Internal Report
DESY F36-76/01
February 1976

SOFPDP - A Deluxe Debugging Aid for PDP-8

by

W. A. McNeely, Jr.

TABLE OF CONTENTS

SOFPDP was inspired by the soft works of Claes Oldenburg.

1. INTRODUCTION

SOFPDP is a PDP-8 program which can interpretively execute the machine code of other PDP-8 programs. An arbitrary PDP-8 program, which will be called here an "object" program, may be checked for errors by running it in software simulation under SOFPDP. The major prerequisite for such operation is that the PDP-8 memory be large enough to accomodate both SOFPDP and the object program at the same time.

An object program running in software simulation takes on a slow-motion appearance, caused by the extra CPU time involved in simulating each individual object instruction. Interaction with such a program, therefore, requires some patience on the part of the user. For this one small price, however, the user acquires an unprecedented degree of control over the object program.

Debugging is facilitated in SOFPDP by the use of "traps", which react whenever pre-defined conditions are fulfilled in the course of execution. When a trap is encountered, SOFPDP ceases to simulate the object program and goes into a "command mode". In this mode, the user may exercise a wide variety of options for studying or changing the object program. Among those options is the possibility to single-step the object program in simulation and thereby follow its development in microscopic detail.

The traps in SOFPDP are unusual in that they operate from <u>outside</u> the object program, not <u>within</u> it (as in the case of "break point" type debugging routines). The advantages which this approach offers are fully exploited in SOFPDP. Thus, the traps which the user arranges prior to simulated execution, called "planned" traps, are singularly versatile. In addition to the planned traps, "involuntary" traps operate automatically to protect SOFPDP from destruction-prone object programs (among other things). Moreover, the user may prompt SOFPDP into the command mode at any time, a non-trivial option which is known here as the "spur-of-the-moment" trap. The net effect of these various traps is to make SOFPDP highly flexible, virtually indestructible, and responsive at all

times to the user's wishes.

An exceptionally powerful feature of SOFPDP is the "traceback" option, under which the last 10 instructions from the object program (and other pertinent data) may be recalled for examination. This allows the user to determine the steps which brought the object program to its current state. This feature, when used in conjunction with traps, offers the user the chance to salvage information from even the grimmest type of software disaster in the object program.

2. APPLICATIONS AND RESTRICTIONS

SOFPDP is probably most useful whenever a new program must be checked out under pressure, e.g., during an experimental run at DESY. (This type of situation normally precludes the use of quiet contemplation, which is still the best deterrent against errors.) SOFPDP also has its place in more relaxed situations, as in helping the user to gain a better understanding of how the PDP-8 works. SOFPDP has already proven its usefulness in the experimental operations of F31 and F36.

SOFPDP occupies a full 4K memory bank in the PDP-8 core storage. As partial compensation for such a large requirement on core space, SOFPDP can run on any available memory bank (other than bank 0). It is necessary that the object program can function meaningfully in the absence of the bank occupied by SOFPDP. That bank is presented to the object program, in certain respects, as a non-existent bank. If the object program uses that bank for data storage, but functions tolerably well when that bank is physically absent, then debugging with SOFPDP is formally possible. At DESY, SOFPDP may be used on banks 1, 2, or 3, depending on the size of the machine (and, in some cases, on the version of the F58 supervisor SUPER).

Simulated execution under SOFPDP is typically about 250 times slower than normal execution. Object programs which are critically sensitive to real time, therefore, may produce misleading results when run under SOFPDP.

For example, any hardware or software clock will appear to run about 250 times faster than normal. In most cases, however, the object program has no explicit real time dependence. The F58 supervisor SUPER is real time independent for all peripheral devices except the DEC tape drives, for which reason SOFPDP simulates the tape drive controller. Input from the teletype keyboard reader, if entered too rapidly by the user, may become lost or garbled. With an object program based on SUPER, the user can be sure that the program is ready for the next keyboard character by checking that the display loop is active.

3. LOADING PROCEDURE

SOFPDP is initially loaded from system tape in the standard way. (The editor text for SOFPDP is available from F58.) On the display appears the message:

MOUNT THE DESIRED SYSTEM TAPE!
WHICH BANK IS FREE?

If the desired object program is on a different system tape, then that tape should be prepared for loading. The user must now decide which memory bank is available for SOFPDP. The corresponding bank number is entered on the teletype. Immediately the above message disappears, and the system tape moves into position for loading a program from its directory. (SOFPDP is simulating the standard tape loader at this point.) The user now enters, in the usual way, the name of the object program, plus RETURN. When the system tape has finished its motions, signifying that the object program is loaded, then SOFPDP goes into the command mode and displays several lines of information, the bottom line of which reads:

LOADING COMPLETED

SOFPDP is ready, at this point, to begin work. The displayed information is called the "status display" and is explained in detail in Section 4 below. Section 5 outlines the acceptable keyboard commands, and Section 6 discusses practical techniques of debugging with SOFPDP.

4. THE STATUS DISPLAY

The "status display" is a concise summary of the state of the object program, and of the traps, which appears automatically when SOFPDP is in the command mode. The format is illustrated below, where fictitious numerical values have been inserted in order to make a clearer example:

```
PROGRAM COUNTER          07043
INSTRUCTION               1034        D.F.    0
INTERPRETED AS:    TAD    00034    (00034)  =    3

ACCUMULATOR      371    LINK     0    MQ       0    SC      0
ION     1  INH.     0   REQ.     0   BUF.      0   MODE    0

WINDOW:     (12420)  =  3243  (OCT)  =  -4535  (OCT)
                     =  1699  (DEC)  =  -2397  (DEC)

TRAPS:
P        06000      07177     (X)
I         1034       1034     (X)
W         3244       3242

TRACEBACK:     STEP     0

PLANNED TRAP ENCOUNTERED
```

The individual items of this display are explained below. Note that the numbers involved, unless otherwise specified, are in octal notation.

PROGRAM COUNTER gives the address of the current object instruction.

INSTRUCTION is followed by the PDP-8 machine code of the current object instruction, and D.F. gives the extended memory data field which applies if the instruction uses indirect addressing.

INTERPRETED AS: indicates the action intended by the current object instruction, an action which has not yet been carried out at this point. Instruction codes 0, 1, 2, 3, 4, and 5 are decoded as AND, TAD, ISZ, DCA, JMP, and JMS, respectively, followed by the absolute address of the storage location which the instruction references. In addition, the contents of that location are displayed in the format (address) = contents. Instruction code 6 is normally indicated here only as IOT. Instruction code 7 microinstructions are decoded in PAL-4 symbolic language, where the time sequence in combined microinstructions runs from left to right.

ACCUMULATOR, LINK, MQ, SC give the current contents of the object program's accumulator, link, multiplier quotient, and step counter, respectively.

ION, INH., REQ., BUF., MODE give further, somewhat more technical, information about the object program. ION is nonzero when the object program interrupt is enabled, INH. is nonzero when that interrupt is inhibited, REQ. is nonzero when an interrupt request is present, and BUF. gives the contents of the object program interrupt buffer. MODE is 0 for mode A, and 1 for mode B, of the PDP-8/e extended arithmetic element.

WINDOW: displays the contents of a preselected address in the format (address) = contents. The contents appear in both octal and decimal notation, and in each case as positive numbers as well as two's-complement negative numbers. (This part of the status display is referred to later as the "window".)

TRAPS: indicates the status of the three types of planned traps which are available to the user. These traps are denoted by the letters P, I, and W, which stand for program counter trap, instruction code trap, and

window trap, respectively. When the user enables one of these traps, then the symbolic name of the trap is followed by two numbers and, possibly, an X in parentheses. The two numbers indicate the range of values for which the trap responds. The X in parentheses indicates that the trap responds only in coincidence with the other traps, if any, which bear this designation. Thus, in the example given above, there are two conditions which will activate a planned trap: (1) when the program counter is in the range from 06000 to 07177, inclusively, and simultaneously the instruction code is 1034, or (2) the (octal) contents of the address 12420, as displayed in the "window", deviate from the value 3243. The latter is explained by the fact that the first number in the range of the window trap, namely, 3244, is greater than the second number, 3242. The corresponding range, in this case, is taken to be from 3244 to 7777, continuing with 0000 to 3242.

TRACEBACK:_ STEP_ gives the number of steps by which the user has translated the status display into the past, by way of using the "traceback" option. Step 0 refers to the present. If the traceback option is not enabled, then this line disappears.

PLANNED TRAP_ENCOUNTERED_ is one of several types of messages which may appear near the bottom of the status display as additional information. In this case, the coincidence condition represented by the program counter trap and the instruction code trap has been fulfilled.

5. KEYBOARD FUNCTIONS

The set of enabled keyboard functions (teletype commands) is described here. A comprehensive discussion of how to apply these functions is given in Section 6 below.

Note that the following commands only apply when SOFPDP is in the command mode. When SOFPDP is simulating the object program, then teletype input is passed directly to the object program for processing. The single exception to this rule is the character '?', which is intercepted

by SOFPDP and taken as a signal to interrupt execution of the object program (the "spur-of-the-moment" trap). The notation (+) means that additional numerical input is requested. The rules governing numerical input are spelled out in the Appendix.

General Control Functions

G	execute
?	interrupt execution
RETURN	execute a single instruction
CTRL/S	execute the bootstrap loader in simulation
CTRL/R	execute the bootstrap loader in reality

Window Commands

E	set the window address (+)
SPACE	increment the current window address
B	decrement the current window address
.	window address = contents of the program counter
A	display the contents of the object program accumulator
D	deposit contents (+) at the address given by the window address
*	deposit contents (+) repeatedly, where the window address is automatically incremented after each entry, until a second asterisk is entered to terminate the series

Trap Commands

P	enable the program counter trap and enter its range (+)
I	enable the instruction code trap and enter its range (+)
W	enable the window trap and enter its range (+)
CTRL/P	disable the program counter trap
CTRL/I	disable the instruction code trap
CTRL/W	disable the window trap

X regulate the coincidence condition among the traps. After entering X, the user may bring a given trap into a coincidence condition by entering the symbolic name of that trap (P, I, or W), or remove that trap from coincidence with CTRL/(P, I, or W). RETURN brings SOFPDP back to the normal command mode.

CTRL/X remove any coincidence condition among the traps. The affected traps, if any, revert to independent operation.

Traceback Commands

T enable the traceback option

CTRL/T disable the traceback option

CTRL/B roll the status display one step backward in time. The number of steps so taken is shown on the status display (maximum: 10 steps).

CTRL/F roll the status display one step forward in time (if it has first been taken backward in time with CTRL/B)

Miscellaneous

L load an address (+) in the object program counter

/ give the next teletype character to the object program for processing. This command can be used to enter the special control character '?' so that it will not be intercepted by SOFPDP. One must first, of course, bring SOFPDP into the command mode for this purpose.

J do not suppress warning messages which ordinarily appear when the bank occupied by SOFPDP is indirectly referenced by the object program using AND, TAD, ISZ, or DCA instructions. These messages warn the user that the bank occupied by SOFPDP is being presented as a non-existent bank to the object program. J is initially in effect by default.

CTRL/J suppress the warning messages (and the involuntary trap associated with them)

6. DEBUGGING WITH SOFPDP

Before turning to SOFPDP, the user should provide himself with some kind of listing of the object program, preferably an assembler listing. If only a PAL-4 listing is available, then it will be necessary to use the assembler-produced listing of the relocatable control section (RCS) origins to establish absolute addresses.

At the completion of the loading procedure, SOFPDP is ready for action in the command mode. The status display shows the first instruction which is up for simulation. The very first instructions in the object program are usually not relevant to the error in question. (For example, the first instructions in any program which uses the F58 supervisor, SUPER, belong to a rather lengthy initialization of that supervisor.) In any case, however, the object program is in a completely virgin state at this point. This opportunity should be used to examine or make changes using the "window", to set up one or more planned traps, or to enable the traceback option. These features of SOFPDP are discussed in separate sub-sections below.

The command for starting simulated execution is 'G'. The status display disappears and is replaced by the display output, if any, from the slow-moving object program. Execution proceeds in this manner until a trap is encountered, which includes the possibility for stopping with the command '?' at any time (the "spur-of-the-moment" trap). When a trap is encountered, SOFPDP returns to the command mode. Until the command mode is reached in this way, all teletype input except '?' is passed on to the object program for processing. In the command mode, single object instructions may be executed with the command RETURN.

It is quite easy to reload the object program under SOFPDP control. The command 'CTRL/S' starts SOFPDP executing the tape bootstrap loader in software simulation. Once the object program is reloaded, the status display returns with the message LOADING COMPLETED appended, and the user may make a fresh start. If, however, the user wishes to exit SOFPDP completely, then he may start the bootstrap loader in normal execution with

the command 'CTRL/R'. (One must, of course, first bring SOFPDP into the command mode.)

Using the Window

The "window" (see Section 4) fulfills a variety of needs in debugging. It is used to examine or change the contents of storage locations in the object program and, to some extent, in SOFPDP itself. For example, the window may be used to check the connection between the editor text for the object program and the assembled code, and simple errors may be corrected by depositing new contents. Important locations may be continuously and automatically monitored by using the "window trap".

The results of the window appear on the status display. The address shown there in parentheses is called the "window address". The window address may be set to an arbitrary extended memory address under the command 'E' (see Appendix for the rules governing numerical input). The commands SPACE and 'B' conveniently increment and decrement, respectively, the current window address. The command '.' sets the window address equal to the address of the current instruction as given by the program counter.

The command 'A' sets the window address equal to the address of the object program accumulator, which is an address inside SOFPDP. In this case, the window address is denoted on the status display by the word ACCUM instead of a numerical address. From this starting point, the command SPACE may be used to bring other elements of the object program (LINK, MQ, SC, ION, etc.) into the window, where the order corresponds to the arrangement of those elements on the status display, where they also appear as permanent features. Again, starting from ACCUM, the command 'B' brings, in turn, the elements D.F., INSTR, and PC into the window. Beyond these elements, in either direction starting from ACCUM, the window address is again denoted by a numerical address, namely, of an ordinary location inside SOFPDP.

The contents of the selected location appear on the status display in several different formats. Those contents may be changed by using the command 'D'. (Of the locations inside SOFPDP itself, only those which are identified as being elements of the object program, by means of symbolic rather than numerical window addresses, are subject to change under 'D'.) The command '*' is more convenient whenever several consecutive locations should be altered. Under '*', the user gives new contents exactly as with 'D', with the difference that each new entry automatically brings up the next consecutive location for alteration. Input continues in this way until the user supplies another '*' in order to terminate the series.

The Planned Traps

The key to successful debugging is judicious use of the planned traps. Suggestions for using the planned traps are given here, although it is impossible to devise a strategy which covers every possible case. It is a matter of intuition and trial-and-error, as well as experience, to find the best configuration of traps for a particular problem.

In most cases, the user knows where to start looking for a possible error, as, for example, in a subroutine which leads to chaos in the object program whenever it is called up. Such cases call for application of the program counter trap, which reacts whenever a predetermined point in the object program is reached in the course of execution. To set such a trap, the absolute address of the desired stopping point is entered uner 'P'. (Remember that the first address encountered in a subroutine is the address immediately following the entry point.) It is also possible to give a range of addresses under 'P' (see Appendix), such that each address in that range becomes a stopping point. This feature can be used, for example, to single-step through a subroutine with the command 'G' (instead of RETURN), whereby any externally called subroutines will not be executed in single-step fashion. Other uses of a finite range under 'P' are mentioned below. The command 'CTRL/P' disables the program counter trap.

The instruction code trap reacts whenever a given machine code (or a range thereof) is encountered as an instruction. The desired machine code is entered under 'I'. This type of trap is useful for seeking out all instances of usage of a relatively rare input-output transfer (IOT) of microinstruction (OPR). For this purpose, it is desirable to have on hand a list of the PDP-8 code 6 and code 7 instructions with their machine codes. 'CTRL/I' disables the instruction code trap.

The instruction code trap is somewhat restricted by the fact that the desired instruction may occur at unexpected points in the object program, especially in the supervisor. By switching the instruction code trap into a coincidence condition with the program counter trap, however, the activity of the instruction code trap can be restricted to the part of the object program corresponding to the range of the program counter trap. This is done by entering 'X', followed by the symbolic trap names, in this case 'I' and 'P'. The traps participating in a coincidence condition are tagged on the status display with an X in parentheses. A trap may be removed from a coincidence condition, once 'X' has been entered, with CTRL/(symbolic name). In any case, RETURN must be given in order to return to the normal command mode. All traps may be at once removed from any coincidence condition, from the command mode, with 'CTRL/X'.

The third type of planned trap, called the "window trap", is by far the most flexible. The window trap reacts whenever the contents of the location monitored by the window become equal to a preselected value, or when those contents lie inside a preselected range of values. The desired value or range of values is entered under 'W'. The range may even be so defined that the window trap triggers whenever the monitored contents change from some preselected value, as in the example of Section 4. When the window displays one of the various object program elements, such as the object program accumulator, then the window trap operates on the contents of that element. When those elements are INSTR and PC, then the window trap becomes a second instruction code and program counter trap, respectively. The window trap is especially useful in a coincidence condition. 'CTRL/W' disables the window trap.

The Involuntary Traps

The "involuntary" traps in SOFPDP react automatically whenever certain pathological conditions arise during execution. When this happens SOFPDP reverts to the command mode and indicates, by means of a message at the bottom of the status display, the nature of the problem. In most cases, the user can continue execution, whereby SOFPDP takes a corrective action against the cause of the interruption. In some cases, however, the problem is of such a severe nature that further execution is impossible without intervention from the user (such as reloading the object program). The various types of involuntary traps, in order of increasing severity, are discussed separately below, preceded in each case by the associated message:

INSTRUCTION = 0000

The machine code 0000 (literally, AND 0) is normally only encountered as an instruction when the object program has strayed from its proscribed region of core. If the traceback option was previously enabled, then the user can try to determine why this happened. Execution may be continued, whereby SOFPDP carries out an AND 0.

INSTRUCTION CAUSES HALT

Any instruction which can be expressed as the inclusive OR of 7402 with any even number causes a CPU halt in normal execution. This may happen when the object program runs astray, or it may be the result of a HLT command in the object program, possibly combined with other microinstructions. In any case, SOFPDP does not execute a CPU halt. If the user decides to continue from this point, then SOFPDP suppresses the bit which is associated with a CPU halt in the microinstruction and executes the instruction.

WARNING: SOFPDP'S BANK IS PROTECTED
(CTRL/J SUPPRESSES THESE WARNINGS)

When the above message appears, it means that the object program is trying to access the bank occupied by SOFPDP using one of the instructions AND, TAD, ISZ, or DCA. Execution may be continued, where the instruction in question will have as much effect as if it were referencing a non-existent bank. (Thus, AND and DCA only clear the accumulator, whereas TAD and ISZ have no effect whatsoever.) In this way, the object program can formally share a bank with SOFPDP. This trap is not really "involuntary", because it may be disabled with the command 'CTRL/J' (and re-enabled with 'J').

SORRY! THIS BANK IS OCCUPIED BY SOFPDP!

This message, which appears under several conditions, means that the object program is trying to access the bank occupied by SOFPDP in a way which precludes further execution. This happens, for example, whenever the object program seeks to transfer control to the forbidden bank with a JMP or JMS instruction. The other conditions which trigger this trap all involve the attempt to store data on SOFPDP's bank using some peripheral device. Those devices are the following: the CAMAC or IBM channels, where execution halts at the associated SVC instruction (octal code 6732), and the DEC tape units, where execution halts at the instruction DTLB (octal code 6774).

DECTAPE ERROR

As mentioned earlier, SOFPDP simulates the DEC tape controller. (It was explained earlier that the object program must not be critically dependent on real time. This affects the DEC tape controller, in particular,

because the DEC tape units are intrinsically dependent on real time.) Thus, SOFPDP only carries out tape actions as they become known from interaction with the object program. When a hardware tape error occurs, therefore, it is quite possible that SOFPDP has already led the object program to believe that the tape transfer occurred flawlessly. Execution may be continued after such an error, although the desired tape transfer has been abandoned. The best course of action is to break off operations with SOFPDP and investigate the reason for the hardware tape error.

The Traceback Option

The user sometimes wishes to know how the object program managed to reach a particular state, such as encountering the instruction 0000 in an unfamiliar part of core. If the traceback option was enabled with the command 'T' prior to such an event, then the user may examine the preceding 10 steps. When the traceback option is enabled, however, the executing speed in SOFPDP drops noticeably, and this is the reason why the traceback is an optional feature.

Under the traceback option, all information on the status display except the information dealing with the window and the traps is saved for possible later examination, up to a maximum of 10 steps. The saved information is viewed by inserting it again in the status display, or, in other words, rolling the status display backward in time. The command 'CTRL/B' rolls the status display one step backward, and simultaneously the number of steps indicated after TRACEBACK: STEP is incremented. 'CTRL/F' rolls the status display one step forward and decrements the number of steps shown on the status display. Once the status display is translated into the past with 'CTRL/B', then any command other than 'CTRL/F' will, in general, cause the status display to return to its original state (step 0) before any other action is taken. The command 'CTRL/T' disables the traceback.

APPENDIX: Numerical Input

Some of the teletype commands in SOFPDP request additional numerical input from the user. The routine which mediates that input has the following properties:

1. The nature of the requested input is always indicated. The current value of the requested quantity appears, in a separate line, after the word CURRENTLY.

2. The requested quantity may be either an address or the contents of an address, depending on the context of the teletype command. In the case of contents, the input consists of up to four digits, possibly preceded by the letter D to indicate decimal notation or by a minus sign to indicate two's-complement negation, or both (in any order). In the case of an address, the input consists of up to five digits, where the left-most digit gives the bank number. If less than five digits are given, then the bank number keeps its current value.

3. If only one quantity is requested, then the input is terminated with RETURN. If two quantities are requested, as in the case of the planned trap ranges, then the first quantity is terminated with a comma, and the second quantity is terminated with RETURN. The first of two quantities may, however, be terminated with RETURN, in which case the second quantity takes its value from the first one by default. In all cases, input which consists only of RETURN causes the requested quantity or quantities to keep their current values.

4. The teletype command '*' starts a series of requests for contents, where each entry in the series is terminated, as usual, with RETURN. If, however, an entry is terminated with '*', then that entry is processed as the final entry in the series.

5. RUBOUT deletes single digits (or literal prefixes) from the input.