IDENTIFICATION

Product Code:	DEC-08-COCO-D
Product Name:	ODT-8
Date Created:	December 30, 1967
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 REQUIREMENTS

2.1 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 ROUTINE

4.1 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 wihout 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.

4.2 Control Characters

a. Slash (/) - Open register preceding/

The register examination character / causes the register addressed by the octal number preceding the slash to be opened and its contents typed out in octal. The open register can then be modified by typing the desired octal number and closing the register. Any octal number from 1 to 4 digits in length is a legal input. Typing a fifth digit is an error and will cause the entire modification to be ignored and a question mark to be typed back by ODT. Typing (/) with no preceding argument causes the latest named register to be opened (again). Typing 0/ is interpreted as / with no argument.

Example:
```
400/6046
400/6046 2468?
400/6046 12345?
/6046
```

b. Carriage Return ( )) - Close register

If the user has typed a valid octal number, after the content of a register was printed by ODT, typing) causes the binary value of that number to replace the original contents of the opened register and the register to be closed. If nothing has been typed by the user, the register is closed but the content of the register is not changed.

Example:
```
400/6046 )          Register 400 is unchanged.
400/6046 2345 )     Register 400 is changed to contain 2345.
/2345 6046 )        Replace 6046 in register 400.
```

Typing another command will also close an opened register.

Example:
```
400/6046 401/6031 2346 )    Register 400 is closed and unchanged and
400/6046 401/2346 )         401 is opened and changed to 2346.
```

c. Line Feed (↓) - Close register, open next sequential register

The line feed has the same effect as the carriage return, but, in addition, the next sequential register is opened and its contents typed.

Example:
```
400/6046↓           Register 400 is closed unchanged and 401
0401/6031 1234↓     is opened. User types change, 401 is
0402/5201 )         closed containing 1234 and 402 is opened.
```

d. Up arrow (↑) - Close register, take contents as memory reference and open same

Up arrow will close an open register just as will carriage return. Further, it will interpret the contents of the register as a memory reference instruction, open the register referenced and type its contents.

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

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:? )
4U? )
406/4671 67K? )
/4671 )
```
ODT opens no register.

ODT ignores modification and closes register 406.

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  }        Breakpoint legal here.
JMS  }
FADD          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 occuring 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 is 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 searchlimit 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

2000/2467
3057/2501           These are 4 ISZ instructions in this section
3124/2032           of core.
4000/2152
```

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 breaktrap 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 breaktraps 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-origined 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 7000 (LF)(CR)                          [mask modified]
7473 /0001 400 (LF)(CR)                      [lower search limit modified]
7474 /7000 500 (CR)(LF)                      [upper search limit modified]
3000W (CR)(LF)                               [quantity for which to search specified and
0404 /3272 (CR)(LF)                                        search begun]
0431 /3277 (CR)(LF)
0437 /3277 (CR)(LF)
0444 /3330 (CR)(LF)
0450 /3277 (CR)(LF)
0453 /3300 (CR)(LF)
0456 /3276 (CR)(LF)
0455 /3277 (CR)(LF)
(LF)                                         [search completed]
```

```
M7000 7777 (LF)(CR)                          [change mask]
7473 /0400 364 (LF)(CR)                      [change lower limit]
7474 /0500 (CR)(LF)                          [upper limit is all right]
7200W (CR)(LF)                               [search for all CLA instructions]
0364 /7200 (CR)(LF)                          [there is only one.  It is at location 364]
(LF)                                         [search is finished]
```

```
M7777 600 (CR)(LF)                           [set mask for indirect and page bits]
400W (CR)(LF)                                [using previous limits search for all references
0377 /7402 (CR)(LF)                                to page zero which occur]
0411 /7450 (CR)(LF)
0414 /7450 (CR)(LF)
0417 /7450 (CR)(LF)
0432 /7402 (CR)(LF)                          [there are none, however, these microinstructions
0440 /7402 (CR)(LF)                            look like indirect references to page zero since
0451 /7402 (CR)(LF)                            they have a 1 in bit 3 and a Ø in bit 4]
0462 /7540 (CR)(LF)
0466 /7402 (CR)(LF)
0472 /7521 (CR)(LF)
(LF)                                         [search completed]
```

```
M0600 0(LF)(CR)                          [set mask to zero so that everything will match]
7473 /0360 407(LF)(CR)                   [set search limits to encompass dump area]
7474 /0500 427(CR)(LF)
W(CR)(LF)                                [since W is typed alone, the word searched for
0407 /1270(CR)(LF)                        is Ø. The result after masking each register
0410 /1272(CR)(LF)                        with Ø is, of course, Ø so all comparisons
0411 /7450(CR)(LF)                        appear to the program equal and hence all
0412 /5253(CR)(LF)                        unmasked contents are typed, constituting
0413 /1273(CR)(LF)                        a dump]
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 /7653(CR)(LF)
0423 /5242(CR)(LF)
0424 /1274(CR)(LF)
0425 /4571(CR)(LF)
0426 /5231(CR)(LF)
0427 /1275(CR)(LF)
(LF)
```

Examples of Register Examination & Modification

```
400/6046 (CR)(LF)                        [Examine Only]
400/6046 2468? (CR)(LF)                  [Non-octal number typed, modification ignored ]
400/6046 12345?(CR)(LF)                  [More than 4 digits typed, modification ignored]
/6046 2345 (CR)(LF)                      [Register 400 modified to 2345]
/2345 6046 (CR)(LF)                      [Modified again]
/6046 401/6031 2346 (CR)(LF)             [Register closed by typing another command]
400/6046 401/2346 (CR)(LF)
/2346 6031 (CR)(LF)
/6031
```

```
400/6046 (LF)(CR)                        [close and examine next ]
0401 /6031 1234 (LF)(CR)                 [modify 401, examine 402]
0402 /5201 (CR)(LF)                      [close 402]
401/1234 6031 (LF)(CR)
0402 /5201 (CR)(LF)
(LF)(CR)
0403 /6036 (CR)(LF)
(LF)(CR)
0404 /7270(CR)(LF)
```

Examples of Register Examination & Modification (continued)

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

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

4: ?(CR)(LF)                                } illegal character. ODT opens no register
4U?(CR)(LF)
6Q?(CR)(LF)
406/4671 Y? (CR)(LF)                        } illegal character. ODT ignores modification
406/4671 67K? (CR)(LF)                      } fifth digit in series. ODT ignores modification
406/4671 67322? (CR)(LF)                    } register 406 still contains original value of 4671
/4671
```

Examples of setting Breakpoints and Executing User's Program

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

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

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

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

```
A0000 1 (CR)(LF)
A0001 (CR)(LF)
/0001 (CR)(LF)
(LF)(CR)
7356 /0001 0 (CR)(LF)
/0000(CR)(LF)
```

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

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

```
446B (CR)(LF)
400G (CR)(LF)
*0446 (0004 (CR)(LF)
C (CR)(LF)
0446 (0010 (CR)(LF)
C0 (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]

```
        /0000           START=/0000
        0004            EPAT=4

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

        /0000           *START

/0000   3073                    DCA I CKSAT     /CLEAR THE CHECKSUM,
/0001   0010            P10,    10              /ARBITRARY CONSTANT

/0002   4337            READ,   JMS CRLF        /END LINE; SET SHUT TO -1
/0003   1073                    TAD I INX       /IRAD
/0004   3367                    DCA WORD        /GET THE TRAP ADDRESS,
/0005   1074                    TAD I IN0       /KEEP
/0006   3767                    DCA I WORD      /RESTORE CONTENT,
/0007   3367            READ3,  DCA WORD        /CLEAR THE INPUT,//TH INST,
/0010   1263                    TAD FM3         /-3
/0011   3374                    DCA TOTE        /SET THE LETTER COUNT,
/0012   0031            REA,    KSF
/0013   5212                    JMP .-1         /WAIT FOR COMMAND,
/0014   0030                    KRB
/0015   3357                    DCA SCHAR
/0016   1357                    TAD SCHAR       /GO TYPE THE CHARACTER,
/0017   4772                    JMS I INY
/0020   1373                    TAD RETN        /INITIALIZE THE PATCH
/0021   3004                    DCA EPAT        /EVERY TIME,
/0022   1243                    TAD BLIST       /COMPUTE ADDRESS OF COMMAND,
/0023   3323                    DCA SPNTR
/0024   1723                    TAD I SPNTR     /SEARCH FOR LEGAL CHARACTER,
/0025   2323                    ISZ SPNTR
/0026   7210            FM270,  SPA             /TEST FOR END OF LIST; MINUS 3
/0027   5277            QUEST,  JMP SEX         /NOT SATISFIED,
/0030   7041                    CIA             /COMPARE THE CHARACTER,
/0031   1357                    TAD SCHAR
/0032   7040            FF240,  SZA CLA         /FOUND
/0033   5224                    JMP .-7         /NO, CONTINUE
/0034   1323                    TAD SPNTR
/0035   1242                    TAD LIABL
/0036   3323                    DCA SPNTR
/0037   1723                    TAD I SPNTR     /LOOK UP THE ADDRESS,
/0040   3323                    DCA SPNTR
/0041   5723                    JMP I SPNTR     /GO PROCESS,

/0042   0214            LIABL,  TABL2-TABL1-1
/0043   0044            BLIST,  TABL1

                        /ODT-8 WILL ALSO CORRECTLY READ SYMBOLIC
                        /TAPES PREPARED FOR II: E.G. 1021/113/*77/5
```

```
                        /COMMAND LIST

        /0044           TABLE=.
/0044   0020                    020             /PUNCH
/0045   0005                    005             /END
/0046   0024                    024             /TRAILER
/0047   0212            LF,     212             /OPEN NEXT
/0050   0215            CR,     215             /CLOSE THIS ONE
/0051   0257            SLA,    257             /OPEN THIS ONE
/0052   0002                    002             /BREAK
/0053   0007                    007             /GO
/0054   0273                    273             /;
/0055   0003                    003             /CONTINUE
/0056   0027                    027             /WORD SEARCH
/0057   0036                    036             /UP-ARROW OPENS INDIRECT(I,E, MEM REF)
/0060   0015                    015             /MASK*UPPER*LOWER*
/0061   0001                    001             /AC*LINK
/0062   0037                    037             /BACK ARROW = OPEN INDIRECTLY
                        /TABLE MUST END WITH A NEG NUMBER
/0063   7773            FIN,    -5

/0064   1307            EXAM,   TAD WORD        /LOAD ADDRESS
/0065   7440                    SZA             /IF ZERO, USE LAST
/0066   3370                    DCA CAD
/0067   1770            EX2,    TAD I CAD
/0070   4771                    JMS I INO       /PNUM   (PRINT CONTENTS)
/0071   3373                    DCA SHUT        /SIGNALS OPEN REG
/0072   5207                    JMP READ5

/0073   7537            INX,    TWAD
/0074   7300            INO,    KEEP
/0075   7303            CRSAI,  CRSA
/0076   7302            IN7,    PROG


                        /PROCESS OCTAL DIGITS,

/0077   7200            SEX,    CLA
/0100   1537                    TAD SCHAR
/0101   1226                    TAD PM260       /(=0)
/0102   7500            CKNUM,  SMA
/0103   5331                    JMP NO          /ILLEGAL CHAR
/0104   1201                    TAD P10         /10
/0105   7510                    SPA
/0106   5331                    JMP NO          /ILLEGAL CHAR
/0107   3323                    DCA SAD
/0110   1307                    TAD WORD        /ASSEMBLE AN ADDRESS
/0111   7104                    RAL CLL
/0112   7006                    RTL
/0113   1323                    TAD SAD
/0114   3307                    DCA WORD
/0115   2374                    ISZ TUTE
/0116   5212                    JMP REA
```

```
                        /TYPE ERROR INDICATOR (?)
/117  /200               NO,     CLA
/120  1227                        TAD QUEST       /???
/121  4772                        JMS I INY       /TYPN
/122  5202                        JMP READ


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

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


      /123              SPNTR=.
      /123              SAV=.

                        /ROUTINE TO HANDLE REG. MODIFICATION AND INCREMENTAL EXAMINE
/123  0000              CRL,    0
/124  1374                        TAD TWTE
/125  7041                        CIA
/126  1203                        TAD FMS         /-5
/127  7650                        SNA CLA
/130  5723                        JMP I CRL       /NO MOD, INFO AVAILABLE
/131  1367                        TAD WORD
/132  2373                        ISZ SHUT        /TEST FOR OPEN AND THEN CLOSE IT.
/133  3770                        DCA I CAD       /MODIFY REGISTER
/134  5723                        JMP I CRL

/135  4323              CRL1,   JMS CRL         /CARRIAGE RETURN TO CLOSE
/136  4357                        JMS CRLF
/137  5267                        JMP READ5

/140  1250              CRL2,   TAD CR          /SINGLE FEED+CR
/141  4772                        JMS I INY
/142  4323                        JMS CRL
/143  4772                        JMS I INY       /TIME FOR CAR TO RET.
/144  2370                        ISZ CAD         /LINE FEED = EXAMINE NEXT
/145  1370              UPAR3,  TAD CAD
/146  4771                        JMS I IN8       /PNUM
/147  1251                        TAD SLA
/150  4772                        JMS I INY       /TYPN
/151  5267                        JMP EX2

/152  4323              UPIN,   JMS CRL         /CLOSE FIRST
/153  1770                        TAD I CAD
/154  3370                        DCA CAD
/155  4357              UPAR2,  JMS CRLF
/156  5345                        JMP UPAR3
```

```
        /127                    SCHAR=.

                                /TYPE A CAR. RET. AND LINE FEED
/127    0000                    CRLF,   0
/100    1250                            TAD CR          /212
/101    4112                            JMS I INY       /TYPN
/102    1247                            TAD LF          /215
/103    4112                            JMS I INY       /TYPN
/104    7040                            CMA             /MINUS ONE
/105    6615                            DCA SHUT        /SIGNALS CLOSED REGISTER
/106    5127                            JMP I CRLF

                                /PAGE ONE PARAMETERS.
/107    0000                    WORD,   0
/110    0000                    CAD,    0               /CURRENT ADDRESS
/111    7440                    INS,    PNUM
/112    1250                    INY,    TYPN
/113    1243                    REIN,   BURP
/114    0000                    IOIE,   0
/115    7777                    SHUT,   7777

/116    1307                    RUNC,   TAD WORD
/117    6016                            DCA I IN/
```

```
                    /0010=0, SECOND CORE PAGE

    /200            *START+200


/200  0111          SP111,  111             /FIRST IN THIS PAGE
/201  5767                  JMP I IN15        /READ5

                    /PUNCH DATA,
/202  7002          PUN1,   CLA HLT
/203  1362                  TAD PROG
/204  4765                  JMS I IN11        /PUNN (PUNCH ORIGIN)
/205  0100                  100
/206  1762          PUN2,   TAD I PROG
/207  4765                  JMS I IN11        /PUNN (PUNCH CONTENTS)
/210  0000                  0
/211  1362                  TAD PROG
/212  7041                  CIA
/213  1764                  TAD I IN10        /WORD
/214  7650                  SNA CLA
/215  5767                  JMP I IN15        /READ5
/216  2362                  ISZ PROG
/217  5206                  JMP PUN2
/220  5767                  JMP I IN15

                    /PUNCH END,
/221  7002          PUN3,   CLA HLT
/222  1363                  TAD CKSA
/223  4765                  JMS I IN11        /PUNN (PUNCH CHECKSUM)
/224  0000                  0
                    /PUNCH LEADER,
/225  1271          PUN4,   TAD SP200
/226  4230                  JMS TYPN
/227  5225                  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
/230  0000          TYPN,   0
/231  6046                  TLS       /(6020) - FOR H,S,
/232  6041                  TSF       /(6021) - FOR H,S,
/233  5232                  JMP .-1
/234  7000          SP7000, 7000      /CLA-GROUP2
/235  5630                  JMP I TYPN
```

```
                        /FEATURES ADDED: INTERUPT TURNED OFF UPON HITTING BREAKPOINT; CAN USE
                        /HI SPEED PUNCH, BREAKPOINT CAN BE PUT ON A JMS FOLLOWED BY ARGUMENTS.
                        /OUT=0 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,

                        /SET A BREAK POINT,
/2300  1704             TRAP,   TAD I IN10      /(WORD)=ADDRESS OF TRAP,
/2301  7450             SNA
/2302  1306             TAD IN12        /CHLP
/2303  3307             DCA TRAD        /TRAP SET (REAL OR DUMMY)
/2304  5320             JMP SPEXIT      /GO TO SECOND PAGE EXIT,

                        /THE TRAP IS SPRUNG

/2305  3325             BURP,   DCA SAC         /SAVE C(AC)
/2306  7004             RAL
/2307  3326             DCA LINK        /SAVE C(L)

/2310  1300             TAD KEEP
/2311  3737             DCA I TRAD      /REPLACE INSTRUCTION WHICH WAS TRAPPED

/2312  7101             IAC CLL
/2313  1307             TAD TRAD
/2314  3361             DCA GAME        /SAVE CONTINUATION ADDRESS (BREAK ADDR+1)

/2315  1300             TAD KEEP        /PICK UP TRAPPED INSTRUCTION
/2316  1372             TAD SP2000      /OVERFLOW TO LINK IF IOT OR OPERATE INSTR,
/2317  0271             AND SP200       /AC=0 IF PAGE 0 REFERENCE
/2320  7600             SZA SNL CLA     /WAS TRAPPED INSTR AN IOT,OPER,PAGE 0 REFERENCE?
/2321  5265             JMP CURPAG      /NO
/2322  4322             JMS TSTJMS      /YES, SEE IF IT WAS A JMS
/2323  7650             SNA CLA
/2324  5267             JMP CURPAG+2    /YES, TREAT AS IF NON=PAGE-ZERO REFERENCE
/2325  1300             TAD KEEP        /NO, PUT ACTUAL INSTR IN "THE" FOR EXECUTION
/2326  5306             JMP LIP4

/2327  1307             CURPAG, TAD TRAD
/2330  0254             AND SP7000
/2331  3362             DCA PROG        /SAVE INITIAL ADDR OF PAGE REFERENCED BY TRAPPED INSTR,

/2332  1300             TAD KEEP
/2333  0200             SP200,  AND SP177       /GET RELATIVE ADDR REFERENCED BY TRAPPED INSTR,
/2334  1362             TAD PROG        /ADD ON TOP OF PAGE
/2335  3362             DCA PROG        /SAVE ABSOLUTE ADDRESS OF MEMORY REFERENCE

/2336  1300             TAD KEEP
/2337  0373             AND SP400
/2340  7650             LPAR,   SNA CLA         /IS IT AN INDIRECT REFERENCE?
/2341  5302             JMP LIP         /NO
/2342  1762             TAD I PROG      /YES, GET ACTUAL REFERENCE
/2343  3362             DCA PROG
```

```
/302  4322          LIP,    JMS TSTJMS      /SEE IF TRAPPED INSTR IS A JMS
/303  7450                  SNA
/304  4771                  JMS I IN21      /YES, IT IS A JMS (JMSER)
/305  1377                  TAD IPROG       /NO     (JMS I PROG) JMS ADDS BACK 4000
/306  3321          LIP4,   DCA THE         /STORE FOR EXECUTION
/307  2765                  ISZ I IN11      /TEST N=CONTINUE
/310  5344                  JMP XCONT       /IGNORE THIS BREAK

/311  6002                  IOF             /STOP INTERRUPTS

/312  1357                  TAD TRAP
/313  4770                  JMS I IN14      /PNUM   (PRINT TRAP ADDRESS)
/314  1276                  TAD LPAR        /LEFT PAREN     (8 BITS=250=ASCII LFT PAREN)
/315  4250                  JMS TYPN
/316  1355                  TAD SAC
/317  4770                  JMS I IN14      /PNUM   (PRINT C(AC))

/320  4766          SPEXIT, JMS I IN12      /CRLF
/321  5767                  JMP I IN13      /READS

/322  0000          TSTJMS, 0
/323  1300                  TAD KEEP        /GET TRAPPED INSTR,
/324  0374                  AND SP7000      /ISOLATE OP CODE
/325  1375                  TAD SP4000      /OVERFLOW TO LINK WITH AC=0 IF JMS (4000)
/326  5722                  JMP I TSTJMS

                    /START AT A LOCATION
/327  1764          JUMP,   TAD I IN10      /(WORD)
/330  3361                  DCA GAME
/331  1352                  TAD JPIGAM      /(JMP I GAME)
/332  3321                  DCA THE
/333  3355                  DCA SAC         /CLEAR THE AC,
/334  7410                  SKP
/335  1764          CONTIN, TAD I IN10      /(WORD)
/336  7040                  CMA
/337  3765                  DCA I IN11      /(PUNN)=EMP COUNTER,
/340  4766                  JMS I IN12      /(CRLF)
                    /PATCH THE NEXT LOCATION WITH NOP(7000)
                    /IF THE PROGRAM BEING DEBUGGED EXPECTS
                    /THE ION FLAG TO BE ON.
/341  6042                  TCF             /CLEAR THE FLAG
/342  1757                  TAD I TRAP      /SAVE TRAP CONTENTS,
/343  3300                  DCA KEEP

/344  1376          XCONT,  TAD BAIT
/345  3757                  DCA I TRAP      /INSERT TRAP INSTRUCTION
/346  1356                  TAD LINK
/347  7110                  RAR CLL         /RESTORE LINK
/350  1355                  TAD SAC         /AND C(AC)
/351  7402          THE,    HLT             /OUT EXECUTION OF TRAPPED INST, AFTER PROCEED
/352  5761          JPIGAM, JMP I GAME
/353  2361                  ISZ GAME        /IMITATE SKIP CONDITION,
/354  5352                  JMP .-2
```

```
                        /VARIABLES MAY BE SCANNED VIA "A",

/355  0000        SAC,    0           /AC
/356  0000        LINK,   0           /LINK BIT
/357  /157        TRAP,   CRLF        /ADDRESS OF TRAP,
/360  0000        KEEP,   0           /CONTENT OF TRAP
/361  0000        GAME,   0           /ADDRESS FOR CONTINUE
/362  0///        PROG,   START-1     /MEMORY REFERENCE,
/363  0000        CKSA,   0           /THE CHECKSUM TO DATE,

                        /INTER COM REGS,
/364  /107        IN10,   WORD
/365  /401        IN11,   PUNN
/366  /157        IN12,   CRLF
/367  /007        IN13,   READ5
/370  /446        IN14,   PNUM
/371  /475        IN21,   JMSER               /PROCESS JMS,

                        /CONSTANTS
/372  2000        SP2000, 2000
/373  0400        SP400,  400
/374  7000        SP7000, 7000
/375  4000        SP4000, 4000
/376  5404        WAIT,   JMP I EPAT
/377  4702        IPROG,  JMS I PROG
```

```
                        /OUTPUT, THIRD CORE PAGE.

        /400            *START+400


                        /PUNCH ROUTINE
/400    0177            TP177,  177             /FIRST IN THIS PAGE.

/401    0000    PUNN,   0
/402    6240            DCA PNUM
/403    1240            TAD PNUM
/404    7012            RTR
/405    7012            RTR
/406    7012            RTR
/407    0354            AND TP77
/410    1001            TAD I PUNN
/411    4250            JMS CKSM
/412    1240            TAD PNUM
/413    0354            AND TP77
/414    4250            JMS CKSM
/415    5001            JMP I PUNN
                        /MEMORY REFERENCE OPENER.
/416    4742    OPAR1,  JMS I IN30              /(CKL)="CLOSER CALL".
/417    1741            TAD I IN27              /LAD
/420    6256            DCA TEM
/421    1656            TAD I TEM
/422    0200    TP200,  AND TP177
/423    6201            DCA TEM2                /SAVE LOWER BITS.
/424    1656            TAD I TEM
/425    0222            AND TP200
/426    7050            SNA CLA                 /TEST FOR PAGE ZERO REF
/427    5252            JMP .+3                 /YES
/430    1741            TAD I IN27
/431    0200            AND TP7000
/432    1201            TAD TEM2
/433    6741            DCA I IN27              /LAD
/434    5053            JMP I .+1
/435    7155            OPAR2

                        /CHECK SUM ACCUMULATOR
/436    0000    CKSM,   0
/437    6273            DCA CKT
/440    1746            TAD I IN20              /CKSA
/441    1273            TAD CKT
/442    6746            DCA I IN20              /CKSA
/443    1273            TAD CKT
/444    4743            JMS I IN19              /TYPN
/445    5636            JMP I CKSM
```

```
                        /ROUTINE TO PRINT OCTAL CONTENTS OF AC
/446  0000               PNUM,   0
/447  6201                       DCA PUNN
/450  1352                       TAD TM4
/451  6256                       DCA CKSM
/452  1201                       TAD PUNN
/453  7004                       RAL
/454  7004               PN2,    RAL
/455  7006                       RTL
/456  6201                       DCA PUNN
/457  1201                       TAD PUNN
/460  0351                       AND TP007       /ONLY 7=DIGITS GUARANTEED.
/461  1355                       TAD TP60        /IN CASE BIT 8 CAME THROUGH.
/462  4745                       JMS I IN19      /TYPN
/463  1201                       TAD PUNN
/464  2256                       ISZ CKSM
/465  5254                       JMP PN2
/466  7000               TP7000, 7000            /DCA=GROUP2
/467  1351                       TAD TP240
/470  4745                       JMS I IN19
/471  5646                       JMP I PNUM


                        /SEARCH VARIABLES.
/472  7777               MASK,   7777
/473  0001               LIMLO,  0001
/474  7000               LIMHI,  START

      /475               CKI=.
/475  0000               JMSER,  0
/476  1747                       TAD I IN22      /(PROG)=ABS MEM REF. (FINAL)
/477  6246                       DCA PNUM
/500  1750                       TAD I IN23      /GAME
/501  6646                       DCA I PNUM      /SIMULATED JMS
/502  2747                       ISZ I IN22      /PROG
/503  1353                       TAD TP1000
/504  5675                       JMP I JMSER
```

```
                        /WORD SEARCH ROUTINE
/3005  4743     WSER,   JMS I IN10      /CRLF
/3006  1273             TAD LIMLO
/3007  3273             DCA CKT
/3010  1073     WSER1,  TAD I CKT
/3011  0272             AND MASK
/3012  7041             CIA
/3013  1744             TAD I IN17      /WORD
/3014  7040             SZA CLA
/3015  3023             JMP WSER2
/3016  1273             TAD CKT
/3017  4246             JMS PNUM
/3020  1357             TAD TP257               /(SLASH)
/3021  4745             JMS I IN19      /TYPN
/3022  1073             TAD I CKT
/3023  4246             JMS PNUM
/3024  4743             JMS I IN10      /CRLF
/3025  1273     WSER2,  TAD CKT
/3026  2273             ISZ CKT
/3027  7041             CIA
/3030  1274             TAD LIMHI
/3031  7040     TP240,  SZA CLA
/3032  3010             JMP WSER1
/3033  4743             JMS I IN10      /CRLF
/3034  3751             JMP I IN23      /READ+5

                        /ROUTINES TO TYPE MASK AND LIMITS
/3035  1336     AUX,    TAD CONSAU
/3036  1300     MASKER, TAD CONSMS
/3037  3744             DCA I IN17      /WORD
/3040  3700             JMP I IN20      /EXAM
```

```
        /401            ITEM2=PUNN
        /450            ITEM=CRSM

/341    /170            IN27,CAU
/342    /123            IN30,CRL

                        /ENTER COM REG
/343    /157            IN10,   CRLF
/344    /167            IN11,   WORD
/345    /230            IN19,   TYPN
/346    /303            IN20,   CRSA
/347    /302            IN22,   PROG
/350    /301            IN23,   GAME
        /351            TP00/=.
/351    /007            IN25,   READ+5

                        /CONSTANTS
/352    ///4            IM4,    -4
/353    1000            TP1000, 1000
/354    00//            TP//,   //
/355    0000            TP00,   00
/356    /003            CONSAC, SAC=MASK
/357    0257            TP257,  257
/360    /472            CONSMS, MASK

        /361            TABL2=.
/361    /202                    PUN1
/362    /221                    PUN3
/363    /225                    PUN4
/364    /140                    CRL2
/365    /135                    CRL1
/366    /004            IN20,   EXAM
/367    /230                    TRAP
/370    /327                    JUMP
/371    /176                    PUNC
/372    /333                    CONTIN
/373    /305                    WSER
/374    /416                    UPAR1
/375    /336                    MASKER
/376    /335                    AUX
/377    /122                    OPIN    /OPEN INDIRECTLY.
```

THERE ARE NO ERRORS

SYMBOL TABLE

AUX	7555
BAIT	7570
BLIST	7045
BURP	7245
CAU	7170
CR$_{NUM}$	7102
CRSA	7505
CRSAI	7073
CRSM	7450
CRT	7473
CUNTIN	7555
CUNSAC	7550
CUNSMS	7500
CR	7052
CRL	7125
CRLP	7157
CRL1	7155
CRL2	7140
CURPAG	7265
EXAM	7004
EX2	7007
FM270	7026
FMS	7005
FP240	7052
FROG	7502
GAME	7501
IFROG	7577
INX	7073
IN0	7074
IN10	7504
IN11	7505
IN12	7500
IN13	7507
IN14	7570
IN15	7545
IN17	7544
IN19	7545
IN20	7546
IN21	7571
IN22	7547
IN23	7550
IN25	7551
IN26	7500
IN27	7541
IN30	7542
IN7	7070
IN8	7171
IN9	7172
JMSER	7473
JPIGAM	7552
JUMP	7527
KEEP	7500
LP	7047

SYMBOL TABLE

LIMHI	7474
LIMLO	7473
LINK	7550
LIP	7502
LIP4	7500
LPAR	7270
LTABL	7042
MASK	7472
MASKER	7550
NU	7111
OPIN	7152
PNUM	7446
PN2	7454
PUNC	7170
PUNN	7401
PUN1	7202
PUN2	7200
PUN3	7221
PUN4	7225
PT0	7001
QUEST	7027
REA	7012
READ	7002
READ5	7007
RETN	7173
SAC	7555
SAD	7123
SCHAR	7157
SEX	7077
SHUT	7175
SLA	7051
SPEXIT	7520
SPNTR	7123
SP177	7200
SP200	7271
SP2000	7372
SP400	7373
SP4000	7375
SP7000	7374
SP7000	7234
START	7000
TABL1	7044
TABL2	7501
TEM	7450
TEM2	7401
TME	7551
TM4	7552
TUTE	7174
TP007	7551
TP1000	7553
TP177	7400
TP200	7422
TP240	7551

SYMBOL TABLE

TP25/	7557
TP60	7555
TP7000	7400
TP77	7554
TRAU	7537
TRAP	7250
TSTJMS	7522
TYPN	7250
UPAR1	7410
UPAR2	7155
UPAR3	7145
WORD	7107
WSER	7505
WSER1	7510
WSER2	7525
XCUNT	7544
ZPAT	0004