

DEC-08-COCO-D

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 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.

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:
$$\begin{array}{l} 404/3270 \uparrow \\ 0470/0212 \text{ 0000 } \downarrow \\ \hline 404/3270 \uparrow \\ 0470/0000 \end{array}$$
 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:
$$\begin{array}{l} 365/5760 \uparrow \\ 0360/0426 \leftarrow \\ \hline 0426/5201 \end{array}$$

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:
$$\begin{array}{l} 4: ? \downarrow \\ 4U ? \downarrow \\ 406/4671 \text{ 67K ? } \downarrow \\ \hline /4671 \downarrow \end{array}$$
 } 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.

DEC-08-COCO-D

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 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

DEC-08-COCO-D

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 1021/1157t 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-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.

DEC-08-COCO-D

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 7322 (LF)(CR)
7473 /0221 422 (LF)(CR)
7474 /7022 522 (CR)(LF)
3000W (CR)(LF)
0404 /3272 (CR)(LF)
0431 /3277 (CR)(LF)
0437 /3277 (CR)(LF)
0444 /3330 (CR)(LF)
0450 /3277 (CR)(LF)
0453 /3330 (CR)(LF)
0456 /3276 (CR)(LF)
0455 /3277 (CR)(LF)
(LF)

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

[search completed]

M7000 7777 (LF)(CR)
7473 /0400 364 (LF)(CR)
7474 /0500 (CR)(LF)
7200W (CR)(LF)
3364 /7200 (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 600 (CR)(LF)
400W (CR)(LF)
0377 /7402 (CR)(LF)
0411 /7450 (CR)(LF)
0414 /7450 (CR)(LF)
0417 /7450 (CR)(LF)
0432 /7402 (CR)(LF)
0440 /7402 (CR)(LF)
0451 /7402 (CR)(LF)
0462 /7540 (CR)(LF)
0466 /7402 (CR)(LF)
0472 /7521 (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]

DEC-08-COCO-D

```

M0500 0(LF)(CR)
7473 /0360 407(LF)(CR)
7474 /0500 427(CR)(LF)
W(CR)(LF)
0407 /1270(CR)(LF)
0410 /1272(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 /523(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 0. The result after masking each register with 0 is, of course, 0 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)

```

404/3270 ↑(CR)(LF)
0470/70212 0000(CR)(LF)
404/3270 ↑(CR)(LF)
0470/70000 (CR)(LF)
/0000 (CR)(LF)
404/3270 3271↑(CR)(LF)
0471/70360(CR)(LF)
404/3271 3270↑(CR)(LF)
0470/0000(CR)(LF)
    
```

[contents of 404 refers to "this page, loc. 70"
 [ODT opens 470. User modifies 470]

[contents of 404 modified to refer to "this page
 [ODT opens 471] loc. 71"]

```

365/5760 ↑(CR)(LF)
0360/70426 (CR)(LF)
0426/5201 (CR)(LF)
    
```

[contents of 365 refers to "this page, loc. 160"
 [ODT opens 360. Contents of 360 become
 [ODT opens 426] address]

```

4:?(CR)(LF)
4U?(CR)(LF)
6Q?(CR)(LF)
406/4671 Y?(CR)(LF)
406/4671 67K?(CR)(LF)
406/4671 67322?(CR)(LF)
/4671
    
```

} illegal character. ODT opens no register

} illegal character. ODT ignores modification
 fifth digit in series. ODT ignores modification
 register 406 still contains original value of 4671

Examples of setting Breakpoints and Executing User's Program

```

475/0000 1 (LF)(CR)
0476/70000 2 (LF)(CR)
0477/0000 (CR)(LF)
432B (CR)(LF)
400G (CR)(LF)
+0432 70000 (CR)(LF)
477/0003
    
```

{ 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 acpts input of "+". Breakpoint
 [477 contains sum of 475 & 476] encountered

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

ODT types break
 address & C(AC)

```

475/0001 3 (LF)(CR)
0476/70002 (CR)(LF)
400G (CR)(LF)
*0432 70000(CR)(LF)
477/0006 (CR)(LF)
    
```

DEC-08-COCO-D

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 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)
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=/000
/0004          ZPAT=4

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

/0000          *START

/0000 0075          DCA I CRSAI          /CLEAR THE CHECKSUM,
/0001 0010          PLO, 10             /ARBITRARY CONSTANT

/0002 4007          READ, JMS CRLEF          /END LINE; SET SHUT TO =1
/0003 1073          TAU I INX           /TRAU
/0004 3367          DCA WURD           /GET THE TRAP ADDRESS,
/0005 1074          TAU I INB           /KEEP
/0006 3767          DCA I WURD          /RESTORE CONTENT,
/0007 3367          READS, DCA WURD      /CLEAR THE INPUT, //TH INST,
/0010 1203          TAU FMS             /=3
/0011 3374          DCA TUTE           /SET THE LETTER COUNT,
/0012 0031          REA, KSF
/0013 0212          JMP ,=1             /WAIT FOR COMMAND,
/0014 0030          KRB
/0015 3357          DCA SCHAK
/0016 1357          TAU SCHAK           /GO TYPE THE CHARACTER,
/0017 4772          JMS I INY
/0020 1373          TAU RETN           /INITIALIZE THE PATCH
/0021 3004          DCA ZPAT           /EVERY TIME,
/0022 1243          TAU BLIST          /COMPUTE ADDRESS OF COMMAND,
/0023 3323          DCA SPNTR
/0024 1723          TAU I SPNTR        /SEARCH FOR LEGAL CHARACTER,
/0025 2323          ISZ SPNTR
/0026 7310          FMZ/0, SPA          /TEST FOR END OF LIST; MINUS 2
/0027 3277          QUEST, JMP SEX      /NOT SATISFIED,
/0030 7041          CJA                /COMPARE THE CHARACTER,
/0031 1357          TAU SCHAK
/0032 7040          FPZ40, SZA CLA       /FOUND
/0033 3224          JMP ,=7            /NO, CONTINUE
/0034 1323          TAU SPNTR
/0035 1242          TAU LIABL
/0036 3323          DCA SPNTR
/0037 1723          TAU I SPNTR        /LOOK UP THE ADDRESS,
/0040 3323          DCA SPNTR
/0041 3723          JMP I SPNTR        /GO PROCESS,

/0042 0314          LIABL, TABL2-TABL1-1
/0043 7044          BLIST, TABL1

/OUT=0 WILL ALSO CORRECTLY READ SYMBOLIC
/TAPES PREPARED FOR IT: E.G. 1021/115/*77/5

```

/COMMAND LIST

/TABLE:

/044	0320		320	/PUNCH
/045	0305		305	/END
/046	0324		324	/TRAILER
/047	0212	LF,	212	/OPEN NEXT
/050	0215	CR,	215	/CLOSE THIS ONE
/051	0257	SLA,	257	/OPEN THIS ONE
/052	0302		302	/BREAK
/053	0307		307	/GO
/054	0275		275	/
/055	0305		305	/CONTINUE
/056	0327		327	/WORD SEARCH
/057	0356		356	/UP-ARROW OPENS INDIRECT(I,E, MEM REF)
/060	0315		315	/MASK+UPPER+LOWER+
/061	0301		301	/AC+LINK
/062	0357		357	/BACK ARROW = OPEN INDIRECTLY

/TABLE MUST END WITH A NEG NUMBER

/MD, -5

/063	1175			
/064	1367	EXAM,	TAU WURD	/LOAD ADDRESS
/065	1440		SEA	/IF ZERO, USE LAST
/066	5570		UCA CAU	
/067	1770	EX2,	TAU I CAU	
/070	4771		JMS I INB	/PNUM (PRINT CONTENTS)
/071	5575		UCA SHUT	/SIGNALS OPEN REG
/072	5207		JMP REAUD	
/073	7557	INX,	TRAD	
/074	7560	IND,	KEEP	
/075	7565	CKSAI,	CKSA	
/076	7562	IN7,	FRUG	

/PROCESS OCTAL DIGITS:

/077	7200	SEX,	CLA	
/100	1357		TAU SCHAR	
/101	1226		TAU FM270	/(=0)
/102	7500	CKNUM,	SMA	
/103	5517		JMP NU	/ILLEGAL CHAR
/104	1201		TAU P10	/10
/105	7510		SPA	
/106	5517		JMP NU	/ILLEGAL CHAR
/107	5525		UCA SAU	
/110	1367		TAU WURD	/ASSEMBLE AN ADDRESS
/111	7104		RAL CLL	
/112	7006		RTL	
/113	1325		TAU SAU	
/114	5567		UCA WURD	
/115	2574		ISE TUTE	
/116	5212		JMP REA	

```

/TYPE ERROR INDICATOR (?)
/117  /200      NU,      CLA
/120  122/      TAD QUEST      /2//
/121  4//2      JMS I INY      /IYFN
/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
/123      SPNTRF,
          SAUF,

/ROUTINE TO HANDLE REG. MODIFICATION AND INCREMENTAL EXAMINE
/123  0000      CR1,      0
/124  10/4      TAD TUTE
/125  /041      CIA
/126  1203      TAD FMS      /FD
/127  /050      SNA CLA
/130  5/23      JMP I CR1      /NO MOD, INFO AVAILABLE
/131  106/      TAD WURD
/132  23/5      ISZ SHUT      /TEST FOR OPEN AND THEN CLOSE IT.
/133  3//0      DCA I CAD      /MODIFY REGISTER
/134  5/23      JMP I CR1

/135  4323      CR11,     JMS CR1      /CARRIAGE RETURN TO CLOSE
/136  435/      JMS CR1F
/137  520/      JMP READ5

/140  1250      CR12,     TAD CR      /SINGLE FEED+CR
/141  4//2      JMS I INY
/142  4323      JMS CR1
/143  4//2      JMS I INY
/144  23/0      ISZ CAD      /TIME FOR CAR TO RET.
/145  10/0      UPARS,    TAD CAD      /LINE FEED = EXAMINE NEXT
/146  4//1      JMS I IN0      /FNUM
/147  1251      TAD SLA
/150  4//2      JMS I INY      /IYFN
/151  526/      JMP EX2

/152  4323      UPIN,     JMS CR1      /CLOSE FIRST
/153  1/70      TAD I CAD
/154  33/0      DCA CAD
/155  435/      UPAR2,    JMS CR1F
/156  5345      JMP UPARS

```

```

/107
/107 0000
/108 1200
/101 4//2
/102 124/
/103 4//2
/104 /040
/105 33/3
/106 2/3/

/107 0000
/110 0000
/111 7446
/112 /230
/113 /243
/114 0000
/115 /777

/116 130/
/117 30/6

SCHAR=,

/TYPE A CAR, RET, AND LINE FEED
CRLF, 0
TAD CR /210
JMS I INY /IYFN
TAD LF /212
JMS I INY /IYFN
CMA /MINUS ONE
DCA SHUT /SIGNALS CLOSED REGISTER
JMP I CRLF

/PAGE ONE PARAMETERS,
WORD, 0
CAD, 0 /CURRENT ADDRESS
INB, PNUM
INY, TYPN
REIN, BURP
IUIE, 0
SHUT, 7777

PUNC, TAD WORD
DCA I IN/

```

/001=0, SECOND CORE PAGE

/200

*START+200

```

/200 0177      SP177, 177          /FIRST IN THIS PAGE
/201 0767      JMP I IN10      /READ

/PUNCH DATA,
PUN1,  CLA HLT
      TAU FRUG
      JMS I IN11      /PUNN (PUNCH ORIGIN)
      100
PUN2,  TAU I FRUG
      JMS I IN11      /PUNN (PUNCH CONTENTS)
      0
      TAU FRUG
      CIA
      TAU I IN10      /WORD
      SNA CLA
      JMP I IN10      /READ
      ISZ FRUG
      JMP PUN2
      JMP I IN10

/PUNCH END.
PUN3,  CLA HLT
      TAU CKSA
      JMS I IN11      /PUNN (PUNCH CHECKSUM)
      0

/PUNCH LEADER,
PUN4,  TAU SP200
      JMS TYPN
      JMP I=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      / (0020) = FOR H,S,
      TSF      / (0021) = FOR H,S,
      JMP I=-1
SP/000, 7000    /CLA=GROUP2
      JMP I TYPN

```

/230 0000

/231 0040

/232 0041

/233 0232

/234 7000

/235 0030

/FEATURES ADDED: INTERRUPT 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,

1236	104	TRAP,	TAU I IN10	/(WORD)=ADDRESS OF TRAP,
1237	1450		SNA	
1240	1300		TAU IN12	/ORLEP
1241	3357		DCA TRAU	/TRAP SET (REAL OR DUMMY)
1242	3320		JMP SPEXIT	/GO TO SECOND PAGE EXIT,

/THE TRAP IS SPRUNG

1243	3355	BURP,	DCA SAC	/SAVE C(AC)
1244	1004		KAL	
1245	3350		DCA LINK	/SAVE C(L)
1246	1300		TAU KEEP	
1247	3757		DCA I TRAU	/REPLACE INSTRUCTION WHICH WAS TRAPPED
1250	1101		IAC CLL	
1251	1357		TAU TRAU	
1252	3361		DCA GAME	/SAVE CONTINUATION ADDRESS (BREAK ADDR+1)
1253	1300		TAU KEEP	/PICK UP TRAPPED INSTRUCTION
1254	1372		TAU SP2000	/OVERFLOW TO LINK IF IOT OR OPERATE INSTR,
1255	0271		AND SP200	/AC=0 IF PAGE 0 REFERENCE
1256	1000		SEA SNA CLA	/WAS TRAPPED INSTR AN IOT,OPER,PAGE 0 REFERENCE?
1257	3205		JMP CURPAG	/NO
1200	4322		JMS TSTJMS	/YES, SEE IF IT WAS A JMS
1201	1050		SNA CLA	
1202	3207		JMP CURPAG*2	/YES, TREAT AS IF NON-PAGE-ZERO REFERENCE
1203	1300		TAU KEEP	/NO, PUT ACTUAL INSTR IN "THE" FOR EXECUTION
1204	3300		JMP LIP4	
1205	1357	CURPAG,	TAU TRAU	
1206	0234		AND SP1000	
1207	3362		DCA FRUG	/SAVE INITIAL ADDR OF PAGE REFERENCED BY TRAPPED INSTR,
1210	1300		TAU KEEP	
1211	0200	SP200,	AND SP177	/GET RELATIVE ADDR REFERENCED BY TRAPPED INSTR,
1212	1302		TAU FRUG	/ADD ON TOP OF PAGE
1213	3362		DCA FRUG	/SAVE ABSOLUTE ADDRESS OF MEMORY REFERENCE
1214	1300		TAU KEEP	
1215	0373		AND SP400	
1216	1050	LPAR,	SNA CLA	/IS IT AN INDIRECT REFERENCE?
1217	3302		JMP LIP	/NO
1300	1702		TAU I FRUG	/YES, GET ACTUAL REFERENCE
1301	3362		DCA FRUG	


```

/302 4322      LIP,      JMS TSTJMS      /SEE IF TRAPPED INSTR IS A JMS
/303  /4300      SNA
/304  4/11      JMS I IN21      /YES, IT IS A JMS (JMSEK)
/305  13/11     DCA I PRUG      /NO (JMS I PRUG) JMS ADDS BACK 4000
/306  3351     LIP4,      DCA THE      /STORE FOR EXECUTION
/307  2/05     JSE I IN11     /TEST N-CONTINUE
/310  5344     JMP XCUNT      /IGNORE THIS BREAK

/311  0002     JUP          /STOP INTERRUPTS

/312  135/     TAU TRAU      /
/313  4/10     JMS I IN14     /PNUM (PRINT TRAP ADDRESS)
/314  12/0     TAU LPAK      /LEFT PAREN (0 BITS=220=ASCII LFT PAREN)
/315  4230     JMS TYPN      /
/316  1355     TAU SAC      /
/317  4/10     JMS I IN14     /PNUM (PRINT C(AC))

/320  4/00     SPEX11, JMS I IN12     /CRLF
/321  5/6/     JMP I IN13     /REAUD

/322  0000     TSTJMS, 0
/323  1300     TAU KEEP      /GET TRAPPED INSTR,
/324  03/4     AND SP/000      /ISOLATE UP CODE
/325  13/5     TAU SP4000     /OVERFLOW TO LINK WITH AC=0 IF JMS (4000)
/326  5/22     JMP I TSTJMS

/START AT A LOCATION
/327  1/04     JUMP,      TAU I IN10     / (WORK)
/330  3301     DCA GAME
/331  1352     TAU JPIGAM     / (JMP I GAME)
/332  3351     DCA THE
/333  3355     DCA SAC      /CLEAR THE AC,
/334  /410     SKP
/335  1/04     CONTIN, TAU I IN10     / (WORK)
/336  /040     CMA
/337  3/05     DCA I IN11     / (PUNN)=EMP COUNTER:
/340  4/66     JMS I IN12     / (CRLF)

/PATCH THE NEXT LOCATION WITH NOP(1000)
/IF THE PROGRAM BEING DEBUGGED EXPECTS
/THE TRY FLAG TO BE UP:
/341  0042     TCF          /CLEAR THE FLAG
/342  1/5/     TAU I TRAU      /SAVE TRAP CONDITIONS,
/343  3300     DCA KEEP

/344  13/6     XCUNT, TAU BAIT
/345  3/5/     DCA I TRAU      /INSERT TRAP INSTRUCTION
/346  1356     TAU LINK
/347  /110     RAR CLL      /RESTORE LINK
/350  1355     TAU SAC      /AND C(AC)
/351  /402     THE,      HLT
/352  5/01     JPIGAM, JMP I GAME     /ODT EXECUTION OF TRAPPED INST, AFTER PROCEED
/353  2301     JSE GAME      /IMITATE SKIP CONDITION,
/354  2352     JMP ,02

```

/VARIABLES MAY BE SCANNED VIA "A",

/355	0000	SAC,	0	/AD
/356	0000	LINK,	0	/LINK BIT
/357	1157	TRAD,	CRLF	/ADDRESS OF TRAP,
/360	0000	KEEP,	0	/CONTENT OF TRAP
/361	0000	GAME,	0	/ADDRESS FOR CONTINUE
/362	0111	FRUG,	START=1	/MEMORY REFERENCE,
/363	0000	CKSA,	0	/THE CHECKSUM TO DATE,

/INTEK COM REGS,

/364	1167	IN10,	WORD	
/365	1401	IN11,	PUNN	
/366	1157	IN12,	CRLF	
/367	1007	IN13,	REAU5	
/370	1446	IN14,	PNUM	
/371	1475	IN21,	JMSER	/PROCESS JMS,

/CONSTANTS

/372	2000	SP2000,	2000
/373	0400	SP400,	400
/374	7000	SP7000,	7000
/375	4000	SP4000,	4000
/376	5404	BAIT,	JMP I ZPA1
/377	4762	IFRUG,	JMS I FRUG

/UD1=0, THIRD CORE PAGE,

/400

*SIAR|+400

/PUNCH ROUTINE

/400 01// IP1// 1// /FIRST| IN THIS PAGE,

```

/401 0000 PUNN, 0
/402 0240 DCA PNUM
/403 1240 TAU PNUM
/404 /012 RTR
/405 /012 RTR
/406 /012 RTR
/407 0004 AND TP//
/410 1001 TAU I PUNN
/411 4200 JMS CKSM
/412 1240 TAU PNUM
/413 0004 AND TP//
/414 4200 JMS CKSM
/415 0001 JMP I PUNN

```

/MEMORY REFERENCE OPENER,

```

/416 4/42 UPAR1, JMS I IN00 /((CRL)="CLOSER CALL",
/417 1/41 TAU I IN2/ /CAU
/420 0200 DCA TEM
/421 1000 TAU I TEM
/422 0200 IP200, AND TP1//
/423 0201 DCA TEM2 /SAVE LOWER BITS,
/424 1000 TAU I TEM
/425 0222 AND TP200
/426 /000 SNA CLA /TEST FOR PAGE ZERO REF
/427 0202 JMP ,+3 /YES
/430 1/41 TAU I IN2/
/431 0200 AND TP/000
/432 1201 TAU TEM2
/433 0/41 DCA I IN2/ /CAU
/434 0000 JMP I ,+1
/435 /100 UPAR2

```

/CHECK SUM ACCUMULATOR

```

/436 0000 CKSM, 0
/437 02/0 DCA CKT
/440 1/40 TAU I IN20 /CKSA
/441 12/0 TAU CKT
/442 0/40 DCA I IN20 /CKSA
/443 12/0 TAU CKT
/444 4/40 JMS I IN19 /|YFN
/445 0000 JMP I CKSM

```

```

/ROUTINE TO PRINT DIGITAL CONTENTS OF AC
/446 00000 PNUM, 0
/447 3201 DCA PUNN
/450 1332 TAU TM4
/451 3236 DCA CKSM
/452 1201 TAU PUNN
/453 7004 RAL
/454 7004 PINZ, RAL
/455 7006 RTL
/456 3201 DCA PUNN
/457 1201 TAU PUNN
/460 0351 AND TP00/ /ONLY 7=DIGITS GUARANTEED,
/461 1335 TAU TP00 /IN CASE BIT 8 CAME THROUGH,
/462 4745 JMS I IN19 /IYFN
/463 1201 TAU PUNN
/464 2236 ISZ CKSM
/465 3254 JMP PINZ
/466 7000 TP/000, 7000 /CLA=GROUPZ
/467 1331 TAU TP240
/470 4745 JMS I IN19
/471 3046 JMP I PNUM

/SEARCH VARIABLES,
/472 7777 MASK, 7777
/473 0001 LIML, 0001
/474 7000 LIMH, START

CRIF,
/475 0000 JMSEK, 0
/476 1747 TAU I IN22 / (FRUG)=ABS MEM REF, (FINAL)
/477 3246 DCA PNUM
/500 1750 TAU I IN23 /GAME
/501 3046 DCA I PNUM /SIMULATED JMS
/502 2747 ISZ I IN22 /FRUG
/503 1333 TAU TP1000
/504 3075 JMP I JMSEK

```

```

/WORD SEARCH ROUTINE
/005 4/43
/006 12/3
/007 32/3
/010 10/3
/011 02/2
/012 7041
/013 1/44
/014 7040
/015 3323
/016 12/3
/017 4240
/020 1337
/021 4/43
/022 10/3
/023 4240
/024 4/43
/025 12/3
/026 22/3
/027 7041
/030 12/4
/031 7040
/032 3310
/033 4/43
/034 3/31

/WSEK1, JMS I IN10 /ORLT
        TAU LIMLU
        DCA CKT
WSEK1, TAU I CKT
        AND MASK
        CIA
        TAU I IN1/ /WORD
        SEA CLA
        JMP WSEK2
        TAU CKT
        JMS PNUM
        TAU TP23/ / (SLASH)
/021 JMS I IN19 /IYFN
        TAU I CKT
        JMS PNUM
WSEK2, JMS I IN10 /ORLT
        TAU CKT
        ISZ CKT
        CIA
        TAU LIMH1
TP240, SEA CLA
        JMP WSEK1
        JMS I IN10 /ORLT
        JMP I IN23 /REAU+3

/ROUTINES TO TYPE MASK AND LIMITS
ACX, TAU CUN3AC
MASKEK, TAU CUN3MS
        DCA I IN1/ /WORD
        JMP I IN20 /EXAM

```

	/401	TEMZ=PUNN
	/430	TEM=UNSM
/341	/170	INZ7,CAU
/342	/123	INS0,CKL
		/INTER COM REG
/343	/157	IN10, CKLF
/344	/167	IN17, WURU
/345	/230	IN19, TYPN
/346	/303	IN20, CKSA
/347	/302	IN22, FRUG
/350	/301	IN23, GAME
	/351	TP00/=,
/351	/007	IN25, READ+3
		/CONSTANTS
/352	//74	IM4, =4
/353	10000	TP1000, 1000
/354	0077	TP77, 77
/355	0000	TP00, 00
/356	/003	UN3AQ, SAC=MASK
/357	0257	TP257, 257
/360	/472	UN3MS, MASK
	/361	TABL2=,
/361	/202	PUN1
/362	/221	PUN3
/363	/225	PUN4
/364	/140	CKL2
/365	/135	CKL1
/366	/004	IN20, EXAM
/367	/236	TRAP
/370	/327	JUMP
/371	/176	PUNC
/372	/335	CUNTI
/373	/305	WSEK
/374	/410	UPAR1
/375	/330	MASKR
/376	/335	AUX
/377	/132	UPIN /OPEN INDIRECTLY.

3

THERE ARE NO ERRORS

SYMBOL TABLE

AUX	7335
BAIT	7370
BLIST	7043
BURP	7243
CAU	7170
CA _N U _M	7102
CKSA	7303
CKSA↓	7073
CKSM	7430
CKT	7473
CUNFIN	7333
CUN3AC	7330
CUN3MS	7300
CK	7050
CKL	7123
CKLF	7137
CKL1	7133
CKL2	7140
CURFAG	7265
EXAM	7004
EX2	7007
FM270	7020
FM5	7003
FM240	7032
FRUG	7302
GAME	7301
↓FRUG	7377
↓INX	7073
↓IN0	7074
↓IN10	7304
↓IN11	7305
↓IN12	7306
↓IN13	7307
↓IN14	7370
↓IN10	7343
↓IN17	7344
↓IN19	7345
↓IN20	7346
↓IN21	7371
↓IN22	7347
↓IN23	7330
↓IN25	7351
↓IN20	7300
↓IN27	7341
↓IN30	7342
↓IN7	7070
↓IN8	7171
↓IN9	7172
JMSER	7473
JP↓GAM	7332
JUMP	7327
KEEP	7300
LF	7047

SYMBOL TABLE

LIMHI	7474
LIMLO	7473
LINK	7350
LIP	7302
LIP4	7300
LPAK	7270
LIABL	7042
MASK	7472
MASKER	7330
NU	7117
UPIN	7152
PNUM	7440
PN2	7454
PUNC	7170
PUNIN	7401
PUN1	7202
PUN2	7200
PUN3	7221
PUN4	7223
PI0	7001
QUEST	7027
KEA	7012
KEAU	7002
KEAU5	7007
KEIN	7173
SAC	7355
SAU	7123
SCHAR	7157
SEX	7077
SHUT	7175
SLA	7051
SPEXIT	7320
SPNTR	7123
SP17	7200
SP200	7271
SP2000	7372
SP400	7373
SP4000	7375
SP7000	7374
SP7000	7254
SIART	7000
TABL1	7044
TABL2	7301
TEM	7430
TEM2	7401
TME	7351
TM4	7352
TUTE	7174
TP007	7351
TP1000	7353
TP177	7400
TP200	7422
TP240	7351

SYMBOL TABLE

TM25/	7331
TM60	7333
TM7000	7400
TM71	7334
TRAD	7331
TRAP	7230
TSTJMS	7322
TYPN	7230
UPAK1	7410
UPAK2	7133
UPAK3	7143
WURU	7101
WSEK	7303
WSEK1	7310
WSEK2	7323
XCOUNT	7344
ZPAT	0004

