501 JMP HALT1 /NO
/YES

07465 607506 CHANGE JMP TITLE
07466 207756 LAC POINT
07467 107536 JMS PRINT /TYPE OUT ADDRESS
07470 760240 LAW 240
07471 107606 JMS TYPE /1 SPACE
07472 207761 LAC TEMP
07473 107536 JMS PRINT /TYPE OUT CORRECT ANSWER
07474 760240 LAW 240
07475 107606 JMS TYPE /1 SPACE
07476 207622 LAC ANSWER
07477 107536 JMS PRINT /TYPE OUT BAD ANSWER
07500 107614 JMS CRLF
07501 750004 HALT1 LAS
07502 741100 SPA /HALT ON ERROR?
07503 740040 XX /YES
07504 207756 LAC POINT /RESTORE AC
07505 627453 JMP* ERROR /EXIT

/
07506 760000 TITLE LAW 0
07507 047465 DAC CHANGE /CHANGE CHANGE
07510 107614 JMS CRLF
07511 107516 JMS TMESS1 /TYPE OUT APPROPRIATE MESSAGE
07512 207770 LAC ZZZ5
07513 047754 DAC PNTR
07514 107516 JMS TMESS1 /TYPE OUT REST OF HEADER
07515 607466 JMP CHANGE+1
.EJECT

```

```

/MESSAGE PRINT SUBROUTINE
/
TMESS1      0
            LAC* PNTR
            RTR
            RTR
            RTR
            RTR
            RAR
            JMS TYPE
            SAD RUR0UT
            JMP* TMESS1
            LAC* PNTR
            JMS TYPE
            SAD RUR0UT
            JMP* TMESS1
            ISZ PNTR
            JMP TMESS1+1
/
/PRINT CONTENTS OF AC IN OCTAL
/PRINT      0
            DAC TEM1
            LAW 17772
            DAC TALLY
            LAC TEM1
            RAL
            RAL
            RTL
            DAC TEM1
            AND SEVEN
            XOR ASKII
            JMS TYPE
            LAC TEM1
            ISZ TALLY
            JMP PRINT+6
            JMP* PRINT
            .EJECT

```

```

07516  000000
07517  227754
07520  742020
07521  742020
07522  742020
07523  742020
07524  740020
07525  107606
07526  547757
07527  627516
07530  227754
07531  107606
07532  547757
07533  627516
07534  447754
07535  607517

```

```

07536  000000
07537  047556
07540  777772
07541  047572
07542  207556
07543  740010
07544  740010
07545  742010
07546  047556
07547  507760
07550  247742
07551  107606
07552  207556
07553  447572
07554  607544
07555  627536

```

```

007556      TEM1=.
07556 000000 /SOME USEFUL SUBROUTINES
07557 750004 /
07560 742010 BELL      0
07561 741100          LAS
07562 627556          RTL
07563 447744          SPA
07564 627556          JMP* BELL
07565 407202          ISZ COUNT
07566 047744          JMP* BELL
07567 760207          XCT FIRST
07570 107606          DAC COUNT
07571 627556          LAW 207
                                JMS TYPE
                                JMP* BELL

007572      /
07572 000000 TALLY=.
07573 750004 BELL1    0
07574 742010          LAS
07575 741100          RTL
07576 627572          SPA
07577 447745          JMP* BELL1
07600 627572          ISZ COUNT1
07601 407200          JMP* BELL1
07602 047745          XCT FIRST-2
07603 760207          DAC COUNT1
07604 107606          LAW 207
07605 627572          JMS TYPE
                                JMP* BELL1

07606 000000      /
07607 507757      TYPE    0
07610 700406          AND RUBOUT
07611 700401          TLS
07612 607611          TSF
07613 627606          JMP .-1
                                JMP* TYPE

07614 000000      /
07615 760215      CRLF   0
07616 107606          LAW 215
07617 760212          JMS TYPE
07620 107606          LAW 212
07621 627614          JMS TYPE
                                JMP* CRLF
                                007622
07622 000000      ANSWER=.
07623 750004      LOOP3  0
07624 507751          LAS
07625 741200          AND MASK3
07626 447622          SNA
07627 627622          ISZ LOOP3
                                JMP* LOOP3
                                .EJECT

```

```

/ERROR MESSAGE 1
/
MESS1      311316      /I,N
           304311      /D,I
           326311      /V,I
           304325      /D,U
           301314      /A,L
           240314      /SP,L
           317303      /O,C
           301324      /A,T
           311317      /I,O
           316240      /N,SP
           324305      /T,E
           323324      /S,T
           215212      /CR,LF
           377000      /RO

/
/ERROR MESSAGE 2
/
MESS2      306317      /F,O
           322327      /R,W
           301322      /A,R
           304240      /D,SP
           327322      /W,R
           311324      /I,T
           305255      /E,-
           322305      /R,E
           301304      /A,D
           240324      /SP,T
           305323      /E,S
           324215      /T,CR
           212377      /LF,RO
.EJECT

```

/ERROR MESSAGE 3

07663 322305
 07664 326305
 07665 322323
 07666 305255
 07667 303317
 07670 315320
 07671 314305
 07672 315305
 07673 316324
 07674 240327
 07675 322311
 07676 324305
 07677 255322
 07700 305301
 07701 304240
 07702 324305
 07703 323324
 07704 215212
 07705 377000

/MESS3
 322305
 326305
 322323
 305255
 303317
 315320
 314305
 315305
 316324
 240327
 322311
 324305
 255322
 305301
 304240
 324305
 323324
 215212
 377000
 .EJECT

/RE
 /V E
 /R S
 /E -
 /C O
 /M P
 /L E
 /M E
 /N T
 /SP W
 /R I
 /T E
 /- R
 /E A
 /D SP
 /T E
 /S T
 /CR LF
 /RO

		/ERROR MESSAGE 4	
		/	
07706	327322	MESS4	327322 /W,R
07707	311324		311324 /I,T
07710	305240		305240 /E,SP
07711	317316		317316 /O,N
07712	305323		305323 /E,S
07713	240311		240311 /SP,I
07714	316240		316240 /N,SP
07715	301240		301240 /A,SP
07716	306311		306311 /F,I
07717	305314		305314 /E,L
07720	304240		304240 /D,SP
07721	317306		317306 /O,F
07722	240332		240332 /SP,z
07723	305322		305322 /E,R
07724	317323		317323 /O,S
07725	215212		215212 /CRLF
07726	377000		377000 /RUBOUT
		/	
		/REST OF HEADER	
07727	301304	MESS5	301304 /A,D
07730	304322		304322 /D,R
07731	305323		305323 /E,S
07732	323240		323240 /S,SP
07733	307317		307317 /G,O
07734	317304		317304 /O,D
07735	240240		240240 /SP,SP
07736	240302		240302 /SP,B
07737	301304		301304 /A,D
07740	215212		215212 /CR.LF
07741	377000		377000 /RO.
			.EJECT

/CONSTANTS AND VARIABLES

```
07742 000260 ASKII 260
07743 007403 CONST EIGHTH+13
07744 000000 COUNT 0
07745 000000 COUNT1 0
07746 607506 JMPT JMP TITLE
07747 040000 MASK1 40000
07750 020000 MASK2 20000
07751 010000 MASK3 10000
07752 004000 MASK4 4000
07753 002000 MASK5 2000
07754 000000 PNTR 0
07755 000000 PNTR1 0
07756 000000 POINT 0
07757 000377 RUBOUT 377
07760 000007 SEVEN 7
07761 000000 TEMP 0
07762 007167 UPLIM FIRST-13
07763 007166 UPLIM1 FIRST-14
07764 007630 ZZZ1 MESS1
07765 007646 ZZZ2 MESS2
07766 007663 ZZZ3 MESS3
07767 007706 ZZZ4 MESS4
07770 007727 ZZZ5 MESS5
000000 .END
NO ERROR LINES
```

ANSWER	007622
ASKII	07742
BELL	07556
BELL1	07572
CHANGE	07465
CLOF	700004
CLON	700044
CLSF	700001
CONST	07743
COUNT	07744
COUNT1	07745
CRLF	07614
EIGHTH	07370
ERROR	07453
FIFTH	07310
FIRST	07202
FOURTH	07272
HALT1	07501
JMPT	07746
KRB	700312
KSF	700301
LOOP3	07622
MASK1	07747
MASK2	07750
MASK3	07751
MASK4	07752
MASK5	07753
MESS1	07630
MESS2	07646
MESS3	07663
MESS4	07706
MESS5	07727
PCF	700202
PNTR	07754
PNTR1	07755
POINT	07756
PRINT	07536
PSA	700204
PSB	700244
PSF	700201
RCF	700102
RRB	700112
RSA	700104
RSB	700144
RSF	700101
RUR0UT	07757
SECOND	07225
SEVEN	07760
SEVENH	07353
SIXTH	07326
TALLY	007572
TCF	700402
TEMP	07761
TEM1	007556
THIRD	07257

TITLE	07506
TLS	700406
TMESS1	07516
TSF	700401
TYPE	07606
UPLIM	07762
UPLIM1	07763
ZZZ1	07764
ZZZ2	07765
ZZZ3	07766
ZZZ4	07767
ZZZ5	07770

FIRST	07202
SECOND	07225
THIRD	07257
FOURTH	07272
FIFTH	07310
SIXTH	07326
SEVENH	07353
EIGHTH	07370
FRROR	07453
CHANGE	07465
HALT1	07501
TITLE	07506
TMESS1	07516
PRINT	07536
BELL	07556
TEM1	007556
BELL1	07572
TALLY	007572
TYPE	07606
CRLF	07614
ANSWER	007622
LOOP3	07622
MESS1	07630
MESS2	07646
MESS3	07663
MESS4	07706
MESS5	07727
ASKII	07742
CONST	07743
COUNT	07744
COUNT1	07745
JMPT	07746
MASK1	07747
MASK2	07750
MASK3	07751
MASK4	07752
MASK5	07753
PNTR	07754
PNTR1	07755
POINT	07756
RUR0UT	07757
SEVEN	07760
TEMP	07761
UPLIM	07762
UPLIM1	07763
ZZZ1	07764
ZZZ2	07765
ZZZ3	07766
ZZZ4	07767
ZZZ5	07770
CLSF	700001
CLOF	700004
CLON	700044
RSF	700101
RCF	700102

RSA	700104
RRB	700112
RSR	700144
PSF	700201
PCF	700202
PSA	700204
PSB	700244
KSF	700301
KRB	700312
TSF	700401
TCF	700402
TLS	700406