

IDENTIFICATION

Product Code: DEC-08-COCO-D  
Product Name: ODT-8  
Date Created: October, 10, 1968  
Maintainer: Software Service Group



1      ABSTRACT

ODT (Octal Debugging Technique) is a debugging aid for the PDP-8, which facilitates communication with, and alteration of, the program being run. Communication between operator and program occurs via the Teletype, using defined commands and octal numbers. This version of ODT has been completely revised and replaces both versions of the former ODT-II program.

2      PRELIMINARY REQUIREMENTS2.1    Equipment

Standard PDP-8 or PDP-5 with basic 4k memory and Teletype.

2.2    Storage

ODT requires 600 (octal) consecutive core locations and one location on page 0 which will be used as an intercom register. It is page relocatable.

3      LOADING OR CALLING PROCEDURE

NOTE: ODT cannot be called as a subroutine.

- a. ODT is normally distributed in binary with the source available on request and is loaded with the Binary Loader.
  1. Place the ODT tape in the reader.
  2. Set 7777 in the SWITCH REGISTER and press LOAD ADDRESS. (If using the high-speed photoelectric reader, put switch 0 down).
  3. Press START.
- b. Load the binary tape of the program to be debugged in the same manner as ODT was loaded. Be sure that the two do not overlap.

4      USING THE PROGRAM OR ROUTINE4.1    Starting Procedure

- a. The starting address of ODT is the address of the symbol START. For standard library versions the high version starts at 7000 and the low at 1000.
- b. Set the starting address in the SWITCH REGISTER. Press LOAD ADDRESS, and START on the console. ODT will issue a carriage return and line feed to indicate that it is now running and awaiting commands from the keyboard.
- c. To restart ODT without clearing the checksum, set the address of START + 1 (usually 7001 high version, or 1001 low version) into the SWITCH REGISTER and press LOAD ADDRESS and START on the console.



Example:      404/3270†                      3270 symbolically is "DCA, this page, relative  
                   0470/0212 0000 )                      location 70," so ODT opens register 470.  
                   404/3270†  
                   0470/0000

e. Back Arrow ( ← ) - Close register, open indirectly.

Back arrow will also close the currently open register and then interrupt its contents as the address of the register whose contents it is to type and open for modification.

Example:      365/5760†  
                   0360/0426 ←  
                   0426/5201

f. Any Illegal Character

Any character that is neither a valid control character nor an octal digit, or is the fifth octal digit in a series, causes the current line to be ignored and a question mark typed.

Example:      4: ? }                      ODT opens no register.  
                   4U ? }                      ODT ignores modification and closes register 406.  
                   406/4671 67K ? }  
                   /4671 )

g. xxxxG - Transfer control to user at location xxxx.

Clear the AC then go to the location specified before the G. All indicators and registers will be initialized and the break-trap, if any, will be inserted. Typing G alone is an error but will nevertheless cause a jump to location 0.

h. xxxxB - Set breakpoint at user location xxxx.

Conditions ODT to establish a breakpoint at the location specified before the B. If B is typed alone, ODT removes any previously established breakpoint and restores the original contents of the break location. A breakpoint may be changed to another location, whenever ODT is in control, by simply typing xxxxB where xxxx is the new location. Only one breakpoint may be in effect at one time; therefore, requesting a new breakpoint removes any previously existing one. The previous restriction on placing a breakpoint on a JMS followed by arguments has been removed as of the June 1967 revision. This means ODT can now be more effectively used, especially in debugging programs which utilize floating point. The only restriction in this regard is that a breakpoint may not be set on any of the floating point instructions which appear as arguments of a JMS.

Example:	TAD } DCA } JMS } FADD	Breakpoint legal here.  Breakpoint illegal here.
----------	---------------------------------	--

The breakpoint (B) command does not make the actual exchange of ODT instruction for user instruction, it only sets up the mechanism for doing so. The actual exchange does not occur until a "go to" or a "proceed from breakpoint" command is executed.

When, during execution, the user's program encounters the location containing the breakpoint, control passes immediately to ODT (via location 0004). The C(AC) and C(L) at the point of interruption are saved in special registers accessible to ODT. The user instruction that the breakpoint was replacing is restored, before the address of the trap and the content of the AC are typed. The restored instruction has not been executed at this time. It will not be executed until the "proceed from breakpoint" command is given. Any user register, including those containing the stored AC and Link, can now be modified in the usual manner. The breakpoint can also be moved or removed at this time.

i. A - Open register containing AC.

When the breakpoint is encountered the C(AC) and C(L) are saved for later restoration. Typing A after having encountered a breakpoint, opens for modification the register in which the AC was saved and types its contents. This register may now be modified in the normal manner (see SLASH) and the modification will be restored to the AC when the "proceed from breakpoint" is given.

↓ after A - Open register containing Link

After opening the AC storage register, typing linefeed (↓) closes the AC storage register, then opens the Link storage register for modification and types its contents. The Link register may now be modified as usual (see SLASH) and that modification will be restored to the Link when the "proceed from breakpoint" is given.

j. C - Proceed (continue) from a breakpoint.

Typing C, after having encountered a breakpoint, causes ODT to insert the latest specified breakpoint (if any), restore the contents of the AC and Link, execute the instruction trapped by the previous breakpoint, and transfer control back to the user program at the appropriate location. The user program then runs until the breakpoint is again encountered.

NOTE: If a trap set by ODT is not encountered while ODT is running the object (user's) program, the instruction which causes the break to occur will not be removed from the user's program.

xxxC - Continue and iterate loop xxx times before break.

The programmer may wish to establish the breakpoint at some location within a loop of his program. Since loops often run to many iterations, some means must be available to prevent a break from occurring each time the break location is encountered. This is the function of xxxC (where xxx is an octal number). After having encountered the breakpoint for the first time, the user specifies, with this command, how many times the loop is to be iterated before another break is to occur. The break operations have been described previously in section h.

k. M - Open search mask.

Typing M causes ODT to open for modification the register containing the current value of the search mask and type its contents. Initially the mask is set to 7777. It may be changed by opening the mask register and typing the desired value after the value typed by ODT, then closing the register.

↓ - Open lower search limit

The register immediately following the mask storage register contains the location at which the search is to begin. Typing line feed (↓) to close the mask register causes this, the lower search limit register to be opened for modification and its contents typed. Initially the lower search limit is set to 0001. It may be changed by typing the desired lower limit after that typed by ODT, then closing the register.

↓ - Open upper search limit

The next sequential register contains the location with which the search is to terminate. Typing line feed (↓) to close the lower search limit register causes this; the upper search limit register to be opened for modification and its contents typed. Initially, the upper search limit is the beginning of ODT itself, 7000 (1000 for low version). It may also be changed by typing the desired upper search limit after the one typed by ODT, then closing the register with a carriage return.

l. xxxxW - Word search.

The command xxxxW (where xxxx is an octal number) will cause ODT to conduct a search of a defined section of core, using the mask and the lower and upper limits which the user has specified, as indicated in section k. Word searching using ODT is similar to word

searching using DDT. The searching operations are used to determine if a given quantity is present in any of the registers of a particular section of memory.

The search is conducted as follows: ODT masks the expression xxxx which the user types preceding the W and saves the result as the quantity for which it is searching. (All masking is done by performing a Boolean AND between the contents of the mask register, C(M), and the register containing the thing to be masked.) ODT then masks each register within the user's specified limits and compares the result to the quantity for which it is searching. If the two quantities are identical, the address and the actual unmasked contents of the matching register are typed and the search continues until the upper limit is reached.

A search never alters the contents of any registers.

Example: Search locations 3000 to 4000 for all ISZ instructions, regardless of what register they refer to (i.e. search for all registers beginning with an octal 2).

M7777	7000↓	Change the mask to 7000, open lower search limit
7453/0001	3000↓	Change the lower limit to 3000, open upper limit
7454/7000	4000↓	Change the upper limit to 4000, close register
2000W		Initiate the search for ISZ instructions
<u>2000/2467</u>		These are 4 ISZ instructions in this section of core.
<u>3057/2501</u>		
<u>3124/2032</u>		
<u>4000/2152</u>		

m. T - Punch leader

ODT is capable of producing leader (code 200) on-line. This is done by typing T and then turning ON the punch. When enough leader has been punched, turn off the punch and hit STOP on the console. It is imperative that the punch be turned OFF before typing again on the keyboard, since anything typed will be punched also, if the punch is left on. To issue any further commands, reload the starting address and press START on the console.

n. xxxx; yyyyP - Punch binary

To punch a binary core image of a particular section of core, the above command is used where xxxx is the initial (octal) address and yyyy is the final (octal) address of the section of core to be punched. The computer will halt (with 7402 displayed) to allow the user to turn ON the punch. Pressing CONTINUE on the console initiates the actual punching of



the block. The punching terminates without having punched a checksum, to allow subsequent blocks to be punched and to allow an all inclusive checksum to be punched at the end by a separate command. This procedure is optional, however, and the user may punch individually checksummed blocks.

It is imperative that the punch be turned OFF before typing another command, since the keyboard and punch are linked.

o. E - Punch checksum and trailer

Given the command E, ODT will halt to allow the punch to be turned on. Pressing CONTINUE on the console will cause it to punch the accumulated checksum for the preceding block(s) of binary output followed by trailer (code 200). When a sufficient length of trailer has been output, turn OFF the punch and press STOP on the console. To continue with ODT reload the starting address and press START on the console.

The binary tape produced in this manner by ODT can now be loaded into core and run. However, the changes should be made to the symbolic source tapes as soon as possible.

### 4.3 Additional Techniques

a. TTY I/O-Flag

Sometimes the program being debugged may require that the TTY flag be up before it can continue output, i.e., the program output routine will be coded as follows:

```
TSF
JMP .-1
TLS
```

Since ODT normally leaves the TTY flag in an off (lowered) state, the above coding will cause the program to loop at the JMP.-1. To avoid this, ODT may be modified to leave the TTY flag in the raised (on) state when transferring control through either a "go to" or a "continue" command. This modification is accomplished by changing location XCONT-3 (normally at 7341) to a NOP (7000). To make the actual change, load ODT as usual.

Open register XCONT-3 and modify it as follows:

```
7341/6042 7000 ) (1341/6042 7000 ) for low version)
```

b. Current Location

The address of the current register or last register examined is remembered by ODT and remains the same, even after the commands G, C, B, T, E, and P. This location may be opened for inspection merely by typing /.

c. Programs Written in ODT Commands

ODT will also correctly read tapes prepared off-line (e.g., a tape punched with 1021/1157† 7775 will cause location 1021 to be opened and changed to 1157; then the memory reference address 157 will be opened and changed to 7775 (-3). This procedure will work with breakpoints, continues, punch commands, etc. Thus, debugging programs may be read into ODT to execute the program, list registers of interest, modify locations, etc.

d. Binary Tape from High Speed Punch

It is possible to obtain a binary tape from the high speed punch, instead of the Teletype, however, this requires switch manipulation. Proceed as follows:

1. Type the punch command xxxx; yyyyP as explained in section 4.2 (n). The computer will halt.
2. Set 7231 (1231 for low version) in the SWITCH REGISTER (SR) and press LOAD ADDRESS.
3. Set 6026 in the SR and press DEPOSIT.
4. Set 6021 in the SR and press DEPOSIT.
5. Set 7225 (1225 for low version) in the SR and press LOAD ADDRESS and START on the console, and leader (code 200) will be output.
6. When a sufficient length of leader has been produced, press STOP on the console.
7. Set 7203 (1203 for low version) in the SR and press LOAD ADDRESS and START on the console, and the section of core specified in the punch command will be output.
8. If another block of data is desired on the same tape, the original contents of the locations changed in steps 3, 4 and 5 must be replaced. (See step 11.) Steps 1, 2, 3, 4, and 8 must then be repeated to output the data block via the high speed punch.
9. Set 7222 (1222 for low version) in the SR and press LOAD ADDRESS and START on the console, and the accumulated checksum will be punched followed by trailer (code 200).
10. When a sufficient amount of trailer has been produced, press STOP on the console and press the TAPE FEED button, then remove the tape from the punch.

11. To continue using ODT, the locations changed in steps 3 and 4 must be restored as follows:

Set 7231 (1231 for low version) in the SR and press LOAD ADDRESS.

Set 6046 in the SR and press DEPOSIT.

Set 6041 in the SR and press DEPOSIT.

12. Set the starting address (7000 or 1000) in the SR and press LOAD ADDRESS and START on the console, and ODT is ready to go again.

e. Interrupt Program Debugging

ODT executes an IOF when a breakpoint is encountered. (It does not do this when more iterations remain in an x-continue command.) This is done so that an interrupt will not occur when ODT types out the breakpoint information. It thus protects itself against spurious interrupts and may be used safely in debugging programs that turn on the interrupt mode.

However, the user must remember that there is no way in which ODT could know whether the interrupt was on when the breakpoint was encountered, and hence it does not turn on the interrupt when transferring control back to the program after receiving a "go" or a "continue" command.

f. Octal Dump

By setting the search mask to zero and typing W, all locations between the search limits will be printed on the Teletype.

g. Indirect References

When an indirect memory reference instruction is encountered, the actual address may be opened by typing † and ←.

4.4 Errors

The only legal inputs are control characters and octal digits. Any other character will cause the character or line to be ignored and a question mark to be typed out by ODT. Typing G alone is an error. It must be preceded by an address to which control will be transferred. This will elicit no question mark also if not preceded by an address, but will cause control be transferred to location 0.

Typing any punch command with the punch ON is an error and will cause ASCII characters to be punched on the binary tape. This means the tape cannot be loaded and run properly.

#### 4.5 Miscellaneous

If a trap set by ODT is not encountered by the user's program, the breakpoint instruction will not be removed. ODT can now be used to debug programs using floating point, since the intercom register is now register 0004, and since breakpoints may now be set on a JMS with arguments following. This version of ODT will operate on a Teletype with an ALT mode key or an ESCAPE key. To restart ODT without clearing the checksum, set the SWITCH REGISTER to the value of start + 1 (7001 or 1001 in library versions) and press LOAD ADDRESS and START on the console. The high speed punch may be used by patching three locations after typing the punch command. (See section 4.3 d.)

### 5 DETAILS OF OPERATION AND STORAGE

#### 5.1 Features

ODT features include register examination and modification; binary punchouts (to the Teletype or high speed punch) of user designated blocks of memory; octal core dumps to the Teletype using the word search mechanism, as in DDT; and instruction breakpoints to return control to ODT (breakpoints). ODT makes no use of the program interrupt facility and will not operate outside of the core memory bank in which it is residing.

The breakpoint is one of ODT's most useful features. When debugging a program, it is often desirable to allow the program to run normally up to a predetermined point, at which the programmer may examine and possibly modify the contents of the accumulator (AC), the Link (L), or various instruction or storage registers within his program, depending on the results he finds. To accomplish this, ODT acts as a monitor to the user program. The user decides how far he wishes the program to run and ODT inserts an instruction in the user's program which, when encountered, causes control to transfer back to ODT. ODT immediately preserves in designated storage registers, the contents of the AC and L at the break. It then prints out the location at which the break occurred, as well as the contents of the AC at that point. ODT will then allow examination and modification of any register of the user's program (or those registers storing the AC and L). The user may also move the breakpoint, and request that ODT continue running his program. This will cause ODT to restore the AC and L, execute the trapped instruction and continue in the user's program until the breakpoint is again encountered or the program terminated normally.

#### 5.2 Storage

ODT requires 600 (octal) locations and, as distributed by the Program Library, resides in memory between 7000 and 7577 (or 1000 and 1577 for the low version). It is, however, page relocatable.

The source tape can be re-originated to the start of any memory page except page 0 and assembled to reside in the three pages following that location, assuming they are all in the same memory bank. ODT also uses location 4 on page 0 as an intercom register between itself and the user's program when executing a breaktrap. If the user wishes to change the location of the intercom register, he may do so by changing the value of ZPAT in the source and reassembling. The intercom register must remain on page 0.

## 6 RESTRICTIONS

- a. ODT will not operate outside of the memory bank in which it is located.
- b. It must begin at the start of a memory page (other than page 0) and must be completely contained in one memory bank.
- c. It will not turn on the program interrupt, since it has no way of knowing if the user's program is using the interrupt. It does, however, turn off the interrupt when a breakpoint is encountered, to prevent spurious interrupts. (See 4.3 (e).)
- d. The user's program must not use or reference any core locations occupied or used by ODT, and vice versa.
- e. Register ZPAT is used as an intercom register by ODT when executing a breakpoint. In library distributed versions ZPAT = 0004. This register must be left free by the user since it is filled with an address within ODT which is used to transfer control between user program and ODT.
- f. Breakpoints are fully invisible to "open register" commands; however, breakpoints may not be placed in locations which the user program will modify in the course of execution or the breakpoint will be destroyed.

## 7 REFERENCES

- a. See DDT Programming Manual (Digital-8-4-S) for a full explanation of the use of debugging programs.
- b. Binary Loader (Digital-8-2-U).

## 8 COMMAND SUMMARY

nnnn/	Open register designated by the octal number nnnn. Reopen latest opened register.
/	Reopen latest opened register.
Carriage Return ( ) )	Close previously opened register.

Line Feed (↓)	Close register and open the next sequential one for modification .
Up Arrow (↑)	Close register, take contents of that register as a memory reference and open it .
Back Arrow (←)	Close register open indirectly .
Illegal character	Current line typed by user is ignored, ODT types "? CR LF" .
nnnnG	Transfer program control to location nnnn .
nnnnB	Establish a breakpoint at location nnnn .
B	Remove the breakpoint .
A	Open for modification the register in which the contents of AC were stored when the breakpoint was encountered .
C	Proceed from a breakpoint .
nnnnC	Continue from a breakpoint and iterate past the breakpoint nnnn times before interrupting the user's program at the breakpoint location .
M	Open the search mask .
(line feed)	Open lower search limit .
(line feed)	Open upper search limit .
nnnnW	Search the portion of core as defined by the upper and lower limits for the octal value nnnn .
T	Punch leader .
nnnn;mmmmP	Punch a binary core image defined by the limits nnnn and mmmm .
E	Punch checksum and trailer .

## 9 EXAMPLES AND/OR APPLICATIONS

Symbols for representing "invisible" Teletype actions:

(CR)	=	Carriage Return
(LF)	=	Line Feed
(H)	=	Computer Halts
(Cont)	=	Key Continue on Console
(PON)	=	Punch On

(POF) = Punch Off  
 (LEAD) = Production of Leader  
 (BIN) = Punching of Binary Text  
 (CKSMT) = Punching of Checksum and Trailer

The following examples are the actual result of using ODT to run the program listed after the examples. Brackets enclose comments local to the description. Underlinings designate that produced by ODT.

M7777 7332 (LF)(CR)  
7473 7332 433 (LF)(CR)  
7474 7733 533 (CR)(LF)  
3333 (CR)(LF)  
0404 73272 (CR)(LF)  
0431 73277 (CR)(LF)  
0437 73277 (CR)(LF)  
0444 73322 (CR)(LF)  
0452 73277 (CR)(LF)  
0453 73322 (CR)(LF)  
0455 73276 (CR)(LF)  
0455 73277 (CR)(LF)  
(LF)

[ mask modified ]  
 [ lower search limit modified ]  
 [ upper search limit modified ]  
 [ quantity for which to search specified and search begun ]

[ search completed ]

M7332 7777 (LF)(CR)  
7473 70433 333 (LF)(CR)  
7474 78533 (CR)(LF)  
72222 (CR)(LF)  
3364 72222 (CR)(LF)  
(LF)

[ change mask ]  
 [ change lower limit ]  
 [ upper limit is all right ]  
 [ search for all CLA instructions ]  
 [ there is only one. It is at location 364 ]  
 [ search is finished ]

M7777 532 (CR)(LF)  
4222 (CR)(LF)  
3377 77422 (CR)(LF)  
0411 77452 (CR)(LF)  
0414 77452 (CR)(LF)  
0417 77452 (CR)(LF)  
0432 77422 (CR)(LF)  
0442 77422 (CR)(LF)  
0451 77422 (CR)(LF)  
0462 77542 (CR)(LF)  
0466 77422 (CR)(LF)  
0472 77521 (CR)(LF)  
(LF)

[ set mask for indirect and page bits ]  
 [ using previous limits search for all references to page zero which occur ]

[ there are none, however, these microinstructions look like indirect references to page zero since they have a 1 in bit 3 and a 0 in bit 4 ]

[ search completed ]

```

00500 0 (LF)(CR)
7473 /1360 427 (LF)(CR)
7474 /0500 427 (CR)(LF)
0 (CR)(LF)
0407 /1270 (CR)(LF)
0410 /1270 (CR)(LF)
0411 /7450 (CR)(LF)
0412 /5253 (CR)(LF)
0413 /1273 (CR)(LF)
0414 /7450 (CR)(LF)
0415 /5234 (CR)(LF)
0416 /1273 (CR)(LF)
0417 /7450 (CR)(LF)
0420 /5227 (CR)(LF)
0421 /7001 (CR)(LF)
0422 /7650 (CR)(LF)
0423 /5242 (CR)(LF)
0424 /1274 (CR)(LF)
0425 /4671 (CR)(LF)
0426 /520 (CR)(LF)
0427 /1275 (CR)(LF)
(LF)

```

```

[ set mask to zero so that everything will match ]
[ set search limits to encompass dump area ]
[ since W is typed alone, the word searched for,
  is Ø. The result after masking each register
  with Ø is, of course, Ø so all comparisons
  appear to the program equal and hence all
  unmasked contents are typed, constituting
  a dump ]

```

#### Examples of Register Examination & Modification

```

400/6046 (CR)(LF)
400/6046 2463? (CR)(LF)
400/6046 12345? (CR)(LF)
/6046 2345 (CR)(LF)
/2345 6046 (CR)(LF)
/6046 401/6031 2346 (CR)(LF)
400/6046 401/2346 (CR)(LF)
/2346 6031 (CR)(LF)
/6031

```

```

[ Examine Only ]
[ Non-octal number typed, modification ignored ]
[ More than 4 digits typed, modification ignored ]
[ Register 400 modified to 2345 ]
[ Modified again ]
[ Register closed by typing another command ]

```

```

400/6046 (LF)(CR)
0401 /6031 1234 (LF)(CR)
0402 /5201 (CR)(LF)
401/1234 6031 (LF)(CR)
0402 /5201 (CR)(LF)
(LF)(CR)
0403 /6036 (CR)(LF)
(LF)(CR)
0404 /73270 (CR)(LF)

```

```

[ close and examine next ]
[ modify 401, examine 402 ]
[ close 402 ]

```



Examples of Register Examination & Modification (continued)

<u>404/3270</u> ↑ (CR)(LF)	[ contents of 404 refers to "this page, loc. 70"]
<u>0470/0212</u> <u>0230</u> (CR)(LF)	[ ODT opens 470. User modifies 470]
<u>404/3270</u> ↑ (CR)(LF)	
<u>0470/0000</u> (CR)(LF)	
<u>0000</u> (CR)(LF)	
<u>404/3270</u> <u>3271</u> ↑ (CR)(LF)	[ contents of 404 modified to refer to "this page
<u>0471/0360</u> (CR)(LF)	[ ODT opens 471] loc. 71"]
<u>404/3271</u> <u>3272</u> ↑ (CR)(LF)	
<u>0470/0000</u> (CR)(LF)	

<u>365/5760</u> ↑ (CR)(LF)	[ contents of 365 refers to "this page, loc. 160"]
<u>0360/0426</u> (CR)(LF)	[ ODT opens 360. Contents of 360 become
<u>0426/5201</u> (CR)(LF)	[ ODT opens 426] address]

<u>4:?</u> (CR)(LF)	} illegal character. ODT opens no register
<u>4U?</u> (CR)(LF)	
<u>6Q?</u> (CR)(LF)	
<u>405/4671</u> Y? (CR)(LF)	} illegal character. ODT ignores modification fifth digit in series. ODT ignores modification register 406 still contains original value of 4671
<u>405/4671</u> 57K? (CR)(LF)	
<u>406/4671</u> 67322? (CR)(LF)	
<u>/4671</u>	

Examples of setting Breakpoints and Executing User's Program

<u>475/0000</u> 1 (LF)(CR)	} user's program expects to find the numbers it is to use in 475 and 476 (see listing) answer will be stored in 477 [Breakpoint is set at location 432] [user's program begins at 400, go there] [user's program accpts input of "+". Breakpoint [477 contains sum of 475 & 476] encountered ODT types break address & C(AC)
<u>0476/0000</u> 2 (LF)(CR)	
<u>0477/0000</u> (CR)(LF)	
<u>432B</u> (CR)(LF)	
<u>400G</u> (CR)(LF)	
<u>+0432/0000</u> (CR)(LF)	
<u>477/0003</u>	

Registers can be changed and the same breakpoint remains in effect.

<u>475/0001</u> 3 (LF)(CR)
<u>0476/0002</u> (CR)(LF)
<u>400G</u> (CR)(LF)
<u>*0432/0000</u> (CR)(LF)
<u>477/0006</u> (CR)(LF)

Examples of examining and modifying AC and L after encountering a breakpoint

A0200 1 (CR)(LF)

A0001 (CR)(LF)

/0001 (CR)(LF)  
(LF)(CR)

7356 /0001 0 (CR)(LF)

/0000 (CR)(LF)

[AC which contained 0 when breakpoint was encountered is modified]

[Link which contained 1 at break is modified to 0]

446B (CR)(LF)

400G (CR)(LF)

\*0446 (0004 (CR)(LF)

C (CR)(LF)

0446 (0010 (CR)(LF)

00 (CR)(LF)

0446 (0014 (CR)(LF)

[Destroys old breakpoint & sets one at 446]

[Breakpoint encountered]

[continue until ...]

[Breakpoint again encountered]

476/0003 7

/0007

446B

400G

\*0446 (0004

2C

0446 (0020

C

0446 (0024

[Breakpoint encountered]

[Continue as before but pass Breakpoint twice before stopping again]

```

/IT IS A VERY PRIMITIVE CALCULATOR WHICH ADDS,
/SUBTRACTS, MULTIPLIES, OR DIVIDES USING TWO PREVIOUSLY STORED
/DECIMAL NUMBERS, THE ONLY INPUT IT ACCEPTS IS AN
/OPERATOR (*,+,-,/), THE NUMBERS IT OPERATES ON
/CAN BE CHANGED BY THE TOGGLES OR BY ODT, THE RESULT IS STORED IN "ANSR"
*360
TYPE, 0
      TSF
      JMP ,-1
      TLS
      CLA
      JMP I TYPE

*400
0400 6346      TLS          /INITIALIZE TELEPRINTER
0401 6031      READ,      KSF
0402 5201      JMP ,-1
0403 6036      KRB          /READ THE CHAR INTO AC
0404 3270      DCA TEMP     /AND STORE
0405 1270      TAD TEMP
0406 4671      JMS I TYPEJ   /ECHO IT
/ROUTINE TO CHECK INPUT CHARACTER AND JMP TO PROPER ROUTINE
/DEPENDING ON WHICH OPERATOR IT WAS,
LEGL, TAD TEMP     /GET OPERATOR
      TAD M257      /IS IT A SLASH (257)?
      SNA
      JMP DVID      /YES, GO TO DIVIDE ROUTINE
      TAD C2        /NO! IS IT A MINUS SIGN (255)?
      SNA
      JMP SUBT      /YES, GO TO SUBTRACT ROUTINE
      TAD C2        /NO! IS IT A PLUS SIGN (253)?
      SNA
      JMP ADD       /YES, GO TO ADDITION ROUTINE
      JAC          /NO! IS IT AN ASTERISK (252)?
      SNA CLA
      JMP MULT      /YES, GO TO MULTIPLY ROUTINE
      TAD C277      /NO IS IT NOT A LEGAL OPERATOR
      JMS I TYPEJ   /TYPE A QUESTION MARK
      JMP READ      /AND GO LISTEN FOR ANOTHER OPERATOR
/ROUTINE TO ADD NUMBER IN "STOR2" TO NUMBER IN
/STOR1 AND DEPOSIT SUM IN "ANSR", (STOR1+STOR2=ANSR)
/PRESSING "CONTINUE" WILL CAUSE PROGRAM TO LISTEN FOR ANOTHER
/OPERATOR,
0427 1275      ADD,       TAD STOR1
0430 1276      TAD STOR2
0431 3277      DCA ANSR
0432 7402      HLT
0433 5201      JMP READ

```

```

/ROUTINE TO SUBTRACT STOR2 FROM STOR1 AND PUT DIFFERENCE IN ANSR,
/STOR1-STOR2=ANSR, PRESS CONTINUE TO ENTER ANOTHER OPERATOR
0434 1276 SUHT, TAD STOR2 /GET NEGATIVE TWO'S COMPLEMENT
0435 7041 CIA /OF STOR2
0436 1275 TAD STOR1
0437 3277 UCA ANSR
0440 7402 HLT
0441 5201 JMP READ

```

```

/ROUTINE TO MULTIPLY STOR1 BY STOR2 AND PUT PRODUCT IN ANSR,
/(STOR1*STOR2=ANSR) THIS IS DONE BY ADDING STOR1 TO STOR1,
/N TIMES WHERE N=STOR2,
/PRESS CONTINUE TO ENTER ANOTHER OPERATOR,
0442 1276 MULT, TAD STOR2 /STORE NEGATIVE TWO'S COMPLEMENT
0443 7041 CIA /IOF STOR2 AS THE NUMBER OF TIMES
0444 3300 UCA CNTR /TO REPEAT THE ADDITION
0445 1275 TAD STOR1
0446 2300 ISZ CNTR
0447 5245 JMP ,=2
0450 3277 UCA ANSR
0451 7402 HLT
0452 5201 JMP READ

```

```

/ROUTINE TO DIVIDE STOR1 BY STOR2 AND STORE IN ANSR (STOR1/ STOR2=ANSR),
/THIS IS DONE BY SUCCESSIVELY SUBTRACTING TO ZERO, COUNTING THE NUMBER
/OF SUBTRACTIONS AND STORING IT IN ANSR
0453 3300 DVID, UCA CNTR /PUT 0 IN COUNTER
0454 1276 TAD STOR2
0455 7041 CIA /NEGATE STOR2
0456 3276 UCA STOR2
0457 1275 TAD STOR1
0460 1276 TAD STOR2 /SUBTRACT STOR2 FROM STOR1
0461 2300 ISZ CNTR
0462 7540 SZA SMA /HAS ZERO BEEN REACHED?
0463 5260 JMP ,=3 /NO; SUBTRACT AGAIN
0464 1300 TAD CNTR /YES, CNTR CONTAINS NO OF SUBTRACTIONS
0465 3277 UCA ANSR /PERFORMED
0466 7402 HLT
0467 5201 JMP READ

```

```

0470 0000 TEMP, 0
0471 0360 TYPEJ, TYPE
0472 7521 M257, -257
0473 0002 C2, 2
0474 0277 C277, 277
0475 0000 STOR1, 0
0476 0000 STOR2, 0
0477 0000 ANSR, 0
0500 0000 CNTR, 0
/THIS IS A SAMPLE PROGRAM FOR ODT
$

```

## SYMBOL TABLE

AJD	0427
ANSR	0477
CNTR	0500
CZ	0473
CZ77	0474
UVID	0453
LEGL	0407
MULT	0442
MZ57	0472
HEAD	0401
STOR1	0475
STOR2	0476
SUBT	0434
TEMP	0470
TYPE	0360
TYPEJ	0471

## SYMBOL TABLE

TYPE	0360
READ	0421
LEGL	0427
AUD	0427
SUBT	0434
MULT	0442
DVID	0453
TEMP	0470
TYPEJ	0471
M257	0472
C2	0473
C277	0474
STOR1	0475
STOR2	0476
ANSR	0477
CNTR	0500

```

1000          START=1000
0004          #PAT=4

          /THIS IS A 3-PAGE, 4K,
          /PAGEWISE-RELOCATABLE,
          /OCTAL DEBUGGING SYSTEM CALLED
          /***OUT-8***

1000          *START

1000 3675          DCA I CKSAI          /CLEAR THE CHECKSUM.
1001 0010          P10, 10            /ARBITRARY CONSTANT

1002 4357          READ, JMS CRLF          /END LINE; SET SHUT TO -1
1003 1673          TAD I INX          /TRAD
1004 3367          DCA WORD          /GET THE TRAP ADDRESS,
1005 1674          TAD I IN0          /KEEP
1006 3767          DCA I WORD          /RESTORE CONTENT,
1007 3367          READ5, DCA WORD          /CLEAR THE INPUT, /7TH INST,
1010 1263          TAD FM5            /-5
1011 3374          DCA TOTE          /SET THE LETTER COUNT,
1012 6031          REA, KSF
1013 5212          JMP ,-1            /WAIT FOR COMMAND,
1014 6036          KRB
1015 3357          DCA SCHAR
1016 1357          TAD SCHAR          /GO TYPE THE CHARACTER,
1017 4772          JMS I IN9
1020 1373          TAD RETN          /INITIALIZE THE PATCH
1021 3004          DCA ZPAT          /EVERY TIME,
1022 1243          TAD BLIST          /COMPUTE ADDRESS OF COMMAND.
1023 3323          DCA SPNTR
1024 1723          TAD I SPNTR          /SEARCH FOR LEGAL CHARACTER,
1025 2323          ISZ SPNTR
1026 7510          FM270, SPA          /TEST FOR END OF LIST; MINUS 5
1027 5277          QUEST, JMP SEX          /NOT SATISFIED,
1030 7041          CIA              /COMPARE THE CHARACTER,
1031 1357          TAD SCHAR
1032 7640          FP240, SZA CLA          /FOUND
1033 5224          JMP ,-7            /NO, CONTINUE
1034 1323          TAD SPNTR
1035 1242          TAD LTABL
1036 3323          DCA SPNTR
1037 1723          TAD I SPNTR          /LOOK UP THE ADDRESS,
1040 3323          DCA SPNTR
1041 5723          JMP I SPNTR          /GO PROCESS,

1042 0514          LTABL, TABL2-TABL1-1
1043 1044          BLIST, TABL1

```

```

/OUT-8 WILL ALSO CORRECTLY READ SYMBOLIC
/TAPES PREPARED FOR IT: F.G. 1021/1157+7775

```

## /COMMAND LIST

1044	0320	320	/PUNCH
1045	0305	305	/END
1046	0324	324	/TRAILER
1047	0212	212	/OPEN NEXT
1050	0215	215	/CLOSE THIS ONE
1051	0257	257	/OPEN THIS ONE
1052	0302	302	/BREAK
1053	0307	307	/GO
1054	0273	273	/;
1055	0303	303	/CONTINUE
1056	0327	327	/WORD SEARCH
1057	0336	336	/UP-ARROW OPENS INDIRECT(I.E. MEM REF)
1060	0315	315	/MASK+UPPER+LOWER+
1061	0301	301	/AC+LINK
1062	0337	337	/BACK ARROW = OPEN INDIRECTLY
/TABLE MUST END WITH A NEG NUMBER			
1063	7773	FM5, -5	
1064	1367	EXAM, TAD WORD	/LOAD ADDRESS
1065	7440	SZA	/IF ZERO, USE LAST
1066	3370	UCA CAD	
1067	1770	EX2, TAD I CAD	
1070	4771	JMS I INB	/PNUM (PRINT CONTENTS)
1071	3375	UCA SHUT	/SIGNALS OPEN REG
1072	5207	JMP READ5	
1073	1357	INX, TRAD	
1074	1360	IN0, KEEP	
1075	1363	CKSA1, CKSA	
1076	1362	IN7, FROG	

## /PROCESS OCTAL DIGITS,

1077	7200	SEX, CLA	
1100	1357	TAD SCHAR	
1101	1226	TAU FM270	/(-8)
1102	7500	CKNUM, SMA	
1103	5317	JMP NO	/ILLEGAL CHAR
1104	1201	TAD P10	/10
1105	7510	SPA	
1106	5317	JMP NO	/ILLEGAL CHAR
1107	3323	UCA SAD	
1110	1367	TAD WORD	/ASSEMBLE AN ADDRESS
1111	7104	RAL CLL	
1112	7006	RTL	
1113	1323	TAD SAD	
1114	3367	OCA WORD	
1115	2374	ISZ TOTL	
1116	5212	JMP REA	



10/10/68 0:35.15

PAGE 2-1

```

/TYPE ERROR INDICATOR (?)
1117 7200      NO,      CLA
1120 1227      TAD QUEST      /277
1121 4772      JMS I IN9      /TYPN
1122 5202      JMP READ

/TO OPEN LOCATION ZERO,
/OPEN 7777 AND TYPE LINEFEED.

/THE ADDRESS OF THE LAST REGISTER
/EXAMINED REMAINS THE SAME AND MAY BE OPENED BY "/"

      1123      SP=TR=,
      1123      SADR=,

/ROUTINE TO HANDLE REG, MODIFICATION AND INCREMENTAL EXAMINE
1123 0700      CRL,      4
1124 1374      TAD TOTE
1125 7041      CIA
1126 1263      TAD FM5      /-5
1127 7650      SNA CLA
1130 5723      JMP I CRL      /NO MOD, INFO AVAILABLE
1131 1367      TAD WORU
1132 2375      ISZ SHUT      /TEST FOR OPEN AND THEN CLOSE IT,
1133 3770      DCA I CAD      /MODIFY REGISTER
1134 5723      JMP I CRL

1135 4323      CRL1,     JMS CRL      /CARRIAGE RETURN TO CLOSE
1136 4357      JMS CRLF
1137 5207      JMP READ5

1140 1250      CRL2,     TAD CR      /SINGLE FEED+CR
1141 4772      JMS I IN9
1142 4323      JMS CRL
1143 4772      JMS I IN9      /TIME FOR CAR TO RET,
1144 2370      ISZ CAD      /LINE FEED - EXAMINE NEXT
1145 1370      UPAR3,    TAD CAD
1146 4771      JMS I IN8      /PNUM
1147 1251      TAD SLA
1150 4772      JMS I IN9      /TYPN
1151 5267      JMP EX2

1152 4323      OPIN,     JMS CRL      /CLOSE FIRST
1153 1770      TAD I CAD
1154 3370      DCA CAD
1155 4357      UPAR2,    JMS CRLF
1156 5345      JMP UPAR3

```

1157		SCHARE,	
		/TYPE A CAR. RET, AND LINE FEED	
1157	1207	CRLF,	0
1160	1250	TAU CR	/215
1161	4772	JMS I IN9	/TYPN
1162	1247	TAU LF	/212
1163	4772	JMS I IN9	/TYPN
1164	7740	GMA	/MINUS ONE
1165	3375	DCA SHUT	/SIGNALS CLOSED REGISTER
1166	5757	JMP I CRLF	
		/PAGE ONE PARAMETERS,	
1167	0000	WORD,	0
1170	0000	CAD,	0
1171	1446	IN8,	PNUM /CURRENT ADDRESS
1172	1230	IN9,	TYPN
1173	1243	RETN,	BURP
1174	0000	TOTE,	0
1175	7777	SHUT,	/777
1176	1367	PUNC,	TAU WORD
1177	3676		DCA I IN7

/DJT-8, SECOND CORE PAGE

```

1200
*START+220

1200 0177      SP177, 177      /FIRST IN THIS PAGE
1201 5767      JMP I IN13   /READ5

/PUNCH DATA,
PUN1,  CLA HLT
      TAD FROG
      JMS I IN11   /PUNN (PUNCH ORIGIN)
      100

PUN2,  TAD I FROG
      JMS I IN11   /PUNN (PUNCH CONTENTS)
      0
      TAD FROG
      CIA
      TAD I IN10   /WORD
      SNA CLA
      JMP I IN13   /READ5
      ISZ FROG
      JMP PUN2
      JMP I IN13

/PUNCH END,
PUN3,  CLA HLT
      TAD CKSA
      JMS I IN11   /PUNN (PUNCH CHECKSUM)
      2

/PUNCH LEADER,
PUN4,  TAD SP200
      JMS TYPN
      JMP ,-2

/TO USE THE HIGH SPEED PUNCH,
/TYPE "XX;YYP" THEN TOGGLE IN
/THE PATCHES INDICATED BELOW,
/THEN LOAD ADDRESS AND START:
/PUN4 - FOR LEADER-TRAILER,
/PUN1+1 - FOR DATA
/PUN3+1 - FOR CHECKSUM AND LEADER,
/RESTORE PATCHES BEFORE RESTARTING,
/RESTART AT START TO CLEAR CHECKSUM,
/RESTART AT START+1 TO RETAIN CHECKSUM,

/TYPE A CHARACTER
TYPN,  0
      TLS      /(6026) - FOR H.S.
      TSF      /(6021) - FOR H.S.
      JMP ,=1
SP7600, 7600   /CLA-GROUP2
      JMP I TYPN

```

10/17/68 0:35,16

PAGE 5-1

/FEATURES ADDED: INTERRUPT TURNED OFF UPON HITTING BREAKPOINT; CAN USE  
 /HI SPEED PUNCH; BREAKPOINT CAN BE PUT ON A JMS FOLLOWED BY ARGUMENTS;  
 /OPT-8 IS RELOCATABLE; IF BREAKPOINT PUT ON INSTR REFERENCING AUTO-INDEX  
 /INDIRECTLY, IT WILL BE INCREMENTED ON CONTINUE; LINK & AC EXAMINE ON  
 /COMMAND; / OPENS LATEST OPENED REGISTER; CLARITY; AUTO LEADER/TRAILER;  
 /OPEN MEM. REF.(+); AND OPEN INDIRECT (BACK ARROW); ALSO XXX C.

1236	1764	/SET A BREAK POINT,		
1237	7450	TRAP,	TAD I IN10	/(WORD)-ADDRESS OF TRAP,
1240	1366		SNA	
1241	3357		TAD IN12	/CRLF
1242	5320		DCA TRAD	/TRAP SET (REAL OR DUMMY)
			JMP SEXIT	/GO TO SECOND PAGE EXIT,
/THE TRAP IS SPRUNG				
1243	3355	CURP,	DCA SAC	/SAVE C(AC)
1244	7004		RAL	
1245	3356		DCA LINK	/SAVE C(L)
1246	1360		TAD KEEP	
1247	3757		DCA I TRAD	/REPLACE INSTRUCTION WHICH WAS TRAPPED
1250	7101		IAC CLL	
1251	1357		TAD TRAD	
1252	3361		DCA GAME	/SAVE CONTINUATION ADDRESS (BREAK ADDR+1)
1253	1360		TAD KEEP	/PICK UP TRAPPED INSTRUCTION
1254	1372		TAD SP2000	/OVERFLOW TO LINK IF IOT OR OPERATE INSTR.
1255	0271		AND SP200	/AC=0 IF PAGE 0 REFERENCE
1256	7660		SZA SNL CLA	/WAS TRAPPED INSTR AN IOT,OPER,PAGE 0 REFERENCE?
1257	5265		JMP CURPAG	/NO
1260	4322		JMS TSTJMS	/YES, SEE IF IT WAS A JMS
1261	7650		SNA CLA	
1262	5267		JMP CURPAG+2	/YES, TREAT AS IF NON-PAGE-ZERO REFERENCE
1263	1360		TAD KEEP	/NO, PUT ACTUAL INSTR IN "THE" FOR EXECUTION
1264	5306		JMP LIP4	
1265	1357	CURPAG,	TAD TRAD	
1266	0234		AND SP7600	
1267	3362		DCA FROG	/SAVE INITIAL ADDR OF PAGE REFERENCED BY TRAPPED INSTR.
1270	1360		TAD KEEP	
1271	0200	SP200,	AND SP177	/GET RELATIVE ADDR REFERENCED BY TRAPPED INSTR.
1272	1362		TAD FROG	/ADD ON TOP OF PAGE
1273	3362		DCA FROG	/SAVE ABSOLUTE ADDRESS OF MEMORY REFERENCE
1274	1360		TAD KEEP	
1275	0373		AND SP400	
1276	7650	LPAR,	SNA CLA	/IS IT AN INDIRECT REFERENCE?
1277	5302		JMP LIP	/NO
1300	1762		TAD I FROG	/YES, GET ACTUAL REFERENCE
1301	3362		DCA FROG	

12/10/68 0:35,17

PAGE 6-1

```

1302 4322      LIP,      JMS TSTJMS      /SEE IF TRAPPED INSTR IS A JMS
1303 7450          SNA
1304 4771          JMS I IN21      /YES, IT IS A JMS (JMSEB)
1305 1377          TAD IFROG      /NO      (JMS I FROG) JMS ADDS BACK 4000
1306 3351      LIP4,      DCA THE      /STORE FOR EXECUTION
1307 2765          ISZ I IN11     /TEST N-CONTINUE
1310 5344          JMP XCONT     /IGNORE THIS BREAK

1311 6002          IOF          /STOP INTERRUPTS

1312 1357          TAD TRAD
1313 4770          JMS I IN14     /PNUM      (PRINT TRAP ADDRESS)
1314 1276          TAD LPAR      /LEFT PAREN      (8 BITS=250=ASCII LFT PAREN)
1315 4230          JMS TYPN
1316 1355          TAD SAC
1317 4770          JMS I IN14     /PNUM      (PRINT C(AC))

1320 4766      SPFXII, JMS I IN12     /CRLF
1321 5767          JMP I IN13     /READ5

1322 0000      TSTJMS, 0
1323 1360          TAD KEEP      /GET TRAPPED INSTR,
1324 0374          AND SP7000     /ISOLATE OP CODE
1325 1375          TAD SP4000     /OVERFLOW TO LINK WITH AC=0 IF JMS (4000)
1326 5722          JMP I TSTJMS

/START AT A LOCATION
1327 1764      JUMP,      TAD I IN10     /(WORD)
1330 3361          DCA GAME
1331 1352          TAD JPIGAM     /(JMP I GAME)
1332 3351          DCA THE
1333 3355          DCA SAC      /CLEAR THE AC,
1334 7410          SKP
1335 1764      CONTIN, TAD I IN10     /(WORD)
1336 7040          CMA
1337 3765          DCA I IN11     /(PUNN)-EMP COUNTER,
1340 4766          JMS I IN12     /(CRLF)

/PATCH THE NEXT LOCATION WITH NOP(7000)
/IF THE PROGRAM BEING DEBUGGED EXPECTS
/THE TTY FLAG TO BE UP,
1341 6042          TCF          /CLEAR THE FLAG
1342 1757          TAD I TRAD     /SAVE TRAP CONTENTS,
1343 3360          DCA KEEP

1344 1376      XCONT,  TAD HAIT
1345 3757          DCA I TRAD     /INSERT TRAP INSTRUCTION
1346 1356          TAD LINK
1347 7110          RAR CLL      /RESTORE LINK
1350 1355          TAD SAC      /AND C(AC)
1351 7402      THE,      HLT          /ODD EXECUTION OF TRAPPED INST. AFTER PROCEED
1352 5761          JPIGAM, JMP I GAME
1353 2361          ISZ GAME      /IMITATE SKIP CONDITION,
1354 5352          JMP ,-2

```



10/10/68 0:35.18

PAGE 7-1

/VARIABLES MAY BE SCANNED VIA "A".

1355	0202	SAC,	0	/AC
1356	0200	LINK,	0	/LINK BIT
1357	1157	TRAD,	CRLF	/ADDRESS OF TRAP.
1360	0200	KEEP,	4	/CONTENT OF TRAP
1361	0200	GAME,	0	/ADDRESS FOR CONTINUE
1362	0777	FROG,	START-1	/MEMORY REFERENCE,
1363	0200	CKSA,	0	/THE CHECKSUM TO DATE,
		/INTER COM REGS,		
1364	1167	IN10,	WORD	
1365	1401	IN11,	PUNN	
1366	1157	IN12,	CRLF	
1367	1207	IN13,	HEAD5	
1370	1446	IN14,	PNUM	
1371	1475	IN21,	JMSER	/PROCESS JMS.
		/CONSTANTS		
1372	2200	SP2000,	2000	
1373	0400	SP400,	400	
1374	7000	SP7000,	7000	
1375	4000	SP4000,	4000	
1376	5404	BAIT,	JMP I ZPAT	
1377	4762	IFROG,	JMS I FROG	

/JOB-8, THIRD CORE PAGE.

1400

\*START+400

```

1400 0177          /PUNCH ROUTINE
                          TP177, 177          /FIRST IN THIS PAGE,

1401 0200          PUNN, 0
1402 3246          JCA PNUM
1403 1246          TAD PNUM
1404 7012          RTR
1405 7012          RTR
1406 7012          RTR
1407 0354          AND TP77
1410 1601          TAD I PUNN
1411 4236          JMS CKSM
1412 1246          TAD PNUM
1413 0354          AND TP77
1414 4236          JMS CKSM
1415 5601          JMP I PUNN

/MEMORY REFERENCE OPENER,
1416 4742          UPAR1, JMS I IN30          /((CRL)-"CLOSER CALL",
1417 1741          TAD I IN27          /CAD
1420 3236          DCA TEM
1421 1636          TAD I TEM
1422 0200          TP200, AND TP177
1423 3201          DCA TEM2          /SAVE LOWER BITS,
1424 1636          TAD I TEM
1425 0222          AND TP200
1426 7650          SNA CLA          /TEST FOR PAGE ZERO REF
1427 5232          JMP ,+3          /YES
1430 1741          TAD I IN27
1431 0266          AND TP7600
1432 1201          TAD TEM2
1433 3741          DCA I IN27          /CAD
1434 5635          JMP I ,+1
1435 1155          UPAR2

/CHECK SUM ACCUMULATOR
1436 0200          CKSM, 0
1437 3275          DCA CKT
1440 1746          TAD I IN20          /CKSA
1441 1275          TAD CKT
1442 3746          DCA I IN20          /CKSA
1443 1275          TAD CKT
1444 4745          JMS I IN19          /TYPN
1445 5636          JMP I CKSM

```

```

1446 0200 /ROUTINE TO PRINT OCTAL CONTENTS OF AC
1447 3201 PNUM, 0
1450 1352 DCA PUNN
1451 3236 TAD TM4
1452 1201 DCA CKSM
1453 7004 TAD PUNN
1454 7004 RAL
1455 7006 PN2, RAL
1456 3201 RTL
1457 1201 DCA PUNN
1460 0351 TAD PUNN
1461 1355 AND TP007 /ONLY 7-DIGITS GUARANTEED,
1462 4745 TAD TP60 /IN CASE BIT 8 CAME THROUGH,
1463 1201 JMS I IN19 /TYPN
1464 2236 TAD PUNN
1465 5254 ISZ CKSM
1466 7600 JMP PN2
1467 1331 TP7600, 7600 /CLA-GROUP2
1470 4745 TAD TP240
1471 5646 JMS I IN19
JMP I PNUM

/SEARCH VARIABLES,
1472 7777 MASK, 7777
1473 0201 LIMLO, 0201
1474 1200 LIMHI, START

1475 1475 CKT=,
1475 0200 JMSEK, 0
1476 1747 TAD I IN22 /(FROG)=ABS MEM REF, (FINAL)
1477 3246 DCA PNUM
1700 1750 TAD I IN23 /GAME
1701 3646 DCA I PNUM /SIMULATED JMS
1702 2747 ISZ I IN22 /FROG
1703 1353 TAD TP1000
1704 5675 JMP I JMSEK

```

```

/WORD SEARCH ROUTINE
1205 4743      WSER,   JMS I IN16      /CRLF
1206 1273      TAD LIMLO
1207 3275      DCA CKT
1210 1675      WSER1,  TAD I CKT
1211 0272      AND MASK
1212 7041      CIA
1213 1744      TAD I IN17      /WORD
1214 7640      SZA CLA
1215 5325      JMP WSER2
1216 1275      TAD CKT
1217 4246      JMS PNUM
1220 1357      TAD TP257      /((SLASH)
1221 4745      JMS I IN19      /TYPN
1222 1675      TAD I CKT
1223 4246      JMS PNUM
1224 4743      JMS I IN16      /CRLF
1225 1275      WSER2,  TAD CKT
1226 2275      ISZ CKT
1227 7041      CIA
1230 1274      TAD LIMHI
1231 7640      TP240,  SZA CLA
1232 5310      JMP WSER1
1233 4743      JMS I IN16      /CRLF
1234 5751      JMP I IN25      /READ+5

/ROUTINES TO TYPE MASK AND LIMITS
1235 1356      ACX,   TAD CON3AC
1236 1360      MASKER, TAD CON3MS
1237 3744      DCA I IN17      /WORD
1240 5766      JMP I IN26      /EXAM

```

	1401	TEM2=PUNV
	1436	TEM=CKSM
1241	1170	IN27,CAU
1242	1123	IN30,CRL
		/INTER COM REG
1243	1157	IN16, CRLF
1244	1167	IN17, WORD
1245	1230	IN19, TYPN
1246	1363	IN20, CKSA
1247	1362	IN22, FROG
1250	1361	IN23, GAME
	1551	TP007=.
1251	1207	IN25, READ+5
		/CONSTANTS
1252	7774	TM4, -4
1253	1200	TP1000, 1000
1254	0077	TP77, 77
1255	0060	TP60, 00
1256	7663	CONJAC, SAC=MASK
1257	0257	TP257, 257
1260	1472	CONJMS, MASK
	1561	TABL2=.
1261	1202	PUN1
1262	1221	PUN3
1263	1225	PUN4
1264	1140	CRL2
1265	1135	CRL1
1266	1264	IN26, EXAM
1267	1236	TRAP
1270	1327	JUMP
1271	1176	PUNC
1272	1335	CONTIN
1273	1505	ASER
1274	1416	JPAR1
1275	1536	MASKER
1276	1535	ACX
1277	1152	OPIN /OPEN INDIRECTLY.

THERE ARE NO ERRORS

## SYMBOL TABLE

ACX	1535
BAIT	1376
BLIST	1043
BURP	1243
CAD	1170
CKNUM	1102
CKSA	1363
CKSAI	1075
CKSM	1436
CKT	1475
CUNFIN	1335
CUNJAC	1556
CUNJMS	1560
CR	1050
CR1	1123
CR1F	1157
CR11	1135
CR12	1140
CURPAG	1265
EXAM	1064
EX2	1067
FM270	1026
FM5	1063
FP240	1032
FMOG	1362
GAME	1361
IFRUG	1377
INX	1073
IN0	1074
IN10	1364
IN11	1365
IN12	1366
IN13	1367
IN14	1370
IN16	1543
IN17	1544
IN19	1545
IN20	1546
IN21	1371
IN22	1547
IN23	1550
IN25	1551
IN26	1566
IN27	1541
IN30	1542
IN7	1076
IN8	1171
IN9	1172
JMSER	1475
JPIGAM	1352
JUMP	1327
KEEP	1350
LF	1047

## SYMBOL TABLE

LIMHI	1474
LIMLO	1473
LINK	1356
LIP	1372
LIP4	1346
LPAR	1276
LIABL	1042
MASK	1472
MASKER	1536
NO	1117
UPIN	1152
PNUM	1446
PN2	1454
PUNC	1176
PUNN	1401
PUN1	1202
PUN2	1206
PUN3	1221
PUN4	1225
P10	1001
QUEST	1027
REA	1012
READ	1002
READ5	1027
RETN	1173
SAC	1355
SAD	1123
SCHAR	1157
SEX	1077
SHUT	1175
SLA	1051
SPEXIT	1320
SPNTR	1123
SP177	1200
SP200	1271
SP2000	1372
SP400	1373
SP4000	1375
SP7000	1374
SP7600	1234
STAKT	1000
TABL1	1044
TABL2	1551
TEM	1436
TEM2	1401
THE	1351
TM4	1552
TUTE	1174
TP007	1551
TP1000	1553
TP177	1400
TP200	1422
TP240	1531



## SYMBOL TABLE

TP257	1557
TP60	1555
TP7000	1466
TP77	1554
TRAD	1357
TRAP	1236
ISTJMS	1322
TYPN	1230
UPAR1	1416
UPAR2	1155
UPAR3	1145
WURD	1167
WSER	1505
WSEK1	1510
WSEK2	1525
XCONT	1344
ZPAT	0074

## SYMBOL TABLE

ZPAT	1074
START	1076
PIV	1071
READ	1042
READ5	1077
REA	1012
FM2/3	1026
NOEST	1027
FP240	1032
LTABL	1042
DLIST	1043
TABL1	1044
LF	1047
CR	1050
SLA	1051
FM5	1063
EXAM	1064
EX2	1067
INX	1073
IN0	1074
CKSAI	1075
IN7	1076
SEX	1077
CKNUM	1102
NU	1117
SPNTR	1123
SAD	1123
CRL	1123
CRL1	1135
CRL2	1140
UPAH3	1145
UPIN	1152
UPAH2	1155
CRLF	1157
SCHAR	1157
WORD	1167
CAD	1170
IN8	1171
IN9	1172
RETN	1173
TUTE	1174
SHUT	1175
PUNC	1176
SP177	1200
PUN1	1222
PUN2	1206
PUN3	1221
PUN4	1225
TYPN	1230
SP7600	1234
TKAP	1236
BURP	1243
CURPAG	1255

## SYMBOL TABLE

SP200	1271
LPAR	1276
LIP	1302
LIP4	1306
SPEXIT	1320
TSTJMS	1322
JUMP	1327
CONTIN	1335
XCONT	1344
THE	1351
JPIGAM	1352
SAC	1355
LINK	1356
TRAD	1357
KEEP	1360
GAME	1361
FROG	1362
CKSA	1363
IN10	1364
IN11	1365
IN12	1366
IN13	1367
IN14	1370
IN21	1371
SP2000	1372
SP400	1373
SP7000	1374
SP4000	1375
BAIT	1376
IFROG	1377
TP177	1400
TEM2	1401
PUNN	1401
UPAR1	1416
TP200	1422
TEM	1436
CKSM	1436
PNUM	1446
PN2	1454
TP7600	1466
MASK	1472
LIMLO	1473
LIMHI	1474
JMSER	1475
CKT	1475
MSER	1505
MSER1	1510
MSER2	1525
TP242	1531
ACX	1535
MASKER	1536
IN27	1541
IN30	1542

