digital

PDP-10
SYSTEMS USER'S GUIDE

PDP-10
SYSTEMS USER'S GUIDE

Order No. DEC-10-NGCA-D from Program Library, Maynard Mass.   Price $10.00

Direct comments concerning this manual to Software Quality Control, Maynard, Mass.

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Printed in U.S.A.

SUPPLEMENT

PDP-10 SYSTEMS USER'S GUIDE
(DEC-10-NGCA-D)

The pages contained in this supplement to the PDP-10 Systems User's Guide are provided to bring your copy of this document up to date. Only the changed pages and additional material have been provided. To revise your manual, simply replace pages of your manual with the corresponding page found in this supplement.

CONTENTS

SCOPE

INTRODUCTION

ORGANIZATION

PDP-10 software can be generally divided into eight functional groupings with respect to common programming activities. These are:

1. Source Program Preparation;
2. Conversational Language Translators;
3. Program Loading and Library Facilities;
4. Debugging;
5. Utilities;
6. Calculators;
7. Batch Processing; and
8. Monitoring.

This Guide is arranged according to the above order.

Source Program Preparation (EDITOR, TECO)

The DECtape Editor and the Text Editor and Corrector (TECO) programs can be used to create (and later correct or modify) text files (e.g., Macro-10 and FORTRAN source language programs) for subsequent assembly or compilation. Editor creates and modifies files on DECtape; TECO performs more complex editing functions on any standard I/O devices.

Conversational Language Translators (MACRO, F40)

The Macro-10 Assembler (MACRO) and the FORTRAN Compiler (F40) translate source programs written in the Macro-10 and FORTRAN IV languages, respectively, into binary machine language for subsequent loading and execution.

relocatable

Program Loading and Library Facilities (LOADER, LIB40, JOBDAT, FUDGE2)

Loading is performed by the Linking Loader, which takes specified relocatable binary programs, loads them in core, links their references to each other, and searches the appropriate subroutine libraries (e.g., LIB40) for required subroutines. A job data area (JOBDAT) is created by the Loader for each program; this area is used to store the current status of the job during execution. Library files of binary programs can be updated when desired by use of the File Update Generator (FUDGE2).

Debugging (DDT, CREF, GLOB)

Once a program is compiled (or assembled), it can be loaded along with the Dynamic Debugging Technique (DDT) program and debugged. DDT allows the user to control program execution and to modify his program in any of several modes, including symbolic. For purposes of further program analysis (and for documentation), the user can elect to use the Cross Reference Listing (CREF) program, which produces a cross-referenced listing of all symbols within his Macro program, and the Global Cross-Reference Listing (GLOB) program, which produces one to three helpful listings of all global symbols encountered in one or more programs.

Utilities (PIP, CONVRT, CODE, SRCCOM, BINCOM)

Digital provides a variety of utility programs for general purpose data handling. Among these programs are: the Peripheral Interchange Program (PIP), which transfers data between any standard I/O devices; DECtape Format Converter (CONVRT), which converts DECtapes from the old PDP-6 format to the new PDP-10 format (or vice versa); Code Translator (CODE), which performs translations between standard ASCII codes and code of other manufacturers; Source Compare (SRCCOM), which compares two versions of an ASCII file; and Binary Compare (BINCOM), which compares two versions of a binary file.

Calculators (AID, DESK)

Two easy-to-use problem solving calculators for scientists and engineers are included as part of the PDP-10 software: the Algebraic Interpretive Dialogue (AID), a problem solving calculator which is based upon the famous RAND JOSS[1] algebraic language; and Desk Calculator (DESK), which provides immediate access to common arithmetic subroutines.

Batch Processing (BATCH, STACK)

The Batch Processor (BATCH) supervises the sequential execution of a series of user jobs with a minimum of operator attention, operates as one of the "users" in a time-sharing environment and runs concurrently with the Batch-controlled jobs (as well as other jobs on the system), and permits constant communication by the operator. Job Stacker (STACK) prepares input stacks for BATCH and processes output stacks from BATCH

Monitors

PDP-10 software includes five separate Monitors, ranging from the single-user 10/10 Monitor, designed for an 8K paper tape system, to the swapping time-sharing 10/50 Monitor, designed for the 32K (and larger) disk systems. In between these are the single-user 10/20 Monitor for an 8K DECtape system, the single-user 10/30 Monitor for larger systems, and the time-sharing 10/40 Monitor for 16K (and larger) multiprogramming systems. System Builder (BUILD) is used to construct a Monitor specifically designed for the user's machine configuration and other requirements.

[1] JOSS is the trademark and service mark of the RAND Corporation for its computer program and services using that program.

SYSTEM OPERATION

Communication with the Monitor and loading and executing Digital-supplied Common User Service Programs (CUSP's), as well as your own programs, is a simple matter once you become familiar with a few basic rules.

To establish communication with the Monitor, ~~you~~ place your Teletype in Monitor mode ~~by entering the Monitor command mode.~~ SMU; Generally, this is accomplished by typing ↑C (i.e., by holding down the CTRL key while striking "C"). Monitor responds with a period (.) and you can then direct Monitor to load and start a program from the System Library (.RUN prog), start a program already loaded in core (.START), discontinue your job (.KJOB), and many other operations.

All CUSP's supplied by Digital are device independent. /e.g. EDITOR Because of this, you must tell the CUSP, via a command string typein, which devices to use. Readiness to receive a command string is signalled by the CUSP via an asterisk (*) /except DDT typeout after loading. For example, when you have called in the FORTRAN IV Compiler and it has responded with an asterisk, you would type in a command string indicating (1) the device containing the source program to be compiled, (2) the device on which the binary output is to be placed, and (3) the device on which the compilation listing is to be written:

*binary-output-device, listing-device←source-device

Devices are specified by a 3-character device name (a fourth character, a digit, specifies the particular unit in the case of DECtapes and magnetic tapes){TTY's SMU;} followed by a colon.

Card reader	CDR:	PTY:
Line printer	LPT:	SYS:
Paper tape reader	PTR:	OPR:
Paper tape punch	PTP:	CTY:
Teletype	TTY:	TTYn:
DECtape	DTAn:	
Magnetic tape	MTAn:	
Disk	DSK:	

For file-oriented devices (DECtape and disk), a filename is also required following the device name to specify either the specific file to be read or the filename to be assigned to the output. A filename can consist of a maximum of six characters. A filename can be further specialized by adding a 3-character extension name to it, preceded by a period (.). Extension names are generally used to classify a file into a particular category and certain standard extensions are used and recognized throughout the system (e.g., .REL for relocatable binary files, .DMP for saved core image files, etc.). A sample FORTRAN command string might appear as follows.

.SAV PDP-6

SMU.

DTA1:BIN.REL,LPT:← DTA0:SOURCE

Compile the file called "SOURCE" on DECtape 0; write the binary output on DECtape 1, calling it "BIN.REL"; print the listing on the line printer.

COLOR CODE CONVENTIONS

All computer typeouts are printed in dark blue.

All operator typeins are printed in black.

All commentary is printed in light blue. half-tone black better

SYMBOLOGY USED IN CONSOLE EXAMPLES

↑C Hold down the CTRL (control) key while striking "C." Normally echoes as ↑C.

↑x Hold down the CTRL (control) key while striking the "x" key, where "x" is any character. Normally echoes as ↑x.

Some special control symbols and their respective key designations for Models 33 and 35 Teletypes are given below.

Key Designation	Shown in This Manual as:	Method of Entry[1]
(TAPE)	↑R	Hold down CTRL key while striking "R."
~~(TAPE)~~ (not-TAPE)	↑T	Hold down CTRL key while striking "T."
(BELL)	↑G	Hold down CTRL key while striking "G."
(TAB) (horizontal)	→| or ↑I	Hold down CTRL key while striking "I."
(FORM)	↑L	Hold down CTRL key while striking "L."
(VT) (vertical tab)	↑K	Hold down CTRL key while striking "K."
(XON)	↑Q	(Initialize paper tape reader input.) Hold down CTRL key while striking "Q".
(XOFF)	↑S	(Terminate paper tape reader input.) Hold down CTRL key while striking "S".

← Hold down the SHIFT key while striking "O."

↵ Strike the RETURN key. Normally echoes back as a carriage return, line feed.

(ALTMODE) or ($) Strike the ALTMODE key (sometimes labelled "ESC" or "PREFIX").

[Unless a "[" key is present on the keyboard, hold down the SHIFT key while striking "K."

] Unless a "]"key is present on the keyboard, hold down the SHIFT key while striking "M."

[1] "Method of Entry" is for Models 33 and 35 Teletypes primarily. On the Model 37 Teletype, many of the special symbols appear on the keyboard and are easily entered.

↑	When appearing alone (as in DDT). Hold down the SHIFT key while striking "N."
<	Hold down the SHIFT key while striking ",".
>	Hold down the SHIFT key while striking ".".
(LINE-FEED)	Strike the LINE-FEED key.
(RUBOUT)	Strike the RUBOUT[1] key. Normally echoes back as a backslash (\\), XXX, or a repeat of the character erased.
\\ or ~~(FORM)~~	Unless a "\\" key is present on the keyboard, hold down the SHIFT key while striking "L."
␣	Strike the space bar to space to indicated position. TAB can also be used in most instances.

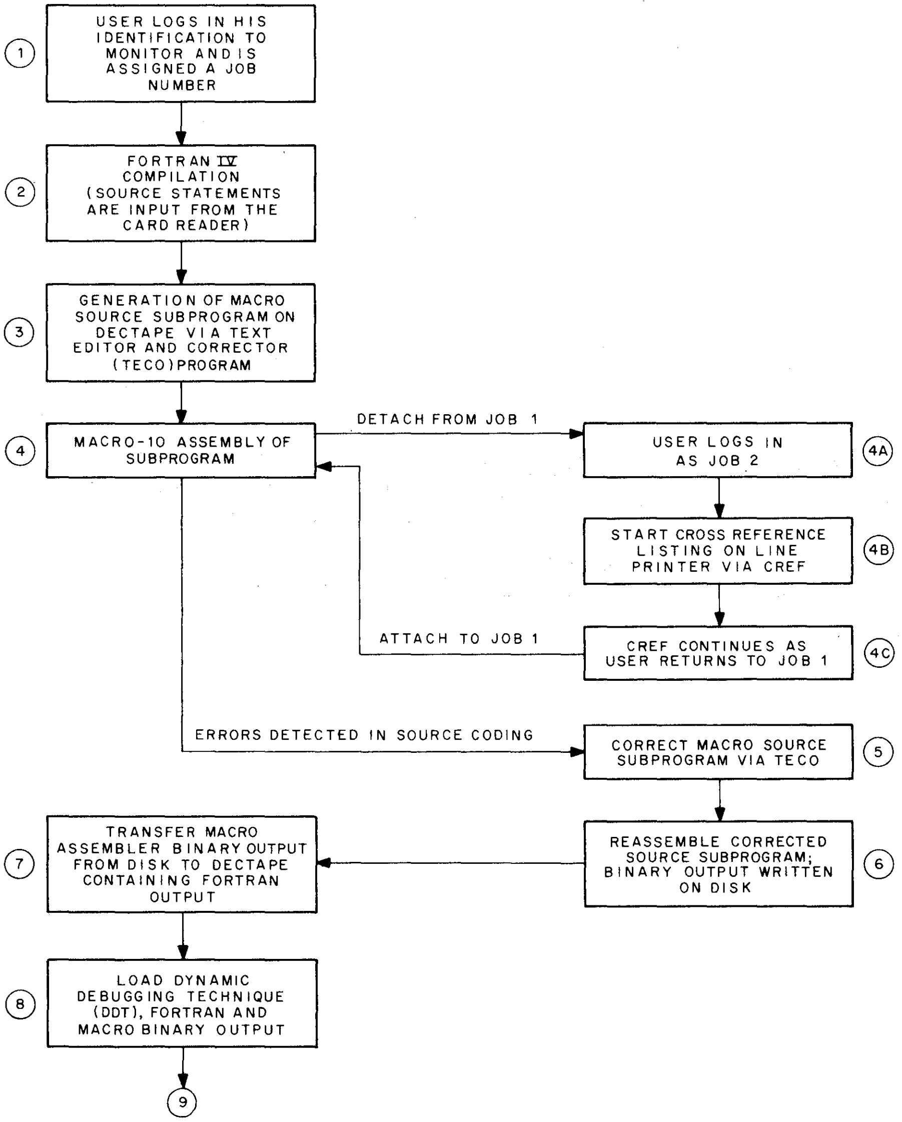
better aspect ratio

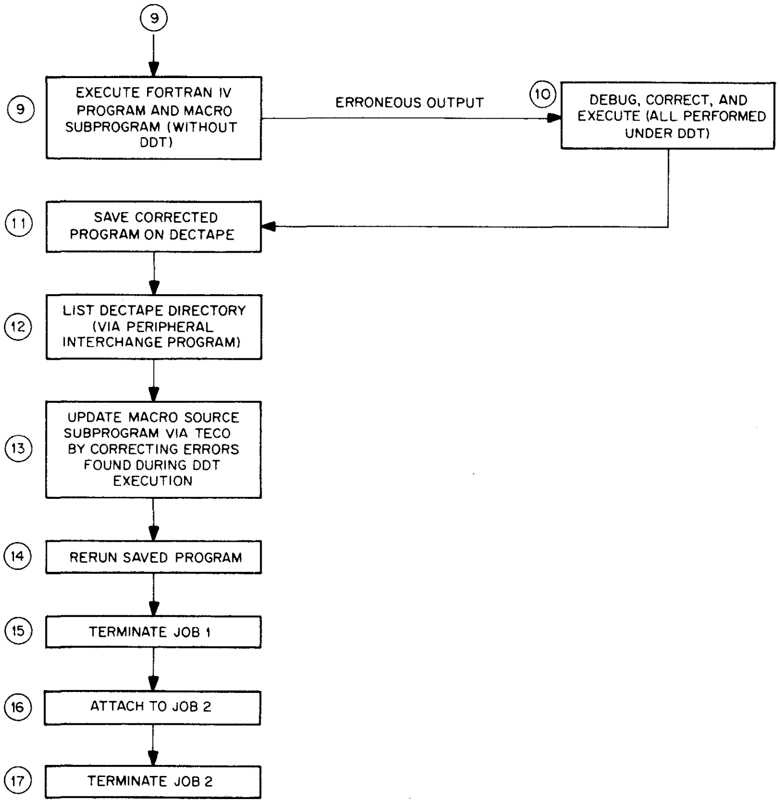
[1] On the Model 37 Teletype, this key is labeled DELETE.

DEMONSTRATION

The following example is designed to demonstrate the ease of operation and flexibility of a PDP-10 software system. Basically, this particular sequence involves the compilation of a FORTRAN source program, the assembly of a Macro-10 subprogram written in conjunction with the FORTRAN program, the loading and linking of the two resultant machine language programs, their execution and correction under control of the Dynamic Debugging Technique (DDT) program, correction of the original source coding, and a final rerun of the corrected programs. Several errors have been purposely introduced in the source coding to demonstrate the ease with which testing, debugging, and updating of programs is accomplished. The procedure for detaching from the current job, logging in and beginning a second job, and then returning to the original job is also illustrated.

FLOW CHART


1
USER LOGS IN HIS IDENTIFICATION TO MONITOR AND IS ASSIGNED A JOB NUMBER
2
FORTRAN IV COMPILATION (SOURCE STATEMENTS ARE INPUT FROM THE CARD READER)
3
GENERATION OF MACRO SOURCE SUBPROGRAM ON DECTAPE VIA TEXT EDITOR AND CORRECTOR (TECO) PROGRAM
4
MACRO-10 ASSEMBLY OF SUBPROGRAM
DETACH FROM JOB 1
USER LOGS IN AS JOB 2
4A
START CROSS REFERENCE LISTING ON LINE PRINTER VIA CREF
4B
ATTACH TO JOB 1
CREF CONTINUES AS USER RETURNS TO JOB 1
4C
ERRORS DETECTED IN SOURCE CODING
CORRECT MACRO SOURCE SUBPROGRAM VIA TECO
5
7
TRANSFER MACRO ASSEMBLER BINARY OUTPUT FROM DISK TO DECTAPE CONTAINING FORTRAN OUTPUT
REASSEMBLE CORRECTED SOURCE SUBPROGRAM; BINARY OUTPUT WRITTEN ON DISK
6
8
LOAD DYNAMIC DEBUGGING TECHNIQUE (DDT), FORTRAN AND MACRO BINARY OUTPUT
9


9
9
EXECUTE FORTRAN IV PROGRAM AND MACRO SUBPROGRAM (WITHOUT DDT)
ERRONEOUS OUTPUT
10
DEBUG, CORRECT, AND EXECUTE (ALL PERFORMED UNDER DDT)
11
SAVE CORRECTED PROGRAM ON DECTAPE
12
LIST DECTAPE DIRECTORY (VIA PERIPHERAL INTERCHANGE PROGRAM)
13
UPDATE MACRO SOURCE SUBPROGRAM VIA TECO BY CORRECTING ERRORS FOUND DURING DDT EXECUTION
14
RERUN SAVED PROGRAM
15
TERMINATE JOB 1
16
ATTACH TO JOB 2
17
TERMINATE JOB 2

CONSOLE OPERATION

```
1   . LOGIN↵
    JOB 1   DEC 3.16K ↵

    12,4↵
    AEDGC$↵
    0426   1-27-68 ↵
    ↑C↵
```

The user begins by identifying himself to the system. Monitor assigns the user a job number (JOB 1) and types the version of the Monitor currently in use (DEC 3.16K). User then responds by typing his project-programmer number (12,4) and his password (typed over a mask generated by Monitor). Monitor indicates acceptance of number and password by responding with the current time and date and ↑C.

```
2   . ASSIGN LPT↵
    LPT ASSIGNED↵
    . ASSIGN CDR↵
    CDR ASSIGNED↵
    . ASSIGN DTA DT↵
    DTA0 ASSIGNED↵
```

User assigns the line printer (LPT) and the card reader (CDR) to his job (both are nonsharable devices). He also requests that an available DECtape also be assigned and given the logical name "DT". Monitor responds by assigning an available DECtape, DTA0, to the job. At this point, the user mounts his DECtape on DECtape unit 0, places his FORTRAN source program deck in the card reader, and checks that both the card reader and printer are in a ready status.

NOTE: If the user is operating from a remote Teletype, he can transmit appropriate instructions to the console operator's Teletype via the Monitor TALK command.

Extremely tricky

```
    . R F40↵
```

Directs the Monitor to load and execute the FORTRAN IV compiler.

```
    * DT:ARRIVE,LPT:←CDR:↵

    MAIN.ERRORS DETECTED:  0↵
    TOTAL ERRORS DETECTED:  0↵
    9K CORE USED↵
```

The compiler responds with an asterisk when loaded. At this point, the user instructs the compiler to (1) read the source program deck from the card reader (←CDR:); (2) place the binary output on his DECtape and assign it a filename of "ARRIVE" (DT:ARRIVE); and (3) print the compilation listing on the line printer (LPT:).

NOTE: A listing of the FORTRAN source program deck is presented in Figure DEMO-1.

```
    *↑C↵
```

The compiler completes the compilation and types the number of errors detected and amount of core used and an * to indicate it is ready to do another compilation. Since the user has no further compilations at this time, he returns to the Monitor command level (↑C).

NOTE: The FORTRAN compilation listing is shown in Figure DEMO-2.

```
    .DEASSIGN LPT↵
    .DEASSIGN CDR↵
```

The user releases the line printer and the card reader so that other time-sharing users can have access to them.

```
(3) .R TECO↵
```

The user now directs the Monitor to load TECO, which he will use to generate a Macro source program directly from his Teletype onto his DECtape so that it will be available for future modification and reassembly.

TECO responds with an * when loaded. The user directs TECO to insert (I) into its output buffer the text which follows the "I" in its command buffer area. For buffer use efficiency, the text is entered in segments; each segment is terminated by two ALTMODE'S ($$), the first delimiting the insertion and the second causing TECO to execute the command string (i.e., transfer the contents of the command buffer into TECO's output buffer), clear the command buffer, and respond with an *. Each segment begins with a new insert command. For those use users incapable of the error-free typing shown in this example, TECO's features include the ability to wipe out one or more characters (by striking the RUBOUT key once for each of the previously typed characters to be erased) and the ability to wipe out the preceding line of characters (by typing 0LK, followed by a new insert command to enter the intended line).

```
*ITITLE  RANDOM NUMBER GENERATING SUBROUTINE↵
SUBTTL  CHARLIE PROGRAMMER     27    JAN    1968↵
↵
;RANDOM NUMBER GENERATING SUBROUTINE↵
$ $↵
*I;THE FORTRAN CALLING SEQUENCE IS --↵
;          CALL RANDOM(ARG)↵
;    WHERE ARG SPECIFIES THE LOCATION AT WHICH THE RESULTING↵
;    SINGLE PRECISION FLOATING POINT RANDOM NUMBER WILL BE↵
;    STORED.   NUMBERS PRODUCED BY THIS ROUTINE ARE "PSEUDO-↵
;    RANDOM NUMBERS"  BUT ARE UNIFORMLY DISTRIBUTED OVER[0, 1].↵
$ $↵
↵
*INTERNAL RANDOM↵
          ACX=5               ;ACCUMULATOR↵
          ACY=6               ; SYMBOLIC↵
          ACZ=ACY+1           ;   DEFINITIONS.↵
```

```
$$↵
 *IRANDOM: 0               ;ENTERED BY JSA 16,RANDOM↵
          CALL    ACX,[SIXBIT/TIMER]          ;GET TIME IN CLOCK TICKS.↵
          ANDI    ACX,3               ;USE TIME TO SELECT 1-4 ITERATIONS.↵
$$↵
*IRLOOP:  MOVE    ACY,RNUMBR          ;FETCH PREVIOUS PSEUDO-RANDOM NUMBER.↵
          MUL     ACY,MAGIC           ;MULTIPLICATIVE RANDOM NUMBER GENERATOR.↵
          MOVEM   ACZ,RNUMBR          ;SAVE NEXT PSEUDO-RANDOM NUMBER.↵
          SOJGE   ACX,RLOOP           ;ITERATE AGAIN ?↵
$$↵
*I        LSH     ACZ,-↑D8            ;CONVERT TO FLOATING POINT FORMAT↵
          TLO     ACZ,20000           ;  IN THE RANGE[0,1] .↵
          FADRI   ACZ,0               ;NORMALIZE.↵
          MOVEM   ACZ,@(16)           ;STORE RESULT, AND --↵
          JRA     16,1(16)            ;***RETURN.***↵
$$↵
↵
*I;THE MULTIPLIER USED IN THIS PSEUDO-RANDOM NUMBER GENERATOR IS 5 RAISED↵
; TO THE 15TH POWER (SEE COMPUTER REVIEWS VOL. 6, NO. 3, REVIEW NUMBER↵
; 7725, AND THE REFERENCED PAPER IN JACM JAN'65 PP83-89).↵
↵
MAGIC:  5*5*5*5*5*5*5*5*5*5*5*5*5*5*5     ;THE MULTIPLIER.↵
↵
RNUMBR: 1                ;THE NEXT RANDOM NUMBER IS ALWAYS HERE↵
;(THE ITERATION STARTS FROM A VALUE OF 1).↵
↵
PATCH:    BLOCK   10     ;PATCHING SPACE.↵
↵
          END↵
$$↵
*HT↵
$$↵
TITLE  RANDOM NUMBER GENERATING ...↵
SUBTTL CHARLIE PROGRAMMER ....↵
          etc.
PATCH: BLOCK    10    ;PATCHING ....↵
       END↵
```

The command "HT" followed by two ALTMODE'S ($ $) directs TECO to type out the entire contents of the output buffer.

The user notices that he has made two typing errors: (1) an "I" is missing from "NTERNAL" (the "I" already there is the insert command); and (2) a space is missssing between "PP" and "83" in his comments line.

```
*BJSNTERNAL Ⓢ-7CII Ⓢ-L2T↵
Ⓢ Ⓢ↵
INTERNAL        RANDOM↵
```

Search for "NTERNAL"; insert an "I" in front of it, and type out the corrected line. For details of this fairly complex command, see the section entitled "TECO".

```
*SPP83 Ⓢ-2CI Ⓢ0LTⓈ Ⓢ↵

; 7725, AND THE REFERENCED PAPER IN JACM JAN '65 PP 83-89.↵
```

Search for "PP83", insert a space between the "PP" and "83", and type out the corrected line.

```
*EWDT:RANDOM.MACⓈPWEFⓈ Ⓢ↵
```

Satisified that his source coding is correct, the user directs TECO to write out the contents of the output buffer on his DECtape and assign it the filename "RANDOM.MAC".

```
*↑C↵
```

Having finished with TECO, the user returns to the Monitor command level.

```
4) .R MACRO↵
```

The user directs Monitor to load the Macro-10 Assembler.

```
*DSK:BIN, /C ← DT:RANDOM.MAC↵
```

Macro-10 responds with an * when loaded. The user directs Macro-10 to assemble the source program file, RANDOM.MAC, located on DTA0. The binary output is to be written on disk under the filename "BIN". A modified assembly listing is written on the disk (assumed device) with /C switch and assigns it the filename CREF.TMP.

```
A   000001/040240   000026'   CALL   ACX,[SIXBIT/TIMER]↵
;GET TIME IN CLOCK TICKS.↵
```

A source language error has been encountered in the input.

```
THERE IS 1 ERROR↵
PROGRAM BREAK IS 000027↵
5K CORE USED↵
*↑C↵
```

User returns to the Monitor command level.

```
A) .DETACH↵
```

User detaches console from this job without affecting the status of the job. He is now free to initiate a new job.

```
 .LOGIN↵
 JOB 2 DEC 3.16K↵
#12, 4↵
 AEDF
→0458 1-27-68↵
 ↑C↵
```

crock

Monitor responds as before with user logged in as job 2 now.

```
(4B).R CREF↵
    *↵
```

User calls CREF to get the cross-reference assembly listing. CREF selects default assumptions of:

output - dev:	LPT:
input - dev:	DSK:
filename - ext	CREF.TMP

The LPT is available. The user does not assign it so that it will be available to other users when his listing is completed.

```
    ↑C↵
```

User interrupts CREF in progress and returns control to the Monitor.

```
(4C).CCONT↵
```

User allows the job to continue running and leave the console in Monitor mode.

```
    .ATTACH 1, (12, 4)↵
```

Automatically detaches the console from the current job and attaches it to job 1 belonging to 12,4 which was the previous user job.

```
(5) .R TECO↵
```

He directs Monitor to load TECO again.

```
    *ERDT:RANDOM.MAC$YBJS/TIMER ($) I/ ($) 0LT ($) ($)↵
              CALL      ACX,[SIXBIT/TIMER/];....↵
```

He directs TECO to read its input from filename RANDOM.MAC, search for "/TIMER", insert a "/" after it, and type out the corrected line.

```
    *EWDT:RANDOM.MAC ($) PWEF ($) ($)↵
```

He directs TECO to write the corrected file over the old file on his DECtape by giving it the same filename.

```
    *↑C↵
```

He then returns to the Monitor command level to call Macro-10 back in and reassemble.

```
(6) .R MACRO↵
    *DSK:BIN,LPT:←DT:RANDOM.MAC↵
```

The user reloads Macro-10, using the same commands as before. The previous contents of the file BIN on the disk are overwritten with the binary output produced by this assembly.

```
    THERE ARE NO ERRORS↵
    PROGRAM BREAK IS 000027↵
    5K CORE USED↵
    *↑C↵
```

NOTE: The assembly listing is shown in Figure DEMO-3.

The user now decides that he would like to transfer the binary output file just produced from the disk to his DECtape so that it will be together with the other program files of this job.

```
(7) .R PIP↵
```

To do this, he directs Monitor to load the Peripheral Interchange Program (PIP).

```
    *DT:RAN.REL←DSK:BIN.REL↵
```

PIP responds with an * when loaded. The user directs PIP to transfer the file BIN.REL from the disk to his DECtape and call it RAN.REL.

```
*↑C↵
```

PIP accomplishes the requested transfer and responds with another asterisk indicating readiness for a new command.

He returns to the Monitor command level to call Linking Loader.

```
8  .R LOADER 7↵
```

He directs Monitor to bring in Linking Loader and assigns it 7K of core.

```
   */D DT:RAN,ARRIVE (ALTMODE) ↵
```

He directs Linking Loader to load DDT (/D) from the SYS device and the two binary programs, RAN and ARRIVE, from his DECtape. Linking Loader automatically searches for files with an .ext of .REL (both the FORTRAN compiler and Macro Assembler create binary output files with this .ext assumed).

not described

```
   LOADER↵
   EXIT↵
   ↑C↵
```

Loading is completed. Linking Loader returns the user to the Monitor command level.

```
9  .START↵
```

The user directs Monitor to begin execution of the loaded programs at the starting address of the last loaded program (i.e., the FORTRAN program, "ARRIVE").

```
   RANDOM INTER-ARRIVAL TIME GENERATOR...↵
   TYPE MEAN WAITING TIME PLEASE:          100.0↵
   NOW TYPE NUMBER OF SAMPLE TIMES DESIRED:10↵
```

The user's program is now in control and prints instructions for entering parameters.

The user decides to try a mean waiting time of 100 and asks for 10 sample random times.

```
   T = 0.77751067E+04↵
   T = 0.78527278E+04↵
   T = 0.78476048E+04↵
   T = 0.78677823E+04↵
   T = 0.78103187E+04↵
   T = 0.77700529E+04↵
   T = 0.77733533E+04↵
   T = 0.77725470E+04↵
   T = 0.80251919E+04↵
   T = 0.77787954E+04↵

   TYPE MEAN WAITING TIME PLEASE:↑C↵
```

Since the program has obviously produced incorrect results (the output is far from random and conspicuously in the wrong range), the user returns to the Monitor command level.

```
(10) .DDT↵
     $RANDOM$:↵
     RLOOP+5/⇥TLO ACZ,20000⇥TLO ACZ,200000↵
```

The user calls DDT to aid him in his debugging. Although DDT is often used to dynamically obtain intermediate results and help the user find his errors, it is assumed for the purposes of this example that he has discovered his error by looking at his assembly listing. He then uses DDT to correct location RLOOP+5 (he had omitted a zero in his source coding).

Note that DDT permits symbolic typeouts and corrections.

```
     MAIN.$:↵
     12P+11$T/   ED:'$    "/ED: / (LINE FEED) ↵
     12P+12/     )        "/ '$)/↵
```

He also discovers that his FORTRAN program failed to space following the second request for input (NOW TYPE NUMBER....). Therefore, he also modifies locations 12P+11 and 12P+12 in his FORTRAN program. For details concerning DDT commands and responses, see "DDT." He then directs DDT to begin execution of his programs.

```
     RANDOM INTER-ARRIVAL TIME GENERATOR FOR POISSON PROCESSES↵
     TYPE MEAN WAITING TIME PLEASE:    100.0 ↵
     NOW TYPE NUMBER OF SAMPLE TIMES DESIRED:    10↵

     T = 0.13360079E+03↵
     T = 0.83559460E+01↵
     T = 0.48267604E+03↵
     T = 0.12974962E+00↵
     T = 0.12997161E+03↵
     T = 0.10218452E+03↵
     T = 0.18005033E+03↵
     T = 0.20455130E+02↵
     T = 0.40742972E+02↵
     T = 0.39184699E+01↵

     TYPE MEAN WAITING TIME PLEASE:↑C ↵
```

Results appear to be correct this time. The user returns to the Monitor command level.

```
(11) .SAVE DT:ARRIVE 7↵


     JOB SAVED↵
     ↑C↵
```

The user directs Monitor to save the core image of his corrected program on his DECtape, calling the saved file ARRIVE (the file extension .SAV is automatically appended) He specifies that 7K of core is to be used any time this program is run. After the image file has been written, Monitor automatically returns to the command level.

```
12) .R PIP↵
    *TTY:←DT:/L ↵
    426.  FREE BLOCKS LEFT ↵

    ARRIVE.REL          27-JAN-68↵
    RANDOM.MAC          27-JAN-68↵
    RAN.REL             27-JAN-68↵
    ARRIVE.SAV          27-JAN-68↵
```

The user now calls PIP and asks for a listing of his DECtape directory on his Teletype. In this way, he verifies that all of the files which should be on the tape are actually there.

PIP lists the number of free blocks left and the name and creation date of each file.

```
13) *↑C↵
    .R TECO↵
    *ERDT:RANDOM.MAC$YBJS20000$I0$0LT$↵

      TLO   ACZ,200000   ; IN THE RANGE[0,1].↵
```

The user returns to Monitor command mode and calls TECO. He updates his Macro source program (RANDOM.MAC) to reflect the correction made to the binary program via DDT, and has TECO type the corrected line (see "TECO" for details of command string).

```
    *EWDT:RANDOM.MAC $PWEF$$↵
```

The corrected file is written over the old version on the DECtape.

```
14) .RUN DT:ARRIVE↵
```

As a final proof that his program now works, he reruns the saved version.

```
    RANDOM INTER-ARRIVAL TIME GENERATOR FOR POISSON PROCESSES ↵
    TYPE MEAN WAITING TIME PLEASE:   50E+1↵
    NOW TYPE NUMBER OF SAMPLE TIMES DESIRED:    4↵

    T = 0.57171754E+03 ↵
    T = 0.90873993E+03 ↵
    T = 0.49264013E+03 ↵
    T = 0.22028286E+03↵

    TYPE MEAN WAITING TIME PLEASE:  ↑C↵
```

```
15) . KJOB ↵
    26.47↵
    26.47↵
    .
```

Having finished his work, the user terminates his job (KJOB). This releases his job number (and any ASSIGNed devices) to the Monitor pool. Monitor responds by typing the number of seconds of computer time used for the entire job and the number of seconds since he last asked for this information (via a TIME command). The user is now logged off the machine and the Teletype is in detached mode.

```
16) .ATTACH 2 [12, 4] ↵
```

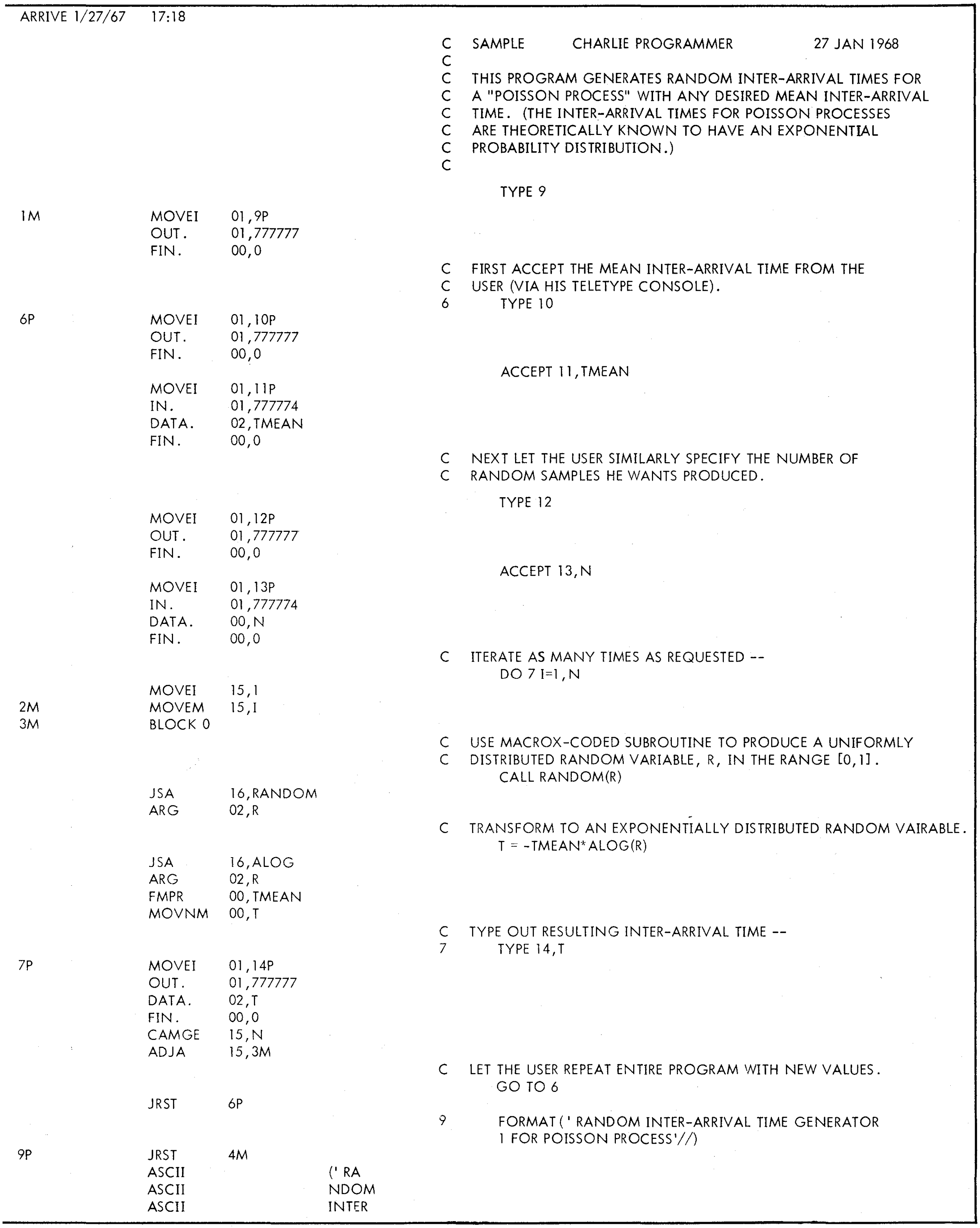
Attach to job which had been running CREF.

```
17) . KJOB↵
     00.30
     00.30
```

Kills job and releases devices.

```
C   SAMPLE PROGRAM --          CHARLIE PROGRAMMER        27 JAN 1968
C
C   THIS PROGRAM GENERATES RANDOM INTER-ARRIVAL TIMES FOR
C   A "POISSON PROCESS" WITH ANY DESIRED MEAN INTER-ARRIVAL
C   TIME. (THE INTER-ARRIVAL TIMES FOR POISSON PROCESSES
C   ARE THEORETICALLY KNOWN TO HAVE AN EXPONENTIAL
C   PROBABILITY DISTRIBUTION.)
C
      TYPE 9
C   FIRST ACCEPT THE MEAN INTER-ARRIVAL TIME FROM THE
C   USER (VIA HIS TELETYPE CONSOLE).
6     TYPE 10
      ACCEPT 11,TMEAN
C   NEXT LET THE USER SIMILARLY SPECIFY THE NUMBER OF
C   RANDOM SAMPLES HE WANTS PRODUCED.
      TYPE 12
      ACCEPT 13,N
C   ITERATE AS MANY TIMES AS REQUESTED --
      DO 7 I=1,N
C   USE MACROX-CODED SUBROUTINE TO PRODUCE A UNIFORMLY
C   DISTRIBUTED RANDOM VARIABLE, R, IN THE RANGE [0,1].
      CALL RANDOM(R)
C   TRANSFORM TO AN EXPONENTIALLY DISTRIBUTED RANDOM VARIABLE.
      T = -TMEAN*ALOG(R)
C   TYPE OUT RESULTING INTER-ARRIVAL TIME --
7     TYPE 14,T
C   LET THE USER REPEAT ENTIRE PROGRAM WITH NEW VALUES.
      GO TO 6
9     FORMAT(' RANDOM INTER-ARRIVAL TIME GENERATOR
      1 FOR POISSON PROCESSES'//)
10    FORMAT(' TYPE MEAN WAITING TIME PLEASE:     '$)
11    FORMAT(E)
12    FORMAT(' NOW TYPE NUMBER OF SAMPLE TIMES DESIRED:  '$)
13    FORMAT(I)
14    FORMAT(' T = ',E15.8)
    END
```

Figure DEMO-1   FORTRAN Source Program Deck

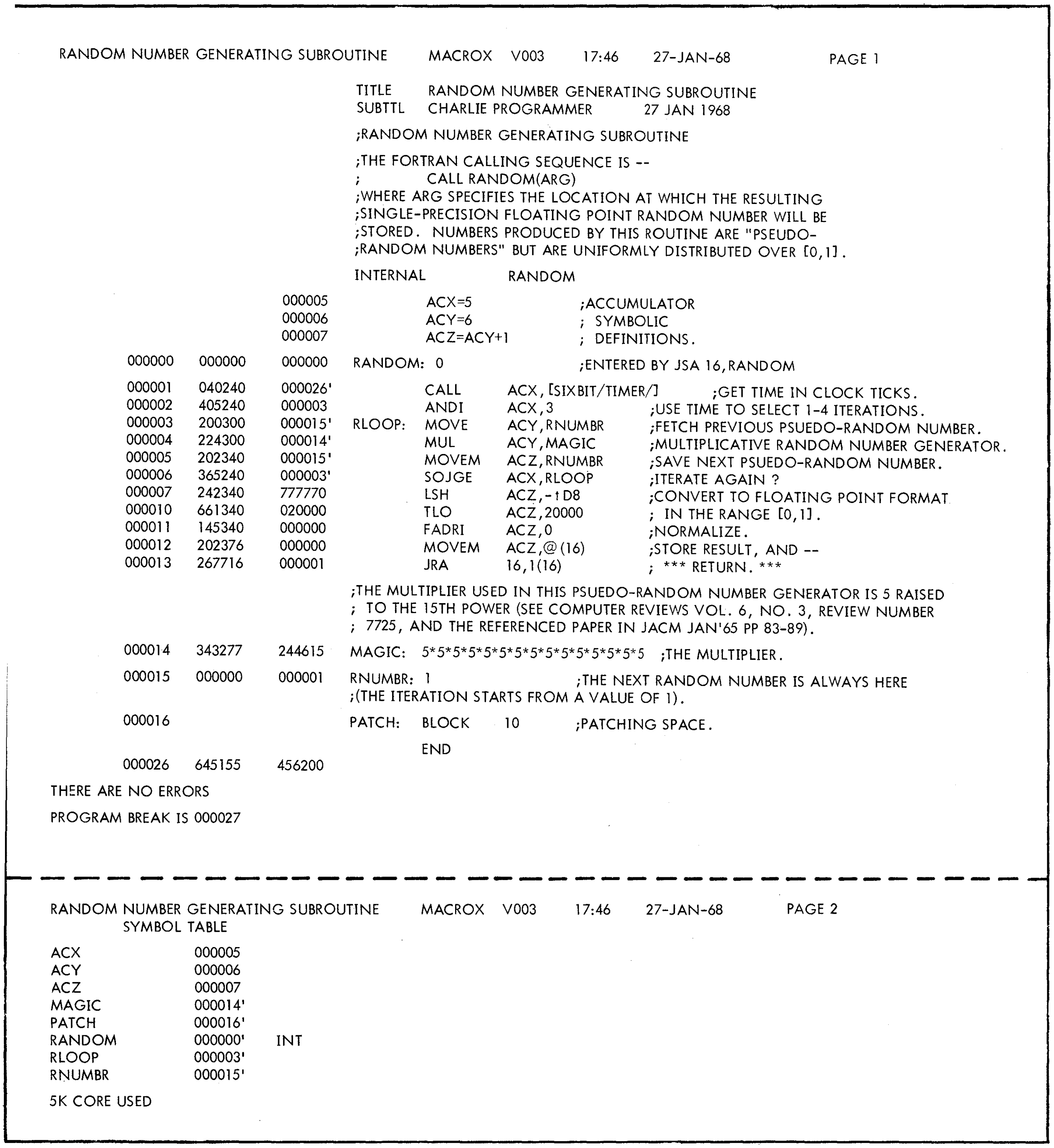


```
ARRIVE 1/27/67  17:18
                                                C   SAMPLE      CHARLIE PROGRAMMER          27 JAN 1968
                                                C
                                                C   THIS PROGRAM GENERATES RANDOM INTER-ARRIVAL TIMES FOR
                                                C   A "POISSON PROCESS" WITH ANY DESIRED MEAN INTER-ARRIVAL
                                                C   TIME.  (THE INTER-ARRIVAL TIMES FOR POISSON PROCESSES
                                                C   ARE THEORETICALLY KNOWN TO HAVE AN EXPONENTIAL
                                                C   PROBABILITY DISTRIBUTION.)
                                                C
                                                        TYPE 9
1M              MOVEI   01,9P
                OUT.    01,777777
                FIN.    00,0
                                                C   FIRST ACCEPT THE MEAN INTER-ARRIVAL TIME FROM THE
                                                C   USER (VIA HIS TELETYPE CONSOLE).
                                                6       TYPE 10
6P              MOVEI   01,10P
                OUT.    01,777777
                FIN.    00,0
                                                        ACCEPT 11,TMEAN
                MOVEI   01,11P
                IN.     01,777774
                DATA.   02,TMEAN
                FIN.    00,0
                                                C   NEXT LET THE USER SIMILARLY SPECIFY THE NUMBER OF
                                                C   RANDOM SAMPLES HE WANTS PRODUCED.
                                                        TYPE 12
                MOVEI   01,12P
                OUT.    01,777777
                FIN.    00,0
                                                        ACCEPT 13,N
                MOVEI   01,13P
                IN.     01,777774
                DATA.   00,N
                FIN.    00,0
                                                C   ITERATE AS MANY TIMES AS REQUESTED --
                                                        DO 7 I=1,N
                MOVEI   15,1
2M              MOVEM   15,I
3M              BLOCK 0
                                                C   USE MACROX-CODED SUBROUTINE TO PRODUCE A UNIFORMLY
                                                C   DISTRIBUTED RANDOM VARIABLE, R, IN THE RANGE [0,1].
                                                        CALL RANDOM(R)
                JSA     16,RANDOM
                ARG     02,R
                                                C   TRANSFORM TO AN EXPONENTIALLY DISTRIBUTED RANDOM VAIRABLE.
                                                        T = -TMEAN*ALOG(R)
                JSA     16,ALOG
                ARG     02,R
                FMPR    00,TMEAN
                MOVNM   00,T
                                                C   TYPE OUT RESULTING INTER-ARRIVAL TIME --
                                                7       TYPE 14,T
7P              MOVEI   01,14P
                OUT.    01,777777
                DATA.   02,T
                FIN.    00,0
                CAMGE   15,N
                ADJA    15,3M
                                                C   LET THE USER REPEAT ENTIRE PROGRAM WITH NEW VALUES.
                                                        GO TO 6
                JRST    6P
                                                9       FORMAT(' RANDOM INTER-ARRIVAL TIME GENERATOR
                                                        1 FOR POISSON PROCESS'//)
9P              JRST    4M
                ASCII              (' RA
                ASCII              NDOM
                ASCII              INTER
```

Figure DEMO-2.   FORTRAN Compilation Listing

```
                ASCII               -ARRI
                ASCII               VAL T
                ASCII               IME G
                ASCII               ENERA
                ASCII               TOR F
                ASCII               OR PO
                ASCII               ISSON
                ASCII               PROC
                ASCII               ESSES

                ASCII               '//)
4M              BLOCK 0
                                                 10    FORMAT(' TYPE MEAN WAITING TIME PLEASE:  '$)
10P             JRST       5M
                ASCII               (' TY
                ASCII               PE ME
                ASCII               AN WA
                ASCII               ITING
                ASCII               TIME
                ASCII               PLEA
                ASCII               SE:
                ASCII               '$)
5M              BLOCK 0
                                                 11    FORMAT(E)
11P             JRST       6M
                ASCII               (E)
6M              BLOCK 0
                                                 12    FORMAT(' NOW TYPE NUMBER OF SAMPLE TIMES DESIRED:'$)
12P             JRST       7M
                ASCII               (' NO
                ASCII               W TYP
                ASCII               E NUM
                ASCII               BER O
                ASCII               F SAM
                ASCII               PLE T
                ASCII               IMES
                ASCII               DESIR
                ASCII               ED:'$
                ASCII               )
7M              BLOCK 0
                                                 13    FORMAT(I)
13P             JRST       8M
                ASCII               (I)
8M              BLOCK 0
                                                 14    FORMAT (' T = ',E15.8)
14P             JRST       9M
                ASCII               (' T
                ASCII               = ',E
                ASCII               15.8)
9M              BLOCK 0

                                                     END
                JSA        16,EXIT
MAIN.%          RESET.     00,0
                JRST       1M

SUBPROGRAMS

FORSE.
RANDOM
ALOG
FLOUT.
FLIRT.
INTO.
INTI.
EXIT

SCALARS

TMEAN                      115
N                          116
I                          117
R                          120
T                          121
```

Figure DEMO-2 (Cont.) FORTRAN Compilation Listing



```
RANDOM NUMBER GENERATING SUBROUTINE      MACROX  V003    17:46    27-JAN-68          PAGE 1

                                        TITLE    RANDOM NUMBER GENERATING SUBROUTINE
                                        SUBTTL   CHARLIE PROGRAMMER        27 JAN 1968

                                        ;RANDOM NUMBER GENERATING SUBROUTINE

                                        ;THE FORTRAN CALLING SEQUENCE IS --
                                        ;        CALL RANDOM(ARG)
                                        ;WHERE ARG SPECIFIES THE LOCATION AT WHICH THE RESULTING
                                        ;SINGLE-PRECISION FLOATING POINT RANDOM NUMBER WILL BE
                                        ;STORED.  NUMBERS PRODUCED BY THIS ROUTINE ARE "PSEUDO-
                                        ;RANDOM NUMBERS" BUT ARE UNIFORMLY DISTRIBUTED OVER [0,1].

                                        INTERNAL         RANDOM

                        000005                  ACX=5                   ;ACCUMULATOR
                        000006                  ACY=6                   ; SYMBOLIC
                        000007                  ACZ=ACY+1               ; DEFINITIONS.

        000000  000000  000000          RANDOM: 0                       ;ENTERED BY JSA 16,RANDOM

        000001  040240  000026'                 CALL    ACX,[SIXBIT/TIMER/]     ;GET TIME IN CLOCK TICKS.
        000002  405240  000003                  ANDI    ACX,3           ;USE TIME TO SELECT 1-4 ITERATIONS.
        000003  200300  000015'         RLOOP:  MOVE    ACY,RNUMBR      ;FETCH PREVIOUS PSUEDO-RANDOM NUMBER.
        000004  224300  000014'                 MUL     ACY,MAGIC       ;MULTIPLICATIVE RANDOM NUMBER GENERATOR.
        000005  202340  000015'                 MOVEM   ACZ,RNUMBR      ;SAVE NEXT PSUEDO-RANDOM NUMBER.
        000006  365240  000003'                 SOJGE   ACX,RLOOP       ;ITERATE AGAIN ?
        000007  242340  777770                  LSH     ACZ,-↑D8        ;CONVERT TO FLOATING POINT FORMAT
        000010  661340  020000                  TLO     ACZ,20000       ; IN THE RANGE [0,1].
        000011  145340  000000                  FADRI   ACZ,0           ;NORMALIZE.
        000012  202376  000000                  MOVEM   ACZ,@(16)       ;STORE RESULT, AND --
        000013  267716  000001                  JRA     16,1(16)        ; *** RETURN. ***

                                        ;THE MULTIPLIER USED IN THIS PSUEDO-RANDOM NUMBER GENERATOR IS 5 RAISED
                                        ; TO THE 15TH POWER (SEE COMPUTER REVIEWS VOL. 6, NO. 3, REVIEW NUMBER
                                        ; 7725, AND THE REFERENCED PAPER IN JACM JAN'65 PP 83-89).

        000014  343277  244615          MAGIC:  5*5*5*5*5*5*5*5*5*5*5*5*5*5*5   ;THE MULTIPLIER.

        000015  000000  000001          RNUMBR: 1                       ;THE NEXT RANDOM NUMBER IS ALWAYS HERE
                                        ;(THE ITERATION STARTS FROM A VALUE OF 1).

        000016                          PATCH:  BLOCK   10              ;PATCHING SPACE.

                                                END

        000026  645155  456200

THERE ARE NO ERRORS

PROGRAM BREAK IS 000027


RANDOM NUMBER GENERATING SUBROUTINE      MACROX  V003    17:46    27-JAN-68          PAGE 2
        SYMBOL TABLE

ACX             000005
ACY             000006
ACZ             000007
MAGIC           000014'
PATCH           000016'
RANDOM          000000'   INT
RLOOP           000003'
RNUMBR          000015'

5K CORE USED
```

Figure DEMO-3 Macro-10 Assembly Listing

DECTAPE EDITOR (EDITOR)

FUNCTION

To create, add to, or delete from sequentially numbered source files recorded in lines of ASCII characters on a DECtape.[1]

- Provides a simple method of creating or modifying Macro or FORTRAN IV source programs

ENVIRONMENT

Monitor	All
Minimum Core	1K
Additional Core	Not used.
Equipment Required	One DECtape unit for the reel containing the file(s) to be modified.

[1] Editor edits the source file; i.e., the input and output files are the same. Fresh source files have editing space in each physical DECtape block. If the user has more edits for a block than will fit in it, an extra block in the DECtape is used and appropriately linked to the preceding and following logical blocks of the file.

INITIALIZATION

```
.R EDITOR↵
```

Loads the DECtape Editor program.

```
*
```

Editor is ready to receive a command.

COMMANDS

INITIALIZE A FILE FOR PROCESSING

```
Sn↑A↵
```

Select DECtape n and zero the directory.

```
Sn,filename.ext↑A ALTMODE
```

Select DECtape n, zero the directory, and create a file called filename.ext.

```
Sn,filename.ext↵
```

Select DECtape n and locate filename.ext for processing.

```
Sn,filename.ext ALTMODE
```

Select DECtape n and add a new file called filename.ext.

NOTE: All the above commands place Editor in the command mode; i.e., the next typein is assumed to be one of the commands given below.

INSERT A LINE

```
Innnnn↵
nnnnn aaaa.......a↵
nnnxx ALTMODE ↵
*
```

Insert the following typed line at line number nnnnn of the currently open file; nnnnn can be specified as a line sequence number or as a point (.). A point refers to the last line typed. If the line number already exists in the file, the line is replaced.

INSERT MULTIPLE LINES

```
Innnnn,increment↵
nnnnn aaaa...aaa↵
nnnxx bbbb...bbb↵
nnnxx ALTMODE↵
*
```

Insert the following typed lines, beginning at line number nnnnn of the currently open file; nnnnn can be specified as a line sequence number or as a point (.). Each time a line is entered, nnnnn is increased by the specified increment (assumed to be 00010 if omitted) and the result becomes the line number for the next insertion. Type ALTMODE after last line insertion.

DELETE A LINE

```
Dnnnnn↵

*
```

Delete line number nnnnn from the currently open file; nnnnn can be specified as a line sequence number or as a point (.).

DELETE A SERIES OF LINES

```
Dmmmmm,nnnnn↵

*
```

Delete lines mmmmm through nnnnn from the currently open file.

PRINT A LINE

```
Pnnnnn↵

nnnnn aaa...aaa↵

*
```

Print line number nnnnn of the currently open file; nnnnn can be specified as a line sequence number or as a point (.).

PRINT A SERIES OF LINES

```
Pmmmmm,nnnnn ↵

mmmmm aaa...aaa↵
.
.
nnnnn bbb...bbb↵

*
```

Print lines mmmmm through nnnnn of the currently open file.

CLOSE THE CURRENT FILE

```
E↵

*
```

Closes the currently open file. Another file can be opened on the same or a different DECtape via an Sn command, or a return can be made to Monitor to terminate Editor.

EXAMPLES

.R EDITOR↵	
*S1,VECTOR ALTMODE	Select DECtape 1 and create a new file on it called VECTOR.
*I20,20↵	Begin inserting at line sequence number 20 and increment this number by 20 each time a line is inserted. Switch to text mode.
00020 DEFINE VMAG(A,B)↵ 00040< MOVE 0,A↵ 00060 FMP 0↵ 00080 MOVE 1,A+1↵ 00100 FMP 1,1↵ 00120 FAD 1↵ 00140 MOVE 1,A+2↵ 00160 FMP 1,1↵ 00180 FAD 1↵ 00200 JSR FSQRT↵ 00220 MOVEM B↵	Editor responds with first line sequence number. Operator types line of coding to be inserted, followed by a carriage return.
00240 ALTMODE↵	Typing ALTMODE terminates insertions and returns Editor to command mode.
*I20↵	Change line number 00020.
00020 DEFINE VMAG(A,B,C)↵	
ILR↵	*ILR* indicates that the indexing increment has resulted in the next line number being equal to that of an already existing line (00040). Note that the indexing increment remains as 20 until explicitly changed.
*I90↵	Insert a line between lines 00080 and 00100.
00090 MOVE 1,C↵	
ILS↵	*ILS* indicates that the indexing increment has resulted in an existing line (00100) being skipped, since the next line addressed would be 00110.
*D180↵	Delete line 00180.

```
*P20,220↵                                   Print lines 00020 through 00220.

00020  DEFINE VMAG(A,B,C)↵
00040 <MOVE 0,A↵
00060  FMP  0↵
00080  MOVE  1,A+1↵
00090  MOVE  1,C↵
00100  FMP  1,1↵
00120  FAD  1↵
00140  MOVE 1,A+2↵
00160  FMP  1,1↵
00200  JSR  FSQRT↵
00220  MOVEM  B↵

*E↵                                         Close the currently open file.

*↑C↵                                        Return to the Monitor.

.KJOB↵                                      Kill the job, deassign the DECtape, and release
                                            core.

.
```

DIAGNOSTIC MESSAGES

Table EDITOR-1 Editor Diagnostic Messages

Message	Meaning
?DDE*	Device data error due to a write error or WRITE LOCK switch. Editor must be restarted.
?DEC*	DECtape directory is full.
?FAU*	A file name assigned to a new file already exists on the DECtape.
?ILC*	Illegal command.
ILR *ILS*	The line sequence increment specified for the insert function will cause the next existing line to be either replaced (R) or skipped (S). This is a warning message only and does not necessarily indicate an error.
?NCF*	Not a current file.
?NFO*	A command requiring an active file has been given but no file is currently open.
?NLN*	A print (P) or delete (D) command refers to a nonexistent line.
?UNA*	The DECtape specified in an Sn command is assigned to another job.

TEXT EDITOR AND CORRECTOR (TECO)[1]

FUNCTION

To edit files recorded in ASCII characters on any standard device.

- Performs simple editing functions as well as highly sophisticated search, match, and substitute operations
- Operates upon arbitrary length character strings under control of commands which are themselves character strings (and contains the mechanisms necessary to exploit this recursiveness)

ENVIRONMENT

Monitor	All
Minimum Core	4K
Additional Core	Takes advantage of any additional core available. Each 1K additional core augments the basic 6,200+- character buffer by 5K additional characters.[2]
Equipment Required	One input device and one output device.

[1] PDP-10 TECO was developed at Project MAC, Massachusetts Institute of Technology. The work of the following people is acknowledged: Daniel L. Murphy, Stewart Nelson, Jack Holloway, Richard Greenblatt.

[2] TECO automatically requests more core to expand its buffer when necessary ~~under any of the following situations~~:

1. An insert by way of the "I" command or "X" (Q Register) will overflow the present memory boundaries.
2. The command acceptance routine needs more core.
3. The total number of characters in the Data Buffer falls below 5000, and an input command from a peripheral device (other than the user console) is executed. Thus, TECO maintains a Data Buffer of at least 5000 characters.

If TECO is successful at obtaining more core, the following message will be typed:

```
*10000<IJ$>$$
STORAGE CAPACITY EXCEEDED
1K NEEDED, 5K CORE IN JOB
*
```

not anymore [nK CORE]

S.M.U. where n is the new core size

If TECO is unsuccessful at obtaining the core request, the following message is typed:

```
STORAGE CAPACITY EXCEEDED
11K NEEDED, NOT AVAILABLE
?
*
```

INITIALIZATION

```
.R TECO↲                              Loads the Text Editor and Corrector program.
*                                     TECO is ready to accept a command.
```

BASIC COMMANDS

NOTES: When typing command strings to TECO, the following points should be noted.

ALTMODE - One ALTMODE is used to terminate the text within a command string, where applicable; two successive ALTMODE's terminate the entire command string sequence and generate a RETURN, LINE-FEED. ALTMODE's type back as $'s.

RUBOUT - The RUBOUT key can be used to erase the preceding typed-in character(s) of a command string. Each character erased is echoed back on the Teletype (e.g., ABD RUBOUT DC...). Successive RUBOUT's can be used to erase more than one character.

N.B. To erase a carriage return (which generates RETURN, LINE-FEED), two RUBOUT's are required, one RUBOUT to erase the LINE-FEED and one to erase the RETURN.

Two successive ↑G's (BELL 's) can be used to wipe out the entire command string currently being typed.

TECO commands in the form ↑x (where "x" is any character) can be entered by either holding down the CTRL key while striking the "x" key or typing up-arrow (shift N) followed by the "x" character. These alternatives are not true where ↑x is a character within a text string (such as in a Search argument); in this case, the CTRL key must be used.

A carriage return, line feed, (↲) is ignored in a TECO command string as long as it does not appear within a particular command, such as Insert. Examples of this are given on the following pages.

SOFTWARE MANUAL UPDATE OCTOBER-DECEMBER, 1968

System User's Guide (DEC-10-NGCB-D)

The first supplement to this document is available. In addition to correcting minor errors and reflecting software changes on the pages included in the original distribution, this supplement contains new sections on AID, Macro-10 Instruction Summary, the Single-User (10/20, 10/30) System Builder and Monitors, and Assembly Instructions for CUSP's.

This supplement is available to all present holders of the System User's Guide and may be obtained from your local sales office or from the Program Library.

Changes not included in this supplement are:

Page INTRO-3 (lines 4 and 5) Change "..., you place......Monitor command mode." to "..., place your Teletype in Monitor command mode."

(line 17) Change "in the case of DECtapes and magnetic tapes)" to "in the case of DECtapes, magnetic tapes, and Teletypes)"

(2 lines from bottom) Change ".DMP" to ".SAV"

Page TECO-1 (lines 7 and 8 from bottom) Change: "STORAGE CAPACITY EXCEEDED 1K NEEDED, 5K CORE IN JOB" to

[nK CORE] where n is new core size of job

(3)

Page TECO-2 (SELECT THE INPUT DEVICE" - line 2) Flag "dev:" with a superscript 1.

dev:^{1}

("SELECT THE INPUT DEVICE" - bottom of box) Insert:

EBdev:filename.ext[proj,prog] (ALTMODE) Edit Backup - Selects an input and file to be edited (the input device, which will also be the output device for the edited file, must be the disk^1). EB is intended to be used to keep a backup of a file during a debugging session, without the user having to invent a new name for each version of the file.

Teco-3a

For example, the command string sequence

```
EBPROG1.MAC ALTMODE
.
.
editing
.
.
EF ALTMODE ALTMODE
```

results in

(1) Reading from file PROG1.MAC on disk

(2) Creating a new output file, which is initially given a temporary name of TECOnn.TMP, where nn is the user's job number in octal; incorporating job number in the filename solves the problem of identifying temporary files belonging to multiple 100,100 users

(3) Performing a number of RENAMEs following the EF command so that the input file becomes PROG1.BAK, any previous PROG1.BAK file is deleted, and the new output file becomes PROG1.MAC.

An ER command can be given following an EB command and before the EF command, but an intervening EW command is illegal and results in the error message ?22 (see Table TECO-2B). Even though an ER command may be given, the name of the final output file is still taken from the EB command.

("SELECT THE OUTPUT DEVICE" - line 5) Flag "dev:" with a superscript 1.

dev:[1]

("SELECT THE OUTPUT DEVICE" - bottom of box) Insert:

EX

EXit - The EX command finishes the current edit operation by (1) writing out all remaining pages of a file, (2) performing an EF (End of File) command, and (3) then exiting to the Monitor. (Thus, EX is equivalent to the command string

```
1000000PEF  BELL)
```

(Bottom of page) Add the footnote:

[1] If dev: is not specified, DSK: is assumed.

Teco 3-6

SELECT THE INPUT DEVICE

ERdev:filename.ext [proj, prog] ALTMODE

Selects the input device and file (if specified).

dev:	DTAn:	(DECtape)
	PTR:	(paper tape reader)
	DSK:	(disk)
	MTAn:	(magnetic tape)
	CDR:	(card reader)
	TTYn:	(Teletype)

filename.ext (DSK: or DTAn: only)

[proj, prog] (DSK: only)

Specified only if file is located in other than user's area.

SELECT THE OUTPUT DEVICE

EWdev:filename.ext [proj, prog] ALTMODE

Selects the output device and file (if specified).

EZdev:filename.ext [proj, prog] ALTMODE

Selects the output device and file (if specified), and rewinds the tape (if magnetic tape) or zeros the directory (if DECtape).

dev:	DTAn:	(DECtape)
	DSK:	(disk)
	MTAn:	(magnetic tape)
	PTP:	(paper tape punch)
	LPT:	(line printer)
	TTYn:	(Teletype)

filename.ext (DSK: or DTAn: only)

[proj, prog] (DSK: only)

Specified only if file is located in other than user's area.

EF

Terminate output on the current output file and close the file without selecting a new output file.

MAGNETIC TAPE POSITIONING

EM

Rewind the currently selected input magnetic tape.

nEM

Depending upon the value of n, perform one of the following operations on the currently selected input magnetic tape.

MAGNETIC TAPE POSITIONING (Cont)

n[1]	Operation
1	Rewind tape to load point.
3	Write end of file.
6	Skip one record.
7	Backspace one record.
8	Skip to logical end of tape.
9	Rewind and unload tape.
11	Erase 3 inches of tape.
14	Advance tape one file.
15	Backspace tape one file.

NOTE 1: Throughout TECO, all numbers in command strings are interpreted as decimal.

INPUT COMMANDS

same thing

Y Read from current input device into buffer until

1. A FORM character is read (i.e., a "page" has been input), or
2. The buffer is more than 2/3 full and one of the following is encountered

   (a) Line Feed

   (b) Form Feed

   or a point 128 characters from the end of the buffer is reached.

NOTES:

1. The FORM character, if read, does not enter the buffer.
2. Any data previously residing in the buffer is destroyed.
3. The pointer is positioned immediately before the first character in the buffer.
4. ~~Representative buffer size for 5K TECO:~~

   ~~Total buffer capacity = approx. 11,200 characters~~

   ~~2/3 buffer capacity = approx. 7,460 characters~~

   ~~1 line-printer page = 7,200+ characters (120 char./line)~~

   ~~(60 lines) 7,800+ characters (130 char./line)~~

omit (if anywhere belongs in TECO ma

A Read from the current input device and append the incoming data to information already residing in the buffer. Terminate reading on the same conditions as in Y.

NOTES:

1. No previous data is destroyed.
2. The pointer is not moved.

OUTPUT COMMANDS

PW Output the entire buffer to the selected output device, with a FORM character appended as the last character. Do not alter the contents of the buffer or move the pointer.

nP n times Equivalent to a PW command followed by a Y command (i.e., output the current contents of the buffer followed by a FORM character, and then read in more data from the input device). P = 1P

If n is specified, repeat this operation n times. If n is omitted, it is assumed to be equal to "1."

m,nP Output the m+1 through the nth character from the buffer to the current output file. Do not append a FORM character at the end. Do not alter the contents of the buffer or move the pointer.

EDITING COMMANDS

Move the Pointer

nJ Move pointer to right of the nth buffer character and give the pointer symbol (.) the value of n. If n is omitted, set pointer in front of the first buffer character (same as 0J).

nC Set the pointer to the right of the nth character beyond the pointer's present position (equal to .+nJ). If n is omitted, 1 is assumed.

nR Set the pointer to the left of the nth character prior to the pointer's present position (equal to .-nJ). If n is omitted, 1 is assumed.

nL +n - Move the pointer to the right, stopping after it has passed over n LINE-FEED characters.

-n - Move the pointer to the left, stopping after it has passed over n+1 LINE-FEED characters, then move to the right of the last LINE-FEED character passed over.
If n is omitted, assume 1L.

Delete Text

nD Delete n characters.

+n - Delete them just to the right of the pointer.

-n - Delete them just to the left of the pointer.
If n is omitted, 1 is assumed.

nK +n - Move the pointer to the right, stopping after it has passed over n LINE-FEED characters. Delete all characters the pointer passes over.

-n - Move the pointer to the left, stopping after it has passed over n+1 LINE-FEED characters, then move it to the right of the last LINE-FEED character passed over. Delete all characters between this point and the pointer's previous position.
If n is omitted, 1 is assumed.

m,nK Delete the m+1 through the nth characters of the buffer. Set the pointer where the deletion occurred.

EDITING COMMANDS (Cont)

Insert Text

Itext... (ALTMODE)

Insert the text following the "I" up to, but not including, the ALTMODE character, beginning at the current pointer position. Move the pointer to the right of the inserted material.

nI — Insert at the pointer location a character whose ASCII code is n (n must be a decimal value). Move the pointer to the right of the inserted character.

n\ — Insert at the current pointer location the ASCII text representation of the decimal value of the expression n. Move the pointer to the right of the inserted text.

→| text... (ALTMODE)

Insert at the current pointer location a TAB (→|) character and the following text up to but not including the ALTMODE character. Move the pointer to the right of the inserted text.

@I/text/ — Insert at the current pointer location the text which follows. The text is delimited by a character, /, which can be any character not appearing in the text.

Type Text

NOTE: T commands do not move the pointer.

nT — Type out the string of characters beginning at the current pointer position and terminating after the nth LINE-FEED character is encountered.

+n n lines to the right of the current pointer position.

-n n lines to the left of the current pointer position.

If n is omitted, the value is assumed to be "1."

m,nT — Type out the m+1 through the nth characters of the buffer.

STAND-ALONE EXAMPLES (BASIC)

Open an Input File

a) ERDTA5:SOURCE.MAC ALTMODE

Open the input file called SOURCE.MAC located on DTA5.

b) ERDSK:SRCE.MAC[12,24] ALTMODE

Open the input file called SRCE.MAC located in area 12,24 on the disk.

c) ERPTR: ALTMODE

Open an input file on the paper tape reader.

Open an Output File

a) EWDTA3:EDITED.MAC ALTMODE

Open an output file on DTA3 and call it EDITED.MAC.

b) EZDTA1:DEBUG.MAC ALTMODE

Zero the directory on DTA1, open an output file on it, and call the file DEBUG.MAC.

Read a Page

a) Y

Read a page into the buffer from the current input file, destroying the previous contents of the buffer.

b) A

Read a page into the buffer, appending the data to the end of the information currently in the buffer.

Output Data

a) PW

Output the entire buffer, followed by a FORM character.

b) 6P

Execute the write and read cycle six times.

c) 12,50P

Write out the 13th through the 50th characters of the buffer.

Pointer Positioning

a) Y18J

b) 5R

a) Read in a page of information and position the pointer after the 18th character of the buffer; b) Then move the pointer left to between characters 13 and 14.

Delete Text

a) J19C3D

Move the pointer to the right of the 19th character in the buffer and then delete the next three characters to the right (characters 20, 21, and 22).

or

b) 19,22K

Delete the 19+1 (20th) through the 22nd characters of the buffer.

Insert Text

a) J2LITAG: MOVE 1,AMT (ALTMODE)

Move the pointer to a position following the second line of the buffer; insert the text "TAG: MOVE 1,AMT" between the second and third lines of the buffer.

b) 69\

S.M.D. ??

Insert the digits "69" in ASCII at the current pointer position (same as I69 or 54I571).

~~NOTE: \ is typed with a SHIFT (FORM)~~.

c) ~~(TAB)~~ ERROR IN JOB (ALTMODE)

Insert a tab followed by the text "ERROR IN JOB" at the current pointer position.

d) @I#ERDSK:PROG (ALTMODE) #

Insert the text "ERDSK:PROG (ALTMODE) at the current pointer position.

NOTE: The use of delimiters is the only method for inserting an ALTMODE in the text.

Typing Text

a) 3T

~~Assuming that the pointer is at the beginning of the buffer, t~~Type out the first three lines of the buffer.

b) 25,100T

Type out the 25+1 (26th) through the 100th character of the buffer.

EXAMPLES (BASIC)

.R TECO↵	
*ERDTA1:SCFILE.MAC (ALTMODE) (ALTMODE)↵	Open the file called SCFILE.MAC on DTA1 for input.
*EWDTA2:EDFILE.MAC (ALTMODE) (ALTMODE)↵	Open an output file on DTA2 and call it EDFILE.MAC.
*Y0,20T (ALTMODE) (ALTMODE)↵ aaaaa.......aaaaa↵	Read a buffer of information from the input file and type the first 20 characters of the buffer.
*3LT (ALTMODE) (ALTMODE)↵ bbbbb.......bbbbb↵	Move the pointer to the right, stopping when three LINE-FEED characters have been encountered; type the text of the fourth line in the buffer.
*ITHIS IS A SAMPLE INSERT↵ (ALTMODE) (ALTMODE)↵	Insert the text "THIS IS A SAMPLE INSERT↵" between the third and fourth lines of the buffer and position the pointer after the inserted material.
*10PT (ALTMODE) (ALTMODE)↵ ccccc......ccccc↵	Write out the current buffer to the output device; read in and write out the next nine "pages" of data; read in the 11th page of data and position the pointer at the beginning of the buffer; type out the first line of the buffer.
*K (ALTMODE) (ALTMODE)↵	Delete this line from the file; position the pointer at the beginning of the (now) first line in the buffer.
*200PPWEF (ALTMODE) (ALTMODE)↵	Repeats the write and read cycle 200 times and writes out the last page before terminating the output file.
*↑G (ALTMODE) (ALTMODE)↵ EXIT↵ ↑C↵	Return control to the Monitor after all output requests have been completed.
.KJOB↵	Kill the job, deassign all devices, release core.

⊙ not anymore

SEARCH COMMANDS

Summary

S text (ALTMODE) - Searches for text in current buffer only.

N text (ALTMODE) - Searches for text through successive buffers by repeatedly writing out current buffer and reading in next buffer (P command).

← text (ALTMODE) - Searches for text through successive buffers by repeatedly reading in new bufferful of information (Y command).

1. All searches begin at the current location of the pointer.

2. Modifiers:

   Each search command can be preceded by the modifier characters, : and/or @.

   : causes the search command to have a numeric value at completion;
   0 if the search has failed (the requested text was not found) or
   -1 if the search was successful (the requested text was found).
   @ indicates that the text to be matched is delimited by some character (same as in the @I command).

3. Numeric Arguments:

   A numeric argument can appear following the modifiers (if any) but preceding the command. If the numeric argument is n, TECO searches for the nth occurrence of the text. If n is not used, the value of n is assumed to be "1."

4. Pointer Positioning:

   If search is successful, the pointer is positioned to the right of the matched text.

   If the search fails, the pointer is positioned at the beginning of the buffer.

5. Use of Special Characters Within Text:

   ↑S - Match any separator character (any character not a letter, number, period, dollar sign, or percent symbol).

   ↑X - Match any (arbitrary) character. Used when the contents of some position within the text is unimportant.

   ↑Nx - Match any character except x.

   ↑Q - Takes the next character literally, even if it is one of these four special characters. For example, S↑Q↑X (ALTMODE) - Find the character↑X.

NOTE: See note on page TECO-2.

Table TECO-1 Search Commands Summary

Command	Action at End of Buffer	Action at End of File	Values		Typeout? if Failure
			Success	Fail	
S	Failure	N/A	N/A	N/A	Yes
:S	Failure	N/A	-1	0	No
N	Performs a P command and resumes search	Failure	N/A	N/A	Yes
:N	Performs a P command and resumes search	Failure	-1	0	No
←	Performs a Y command (read only) and resumes search	Failure	N/A	N/A	Yes
:←	Performs a Y command (read only) and resumes search.	Failure	-1	0	No

Q-REGISTER COMMANDS

Q registers are provided for storing quantities, command strings, or buffer contents for later use. Thirty-six Q registers, labeled 0 through 9 and A through Z, are available.

nUi Places the numeric value n in Q-register i.

Qi Represents the current value in Q-register i.

%i Adds 1 to the value in Q-register i and represents the new value.

m,nXi Copies characters m+1 through the nth character of the buffer into Q-register i. Does not alter buffer contents or pointer.

nXi Copies the buffer characters between the current pointer position and the nth LINE-FEED character in Q-register i.

Gi Inserts the text contained in Q-register i into the buffer beginning at the current pointer location. Set the pointer to the right of the insertion.

[i Pushes the contents of Q-register i onto the Q-register pushdown list.

]i Pops the top entry of the Q-register pushdown list into Q-register i. The Q-register pushdown list is cleared each time two successive ALTMODE's are typed.

MACRO, ITERATION, AND CONDITIONAL COMMANDS

Mi Perform the text in Q-register i as a series of commands.

< > Iteration brackets. When > is encountered, command interpretation is sent back to <.

n < > Perform the commands within the iteration brackets n times.

; If not in an iteration, an error results. If most recent search failed, send command interpretation to just beyond the matching > on the right; otherwise, no effect.

n; If not in an iteration, an error results. If the value of n is 0 or greater, send command interpretation just past the matching > to the right; otherwise, no effect.

!tag! Tag definition. Tag is the name of the location in which it appears in a command string.

Otag ALTMODE Go to the named tag, which must appear in the current macro or command string.

n"G If n ≤ 0, ~~send~~ skip command interpretation to the next matching '; if n >0, ~~no effect~~ continue.

n"L If n ≥ 0, ~~send~~ command interpretation to the next matching'; if n <0, ~~no effect~~.

n"N If n = 0, ~~send~~ command interpretation to the next matching'; if n ≠ 0, ~~no effect~~.

n"E If n ≠ 0, ~~send~~ command interpretation to the next matching'; if n = 0, ~~no effect~~.

n"C If n is not one of the symbol constituents (a letter, number, period, dollar sign, or percent symbol), ~~send~~ skip command interpretation to the next matching '; otherwise, ~~no effect~~ continue.

NOTE: The " and ' symbols are matched in the same way as the (and) symbols.

NUMERIC VALUES AND ARGUMENTS IN COMMAND STRINGS

Many command string formats permit arguments with numeric values. The following characters may appear in a command string to develop these values in any instance where a numeric value is permissable.

0 through 9 Represent their corresponding numeric values.

B Equivalent to 0.

Z Equivalent to the number of characters in the buffer.

. Equivalent to the number of characters to the left of the current pointer position (or in other words, equal to the current pointer position).

Qi Equivalent to most recent numeric value placed in Q-register i.

nA Equivalent to ASCII value of character to right of pointer; n is used to differentiate this argument from an Append command (A) and has no other significance.

↑H Equivalent to value of elapsed time in 60ths of a second since midnight.

↑F Equivalent to the value of the console data switches.

NUMERIC VALUES AND ARGUMENTS IN COMMAND STRINGS (Cont)

↑E	Has the value of the form feed switch. If, during the last Y or A command execution, data transmission was terminated by a form feed character, ↑E has a value of -1, otherwise, the value is 0.
↑↑	(On Models 33 and 35, hold down both the CTRL and SHIFT keys and type "N".) Equivalent to the ASCII value of the next character in the command string; this character is not interpreted as a command.
↑T	Stops command execution until user types a character on the Teletype; ↑T then becomes equivalent to the ASCII value of the character typed.
\	Equivalent to the value represented by the digits (or minus sign) immediately following the current pointer position. The value is terminated by the first nonnumeric character encountered. The pointer is positioned immediately following the value.
m+n m-n	Add Subtract } Take one or two arguments. A space is equal to +.
m*n m/n	Multiply Divide (truncates) } Take one or two arguments.
m&n	Logical AND; bitwise AND of binary representations m and n.
m#n	Logical IOR; bitwise inclusive OR of binary representations m and n.
( )	Operators +, -, *, /, #, and & are normally performed left to right. This sequence can be overruled by use of parentheses. NOTE: TECO does not assume that multiplication and division are always performed before addition and subtraction. Thus, to obtain the equivalent of a + (b * c), one must use the parentheses; otherwise, (a + b) * c is assumed.
n=	Causes the value of n to be typed out.
H	Abbreviation for B, Z. (0 through the last location of the buffer; in other words, the whole buffer).

NOTES: **If a command takes two numeric arguments, a comma is used to separate them.**

TECO TERMINATION COMMANDS

↑C	Returns control to the Monitor without waiting for any I/O operations to finish.
↑G (BELL)	Returns control to the Monitor after completing all current output requests.

STAND-ALONE EXAMPLES (ADVANCED)

Search

J3SMOVE (ALTMODE) IM (ALTMODE)

Within the current buffer, search for the third occurrence (3S) of the text "MOVE", position the pointer immediately after it, and insert an "M" at that point.

Search for a Special Character

a) S↑NA (ALTMODE)

Search for any character except A within the current buffer.

b) S↑S (ALTMODE)

Search for any separator character within the current buffer.

Q-Registers, Macros, Iterations, and Conditionals

a) J0UN<S (LINE-FEED) (ALTMODE) ; %N>QN=

Count the number of LINE-FEED characters in the buffer as follows:

1. Position the pointer at the beginning of the buffer (J),
2. Place 0 in Q-register N (0UN),
3. Perform a search for a LINE-FEED character (S LINE-FEED ALTMODE); if one is found, add 1 to Q-register N (;%N). Go back (<>) and repeat this cycle until the end of the buffer is reached and the test fails (;); at this point type out the contents of Q-register N (QN=).

b) J<SJUMPA (ALTMODE) ; -4DIRST (ALTMODE) >

Replace all instances of the text "JUMPA" with "JRST" in the current buffer.

1. Position the pointer at the beginning of the buffer (J).
2. Search for JUMPA; when found, backspace the pointer four positions and delete the four characters passed over (;-4D).
3. Replace these four characters with the characters "RST" (IRST).
4. Repeat this routine (<>) until the test fails (end of the buffer has been reached) and exit (;) to >.

To Place a Command in a Q-Register for Later Execution

@I#J0UN<S (LINE-FEED) (ALTMODE) ↵ ;%N>QN=#HXP	1. Insert the text "J0UN<S (LINE-FEED) (ALTMODE) ;%N>QN=" into the buffer (@I#.........#) 2. Copy the contents of the buffer into Q-register P (HXP).

To Execute the Command:

	1. Read in a page of a file to search. (ERDTA3:FN.EX (ALTMODE) Y)
ERDTA3:FN.EX (ALTMODE) YMP	2. Execute the command stored in Q-register P (MP).

To Read in Text to be Inserted in Several Places in a File and to Store it in a Q-Register

ERPTR: (ALTMODE) YHXP↵	1. Assume that the text to be inserted is on paper tape. Open an input file on the paper tape reader (ERPTR:); read the text into the buffer (Y); copy the contents of the buffer into Q-register P (HXP).
ERDTA4:TXTEDT (ALTMODE)↵ EWDSK:TXTEDU (ALTMODE)↵	2. Open the input file to be edited and the output file to contain the edited version.
YNCALC: (ALTMODE) GP↵	3. Read a page from the input file and initiate a search for the text "CALC:". When found, insert the text stored in Q-register P at that point (GP).
NTOT: (ALTMODE) GP	4. Search for the text "TOT:" and, when found, insert the text stored in Q-register P after it.

EXAMPLES (ADVANCED)

```
.R TECO ↵
*ERMTA1: (ALTMODE) EM14EM (ALTMODE)(ALTMODE) ↵
```

Select MTA1 for input; rewind the tape (EM) and advance the tape one file (14EM).

```
*EZDTA1:REVFIL (ALTMODE)(ALTMODE) ↵
```

Select DTA1 for output; zero the directory; open a file and call it REVFIL.

```
*YNTAXRT (ALTMODE) 0LT↵
1X1 (ALTMODE) (ALTMODE) ↵
aaaa...TAXRT aaaa......aaaaa ↵
```

Read in the first page from the input file; search for the text "TAXRT"; if not found, write the buffer out, read in the next page, search again, etc.; continue this cycle until either TAXRT is found or end of file is reached. If TAXRT is found, position the pointer at the beginning of the line containing it, type the line, and place the line in Q-register 1.

```
*JNTXRTE (ALTMODE) 0LT↵
G1 (ALTMODE) (ALTMODE) ↵
bbb...TXRTE bbb......bbbbb ↵
```

Search the buffer for the text "TXRTE"; if not found, write out the buffer, read the next page, search again; continue this cycle until either TXRTE is found or end of file is reached. If TXRTE is found, position the pointer at the beginning of the line containing it, type the line, and insert the contents of Q-register 1 immediately before that line.

```
*NTXTEND: (ALTMODE) ↵
J<SA (ALTMODE) ;1A-47"G1A-58"L-DIB (ALTMODE) ' '>↵
PWEF (ALTMODE)(ALTMODE) ↵
```

Read pages from the input file and write them on the output file until end of file (marked by the text "TXTEND:") is found. At that point, move the pointer to the beginning of the buffer (J), and search for all A's in the buffer (SA); if the character following the "A" is a digit, 0 through 9 (ASCII codes 48_{10} through 57_{10}), change the "A" to a "B" (IB); continue searching and modifying until end of buffer is reached; write out last page and write end of file on output device.

```
*↑G (ALTMODE) (ALTMODE) ↵
EXIT ↵
↑C↵
```

Return control to the Monitor after all output requests have been completed.

```
.KJOB↵
.
```

Kill the job, deassign all devices, release core.

Pages SRCCOM-1 through SRCCOM-4 - Replace with attached pages.

Page BINCOM-3 — (line 5, command column) Change "file 1-word....XOR" to "loc file-1-word file 2-word XOR"

Page BINCOM-4 — (Table BC-1) Replace this table with Table BINCOM-1 on the following page.

(last line) Change to:

"octal loc. file1-word file2-word XOR of both words"

Page DESK-6 — (line 3, command column) Change "0.54402" to "-0.54402"

(line 6, command column) Change "0.839" to "-0.839"

Page BATCH-3,
BATCH-4 — Note that commands can be abbreviated to just the two or three letters necessary to make them unique (e.g., IN, SK, EN, EF, etc.)

STACK Section — A library writeup, "PDP-10 Job Stacker (STACK)", is now available. You may want to place this writeup in this section of your manual until the next System User's Guide Supplement is available.

Page BUILD-4 — (third line from bottom) Change

"Standard = 2 (200 bpi, odd)" to

"Standard = 2 (556 bpi, odd)"

Page MONITOR-6 — (line 4, Format Column) Change "proj,prog" to "[proj,prog]"

Page MONITOR-7 — (line 14, Diagnostic Messages Column) Change "An error was detected while" to "An error was detected while reading or"

Page CUSP-4 — (line 13 - Command Column) Change "...3616)↲ " to "...3616↲ "

(line 14 - Command Column) The "↲" should be blue.

(line 16 - Command Column) The "↲" should be blue.

DIAGNOSTIC MESSAGES

Table TECO-2A TECO Diagnostic Messages

Message	Meaning
?n	n is a decimal number associated with one of the list of error messages given in Table TECO-2B. TECO ignores the remainder of the command string and returns to the idle state. At this point, the user can type back ?, causing TECO to type out the command string terminated by the bad command.

Table TECO-2B Error List for ?n Messages

n_{10}	Meaning
1	TECO attempted to read commands beyond the terminating ALTMODE ALTMODE. This error is probably due to an unterminated @I or @S command, or to an unsatisfied O command.
2	Same as 1.
3	An attempt was made to supply more than two arguments to a command, either by the use of two commas or by "H,".
4	Too many right parentheses.
5	= command with no argument.
6	U command with no argument.
7	Q, U, X, or G command specifies an illegal Q-register (i.e., other than A through Z or 0 through 9).
8	In an X command, the second argument is not greater than the first.
9	In a G command, the Q-register does not contain text.
10	In a G command, the data in the Q-register is not in correct form (this is an internal error).
11	In an Ec command (e.g., ER, EW, EF, etc.), c is illegal.
12	File not found on LOOKUP.
13	Blank filename specified for directory device.
14	Project-programmer number specified does not have UFD.

Table TECO-2B (Cont) Error List for ?n Messages

n_{10}	Meaning
15	Protection failure on disk.
16	File cannot be accessed because it is currently being written.
17	LOOKUP or ENTER returned error type 6 (not defined).
18	LOOKUP or ENTER returned error type 7 (no device).
19	Directory full on ENTER.
20	Requested I/O device not available.
21	Not assigned.
22	EW command between an EB command and its EF.
23	EM command given, but no input file open.
24	nEM command, where n is not in the range 1 to 16.
25	Internal error: EF after EB, but no input file is open.
26	Illegal character in filename.
27	Illegal character in project-programmer number.
28	Attempt to read an input page when no file has been opened for input.
29	I/O error on input device.
30	Attempt to output a page when no file has been opened for output.
31	Two arguments were supplied for an L command.
32	Attempt to move pointer beyond page.
33	A 2-argument command has its second argument less than the first argument.
34	Attempt to search for too long a character string.
35	Search command did not find the required string.
36	In an M command, the Q-register does not contain text.
37	In an M command, the data in the Q-register is not in correct form (this is an internal error).
38	Unmatched right angle bracket.

Table TECO-2B (Cont) Error List for ?n Messages

n_{10}	Meaning
39	; encountered when not in iteration.
40	" command with no numeric argument, or "x where x is not G, L, E, N, or C.
41	This is the number typed out at the end of the ? command's dump of the command string in error. Refer to the number of the previous error.
42	A character has been encountered as an undefined command.
43	A ↑D command, when DDT has not been loaded with TECO.
44	Not enough core available from the Monitor.
45	A RENAME attempted with either a blank name or one already in use. Presumably due to a fault in the EB command.

Debugging Aids

As an aid in debugging macros and iterations, TECO can be set in the trace mode by typing ? as any character other than the first in a command string. When in trace mode, TECO types out each command as it is interpreted, interspersed with requested output. Typing a second ? in the same manner takes TECO out of trace mode; the ? can be typed each time it is desired to change the current mode.

The user can also type comments on his Teletype sheet as he executes TECO by typing:

↑Atext ↑A

This causes all text entered to be printed on the Teletype (with the exception of terminating ↑A character).

NOTE

Since the terminator ↑A is not a command, it must be typed by holding the CTRL key down while typing "A"; it cannot be entered as "up arrow, A."

If DDT (Dynamic Debugging Technique program) has been loaded along with TECO by Linking Loader, control can be transferred to DDT by using the command

↑D

MACRO—10 ASSEMBLER (MACRO)

FUNCTION

To assemble source programs coded in the Macro-10 programming language and produce machine language programs which are compatible with the Linking Loader and the Dynamic Debugging Technique program.

- Sophisticated 2-pass assembly
- Device independent
- Complete macro instruction facilities
- Symbolic linkage to other independently-generated programs
- Unlimited indexed/indirect addressing and expanded address arithmetic
- Ten data-generating pseudo-operations and eleven conditional assembly pseudo-operations
- Object coding produced in either relocatable or absolute address format
- Accepts input from any input device

ENVIRONMENT

Monitor	All
Minimum Core	5K
Additional Core	Automatically requests additional core assignments from the time-sharing monitor as needed.
Equipment Required	One input device (source program input). Two output devices (machine language program output and listing output). If the listing output is to be used as input to the Cross Reference (CREF) program, it must be written on either DECtape, magnetic tape, or disk.

INITIALIZATION

.R MACRO↵	Loads the Macro-10 Assembler into core.
*	The assembler is ready to receive a command.

COMMANDS

General Command Format

```
objprog-dev:filename.ext,list-dev:filename.ext ←— source-dev:filename.ext,
                                                  ......source-n↵
```

objprog-dev: The device on which the object program is to be written.

MTAn:	(magnetic tape)
DTAn:	(DECtape)
PTP:	(paper tape punch)
DSK:	(disk)

list-dev: The device on which the assembly listing is to be written.

MTAn:	(magnetic tape)	Must be one of these if input to CREF.
DTAn:	(DECtape)	
DSK:	(disk)	
LPT:	(line printer)	
TTY:	(Teletype)	
PTP:	paper tape punch	S.M.D.

source-dev: The device(s) from which the source-program input to assembly is to be read.

MTAn:	(magnetic tape)
CDR:	(card reader)
DTAn:	(DECtape)
DSK:	(disk)
PTR:	(paper tape reader)
TTY:	(Teletype)

If more than one file is to be assembled from a magnetic tape, card reader, or paper tape reader, dev: is followed by a comma for each file beyond the first.

Input via the Teletype is terminated by typing CTRL Z (↑Z) to enter pass 1; the entries must be retyped at the beginning of pass 2.

filename.ext (DSK: and DTAn: only)

The filename and filename extension of the object program file, the listing file, and the source file(s).

←

The object program and listing devices are separated from the source device by the left arrow ~~symbol.~~ ←

Disk File Command Format

DSK:filename.ext [proj,prog]

[proj,prog]

Project-programmer number assigned to the disk area to be searched for the source file(s) if other than the user's project-programmer number.

The standard protection[1] is assigned to any disk file specified as output.

Notes:

If object coding output is not desired (as in the case where a program is being scanned for source language errors), objprog-dev: is omitted.

If an assembly listing is not desired, list-dev: is omitted.

[1] Standard protection (055) designates that the owner is permitted to read or write, or change the protection of, the file while others are permitted only to read the file.

EXAMPLES

.R MACRO↲	
*DTA3:OBJPRG,LPT:←CDR:↲	Assemble one source program file from the card reader; write the object code on DTA3 and call the file OBJPRG; write the assembly listing on the line printer.
END OF PASS 1↲	The source program cards must be manually refed for pass 2.
THERE ARE 2 ERRORS↲ PROGRAM BREAK IS 002537↲ 5K CORE USED↲	Number of source errors. Size of object program. Core used by assembler.
*↑C↲	Return to the Monitor.
.KJOB↲	Kill the job, deassign all devices, and release core.
.	

.R MACRO↲	
*MTA3:,MTA2:←MTA1:,,↲ THERE ARE NO ERRORS↲ PROGRAM BREAK IS 003552↲ 6K CORE USED↲	Assemble the next three source files located at the present position of MTA1; write the object program on MTA3; write the listing on MTA2 for later printing.
*,LPT:←DTA1:FILE1,FILE2,FILE5↲ THERE ARE NO ERRORS↲ PROGRAM BREAK IS 001027↲ 6K CORE USED↲	Assemble the source files named FILE1, FILE2, and FILE5 from DTA1; produce no object coding; write the listing on the line printer.
*,←DSK:FILE1.MAC [14,12]↲ THERE ARE NO ERRORS↲ PROGRAM BREAK IS 000544↲ 5K CORE USED↲	? user files 14/12 Scan the source program called FILE1.MAC, located in area 14,12 on the disk, for source language errors; produce no object coding or assembly listing; print all error diagnostics on the Teletype.
*↑C↲	Return to the Monitor.
.KJOB↲	Kill the job, deassign all devices, and release core.
.	
.R MACRO	
*MTA1:,TTY:← TTY:↲	Assemble a source file from the Teletype; write the object code program on MTA1 and print the assembly listing on the Teletype.
JMP R↲ R: AOS G↲ G: JFCL↲ END↲ } Enter the source statements	
↑Z↲	Terminate input.
END OF PASS 1↲	Reenter Teletype input.
JMP R↲	Reenter the first statement.

```
.MAIN     MACROX.H9         10:14    20-DEC-67    PAGE 1↲        Page heading.

O         000000  000000  000001'         JMP     R↲             First assembled.

R:        AOS               G↲                                   Reenter second.

          000001  350000  000002'  R:     AOS     G↲             Second assembled.

G:        JFCL↲                                                  Reenter third.

          000002  255000  000000   G:     JFCL↲                  Third assembled.

          END↲                                                   Reenter fourth.

                                          END↲                   Fourth assembled.

THERE IS 1 ERROR↲

PROGRAM BREAK IS 000003↲

.MAIN     MACROX.H9         10:14    20-DEC-67    PAGE 2↲        Typeout of symbol table.
          SYMBOL TABLE↲

G         000002'↲
R         000001'↲

5K CORE USED↲

*↑C↲                                      Return to the Monitor.

.KJOB↲                                    Kill the job, deassign all devices and release core.

.
```

SWITCHES

Switches are used to specify such options as:

1. Magnetic tape control,
2. Macro call expansion,
3. Listing suppression,
4. Pushdown list expansion, and
5. Cross-reference file output.

All switches are preceded by a slash (/) (or enclosed in parentheses) and usually occur prior to the left arrow.

Table MACRO-1 Macro-10 Switch Options

Switch	Meaning
A[1]	Advance magnetic tape reel by one file.
B[1]	Backspace magnetic tape reel by one file.
C[1]	Produce listing file in a format acceptable as input to CREF.[2]
E	List macro expansions (same function as LALL pseudo-op).
L	Reinstate listing (used after list suppression by XLIST pseudo-op or S switch).
N	Suppress error printouts on the Teletype.
P	Increase the size of the pushdown list. This switch may appear as many times as desired (pushdown list is initially set to a size of 80_{10} locations; each /P increases its size by 80_{10}).
Q	Suppress Q (questionable) error indications on the listing; Q messages indicate assumptions made during pass 1.
S	Suppress listing (same function as XLIST pseudo-op).
T[1]	Skip to the logical end of the magnetic tape.
W[1]	Rewind the magnetic tape.
X	Suppress all macro expansions (same function as XALL psuedo-op).
Z[1]	Zero the DECtape directory.
NOTES:	1. Must immediately follow the device or file to which it refers. 2. Unless the file is named, CREF.TMP is assigned as the filename; if no extension is given, .TMP is assigned; if no list-dev: is specified, DSK: is assumed.

EXAMPLES

```
.R MACRO↲
*MTA1:,DTA3:/C ←— PTR:↲
```

Assemble one source file from the paper tape reader; write the object code on MTA1; write the assembly listing on DTA3 in cross-reference format and call the file CREF.TMP.

```
END OF PASS 1↲
```

The paper tape must be refed by the operator for pass 2.

```
THERE ARE 3 ERRORS↲
PROGRAM BREAK IS 000401↲
5K CORE USED↲
```

End-of-assembly messages.

```
*DTA2:ASSEMB.ONE/Z,LPT: ←— MTA4:/W,↲

THERE ARE NO ERRORS↲
PROGRAM BREAK IS 005231↲
6K CORE USED↲
```

Rewind MTA4 and assemble the first two source files on it; write the object code on DTA2, after zeroing the directory, and call the file ASSEMB.ONE; write the assembly listing on the line printer.

```
*MTA1:/W,LPT: ←— MTA3:/W,(AA),(BB)↲

THERE IS 1 ERROR↲
PROGRAM BREAK IS 000655↲
5K CORE USED↲
```

Rewind MTA1 and MTA3 and assemble files 1, 4, and 3 (in that order) from MTA3. Print the assembly listing on the line printer. Write the object code on MTA1.

```
*↑C↲
```

Return to the Monitor .

```
.KJOB↲
.
```

Kill the job, deassign all devices, and release core.

DIAGNOSTIC MESSAGES

Table MACRO-2 Macro-10 Diagnostic Messages

Message	Meaning
?CANNOT ENTER FILE filename.ext	DTA or DSK directory is full; file cannot be entered.
?CANNOT FIND filename.ext	The file cannot be found on the device specified.
?COMMAND ERROR	The last command string is in error.
?DATA ERROR ON DEVICE dev:	Output error has occurred on the device.
BELL END OF PASS1	This message is issued prior to pass 2 whenever the input source file is on a medium which must be manually re-entered by the operator (PTR:, CDR:, TTY:). When this message appears, the operator must refeed the tape or cards or retype the entries.
?IMPROPER INPUT DATA	The input data is not in the proper format.
?INPUT ERROR ON DEVICE dev:	Data cannot be read.
?INSUFFICIENT CORE	An insufficient amount of core is available for assembly.
nK CORE USED	Amount of core used for this assembly.
BELL LOAD THE NEXT FILE	Manual loading is required for the next card or paper tape file.
?NO END STATEMENT ENCOUNTERED ON INPUT FILE	The END statement is missing at the end of the source program file.
?dev: NOT AVAILABLE	The device is assigned to another user or does not exist.
?PDP OVERFLOW, TRY/P	A pushdown list overflow has occurred.
PROGRAM BREAK IS nnnnn	The highest relative location occupied by the object program produced.
?THERE ARE n ERRORS THERE ARE NO ERRORS ?THERE IS 1 ERROR	Number of source language errors found.

INSTRUCTION OPERATIONS

The extensive PDP-10 instruction repertoire is structured around a design that implements all possible variations at the object code level. Thus, instructions which add or subtract may store the results either in an accumulator or in memory or both. Boolean operations admit all 16 combinations of two variables, while arithmetic compare and modify codes permit branching on the eight possible results. This logically complete instruction set contributes significantly toward reduction in program length and running time.

Instruction List

Instructions are organized around basic operations. A mode is appended to the instruction mnemonic to specify the result destination, test conditions, and other options on the basic operation.

Arithmetic and Logical

Fixed point arithmetic is single precision and negative numbers are in 2s complement form. Double precision is facilitated by using the CRY0 and CRY1 flags which record the hardware-generated carries from bits 0 and 1 of the arithmetic unit. Overflow is recorded in the AROV flag.

ADD	(270)	add
SUB	(274)	subtract
MUL	(224)	multiply
IMUL	(220)	integer multiply
DIV	(234)	divide
IDIV	(230)	integer divide

MODES	
	combine (AC) and (E) results → (AC)
I	Immediate combine (AC) and E results → (AC)
M	Memory combine (AC) and (E) results → (E)
B	Both combine (AC) and (E) results → (AC) and (E)

Instruction Codes

ADD	(270)	(AC) + (E) → (AC)
ADDI	(271)	(AC) + E → (AC)
ADDM	(272)	(AC) + (E) → (E)
ADDB	(273)	(AC) + (E) → (AC), (E)
SUB	(274)	(AC) - (E) → (AC)
SUBI	(275)	(AC) - E → (AC)
SUBM	(276)	(AC) - (E) → (E)
SUBB	(277)	(AC) - (E) → (AC), (E)
		High order part of the product is lost for all IMUL instructions. The overflow flag is set if the high order part is non-zero.
IMUL	(220)	(AC) x (E) → (AC)
IMULI	(221)	(AC) x E → (AC)
IMULM	(222)	(AC) x (E) → (E)
IMULB	(223)	(AC) x (E) → (AC), (E)
MUL	(224)	(AC) x (E) → (AC), (AC + 1)
MULI	(225)	(AC) x E → (AC), (AC + 1)
MULM	(226)	(AC) x (E) → (E) low order part discarded
MULB	(227)	(AC) x (E) → (AC), (E) low order part → (AC + 1)
		overflow occurs during IDIV only if the divisor is zero
IDIV	(230)	(AC) ÷ (E) → (AC) remainder → (AC + 1)
IDIVI	(231)	(AC) ÷ E → (AC) remainder → (AC + 1)
IDIVM	(232)	(AC) ÷ (E) → (E) remainder is discarded
IDIVB	(233)	(AC) ÷ (E) → (AC), (E) remainder → (AC + 1)
DIV	(234)	$\left[(AC) + 2^{-35}_{\times} (AC + 1)\right] \div (E) \rightarrow (AC)$ remainder → (AC + 1)

DIVI	(235)	$\left[(AC) + 2^{-35}_{x} (AC + 1)\right]$ $\div E \rightarrow (AC)$ remainder (AC + 1
DIVM	(236)	$\left[(AC) + 2^{-35} (AC + 1)\right]$ $\div (E) \rightarrow (E)$ remainder is discarded
DIVB	(237)	$\left[(AC + 2^{-35} (AC + 1)\right]$ $\div (E) \rightarrow (AC), (E)$ remainder → (AC + 1)

Floating Point Arithmetic - Exponents are excess 200_8 except that a 0 fraction will result in a 0 exponent. Results are normalized except for UFA. Arithmetic flag indications are:

AROV	FOV	FXU	DCK	
1	1	0	0	floating exponent overflow
1	1	0	1	floating divide check
1	1	1	0	floating exponent underflow

Add, Subtract, Multiply

FAD	(140)	floating add
FSB	(150)	floating subtract
FMP	(160)	floating multiply

MODES	
	result → (AC)
L	Long double precision result → (AC), (AC + 1) Sign and exponent of (AC + 1) are positive, 27_{10} less than (AC) exponent, and not normalized
M	Memory result → (E)
B	Both result → (AC), (E)

Instruction Codes

FAD	(140)	(AC) + (E) → (AC)
FADL	(141)	(AC) + (E) → (AC), (AC + 1)
FADM	(142)	(AC) + (E) → (E)
FADB	(143)	(AC) + (E) → (AC), (E)
FSB	(150)	(AC) - (E) → (AC)
FSBL	(151)	(AC) - (E) → (AC), (AC + 1)
FSBM	(152)	(AC) - (E) → (E)
FSBB	(153)	(AC) - (E) → (AC), (E)
FMP	(160)	(AC) x (E) → (AC)
FMPL	(161)	(AC) x (E) → (AC), (AC + 1)
FMPM	(162)	(AC) x (E) → (E)
FMPB	(163)	(AC) x (E) → (AC), (E)

Divide FDV (170) floating divide

MODES	
	quotient → (AC)
L	Long quotient → (AC) remainder → (AC + 1) sign of the remainder is the sign of the dividend Exponent of the remainder is 27_{10} less than that of the dividend.
M	Memory quotient → (E)
B	Both quotient → (AC), (E)

Instruction Codes

FDV	(170)	(AC) ÷ (E) → (AC)
FDVL	(171)	(AC) ÷ (E) → (AC)
		remainder → (AC + 1)
FDVM	(172)	(AC) ÷ (E) → (E)
FDVB	(173)	(AC) ÷ (E) → (AC), (E)

Rounded

FADR	(144)	floating add and round
FSBR	(154)	floating subtract and round
FMPR	(164)	floating multiply and round
FDVR	(174)	floating divide and round

MODES	
	rounded result → (AC)
I	Immediate the second operand is E, left justified with zeros in the right half of the word rounded result → (AC)
M	Memory rounded result → (E)
B	Both rounded result → (AC), (E)

Instruction Codes

FADR	(144)	(AC) + (E) → (AC)
FADRI	(145)	(AC) + E → (AC)
FADRM	(146)	(AC) + (E) → (E)
FADRB	(147)	(AC) + (E) → (AC), (E)
FSBR	(154)	(AC) - (E) → (AC)
FSBRI	(155)	(AC) - E → (AC)
FSBRM	(156)	(AC) - (E) → (E)
FSBRB	(157)	(AC) - (E) → (AC), (E)
FMPR	(164)	(AC) x (E) → (AC)
FMPRI	(165)	(AC) x E → (AC)
FMPRM	(166)	(AC) x (E) → (E)
FMPRB	(167)	(AC) x (E) → (AC), (E)
FDVR	(174)	(AC) ÷ (E) → (AC)
FDVRI	(175)	(AC) ÷ E → (AC)
FDVRM	(176)	(AC) ÷ (E) → (E)
FDVRB	(177)	(AC) ÷ (E) → (AC), (E)

Miscellaneous Floating Point

UFA	(130)	unnormalized floating add. Same as FAD except that the result is not normalized. (AC) + (E) → (AC + 1)
DFN	(131)	double precision floating negate $- \left[(AC), (E)\right] \rightarrow (AC), (E)$ the exponents are not changed
FSC	(132)	floating scale $2^E x\ (AC) \rightarrow (AC)$

Boolean

All 16 boolean operations on two variables are provided. The contents of the accumulator and the operand are combined on a bit for bit basis. The table below gives the results for all operations.

Operand Bit	0	0	1	1
Accumulator Bit	0	1	0	1
SETZ	0	0	0	0
AND	0	0	0	1
ANDCA	0	0	1	0
SETM	0	0	1	1
ANDCM	0	1	0	0
SETA	0	1	0	1
XOR	0	1	1	0
IOR	0	1	1	1
ANDCB	1	0	0	0
EQV	1	0	0	1
SETCA	1	0	1	0
ORCA	1	0	1	1
SETCM	1	1	0	0
ORCM	1	1	0	1
ORCB	1	1	1	0
SETO	1	1	1	1

MODES	
	combine (AC) and (E) results → (AC)
I	Immediate combine (AC) and E zeros are assumed in the left half of the operand result → (AC)
M	combine (AC) and (E) results → (E)
B	combine (AC) and (E) results → (AC) and (E)

Instruction Codes

SETZ	(400)	0 → (AC)
SETZI	(401)	0 → (AC)
SETZM	(402)	0 → (E)
SETZB	(403)	0 → (AC), (E)
SETM	(414)	(E) → (AC)
SETMI	(415)	E → (AC)
SETMM	(416)	(E) → (E)
SETMB	(417)	(E) → (AC), (E)
SETA	(424)	(AC) → (AC)
SETAI	(425)	(AC) → (AC)
SETAM	(426)	(AC) → (E)
SETAB	(427)	(AC) → (AC), (E)
SETCA	(450)	(AC)' → (AC)
SETCAI	(451)	(AC)' → (AC)
SETCAM	(452)	(AC)' → (E)
SETCAB	(453)	(AC)' → (AC), (E)
SETCM	(460)	(E)' → (AC)
SETCMI	(461)	E' → (AC)
SETCMM	(462)	(E)' → (E)
SETCMB	(463)	(E)' → (AC), (E)

SETO	(474)	$777777777777_8 \rightarrow (AC)$
SETOI	(475)	$777777777777_8 \rightarrow (AC)$
SETOM	(476)	$777777777777_8 \rightarrow (E)$
SETOB	(477)	$777777777777_8 \rightarrow (AC), (E)$
AND	(404)	$(AC) \wedge (E) \rightarrow (AC)$
ANDI	(405)	$(AC) \wedge E \rightarrow (AC)$
ANDM	(406)	$(AC) \wedge (E) \rightarrow (E)$
ANDB	(407)	$(AC) \wedge (E) \rightarrow (AC), (E)$
ANDCA	(410)	$(AC)' \wedge (E) \rightarrow (AC)$
ANDCAI	(411)	$(AC)' \wedge E \rightarrow (AC)$
ANDCAM	(412)	$(AC)' \wedge (E) \rightarrow (E)$
ANDCAB	(413)	$(AC)' \wedge (E) \rightarrow (AC), (E)$
ANDCM	(420)	$(AC) \wedge (E)' \rightarrow (AC)$
ANDCMI	(421)	$(AC) \wedge E' \rightarrow (AC)$
ANDCMM	(422)	$(AC) \wedge (E)' \rightarrow (E)$
ANDCMB	(423)	$(AC) \wedge (E)' \rightarrow (AC), (E)$
ANDCB	(440)	$(AC)' \wedge (E)' \rightarrow (AC)$
ANDCBI	(441)	$(AC)' \wedge E' \rightarrow (AC)$
ANDCBM	(442)	$(AC)' \wedge (E)' \rightarrow (E)$
ANDCBB	(443)	$(AC)' \wedge (E)' \rightarrow (AC), (E)$
IOR	(434)	$(AC) \vee (E) \rightarrow (AC)$
IORI	(435)	$(AC) \vee E \rightarrow (AC)$
IORM	(436)	$(AC) \vee (E) \rightarrow (E)$
IORB	(437)	$(AC) \vee (E) \rightarrow (AC), (E)$
IORCA	454	$(AC)' \vee (E) \rightarrow (AC)$
IORCAI	5	$(AC)' \vee E \rightarrow (AC)$
IORCAM	6	$(AC)' \vee (E) \rightarrow (E)$
IORCAB	7	$(AC)' \vee (E) \rightarrow (AC), (E)$
IORCM	464	$(AC) \vee (E)' \rightarrow (AC)$
IORCMI	5	$(AC) \vee (E)' \rightarrow (AC)$
IORCMM	6	$(AC) \vee (E)' \rightarrow (E)$
IORCMB	7	$(AC) \vee (E)' \rightarrow (AC), (E)$

IORCB	470	$(AC)' \vee (E)' \rightarrow (AC)$
IORCBI	1	$(AC)' \vee E' \rightarrow (AC)$
IORCBM	2	$(AC)' \vee (E)' \rightarrow (E)$
IORCBB	3	$(AC)' \vee (E)' \rightarrow (AC), (E)$
XOR	(430)	$(AC) \oplus (E) \rightarrow (AC)$
XORI	(431)	$(AC) \oplus E \rightarrow (AC)$
XORM	(432)	$(AC) \oplus (E) \rightarrow (E)$
XORB	(433)	$(AC) \oplus (E) \rightarrow (AC), (E)$
EQV	(444)	$[(AC) \oplus (E)]' \rightarrow (AC)$
EQVI	(445)	$[(AC) \oplus E]' \rightarrow (AC)$
EQVM	(446)	$[(AC) \oplus (E)]' \rightarrow (E)$
EQVB	(447)	$[(AC) \oplus (E)]' \rightarrow (AC), (E)$

Shifting

Arithmetic shifts perform 2's complement multiplication. Overflow may occur when shifting left. Logical shift inserts 0's at one end and bits at the other end are lost. Rotate forms a ring so that bits are recycled. The number of shifts is designated by E (considered as a 2's complement number). The direction is to the left if E is positive; to the right if E is negative.

ASH	(240)	arithmetic shift
ROT	(241)	rotate
LSH	(242)	logical shift

MODES	
	shift (AC)
C	shift Combined accumulators, (AC) and (AC+1). (AC) is in the high order position. During arithmetic shifts the sign of AC+1 is set to the sign of AC, if the shift is non-zero.

Instruction Codes

ASH	(240)	arithmetic shift
ASHC	(244)	arithmetic shift combined accumulators
ROT	(241)	rotate

ROTC	(245)	rotate combined accumulators
LSH	(242)	logical shift
LSHC	(246)	logical shift combined accumulators

DATA TRANSMISSION

Full Word

MOVE	(200)	move full word
MOVS	(204)	move swapped word (left and right halves exchanged)
MOVN	(210)	move full word negative (2's complement)
MOVM	(214)	move full word magnitude

MODES	
	source: (E) destination: (AC)
I	Immediate source: (O,E) destination: (AC)
M	Memory source: (AC) destination: (E)
S	Self source: (E) destination: (E) and also (AC) if the AC address is non-zero

Instruction Codes

MOVE	(200)	(E) → (AC)
MOVEI	(201)	$E \rightarrow (AC)_R$ $0 \rightarrow (AC)_L$
MOVEM	(202)	(AC) → (E)

MOVES	(203)	$(E) \rightarrow (E)$ $(E) \rightarrow (AC)$ if $AC \neq 0$
MOVS	(204)	$(E)_L \rightarrow (AC)_R$ $(E)_R \rightarrow (AC)_L$
MOVSI	(205)	$0 \rightarrow (AC)_R$ $E \rightarrow (AC)_L$
MOVSM	(206)	$(AC)_L \rightarrow (E)_R$ $(AC)_R \rightarrow (E)_L$
MOVSS	(207)	$(E)_L \rightarrow (E)_R$ $(E)_R \rightarrow (E)_L$ $(E)_L \rightarrow (AC)_R$ and $(E)_R \rightarrow (AC)_L$ if $AC \neq 0$
MOVN	(210)	$-(E) \rightarrow (AC)$
MOVNI	(211)	$-E \rightarrow (AC)$
MOVNM	(212)	$-(AC) \rightarrow (E)$
MOVNS	(213)	$-(E) \rightarrow (E)$ $-(E) \rightarrow (AC)$ if $AC \neq 0$
MOVM	(214)	$\|(E)\| \rightarrow (AC)$
MOVMI	(215)	$E \rightarrow (AC)$
MOVMM	(216)	$\|(AC)\| \rightarrow (E)$
MOVMS	(217)	$\|(E)\| \rightarrow (E)$ $\|(E)\| \rightarrow (AC)$ if $AC \neq 0$

Miscellaneous Full Word

EXCH	(250)	$(E) \longleftrightarrow (AC)$
BLT	(251)	Block Transfer $E-(AC)_R+1$ words from locations starting at $(AC)_L$ to locations starting at $(AC)_R$ Note: if a priority interrupt occurs during this instruction then (AC) are indeterminate.

Half-Word

Half word transmission instruction codes are composed of a modifier and the mode.

HLL	(500)	half word left to left
HRR	(540)	half word right to right
HRL	(504)	half word right to left
HLR	(544)	half word left to right

MODIFIER	
	none
Z	set other half of destination word to Zeros
O	set other half of destination word to Ones
E	Extend sign of destination half-word to other half

MODES	
	source: (E) destination: (AC)
I	Immediate source: (0, E) destination: (AC)
M	Memory source: (AC) destination: (E)
S	Self source: (E) destination (E) and also (AC) if the AC field is non-zero

Instruction Codes

HLL	(500)	$(E)_L \rightarrow (AC)_L$
HLLI	(501)	$0 \rightarrow (AC)_L$
HLLM	(502)	$(AC)_L \rightarrow (E)_L$
HLLS	(503)	$(E)_L \rightarrow (E)_L$
HRR	(540)	$(E)_R \rightarrow (AC)_R$
HRRI	(541)	$E \rightarrow (AC)_R$
HRRM	(542)	$(AC)_R \rightarrow (E)_R$
HRRS	(543)	$(E)_R \rightarrow (E)_R$
HRL	(504)	$(E)_R \rightarrow (AC)_L$
HRLI	(505)	$E \rightarrow (AC)_L$
HRLM	(506)	$(AC)_R \rightarrow (E)_L$
HRLS	(507)	$(E)_R \rightarrow (E)_L$
HLR	(544)	$(E)_L \rightarrow (AC)_R$
HLRI	(545)	$0 \rightarrow (AC)_R$
HLRM	(546)	$(AC)_L \rightarrow (E)_R$
HLRS	(547)	$(E)_L \rightarrow (E)_R$
HLLZ	(510)	$(E)_L \rightarrow (AC)_L$ $0 \rightarrow (AC)_R$
HLLZI	(511)	$0 \rightarrow (AC)_L$ $0 \rightarrow (AC)_R$
HLLZM	(512)	$(AC)_L \rightarrow (E)_L$ $0 \rightarrow (E)_R$
HLLZS	(513)	$(E)_L \rightarrow (E)_L$ $0 \rightarrow (E)_R$

HRRZ	(550)	$(E)_R \rightarrow (AC)_R$
		$0 \rightarrow (AC)_L$
HRRZI	(551)	$E \rightarrow (AC)_R$
		$0 \rightarrow (AC)_L$
HRRZM	(552)	$(AC)_R \rightarrow (E)_R$
		$0 \rightarrow (E)_L$
HRRZS	(553)	$(E)_R \rightarrow (E)_R$
		$0 \rightarrow (E)_L$
HRLZ	(514)	$(E)_R \rightarrow (AC)_L$
		$0 \rightarrow (AC)_R$
HRLZI	(515)	$E \rightarrow (AC)_L$
		$0 \rightarrow (AC)_R$
HRLZM	(516)	$(AC)_R \rightarrow (E)_L$
		$0 \rightarrow (E)_R$
HRLZS	(517)	$(E)_R \rightarrow (E)_L$
		$0 \rightarrow (E)_R$
HLRZ	(554)	$(E)_L \rightarrow (AC)_R$
		$0 \rightarrow (AC)_L$
HLRZI	(555)	$0 \rightarrow (AC)_R$
		$0 \rightarrow (AC)_L$
HLRZM	(556)	$(AC)_L \rightarrow (E)_R$
		$0 \rightarrow (E)_L$
HLRZS	(557)	$(E)_L \rightarrow (E)_R$
		$0 \rightarrow (E)_L$
HLLO	(520)	$(E)_L \rightarrow (AC)_L$
		$777777_8 \rightarrow (AC)_R$

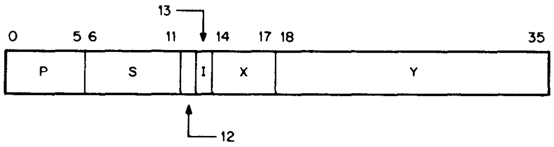
HLLOI	(521)	$0 \rightarrow (AC)_L$ $777777_8 \rightarrow (AC)_R$
HLLOM	(522)	$(AC)_L \rightarrow (E)_L$ $777777_8 \rightarrow (E)_R$
HLLOS	(523)	$(E)_L \rightarrow (E)_L$ $777777_8 \rightarrow (E)_R$
HRRO	(560)	$(E)_R \rightarrow (AC)_R$ $777777_8 \rightarrow (AC)_L$
HRROI	(561)	$E \rightarrow (AC)_R$ $777777_8 \rightarrow (AC)_L$
HRROM	(562)	$(AC)_R \rightarrow (E)_R$ $777777_8 \rightarrow (E)_L$
HRROS	(563)	$(E)_R \rightarrow (E)_R$ $777777_8 \rightarrow (E)_L$
HRLO	(524)	$(E)_R \rightarrow (AC)_L$ $777777_8 \rightarrow (AC)_R$
HRLOI	(525)	$E \rightarrow (AC)_L$ $777777_8 \rightarrow (AC)_R$
HRLOM	(526)	$(AC)_R \rightarrow (E)_L$ $777777_8 \rightarrow (E)_R$
HRLOS	(527)	$(E)_R \rightarrow (E)_L$ $777777_8 \rightarrow (E)_R$
HLRO	(564)	$(E)_L \rightarrow (AC)_R$ $777777_8 \rightarrow (AC)_L$
HLROI	(565)	$E \rightarrow (AC)_R$ $777777_8 \rightarrow (AC)_L$

HLROM	(566)	$(AC)_L \rightarrow (E)_R$
		$777777_8 \rightarrow (E)_L$
HLROS	(567)	$(E)_L \rightarrow (E)_R$
		$777777_8 \rightarrow (E)_L$
HLLE	(530)	$(E)_L \rightarrow (AC)_L$
		$(E)_0 \rightarrow (AC)_R$
HLLEI	(531)	$0 \rightarrow (AC)_L$
		$0 \rightarrow (AC)_R$
HLLEM	(532)	$(AC)_L \rightarrow (E)_L$
		$(AC)_0 \rightarrow (E)_R$
HLLES	(533)	$(E)_L \rightarrow (E)_L$
		$(E)_0 \rightarrow (E)_R$
HRRE	(570)	$(E)_R \rightarrow (AC)_R$
		$(E)_{18} \rightarrow (AC)_L$
HRREI	(571)	$E \rightarrow (AC)_R$
		$E_{18} \rightarrow (AC)_L$
HRREM	(572)	$(AC)_R \rightarrow (E)_R$
		$(AC)_{18} \rightarrow (E)_L$
HRRES	(573)	$(E)_R \rightarrow (E)_R$
		$(E)_{18} \rightarrow (E)_L$
HRLE	(534)	$(E)_R \rightarrow (AC)_L$
		$(E)_{18} \rightarrow (AC)_R$
HRLEI	(535)	$E \rightarrow (AC)_L$
		$E_{18} \rightarrow (AC)_R$
HRLEM	(536)	$(AC)_R \rightarrow (E)_L$
		$(AC)_{18} \rightarrow (E)_R$

HRLES	(537)	$(E)_R \rightarrow (E)_L$ $(E)_{18} \rightarrow (E)_R$
HLRE	(574)	$(E)_L \rightarrow (AC)_R$ $(E)_0 \rightarrow (AC)_L$
HLREI	(575)	$0 \rightarrow (AC)_R$ $0 \rightarrow (AC)_L$
HLREM	(576)	$(AC)_L \rightarrow (E)_R$ $(AC)_0 \rightarrow (E)_L$
HLRES	(577)	$(E)_L \rightarrow (E)_R$ $(E)_0 \rightarrow (E)_L$

Byte Manipulation

Byte manipulation instructions permit easy access to any number of contiguous bits anywhere in a single word. The size (S) and location (P) of the byte are specified by a pointer word designated by the effective address:



Byte Pointer Word

The I, X, and Y-fields of the pointer word are used in the usual manner to computer an effective address which is the location of the memory word containing the byte. The P-field specifies the number of bits between the right end of the word and the farthest right bit of the byte. The S-field specifies the size of the byte, up to 36 bits. To store and retrieve successive bytes, the pointer word incrementing instructions automatically subtract the size of the byte from P, moving the position to the right by one byte. If there is insufficient room in the memory word for the next byte (P-S<0), the Y-field of the pointer word is incremented by 1 and P is reset to 36-S.

This instruction may be interrupted between incrementing and the byte operation. The byte increment suppression (BIS) flag records the incrementing operation, and prevents its occurrence when returning from an interrupt. Note that this flag is under program control, and is restored by the JRST instruction.

Instruction Codes

LDB	(135)	LoaD Byte The byte is loaded into the accumulator right justified. Unused bits in the accumulator are cleared.
DPB	(137)	DePosit Byte The byte in the accumulator is deposited in the memory word. Unused bits in the memory word are not affected, and the unused bits in the accumulator need not be zero.

IBP	(133)	Increment Byte Pointer
		if P-S > 0 then P-S → P
		if P-S $\overline{<}$ 0 then Y+1 → Y
		and 36-S → P

ILDB	(134)	Increment and LoaD Byte Same as IBP followed by LDB
IDPB	(136)	Increment and DePosit Byte Same as IBP followed by DPB

EXECUTIVE

Arithmetic Compare and Modify

SKIP tests the (E) against 0 and skips 1 instruction location if the conditions specified by the mode are satisfied. AOS and SOS add 1 or subtract 1 from (E) before testing. JUMP tests the designated accumulator. AOJ and SOJ add 1 or subtract 1 from (AC) and then jump if the test is satisfied.

SKIP	(330)	SKIP if conditions specified by the mode are satisfied by (E). (E) → (AC) if AC≠0.
JUMP	(320)	JUMP if conditions specified by the mode are satisfied by (AC).
AOS	(350)	Add One to memory and Skip. (E)+1 → (AC) if AC≠0.
SOS	(370)	Subtract One from memory and Skip. (E)-1 → (AC) if AC≠0.
AOJ	(340)	Add One to (AC) and Jump.
SOJ	(360)	Subtract One from (AC) and Jump.

CAM	(310)	Compare Accumulator to Memory and skip.
CAI	(300)	Compare Accumulator Immediate and skip.

MODES	
	never
L	Less than
E	Equal to
LE	Less than or Equal
A	Always
GE	Greater than or Equal
N	Not equal
G	Greater than

Instruction Codes

SKIP	(330)	$(E) \rightarrow (AC)$ if $AC \neq 0$
SKIPL	(331)	$(E) \rightarrow (AC)$ if $AC \neq 0$ skip if $(E) < 0$
SKIPE	(332)	$(E) \rightarrow (AC)$ if $AC \neq 0$ skip if $(E)=0$
SKIPLE	(333)	$(E) \rightarrow (AC)$ if $AC \neq 0$ skip if $(E) \leq 0$
SKIPA	(334)	$(E) \rightarrow (AC)$ if $AC \neq 0$ always skip
SKIPGE	(335)	$(E) \rightarrow (AC)$ if $AC \neq 0$ skip if $(E) \geq 0$
SKIPN	(336)	$(E) \rightarrow (AC)$ if $AC \neq 0$ skip if $(E) \neq 0$
SKIPG	(337)	$(E) \rightarrow (AC)$ if $AC \neq 0$ skip if $(E) > 0$
JUMP	(320)	No action
JUMPL	(321)	$E \rightarrow PC$ if $(AC) < 0$
JUMPE	(322)	$E \rightarrow PC$ if $(AC) = 0$
JUMPLE	(323)	$E \rightarrow PC$ if $(AC) \leq 0$

JUMPA	(324)	E → PC
JUMPGE	(325)	E → PC if (AC) ≥ 0
JUMPN	(326)	E → PC if (AC) ≠ 0
JUMPG	(327)	E → PC if (AC) > 0
AOS	(350)	(E) + 1 → (E) result → (AC) if AC ≠ 0
AOSL	(351)	(E) + 1 → (E) result → (AC) if AC ≠ 0 skip if result < 0
AOSE	(352)	(E) + 1 → (E) result → (AC) if AC ≠ 0 skip if result = 0
AOSLE	(353)	(E) + 1 → (E) result → (AC) if AC ≠ 0 skip if result ≤ 0
AOSA	(354)	(E) + 1 → (E) result → (AC) if AC ≠ 0 always skip
AOSGE	(355)	(E) + 1 → (E) result → (AC) if AC ≠ 0 skip if result ≥ 0
AOSN	(356)	(E) + 1 → (E) result → (AC) if AC ≠ 0 skip if result ≠ 0
AOSG	(357)	(E) + 1 → (E) result → (AC) if AC ≠ 0 skip if result > 0
SOS	(370)	(E) - 1 → (E) result → (AC) if AC ≠ 0
SOSL	(371)	(E) - 1 → (E) result → (AC) if AC ≠ 0 skip if result < 0
SOSE	(372)	(E) - 1 → (E) result → (AC) if AC ≠ 0 skip if result = 0

SOSLE	(373)	(E) - 1 → (E) result → (AC) if AC ≠ 0 skip if result ≤ 0
SOSA	(374)	(E) - 1 → (E) result → (AC) if AC ≠ 0 always skip
SOSGE	(375)	(E) - 1 → (E) result → (AC) if AC ≠ 0 skip if result ≥ 0
SOSN	(376)	(E) - 1 → (E) result → (AC) if AC ≠ 0 skip if result ≠ 0
SOSG	(377)	(E) - 1 → (E) result → (AC) if AC ≠ 0 skip if result > 0
AOJ	(340)	(AC) + 1 → (AC)
AOJL	(341)	(AC) + 1 → (AC) E → PC if result < 0
AOJE	(342)	(AC) + 1 → (AC) E → PC if result = 0
AOJLE	(343)	(AC) + 1 → (AC) E → PC if result ≤ 0
AOJA	(344)	(AC) + 1 → (AC) E → PC
AOJGE	(345)	(AC) + 1 → (AC) E → PC if result ≥ 0
AOJN	(346)	(AC) + 1 → (AC) E → PC if result ≠ 0
AOJG	(347)	(AC) + 1 → (AC) E → PC if result > 0
SOJ	(360)	(AC) - 1 → (AC)
SOJL	(361)	(AC) - 1 → (AC) E → PC if result < 0
SOJE	(362)	(AC) - 1 → (AC) E → PC if result = 0

SOJLE	(363)	(AC) - 1 → (AC) E → PC if result ≤ 0
SOJA	(364)	(AC) - 1 → (AC) E → PC
SOJGE	(365)	(AC) - 1 → (AC) E → PC if result ≥ 0
SOJN	(366)	(AC) - 1 → (AC) E → PC if result $\neq 0$
SOJG	(367)	(AC) - 1 → (AC) E → PC if result > 0

CAM	(310)	No action
CAML	(311)	skip if $(AC) < (E)$
CAME	(312)	skip if $(AC) = (E)$
CAMLE	(313)	skip if $(AC) \leq (E)$
CAMA	(314)	always skip
CAMGE	(315)	skip if $(AC) \geq (E)$
CAMN	(316)	skip if $(AC) \neq (E)$
CAMG	(317)	skip if $(AC) > (E)$

CAI	(300)	No action
CAIL	(301)	skip if $(AC) < E$
CAIE	(302)	skip if $(AC) = E$
CAILE	(303)	skip if $(AC) \leq E$
CAIA	(304)	always skip
CAIGE	(305)	skip if $(AC) \geq \underline{E}$
CAIN	(306)	skip if $(AC) \neq E$
CAIG	(307)	skip if $(AC) > E$

The following two instructions add 1 to both halves of (AC) and then test and jump. There are no mode codes.

AOBJP	(252)	Add One to Both and Jump if Positive $(AC) + 1000001_8 \rightarrow (AC)$ E → PC if result ≥ 0
AOBJN	(253)	Add One to Both and Jump if Negative $(AC) + 1000001_8 \rightarrow (AC)$ E → PC if result < 0

Logical Compare and Modify

Accumulator bits corresponding to 1-bits in a mask can be modified or tested to determine a skip. The source of the mask is designated by the second letter of the instruction code mnemonic:

D	Direct memory word specified by (E)
S	(E) with right and left halves Swapped
L	A word with E in the Left half, zeros in the right half.
R	A word with E in the Right half, zeros in the left half.

The masked bits of the accumulator may be modified as indicated by the third letter of the instruction mnemonic:

N	No modification
Z	set masked bits to Zero
O	set masked bits to One
C	Complement masked bits

The skip condition is specified by the mode (fourth letter of mnemonic).

MODES	
	never skip
E	all masked bits are Equal to zero
A	Always skip
N	Not all masked bits equal to are zero

Instruction Codes

TDN	(610)	No action
TDNE	(612)	skip if $(AC) \wedge (E) = 0$
TDNA	(614)	always skip
TDNN	(616)	skip if $(AC) \wedge (E) \neq 0$
TSN	(611)	No action
TSNE	(613)	skip if $(AC) \wedge (E)_S = 0$
TSNA	(615)	always skip
TSNN	(617)	skip if $(AC) \wedge (E)_S \neq 0$
TLN	(601)	No action
TLNE	(603)	skip if $(AC)_L \wedge E = 0$
TLNA	(605)	always skip

TLNN	(607)	skip if $(AC)_L \wedge E \neq 0$
TRN	(600)	No action
TRNE	(602)	skip if $(AC)_R \wedge E = 0$
TRNA	(604)	always skip
TRNN	(606)	skip if $(AC)_R \wedge E \neq 0$
TDZ	(630)	$(AC) \wedge (E)' \rightarrow (AC)$
TDZE	(632)	skip if $(AC) \wedge (E) = 0$ $(AC) \wedge (E)' \rightarrow (AC)$
TDZA	(634)	always skip $(AC) \wedge (E)' \rightarrow (AC)$
TDZN	(636)	skip if $(AC) \wedge (E) \neq 0$ $(AC) \wedge (E)' \rightarrow (AC)$
TSZ	(631)	$(AC) \wedge (E)'_S \rightarrow (AC)$
TSZE	(633)	skip if $(AC) \wedge (E)_S = 0$ $(AC) \wedge (E)'_S \rightarrow (AC)$
TSZA	(635)	always skip $(AC) \wedge (E)'_S \rightarrow (AC)$
TSZN	(637)	skip if $(AC) \wedge (E)_S \neq 0$ $(AC) \wedge (E)'_S \rightarrow (AC)$
TLZ	(621)	$(AC)_L \wedge E' \rightarrow (AC)_L$
TLZE	(623)	skip if $(AC)_L \wedge E = 0$ $(AC)_L \wedge E' \rightarrow (AC)_L$
TLZA	(625)	always skip $(AC)_L \wedge E' \rightarrow (AC)_L$
TLZN	(627)	skip if $(AC)_L \wedge E \neq 0$ $(AC)_L \wedge E' \rightarrow (AC)_L$
TRZ	(620)	$(AC)_R \wedge E' \rightarrow (AC)_R$
TRZE	(622)	skip if $(AC)_R \wedge E = 0$ $(AC)_R \wedge E' \rightarrow (AC)_R$
TRZA	(624)	always skip $(AC)_R \wedge E' \rightarrow (AC)_R$
TRZN	(626)	skip if $(AC)_R \wedge E \neq 0$ $(AC)_R \wedge E' \rightarrow (AC)_R$

TDO	(670)	$(AC) \vee (E) \rightarrow (AC)$
TDOE	(672)	skip if $(AC) \wedge (E) = 0$ $(AC) \vee (E) \rightarrow (AC)$
TDOA	(674)	always skip $(AC) \vee (E) \rightarrow (AC)$
TDON	(676)	skip if $(AC) \wedge (E) \neq 0$ $(AC) \vee (E) \rightarrow (AC)$
TSO	(671)	$(AC) \vee (E)_S \rightarrow (AC)$
TSOE	(673)	skip if $(AC) \wedge (E)_S = 0$ $(AC) \vee (E)_S \rightarrow (AC)$
TSOA	(675)	always skip $(AC) \vee (E)_S \rightarrow (AC)$
TSON	(677)	skip if $(AC) \wedge (E)_S \neq 0$ $(AC) \vee (E)_S \rightarrow (AC)$
TLO	(661)	$(AC)_L \vee E \rightarrow (AC)_L$
TLOE	(663)	skip if $(AC)_L \wedge E = 0$ $(AC)_L \vee E \rightarrow (AC)_L$
TLOA	(665)	always skip $(AC)_L \vee E \rightarrow (AC)_L$
TLON	(667)	skip if $(AC)_L \wedge E \neq 0$ $(AC)_L \vee E \rightarrow (AC)_L$
TRO	(660)	$(AC)_R \vee E \rightarrow (AC)_R$
TROE	(662)	skip if $(AC)_R \wedge E = 0$ $(AC)_R \vee E \rightarrow (AC)_R$
TROA	(664)	always skip $(AC)_R \vee E \rightarrow (AC)_R$
TRON	(666)	skip if $(AC)_R \wedge E \neq 0$ $(AC)_R \vee E \rightarrow (AC)_R$
TDC	(650)	$(AC) \oplus (E) \rightarrow (AC)$
TDCE	(652)	skip if $(AC) \wedge (E) = 0$ $(AC) \oplus (E) \rightarrow (AC)$

TDCA	(654)	always skip $(AC) \oplus (E) \rightarrow (AC)$
TDCN	(656)	skip if $(AC) \wedge (E) \neq 0$ $(AC) \oplus (E) \rightarrow (AC)$
TSC	(651)	$(AC) \oplus (E)_S \rightarrow (AC)$
TSCE	(653)	skip if $(AC) \wedge (E)_S = 0$ $(AC) \oplus (E)_S \rightarrow (AC)$
TSCA	(655)	always skip $(AC) \oplus (E)_S \rightarrow (AC)$
TSCN	(657)	skip if $(AC) \wedge (E)S \neq 0$ $(AC) \oplus (E)_S \rightarrow (AC)$
TLC	(641)	$(AC)_L \oplus E \rightarrow (AC)_L$
TLCE	(643)	skip if $(AC)_L \wedge E = 0$ $(AC)_L \oplus E \rightarrow (AC)_L$
TLCA	(645)	always skip $(AC)_L \oplus E \rightarrow (AC)_L$
TLCN	(647)	skip if $(AC)_L \wedge E \neq 0$ $(AC)_L \oplus E \rightarrow (AC)_L$
TRC	(640)	$(AC)_R \oplus E \rightarrow (AC)_R$
TRCE	(642)	skip if $(AC)_R \wedge E = 0$ $(AC)_R \oplus E \rightarrow (AC)_R$
TRCA	(644)	always skip $(AC)_R \oplus E \rightarrow (AC)_R$
TRCN	(646)	skip if $(AC)_R \wedge E \neq 0$ $(AC)_R \oplus E \rightarrow (AC)_R$

Inter-Program Transfer

In the following descriptions, a reference to flags refers to a half-word quantity described as follows:

bit 0:	AROV	bit 5:	User Mode
bit 1:	CRY0	bit 6:	User Mode IOT
bit 2:	CRY1	bit 11:	FXU
bit 3:	FOV	bit 12:	DCK
bit 4:	BIS	and 0s elsewhere.	

JSR	(264)	Jump to SubRoutine flags $\rightarrow (E)_L$ PC $\rightarrow (E)_R$ then $E + 1 \rightarrow$ PC
JRST	(254)	Jump and ReSTore This instruction may be used as the return from a JSR or JSP. The AC-field determines additional functions: If AC10 (bit 9 = 1) reset the current priority interrupt channel. If AC4 (bit 10 = 1) then halt If AC2 (bit 11 = 1) then restore flags. Indirect addressing or indexing must be specified if AC2 is designated. If AC1 (bit 12 = 1) then set user mode flag.
JSP	(265)	Jump and Save Program counter PC $\rightarrow (AC)_R$ flags $\rightarrow (AC)_L$ $E \rightarrow$ PC
JSA	(266)	Jump and Save Accumulator (see JRA for return) $(AC) \rightarrow (E)$ PC $\rightarrow (AC)_R$ $E \rightarrow (AC)_L$ $E + 1 \rightarrow$ PC
JFFO	(243)	Jump if Find First One If $(AC) = 0$, $0 \rightarrow AC + 1$ If $(AC) \neq 0$, [the bit position of the leftmost 1 of (AC)] $\rightarrow AC + 1$ $E \rightarrow$ PC

JRA	(267)	Jump and Restore Accumulator (used as a return for JSA) $E \rightarrow PC$ $((AC)_L) \rightarrow (AC)$
JFCL	(255)	Jump and Clear Flags if any flag selected by the AC field is a 1 then E → PC and the selected flags are cleared AC-field flag assignments. bit 9: arithmetic overflow bit 10: carry 0 bit 11: carry 1 bit 12: floating overflow

Note: See PUSHJ and POPJ under pushdown operations for completely recursive subroutine operations.

Push Down List

The push down overflow flag is set if a carry from bit 0 of the accumulator occurs during incrementing or decrementing. Bit assignments for flags are shown under the inter-program transfer instructions.

PUSH	(261)	PUSH $(AC) + 1000001_8 \rightarrow (AC)$ then $(E) \rightarrow ((AC)_R)$
POP	(262)	POP $((AC_R) \rightarrow (E)$ then $(AC) - 1000001_8 \rightarrow (AC)$
PUSHJ	(260)	PUSH and Jump $(AC) + 1000001_8 \rightarrow (AC)$ then $PC \rightarrow ((AC)_R)$ flags $\rightarrow ((AC)_L)$ $E \rightarrow PC$
POPJ	(263)	POP and Jump $((AC)_R)_R \rightarrow PC$ then $(AC) - 1000001_8 \rightarrow (AC)$

Execute

XCT	(256)	execute the instruction at (E). This may be another XCT instruction

INPUT/OUTPUT

Input/output instructions address devices connected to the input/output bus. Each device contains a device status register (DSR) which records commands from the central processor and contains the status of the device (e.g. interrupt request, activity, errors). Data is transferred through the data buffer register (DBR) located at the device, and may consist of from one to 36 bits depending upon the nature of the device. The central processor and the priority interrupt system are regarded as devices (addresses 000 and 004_8 respectively).

CONO	(700)	CONditions Out $E \rightarrow (DSR)$
CONI	(700)	CONditions In $(DSR) \rightarrow (E)$
DATAO	(700)	DATA Out $(E) \rightarrow (DBR)$
DATAI	(700)	DATA In $(DBR) \rightarrow (E)$
CONSZ	(700)	CONditions in and Skip if Zero skip if $(DSR) \wedge E = 0$
CONSO	(700)	CONditions in and Skip if One skip if $(DSR) \wedge E \neq 0$
BLKI	(700)	BLocK In The effective address is used to locate a pointer word. The left half of the pointer word contains the word count and the right half holds the operand address. $(E) + 1000001_8 \rightarrow (E)$ then $(DBR) \rightarrow ((E)_R)$ If not executed in a priority interrupt cycle, then skip if a carry occurred from bit 0 when incrementing the pointer word. If in priority interrupt cycle and overflow occurred then execute next instruction after trap location. If in program interrupt cycle and overflow did not occur, then dismiss interrupt and return to the interrupted sequence.

BLKO	(700)	BLocK Out same as BLKI except $((E)_R) \rightarrow (DBR)$

Central Processor

The central processor is addressable as an input/output device with address 000_8. The input/output instructions are used for console operations, loading the memory protection and relocation registers, and processing machine flags. MACRO-10 assembly language notation is used in the following discussion.

DATAI APR, ADDRESS

The contents of the console data switches are stored in the effective address.

DATAO APR, ADDRESS

Bits 18-25 of the contents of the effective address are stored in bits 18-25 of the relocation register. Bits 0-7 are stored in bits 18-25 of the memory protection register.

CONI APR, ADDRESS

The contents of the effective address bits 19-35 are replaced. Unused bit positions are cleared.

(E) bit	Value
33-35	processor interrupt channel assignment
32	arithmetic overflow flag
31	arithmetic overflow interrupt enable flag
30	not used
29	floating overflow flag
28	floating overflow interrupt enable flag
27	not used
26	clock flag
25	clock interrupt enable flag
24	not used
23	non-existent memory flag
22	memory protection violation flag
21	address break flag
20	user mode IOT flag
19	push down overflow flag

CONO APR, ADDRESS

The bits of the effective address, E, perform the indicated functions.

(E) bit	Function
33-35	assigns the processor to an interrupt channel
32	clears arithmetic overflow flag
31	sets arithmetic overflow interrupt enable flag
30	clears arithmetic overflow interrupt enable flag
29	clears floating overflow flag
28	sets floating overflow interrupt enable flag
27	clears floating overflow interrupt enable flag
26	clears clock flag
25	sets clock interrupt enable flag
24	clears clock interrupt enable flag
23	clears non-existent memory flag
22	clears memory protection violation flag
21	clears address break flag
20	not used
19	transmits a reset signal to all I/O devices
18	clears push down overflow flag

Priority Interrupt System

The priority interrupt system is addressable as device 004_8 (PI).

DATAI PI, ADDRESS

Zeros are stored in (E).

DATAO PI, ADDRESS

The (E) are displayed at the console on the memory indicator lights.

CONI PI, ADDRESS

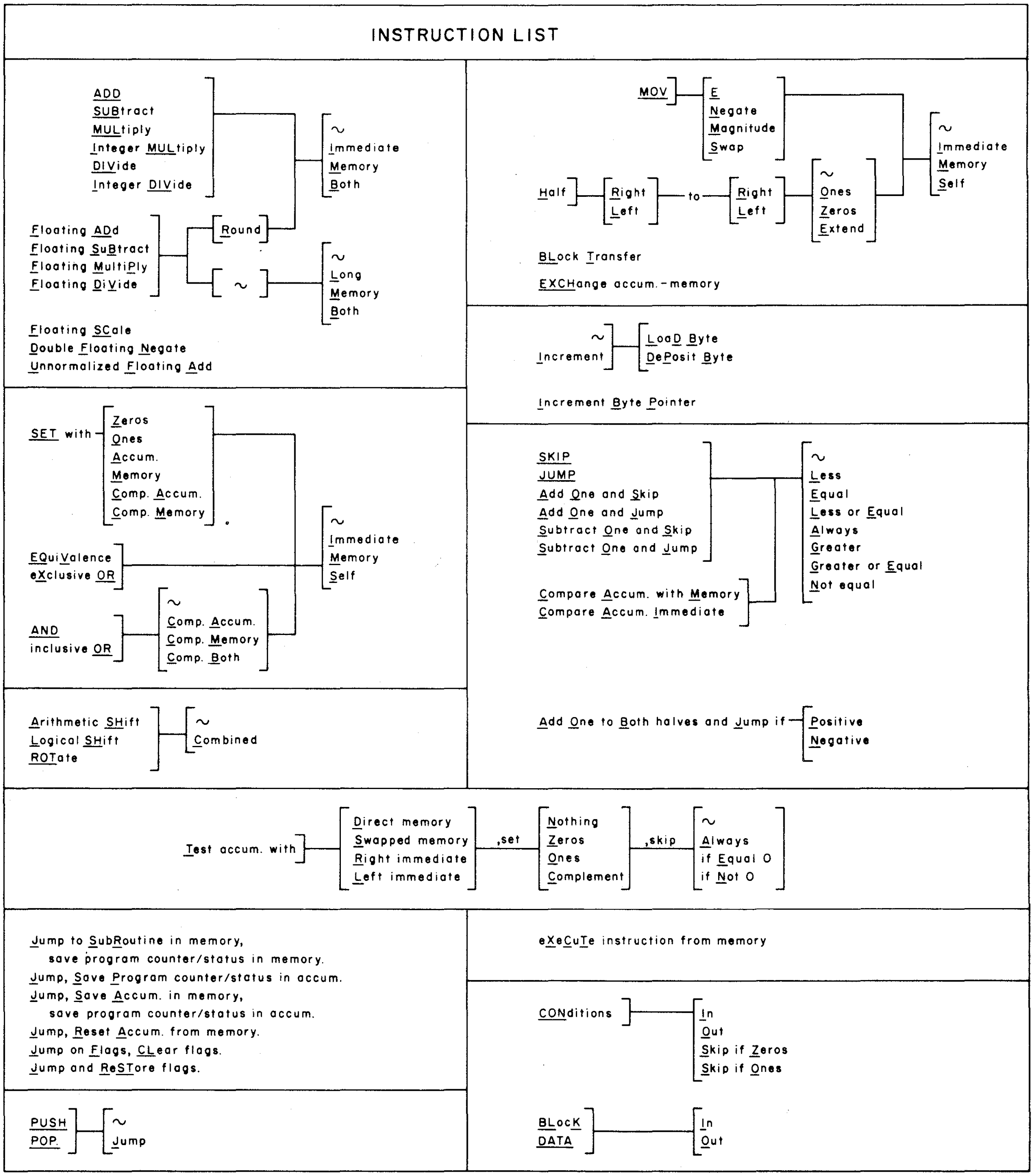
The (E) are replaced. Unused bit positions are cleared.

(E) bit	Value
35	channel 7 has been enabled
•	•
•	•
•	•
29	channel 1 has been enabled
28	priority interrupt system is on
27	interrupt in progress on channel 7
•	•
•	•
•	•
21	interrupt in progress on channel 1
20	memory parity error interrupt enable flag
19	memory parity error flag
18	power failure flag

CONO PI, ADDRESS

The $(E)_R$ perform the indicated functions.

(E) bit	Function
35	selects channel 7 for bits 24, 25, 26
•	•
•	•
•	•
29	selects channel 1 for bits 24, 25, 26
28	turn on priority interrupt system
27	turn off priority interrupt system
26	disable selected channel
25	enable selected channel
24	initiate an interrupt on selected channel
23	clear (reset) priority interrupt system
22	not used
21	set memory parity error interrupt enable flag
20	clear memory parity error interrupt enable flag
19	clear memory parity error flag
18	clear power failure flag



INSTRUCTION LIST

ADD
SUBtract
MULtiply
Integer MULtiply
DIVide
Integer DIVide
~
Immediate
Memory
Both

Floating ADd
Floating SuBtract
Floating MultiPly
Floating DiVide
Round
~
Long
Memory
Both

Floating SCale
Double Floating Negate
Unnormalized Floating Add

SET with
Zeros
Ones
Accum.
Memory
Comp. Accum.
Comp. Memory

EQuiValence
eXclusive OR

AND
inclusive OR
~
Comp. Accum.
Comp. Memory
Comp. Both

~
Immediate
Memory
Self

Arithmetic SHift
Logical SHift
ROTate
~
Combined

MOV
E
Negate
Magnitude
Swap

Half
Right
Left
to
Right
Left
~
Ones
Zeros
Extend

~
Immediate
Memory
Self

BLock Transfer
EXCHange accum.-memory

~
Increment
LoaD Byte
DePosit Byte

Increment Byte Pointer

SKIP
JUMP
Add One and Skip
Add One and Jump
Subtract One and Skip
Subtract One and Jump

Compare Accum. with Memory
Compare Accum. Immediate

~
Less
Equal
Less or Equal
Always
Greater
Greater or Equal
Not equal

Add One to Both halves and Jump if
Positive
Negative

Test accum. with
Direct memory
Swapped memory
Right immediate
Left immediate
,set
Nothing
Zeros
Ones
Complement
,skip
~
Always
if Equal 0
if Not 0

Jump to SubRoutine in memory,
save program counter/status in memory.
Jump, Save Program counter/status in accum.
Jump, Save Accum. in memory,
save program counter/status in accum.
Jump, Reset Accum. from memory.
Jump on Flags, CLear flags.
Jump and ReSTore flags.

eXeCuTe instruction from memory

CONditions
In
Out
Skip if Zeros
Skip if Ones

PUSH
POP
~
Jump

BLocK
DATA
In
Out

SYMBOLS

(A)	contents of A
$(A)_L$	left half of (A)
$(A)_R$	right half of (A)
$(A)_S$	contents of A with left and right halves swapped
$\wedge$	boolean AND
$\vee$	boolean inclusive OR
$\oplus$	boolean exclusive OR
'	boolean inversion
$+$	arithmetic addition
$-$	arithmetic subtraction
$\times$	arithmetic multiplication
$\div$	arithmetic division
$\lvert\;\rvert$	arithmetic magnitude
$(A) \longrightarrow (B)$	contents of A replaces contents of B
AC	accumulator
E	effective address
$=$	equal
$\neq$	not equal

MACRO ASSEMBLY STATEMENTS

Symbols

1. One to six characters in length
2. Composed of characters from the set:
   - A through Z
   - 0 through 9
   - $ (dollar sign)
   - % (percent symbol)
   - . (point)

   First character cannot be a digit; if first character is a period, second character cannot be a digit. No imbedded spaces are allowed.

Direct Assignment Statement

symbol = value

Numbers

$2 \leq \text{radix} \leq 10$

RADIX statement - Used to set radix for numbers which follow

To set radix for single numeric term:

↑Dnn	Decimal
↑Onn	Octal
↑Bnn	Binary

Binary shifting:

nnBb — Place the number nn in a word, placing the rightmost bit of the number in bit "b"

-nnBb — Same as above but store the 2's complement of the number

Floating point decimal numbers:

nnnEe — Store the number as a floating point decimal number with the signed exponent "e"

Any string of digits containing a decimal point is also stored as a floating point decimal number.

Fixed point decimal numbers:

↑Fnnn.nnBb — Assembler places value into two 36-bit registers, with the integer portion in the left register and the fractional portion in the right register. The value is then stored in one word, with the assumed decimal point set after bit "b" in the storage word.

Arithmetic and Logical Operators:

+	Add
-	Subtract
*	Multiply
/	Divide
&	AND
!	Inclusive OR

Order of Evaluation:

Expressions enclosed in angle brackets, beginning with the innermost pair

Unary operations (leading + or −)

Binary shifts

Logical operations (left to right)

Multiplication and division (left to right)

Addition and subtraction (left to right)

Numeric term: a digit, a string of digits, or an expression enclosed in angle brackets

Address Assignments

Assigns addresses consecutively

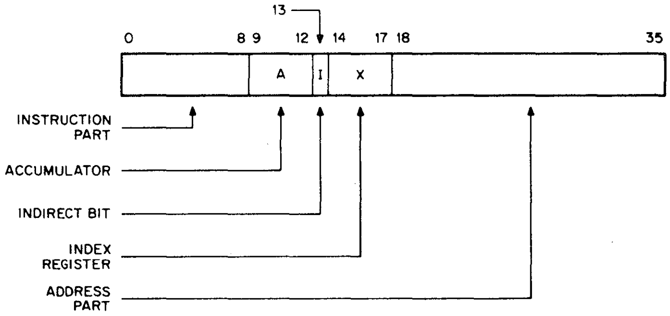
Location counter: . (point)

Indirect addressing: @ adr

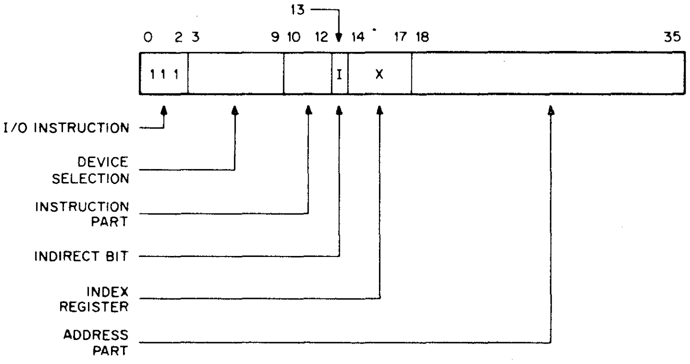
Indexing adr(n) where n = 1 through 17

Literal: [data-generating expression]

Primary Instruction Format



Input/Output Instruction Format



SUMMARY OF PSEUDO-OPS

ASCII	Seven-bit ASCII text.
ASCIZ	Seven-bit ASCII text, with null character guaranteed at end.
BLOCK	Reserves block of storage cells.
BYTE	Input bytes of length 1 through 36 bits.
DEC	Input decimal numbers.
DEFINE	Defines macro.
DEPHASE	Terminates PHASE relocation mode.
ENTRY	Enters subroutine library.
EXP	Input expressions.
EXTERN	Identifies external symbols.

Conditional Assembly Statements

	Assemble if:
IF1	Encountered during pass 1.
IF2	Encountered during pass 2.
IFB	Blank
IFDEF	Defined
IFDIF	Different
IFE	Zero
IFG	Positive
IFGE	Zero, or positive
IFIDN	Identical
IFL	Negative
IFLE	Zero, or negative
IFN	Non-zero
IFNB	Not blank
IFNDEF	Not defined
INTERN	Define internal symbols.
IOWD	Set up I/O transfer word.
IRP	Indefinite repeat of macro arguments.
LALL	List all; expanded listing.
LIST	Implied at end of pass 2.

LIT	Assemble literals
LOC	Assign absolute addresses.
NOSYM	Suppress symbol table listing.
OCT	Input octal numbers.
OPDEF	Defines user-created operator. Generates only one word.
PAGE	Skip to top of next page.
PASS2	Terminates pass 1 remaining statements are processed pass 2 only.
PHASE	Following coding relocated at execution time.
POINT	Sets up byte pointer word.
PRINTX	Prints when encountered during pass 1.
PURGE	Purge symbols
RADIX	Sets prevailing radix to 2-10.
RADIX50	Compresses 36-bit words, primarily for system use.
RELOC	Implied first statement; assigns relocatable addresses.
REMARK	Comments only statement.
REPEAT	Repeat n times
RIM	Prepare output in RIM paper-tape format.
RIM10	Absolute, unblocked, output format. No checksums.
RIM10B	Absolute, blocked, checksummed output format.
SIXBIT	Input text in compressed 6-bit ASCII.
STOPI	Stop indefinite repeat of macro arguments.
SUBTTL	Subtitle on listing.
SYN	Make synonomous
TITLE	Title on listing.
VAR	Assemble variables suffixed with #.
XALL	Stop expanded listing.
XLIST	Stop listing.
XWD	Input two 18-bit half words.
Z	Input zero word.

SUMMARY OF CHARACTER INTERPRETATION

The characters listed below have special meaning in the contexts indicated. These interpretations do not apply when these characters appear in text strings, or in comments.

Character	Meaning		Example
:	Colon. Immediately follows all labels.		LABEL: Z
;	Semi-colon. Precedes all comments.		;THIS IS A COMMENT
.	Point. Has current value of the location counter.		JRST .+5 Jump forward five locations.
,	Comma. General operand or argument delimiter.		DEC 10, 5, 6 EXP A+B, C-D
	Accumulator field delimiter		MOVEI 1, TAG
	References accumulator 0. The comma is optional.		MOVEI, TAG
	Delimits macro arguments.		MACRO (A,B,C)
!	Inclusive OR	Logical Operators	
&	AND		
*	Multiplication	Arithmetic Operators	
/	Division		
+	Add		
-	Subtract		
1st character of text string	In ASCII, ASCIZ and SIXBIT test strings, the first non-blank character is the delimiter.		ASCII/STRING/;
B	Follows number to be shifted and precedes binary shift count.		7B2
E	Exponent. Precedes decimal exponent in floating-point numbers.		F22.1E5 Exponent is 5.
( )	Parentheses. Use to enclose index fields.		ADD AC1, X(7)
	Enclose the byte size in BYTE statements.		BYTE (6) 8, 8, 7
	Enclose the dummy argument string in macro DEFINE statements.		DEFINE MAC (A,B,C)

Character	Meaning	Example
< >	Angle brackets. In an expression, enclose a numeric quantity.	<A-B+500/C>
	In conditional assembly statements, contain a single argument, and the conditional coding.	IF1 <MOVE AC0, TAX>
	In REPEAT statements, contain coding to be repeated.	REPEAT 3, <SUB 17, TAG>
	In macros, enclose the macro definition.	DEFINE PUNCH <DATAO PTP, PUNBUF (4)>
[ ]	Square brackets. Delimits literals.	ADD 5, [MOVEI 3,TAX]
	In OPDEF statement, contain new operator.	OPDEF CAL [MOVE]

=	Equal sign, direct assignment.	SYM=6
		SYM=A+B*D

"..."	Quotation marks enclose 7-bit ASCII text, from one to five characters.	"ABCDE"
#	Number sign. Defines a symbol used as a tag. Variable.	ADD 3,TAG#
'	Apostrophe or single quote. Catenation character used only within macro definitions.	DEFINE MAC (A,B,C); < JUMP'A,B,C>
\	Reverse slash. If used as the first character in a macro call, the value of the following symbol is converted to an ASCII symbol in the current radix.	MAC \A if A=500, this generates three 7-bit ASCII characters.
←	Left arrow. Line continuation.	

FORTRAN IV COMPILER (F40)

FUNCTION

To compile source programs coded in FORTRAN IV programming language and produce machine language programs which are compatible with the Linking Loader and the Dynamic Debugging Technique program.

- One-pass compiler
- Accepts input from any input device
- Includes features far in advance of ASA FORTRAN X3.9, 1966, requirements

ENVIRONMENT

Monitor	All
Minimum Core	FORTRAN IV - (9K permits 4 continuation lines; 10K or more permits 19 continuation lines) FORTRAN IV Subset - 5.5K (4 continuation lines only)
Additional Core	Requests additional core from the Monitor when needed.
Equipment Required	One input device (source program input). Two output devices (if both machine language output and a listing are desired).

INITIALIZATION

.R F40 core ↵	Loads the FORTRAN IV Compiler into core.
*	The FORTRAN IV Compiler is ready to accept a command.

COMMANDS

General Command Format

objprog-dev:filename.ext,list-dev:filename.ext←source-dev:filename.ext,........source-n ↵

objprog-dev:

The device on which the machine language coding is to be written. Only one such file is produced.

MTAn:	(magnetic tape)
DTAn:	(DECtape)
DSK:	(disk)
PTP:	(paper tape punch)

list-dev:[2]

The device on which the compilation listing is to be written. Only one such file is produced.

MTAn:	(magnetic tape)
DTAn:	(DECtape)
LPT:	(line printer)
DSK:	(disk)
TTY:	(Teletype)

source-dev:[2]

The device(s) from which the source program(s) input to compilation is to be read.[1]

MTAn:	(magnetic tape)
DTAn:	(DECtape)
TTY:	(Teletype)
CDR:	(card reader)
PTR:	(paper tape reader)

[1] Note: Each source file may contain any number of source programs and need not contain an integral number of source programs as END OF FILE is ignored. However, statements may not be split between files.

[2] If this device is omitted, the last device named previous to it in the command string is assumed.

If more than one file is to be compiled from a magnetic tape, card reader, or paper tape reader, dev: is followed by a comma for each file beyond the first.

If more than one file is to be compiled from DSK: or DTAn:, subsequent filename.ext's after the first are delimited by commas; dev: need not be repeated.

Input via the Teletype is terminated by typing the END statement followed by a carriage return. & terminating file with ↑Z.

S.M.U.

filename.ext (DSK: and DTAn: only)

The filename and filename extension of the object program file, the listing file, and the source file(s). If .ext is omitted, .REL is assumed for the object file, .LST is assumed for the listing file, and .F4 is assumed for the source file.

←

The object program and listing devices are separated from the source device(s) by a left arrow.

Disk File Command Format

DSK:filename.ext[proj,prog]

[proj,prog]

The project-programmer number assigned to the disk area to be searched for the source file(s) or in which the destination file is to be written if other than the user's project-programmer number.

The standard protection[1] is assigned to any disk file specified as output.

Notes

If object coding output is not desired (as in the case where a program is being scanned for source language errors), objprog-dev: is omitted.

If a compilation listing is not desired, list-dev: is omitted.

[1] Standard protection (055) designates that the owner is permitted to read or write, or change the protection of, the file while others are permitted only to read the file.

EXAMPLES

```
.R F40↲
*DSK:OBJPRO,LPT:←TTY:↲
```

Compile one source file from the Teletype; write the object code on the disk and assign the filename.ext OBJPRO.REL; write the compilation listing on the line printer.

```
 ⇥COMMON A,B,C,D↲                    Note 1
 ⇥DOUBLE PRECISION A,B,C,D↲
 ⇥TYPE 10↲
 ⇥TYPE 20,A,B,C,D↲
 ⇥CALL RESID (1,1,'P2222')↲
10 ⇥ FORMAT ('--PHASE1,MAIN--'/)↲
20 ⇥ FORMAT (1H0,D/)↲
 ⇥END↲
↑Z↲
MAIN. ERRORS DETECTED:  0

TOTAL ERRORS DETECTED:  0
9K CORE USED
```

Typein of source coding

```
*DTA3:TSTA,←CDR:,,↲

MAIN. ERRORS DETECTED:  0
SUB1 ERRORS DETECTED:  0
SUB2 ERRORS DETECTED:  0

TOTAL ERRORS DETECTED:  0
9K CORE USED
```

Compile three source program files from the card reader; write the object code on DTA3 and call the file TSTA.REL; produce no compilation listing.

NOTE: This is not batch compilation. There may be many source programs but only one object code file is produced. Assume there is one main program and two subroutines, SUB1 and SUB2.

```
*↑C↲
```

Return to the Monitor.

```
.KJOB

.
```

Kill the job, deassign all devices, and release core.

next field

nonsense

NOTE 1: The use of TAB (⇥) effectively positions the Teletype at column 7 of the FORTRAN Programming Coding Form (column 6 if a continuation Line). TAB is typed by holding down the CTRL key at the same time the TAB key is depressed.

BATCH COMPILATION

If several independent program files being compiled from the same input device require independent listings on the same device, and require their (independent) machine language output on the same device, the length of the command strings required for each compilation can be shortened by assigning the three common devices before running the compiler. The command strings will then consist only of the filenames of the source program files. As in the preceding section, each source file may contain several source programs but each file will be assumed to contain an integral number of programs.

Example

The files TESTA, TESTB, TESTC, and TESTD, all of which reside on DTA1 are to be compiled. File TESTA contains two source programs. The object coding for each file is to be written on DTA2 and assigned the filenames of TESTA.REL, TESTB.REL, TESTC.REL, and TESTD.REL, respectively.

```
.AS DTA1 SRC ↵                       Assign DTA1 as the common source device.

DEVICE DTA1 ASSIGNED ↵

.AS DTA2 BIN ↵                       Assign DTA2 as the common binary coding output
                                     device.

DEVICE DTA2 ASSIGNED ↵

.AS LPT LST ↵                        Assign the line printer as the common listing output
                                     device.

DEVICE LPT ASSIGNED ↵

. F40 ↵

*TESTA,TESTB,TESTC,TESTD ↵           No devices need be specified in the command; the
MAIN. ERRORS DETECTED: 0             devices assigned are assumed.
SUBA ERRORS DETECTED: 0              The object code from the two source programs in file
SUBB ERRORS DETECTED: 0              TESTA will be written as file TESTA.REL on DTA2
SUBC ERRORS DETECTED: 0              and so on for the other source files. [1]Assume TESTA
SUBD ERRORS DETECTED: 0              contains a main program and subroutine SUBA,
                                     TESTB, TESTC, TESTD each contain one subroutine.
TOTAL ERRORS DETECTED: 0
9K CORE USED
*↑C ↵                                Return to the Monitor (or, if desired, more filenames
                                     could be specified for another compilation).

 .KJOB ↵
```

[1] File TESTA.F4, TESTB.F4, etc., are assumed as the filenames and extensions of the source files; if TESTA.F4 cannot be found, filename TESTA is searched for (same for TESTB, TESTC, TESTD).

SWITCHES

Switches are used to specify such options as:

1. Magnetic tape control,
2. Device directory manipulation, and
3. Types of listings and message typeouts.

All switches are either preceded by a slash or enclosed in parentheses. Switches C, E, M and N are complementary.

Table FORTRAN-1 FORTRAN Switch Options

Switch	Meaning
A[1]	Advance magnetic tape reel by one file.
B[1]	Backspace magnetic tape reel by one file.
C[2]	Generate a CREF-type cross-reference listing. Complement: Do not produce cross-reference information (standard procedure).
E[2]	Print an octal listing of the binary program produced by the compiler in addition to the symbolic listing output. Complement: Do not produce octal listing (standard procedure).
M[2]	Eliminate the macro coding from the output listing. Complement: Include macro coding in the output listing (standard procedure).
N[2]	Suppress output of error messages on the Teletype. Complement: Output error messages on TTY (standard procedure).
S	If the compiler is running on the PDP-10, produce code for execution on the PDP-6 and vice-versa.
T[1]	Skip to the logical end of the magnetic tape reel.
W[1]	Rewind the magnetic tape reel.
Z[1]	Zero the DECtape directory.

NOTES: 1. Must immediately follow the device name or filename.ext to which it applies.

2. Standard listing procedures are - list error messages on the TTY and the source program with macro coding on the listing device. ~~This is done~~ unless a switch is used to override the standard listing. If the switch is complementary, a second occurrence of it in a command string means revert to standard ~~listing procedures~~ as described for the complement of the particular switch. A third occurrence of a switch acts as the first, and so on.

complemented

EXAMPLES

```
.R F40 ↲
*DTA2:/ZOBJTST,LPT:←DSK:/EFORTAA[12,20] ↲
```

Zero the directory on DTA2 and create a file called OBJTST.REL as the binary object program file; compile the source file, FORTAA, located in area 12,20 of the disk; produce a binary program listing (with octal code) as well as the regular source listing on the line printers. Assume FORTAA contains one main program.

```
     20        FORMMAT (1H0,D/) ↲
                  ↑
  1) SYNTAX ↲
```

A syntax error has been detected in the source language. Such typeouts can be suppressed by use of the /N switch.

```
MAIN. ERRORS DETECTED:  1

? TOTAL ERRORS DETECTED:  1
9K CORE USED

*DTA3:JOB1,DSK:JOB1←MTA1:/W,, ↲

SUB1 ERRORS DETECTED:  0
SUB2 ERRORS DETECTED:  0
SUB3 ERRORS DETECTED:  0

TOTAL ERRORS DETECTED:  0
9K CORE USED
```

Create a new file called JOB1.REL on DTA3 as the binary object program file; create a file called JOB1.LST on the disk as the listing file; rewind MTA1, and compile the first three program files. Assume each contains one subroutine, SUB1, SUB2, SUB3.

```
*↑C↲
```

Return to the Monitor.

```
.KJOB ↲

.
```

Kill the job, deassign all devices, and release core.

DIAGNOSTIC MESSAGES

After each source program is compiled the message

"program-name"   ERRORS DETECTED: n

is printed where

"program-name" is 1) MAIN. for a main program
2) DAT. for a BLOCK DATA subroutine
3) Subroutine-name for a subroutine

and after all the files specified in the command string have been compiled, the messages

TOTAL ERRORS DETECTED: m
nK CORE USED

are printed, where m is the total number of errors detected and is preceded by "?" if m>0, and n is the number of 1K blocks required to compile the programs.

Table FORTRAN-2 FORTRAN Diagnostic Messages

Message	Meaning
?BINARY OUTPUT ERROR dev:filename.ext	An output error has occurred on the device specified for the binary program output.
?CANNOT FIND dev:filename.ext	Filename.ext cannot be found on this device.
?INPUT DATA ERROR dev:filename.ext	A read error has occurred on the source device.
?x IS A BAD SWITCH	The specified switch is not recognizable.
?x IS AN ILLEGAL CHARACTER	A character in a command string typein is not recognizable (e.g., FORM-FEED).
? dev: IS NOT AVAILABLE	Either the device does not exist or it has been assigned to another job.
?LISTING OUTPUT ERROR dev:filename.ext	An output error has occurred on the device specified for the binary program output.
?NO ROOM FOR dev:filename.ext	The directory on dev: DTAn is full and cannot accept filename.ext as a new file, or a protection failure occurred for a DSK output file.
?SYNTAX ERROR IN COMMAND STRING	A syntax error has been detected in a command string typein (e.g., the ← has been omitted).
?INSUFFICIENT CORE - COMPILATION TERMINATED	The compiler has insufficient table space to compile the program.

SMALL FORTRAN IV COMPILER

The instructions below apply to the small FORTRAN IV (F40S) Compiler, which handles a subset of the standard FORTRAN IV Compiler (F40).

INITIALIZATION

Logical Assignments (see "Batch Compilation")

.R F40S↵	Loads the small FORTRAN IV Compiler into core.
*	The Compiler is ready to accept a command.

COMMANDS

1. Logical assignments must be made as shown for batch compilation (page FORTRAN-5).
2. The command string can contain many source files but each source file is assumed to contain only one source program. Every source file on a directory device must have the extension ".F4" as this is assumed by the compiler. The command string, then, consists of filenames only.
3. One binary output file and one listing file are produced if logical assignments have been made for them.
4. If no listing device is specified, a source code error produces "?E" on the user's Teletype. Compilation of that file is terminated.
5. No switches are allowed.

Example:

.AS DTA2 SRC ↵ DEVICE DTA2 ASSIGNED↵	Assign DTA2 as the source device.
.AS DTA3 BIN↵ DEVICE DTA3 ASSIGNED↵	Assign DTA3 as the binary output device.
.R F40S↵	Run the small compiler
*FILA, FILB↵	Compile file FILA.F4 and FILB.F4. the object code for FILA.F4 will be written as file FILA.REL (and FILB.F4 as FILB.REL) on DTA3.
*↑C ↵	Return to Monitor (or, if desired, more files can be compiled).

LINKING LOADER (LOADER)

FUNCTION

To load and link relocatable binary (.REL) programs generated by Macro-10 or FORTRAN IV preparatory to execution. Generates a symbol table in core for execution under the Dynamic Debugging Technique program.

- Provides automatic loading and relocation of Macro- and FORTRAN-generated binary programs
- Produces an optional storage map
- Storage used by the Linking Loader is recoverable after loading
- Performs loading and library searching regardless of the input medium

ENVIRONMENT

Monitor	All
Minimum Core	2K
Additional Core	Automatically requests additional core from the Monitor as required.
Equipment Required	User Teletype for control; one or more input devices for binary programs to be loaded; output device for loader map (optional); one systems device containing library files (optional).

INITIALIZATION

.R LOADER core↵

Loads the Linking Loader into core. Core allocated equals 2K plus core required by binary programs; "core" is optimal.

*

Indicates that the program is ready to receive a command.

COMMANDS

General Command Format

list-dev:filename.ext← source-dev1:filename.ext,dev2:....source-nALTMODE

list-dev:

The device on which any storage maps or undefined globals are to be written.

LPT:	(line printer)
TTY:	(Teletype)
DTAn:	(DECtape)
DSK:	(disk)
MTAn:	(magnetic tape)

If the Teletype is to be assumed as the output device, omit

list -dev:filename.ext←

source-dev:

The device(s) from which the binary relocatable programs are to be loaded.

DSK:	(disk)
DTAn:	(DECtape)
MTAn:	(magnetic tape)
PTR:	(paper tape reader)

If more than one file is to be loaded from a magnetic tape, card reader, or paper tape reader, dev: is followed by a comma (or the device name or : can be repeated) for each file after the first.

filename.ext (DSK: and DTAn: only)

The filename.ext of each relocatable binary file to be loaded. If .ext is omitted, it is assumed to be .REL. If a search for filename.REL is unsuccessful, a second search for the same filename with the null extension is performed.

The filename.ext of the output listing file. If .ext is omitted, .MAP is used.

←	The storage map device is separated from the source device(s) by the left arrow symbol.

Notes

1. Each time RETURN (↵) is typed, loading is performed for all files listed on that line.

2. Each time ALTMODE is typed, all remaining loading, library searches, and output operations are completed, and an exit is made to the monitor.

3. The source device, once stated, continues as the source device until a new source device or destination device is specified, or until ALTMODE is typed.

4. Files are loaded in the order they appear in the command string. The file requiring the largest COMMON area must be specified first in any loading operation.

5. When loading is terminated (by ALTMODE or switches /C, /G, or /R), the following steps are executed.

   a. A FORTRAN library search is performed if any undefined globals remain (unless prevented by the /P switch).

   b. If undefined globals still remain, they are listed on the Teletype or other specified listing device.

   c. The number of multiply defined globals (if any) and the number of undefined globals (if any) are printed on both the Teletype and on the specified listing device (if given).

   d. A Chain file, if requested, is written.

   e. The loaded program is relocated down to the actual locations into which it is to be loaded.

   f. The message

      ```
      LOADER
      EXIT
      ↑C
      ```

      is printed on the Teletype.

Save and Execute Commands

After loading is completed, to write the loaded program onto an output device so that it can be executed at some future date without rerunning Linking Loader:

```
LOADER↵                                          Loading is completed.
EXIT↵                                            Automatic exit to the Monitor.
↑C↵
```

`.SAVE dev:filename.ext core↵`	Write out the user's area of core onto the specified output device and, if the device is DTAn: or DSK: , assign it the specified filename.ext. If .ext is omitted, .SAV is assumed. The value for core may be given when the user wishes to run the program in more core than it will be saved in; this might be done to gain more space for dynamic allocation of buffers.

`JOB SAVED↵`
`↑C↵`
	Save operation completed. Core is unchanged and still contains loaded program. Automatic return is made to the Monitor.

`.START↵`	Start execution of loaded program. Return is made to user's level.
`EXIT↵` `↑C↵`	User's program execution is completed. Automatic return is made to the Monitor.
`.`	

EXAMPLES

```
.R LOADER 5↵
```

Load Linking Loader and assign it 5K of core.

```
*DSK:MARK1,MARK3,DTA3:SUBRTE ↵
*CALC,PTR:ALTMODE ↵
```

Load and link the .REL files MARK1 and MARK3 from the disk, .REL files SUBRTE and CALC from DTA3, and one .REL file from the paper tape reader.

```
LOADER ↵
EXIT↵
↑C ↵
```

Link-loading is completed; and automatic return is made to the Monitor.

```
.SAVE DSK MARKET↵
```

Write out the user's program as an executable program on the disk and call the file MARKET.DMP. Core assigned to the user remains unchanged.

NOTE: Saving a job is optional.

```
JOB SAVED↵
↑C ↵
```

Save process is completed; an automatic return is made to the Monitor.

```
.START↵
```

Begin execution of job.

```
EXIT↵
↑C↵
.
```

Program execution is completed; automatic return is made to the Monitor.

SWITCHES

Switches are used to:

1. Specify the types of symbols to be loaded or listed,
2. Set the library search mode,
3. Load the Dynamic Debugging Technique (DDT) program, and
4. Clear and restart Linking Loader.

All switches are either preceded by a slash (/) or enclosed in parentheses.

Table LOADER-1 Linking Loader Switch Options

Switch	Meaning	Complement Switch
A	List all global symbols in storage map regardless of program length.	Ⓧ
nnnnnC	Create Chain file; use first block data for program break; nnnnn (if nonzero) is starting address. Terminate Linking Loader.	
D	Load DDT; enter "load with symbols: mode (S); turn off library search mode (N). Terminates specification.	
E	Upon termination of loading, control will be transferred to user's program starting address (starting address of last program loaded).	
F	Perform a library search of LIB40; exit from "load with symbols" mode. Terminates specification.	
nnnnnG	Perform an automatic search of LIB40 if any undefined globals remain (unless the /P switch is used); list any still-undefined globals; set the starting address of the program as nnnnn; exit to the Monitor. Use ALTMODE, instead, if starting address to be used is the one originally specified.	
I	Set the loader to ignore the starting addresses in binary input.	Ⓙ
Ⓙ	Set the loader to accept the starting address of this binary input program.	I
L	Enter the library search mode.	Ⓝ
NOTE: ○	**indicates those switches set when loader is in its initial state.**	

Table LOADER-1 (Cont) Linking Loader Switch Options

Switch	Meaning	Complement Switch
M	Print the storage map and undefined globals. Terminate specification.	
Ⓝ	Turn off the library search mode.	
nnnnnO	Load beginning at numeric argument (octal) if nonzero.	
P	Prevent an automatic library search.	Ⓠ
Ⓠ	Allow an automatic library search. Turn off the "load with local symbols" switch.	P
nnnnnR	Create Chain file; use first FORTRAN IV program break; nnnnn (if nonzero) is starting address. Terminate Linking Loader.	
S	Load with local symbols.	Ⓦ
T	Loads SYS:DDT.REL; turns on "load with local symbols (S) switch; upon termination of loading transfers control to DDT for program testing.	
U	List undefined global symbols on the output list device. Terminates specification.	
Ⓦ	Load without local symbols.	S
Ⓧ	Suppress listing of global symbols for zero-length programs.	A
Y	Rewind magnetic tape before use.	
Z	Clear user's core area; reset the loader to its initial state; restore the Teletype; restart loading. Terminates line.	
NOTE ○	indicates those switches set when loader is in its initial state.	

The effect of a switch on adjacently named files in the command string depends upon whether the switch is a status switch or an action switch.

Status Switches (A, I, J, L, N, O, P, Q, S, W, X) set the loader to a particular status and have an effect on the file in whose specification it appears and on any subsequently name files in the command string (unless the switch is reset). A file specification is terminated and processed whenever a comma, or a colon (if the previous delimiter was a colon), a RETURN, or ALTMODE is encountered.

*DTA5:RESID/S,/M	Local symbols are loaded for this and any following files. A storage map is printed for this file.
*DTA5:RESID,/M/S	A storage map is printed for this file; however, local symbols are not loaded for this file since the /S switch appears outside the file specification (which is terminated by the comma). Local symbols are loaded for any following files.

*DTA5:RESID,/S — Local symbols are not loaded for this file since the /S switch appears outside the file specification (which is terminated by the comma).

Action Switches (C, D, E, F, G, M, R, T, U, Y) request an immediate or file-independent action to be performed by the Loader and are not directly related to any specific file specification(s).

Chain Feature

The Chain feature is used to segment FORTRAN programs which are too large to be loaded into core as one unit. When switch /C or /R is specified, loading is terminated and a file acceptable to the Chain program is written.

Examples: *DSK:CHNPRG←/R or *DTA1:SEGF4←/C

If .ext is omitted for the output Chain filename, .CHN is used.

The Chain file contains:

1. The contents to be loaded into JOBDDT, JOBSA, JOBFF, and JOBSYM.
2. The data, beginning from the Chain address through the top of the core area used in loading.

The Chain address is set from JOBCHN as loaded; switch /C specifies the right half and switch /R specifies the left half. Location JOBCHN is loaded as follows: (1) the right half contains the program break of the first FORTRAN IV BLOCK DATA program; (2) the left half contains the program break of the first FORTRAN IV program. If switch /C or /R contains a nonzero numeric argument, this becomes the starting address of the loaded program. After the Chain file has been written correctly, the messages below are output to the Teletype.

```
CHAIN↲
EXIT↲
↑C↲
```

EXAMPLES

.R LOADER 6↵	Run Linking Loader and assign it 6K of core.
*DTA5:RESID,SUB1,SUB2,DTA3:COMPLX↵	Load and link binary program files RESID.REL, SUB1.REL, and SUB2.REL from DTA5, and the file COMPLX.REL, DTA3. Carriage return initiates loading.
*/F↵	Force a premature search of LIB40 to resolve any undefined globals up to this point.
*/U↵ ?000001 UNDEFINED GLOBALS↵ ? SUB4A 000153↵	List on the Teletype (since no output device was specified in the first command line) all globals which are still undefined. Undefined global and location containing instruction which calls it are listed.
*DTA5:SUB4↵	Knowing that the undefined global is in the binary program file SUB4, the user requests that it be loaded also.
*/U↵	Check if undefined global has now been resolved.
*LPT:/M ← ALTMODE↵ LOADER↵ EXIT↵ ↑C↵	All globals are defined; print storage map on the line printer and exit to the Monitor.
.KJOB↵ .	Kill the job, deassign all devices, and release core.

DIAGNOSTIC MESSAGES

Table LOADER-2 Linking Loader Diagnostic Messages

Message	Meaning
?CANNOT FIND filename.ext	The filename.ext specified is not in the file directory. If no .ext is specified for a file, the file is first searched for with the name filename.REL, and if not found, is then searched for under the null filename extension.
?CHAIN DEV ERROR	A device error has occurred while writing the Chain file. Chain file is terminated.
?x CHAR. ERROR IN LOADER COMMAND	An illegal character was entered in a command string.
?DIR. FULL	The file directory of the specified list device is full and cannot contain an additional file, or a null file name was specified.
EXIT	If this message appears at the beginning of the run, either insufficient core has been assigned for loading or no console is attached to the job. EXIT normally is typed at the end of the loading process (after ALTMODE or /G) before exiting to the monitor.
?ILL. COMMON filename.ext	A file other than the first contains a program which has attempted to expand the already established COMMON area. This program must be loaded first.
?ILL. FORMAT filename.ext	The input source file is in proper checksummed binary format, but not in proper link format.
?INPUT ERROR filename.ext	A read error has occurred on an input source device. Use of that device is terminated.
?symbol ignored-value old-value MUL.DEF.GLOBAL filename.ext	A global symbol definition having a value different from that of a previous definition of the same symbol has been encountered. The new value is ignored and the symbol appears in the symbol table only once.
?NO CHAIN DEVICE	No device has been specified for the Chain file.

~~Page LOADER-6 (Switch E, "Meaning" Column) Add: "Equivalent to typing START after exit from Loader."~~

~~Page LOADER-7 (Switch N, "Complement Switch" Column) Add: "L"~~

~~(Switch T, "Meaning" Column) Add: "Equivalent to typing /D in command string and, then, after exit from Loader, typing DDT."~~

~~Page LOADER-8 (Line 13) Change "1. The contents to be loaded into JOBDDT, ..." to "1. The contents to be loaded into JOB41, JOBDDT, ..."~~

Page LOADER-9 (Bottom of page) Insert:

Use of /E Switch:

```
.R LOADER ↲
*DSK:PROG1,PROG2/E (ALTMODE)
LOADER CORE 7 ↲              (Typeout from Loader)
...program execution occurs here....
EXIT ↲
↑C ↲
```

(Following "? DIR. FULL") Insert:

ERROR WHILE READING COMMAND FILE	Loader was unable to read part of a command string stored on the disk. Program was partially loaded.
EXECUTION DELETED	Loader has loaded a program because the user gave an EXECUTE or DEBUG command, however, the program, or DDT, was not started because there were compilation errors in at least one .REL file.

(Following "? symbol ignored value....") Insert:

MORE CORE NEEDED	Loader requested additional core from Monitor, but none was available.

~~Page LOADER-10 (prior to last ~~line of~~ Table LOADER-2) Insert:~~

~~MORE CORE NEEDED — Loader requested additional core from Monitor, but none was available.~~

Page LIB40-1 — Change all references from "Scientific Library" to "Science Library" Change all references from "subroutines" to "subprograms"

Page FUDGE-5 — (line 4 and 5, command column) Change to:

```
*DSK:LIB4BB←DTA2:LIB4AA<EXP.3>_,DTA1:F1 ↲
<EXP.3A,EXP.3B>_,DTA3:EXPNT (R) ALTMODE ↲
```

Page DDT-1 — (line 12) Change "test" to "text"

Page DDT-2 — (lines 6, 7, 8) Asterisks should be blue. Change "*DTA5:TEST" to "*dev: filename.ext"

NOTE: The following changes reflect DDT.V11 and later.

Page DDT-8 — ("Type-Out Modes" - lines 9 and 10 under "Sample Output(s)") - Change

(4002)4005	to	4002,,4005
(X+1)X+4	to	X+1,,X+4

Page DDT-9 — ("Examining Storage Words" line 1 under "Sample Output(s)")

(254020) DDTEND	to	254020,,DDTEND

(line 3 under "Sample Output(s)") Change

254020003454	to	254020,,3454

Page DDT-10 — ("Typing In" - lines 4 and 5) Change

"To type in half words, enclose. . . . " (402)403

to

"To type in half words, separate the left and right halves with two commas." 402,,403

Page DDT-11 — ("Arithmetic Operators Permitted in Forming Expressions")

Change "Two's complement addition" to "Two's complement integer addition"

Change "Two's complement subtraction" to "Two's complement integer subtraction"

Delete the entire line on "Hierarchial parentheses"

(Field Delimiters in Symbolic Typeins") Insert:

To delimit two half words, type left,,right

Page DDT-12 — ("Breakpoints (Cont.)") Change

adr(x)$nB	Z+6(AC3)$5B	to	x,,adr$nB	AC3,,Z+6$5B
adr(x)$B	ABLE(AC4)$B		x,,adr$B	AC4,,ABLE$B
adr(x)$$nB	Z+6(AC3)$$5B		x,,adr$$nB	AC3,,Z+6$$5B
adr(x)$$B	ABLE(AC4)$$B		x,,adr$$B	AC4,,ABLE$$B

Table LOADER-2 (Cont)   Linking Loader Diagnostic Messages

Message	Meaning
?x SWITCH ERROR IN LOADER COMMAND	An improper switch designation has been entered in a command string.
?x SYNTAX ERROR IN LOADER COMMAND	A syntax error has been encountered in a command string.
?dev: UNAVAILABLE	Either the device does not exist or it is assigned to another job.
?UNCHAINABLE AS LOADED	The Chain address (the half of JOBCHN selected by /C or /R) is zero.
?nnnnnn UNDEFINED GLOBALS	nnnnnn undefined globals were found.
?SYMBOL TABLE OVERLAP file.ext ?nnnnnn WORDS OF OVERLAP file.ext	nnnnnn additional words (octal) are required to load everything requested in the last command string line.

FORTRAN LIBRARY

- FORTRAN OPERATING SYSTEM (FORSE.)
- SCIENTIFIC SUBROUTINES

FUNCTION

The FORTRAN Library (LIB40) contains the FORTRAN Operating System (FORSE.) and its subroutines, as well as the ~~Scientific~~ Science Subroutine Library. Some of these library routines are always required by FORTRAN-compiled programs at run time.

The FORTRAN Operating System performs two major functions: (1) all I/O operations and format conversions, and (2) error checking and output of error messages on the Teletype.

COMPILATION TIME

At compilation time, the FORTRAN compiler generates user UUO's in the range of 00 through 37_8 to communicate the necessary I/O requests to FORSE. These UUO's are interpreted at run time by FORSE., which in turn executes Monitor UUO's (40 through 77_8) to perform the actual I/O operations called for.

LOADING TIME

The loading of the appropriate subroutines from the FORTRAN Library is performed for each FORTRAN-compiled program by the Linking Loader. When loading is terminated, an automatic search of the FORTRAN Library file (LIB40) is performed for any programs referred to by the FORTRAN-compiled program but not explicitly loaded previously by the user. As a result of this search, FORSE., its related subroutines, and any required ~~Scientific~~ Library subroutines are loaded.
Science

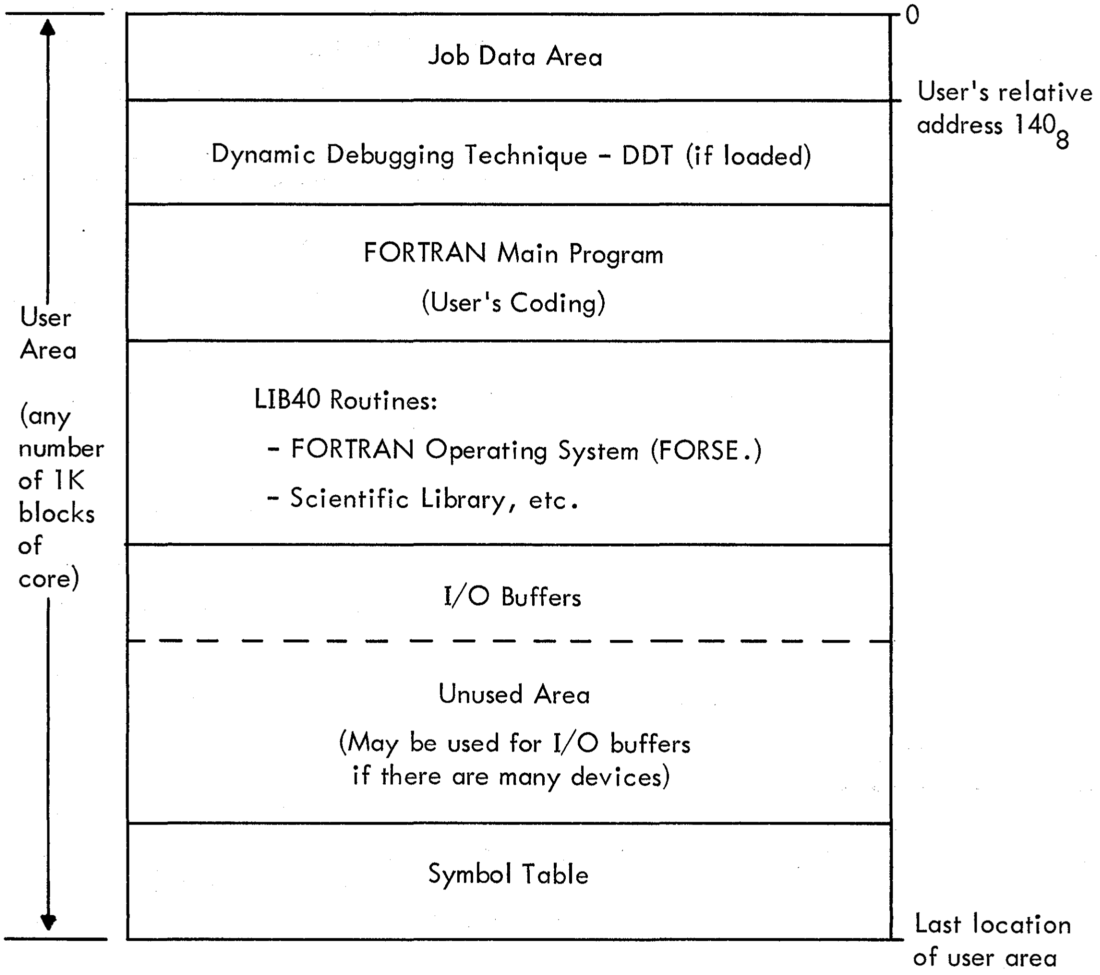

0
Job Data Area
User's relative address 140_8
Dynamic Debugging Technique - DDT (if loaded)
FORTRAN Main Program
(User's Coding)
User Area
(any number of 1K blocks of core)
LIB40 Routines:
- FORTRAN Operating System (FORSE.)
- Scientific Library, etc.
I/O Buffers
Unused Area
(May be used for I/O buffers if there are many devices)
Symbol Table
Last location of user area

Figure LIB40-1 Core Storage Map of DDT, FORTRAN Main Program, and LIB40

DIAGNOSTIC MESSAGES

Table LIB40-1 FORTRAN Operating System Diagnostic Messages

Message	Meaning
?DEVICE dev: NOT AVAILABLE	FORSE. tried to initialize a device which either does not exist or has been assigned to another job.
?DEVICE NUMBER n IS ILLEGAL	A nonexistent device number was selected.
?DOUBLE PRECISION OVER OR UNDERFLOW	An overflow or underflow error occurred while adding, subtracting, multiplying, or dividing two double-precision numbers.
?END OF FILE ON dev:	A premature end of file has occurred on an input device.
?END OF TAPE ON dev:	The end of tape marker has been sensed during input or output.
?FILE NAME filename.ext NOT ON DEVICE dev:	Filename.ext cannot be found in the directory of the specified device.
?ILLEGAL CHARACTER, x, IN FORMAT	The illegal character x is not valid for a FORMAT statement.
?ILLEGAL CHARACTER, x, IN INPUT STRING	The illegal character, x, is not valid for this type of input.
?ILLEGAL MAGNETIC TAPE OPERATION, TAPE dev:	An attempt was made to skip a record after performing output on a magnetic tape.
?ILLEGAL PHYSICAL RECORD COUNT, TAPE dev:	FORSE. has encountered an inconsistency in the physical record count on a magnetic tape.
?ILLEGAL USER UUO uuu AT USER loc	An illegal user UUO to FORSE. was encountered at location loc.
?INPUT DEVICE ERROR ON dev:	A data transmission error has been detected in the input from a device.
?MORE THAN 15 DEVICES REQUESTED	Too many devices have been requested.
?NAMELIST SYNTAX ERROR	Improper mode of I/O (octal or Hollerith), incorrect variable name.
?NO ROOM FOR FILE filename.ext ON DEVICE dev:	There is no room for the file in the directory of the named device.

Table LIB40-1 (cont.) FORTRAN Operating System Diagnostic Messages

Message	Meaning
programname NOT LOADED	A dummy routine was loaded instead of the real one. Generally, this error occurs when a loaded program is patched to include a call to a library program which was not called by the original program at load time.
?OUTPUT DEVICE ERROR ON dev:	A data transmission error has been detected during output to a device.
?PARITY ERROR ON dev:	A parity error has been detected.
?REREAD EXECUTED BEFORE FIRST READ	A reread was attempted before initializing the first input device.
?TAPE RECORD TOO SHORT ON UNIT n	The data list is too long on a binary tape read operation.
?dev: WRITE PROTECTED	The device is write locked.

NOTE: With the exception of the messages "DOUBLE PRECISION OVER OF UNDERFLOW" and "ILLEGAL USER UUO uuu AT USER loc," all messages are followed by a second message

?LAST FORTRAN I/O AT USER LOC adr

OR

FUNCTION

This area provides storage for items of interest to both the Monitor and the user. The job data area is automatically allocated space just prior to the core area occupied by the program coding. The area occupies 96_{10} (140_8) locations.

JOBDAT exists in binary form in the Systems Library for loading with user programs which refer to JOBDAT locations symbolically. JOBDAT is loaded automatically, if needed, during the Loader's library search.

Table JOBDAT-1 Job Data Area Locations

Name	Relative Location(s)		Description
	Octal	Decimal	
JOBUUO	40	32	User's location 40_8. Used for processing user UUO's (001 through 037).
JOB41	41	33	User's location 41_8. Contains the beginning address of the user's programmed operator service routine.
JOBREL	44	36	Left half: 0 Right half: The highest relative core location available to the user (i.e., the contents of the memory protection register when this user is running).
JOBDDT	74	60	Contains the starting address of DDT. If contents are 0, DDT has not been loaded.
JOBSYM	116	78	Contains a pointer to the symbol table created by Linking Loader. Left half: Negative count of the length of the symbol table. Right half: Lowest register used.
JOBSA	120	80	Left half: First free location in user area (set by Loader). Right half: Starting address of the user's program.
JOBFF	121	81	Left half: 0 Right half: Address of the first free location following the user's program. Set to $C(JOBSA)_{LH}$ by RESET UUO.

Table JOBDAT-1 (Cont) Job Data Area Locations

Name	Relative Location(s)		Description
	Octal	Decimal	
JOBREN	124	84	Set by user and used by REENTER command as an alternate entry point.
JOBAPR	125	85	Left half: 0 Right half: Set by user program to trap address when user is enabled to handle APR traps such as illegal memory, pushdown overflow, arithmetic overflow, and clock. See CALL APRENB UUO.
JOBCNI	126	86	Contains state of APR as stored by CONI APR when a user-enabled APR trap occurs.
JOBTPC	127	87	Monitor stores PC of next instruction to be executed when a user-enabled APR trap occurs.
JOBOPC	130	88	The previous contents of the user's program counter are stored here by Monitor upon execution of a DDT, REENTER, START, or CSTART command.
JOBCHN	131	89	Left half: 0 Right half: Address of first location after first FORTRAN IV Block Data.

NOTE: Only those JOBDAT locations of significant importance to the user are given in this table. JOBDAT locations not listed include those which are used by the Monitor and those which are unused at the present time. User programs should not refer to any locations not listed above since such locations are subject to change without notice.

nonsense

FILE UPDATE GENERATOR (FUDGE2)

FUNCTION

To update files containing one or more relocatable binary programs.

- Permits user to manipulate individual programs within program files

ENVIRONMENT

Monitor	All
Minimum Core	2K
Additional Core	Dynamically allocates its buffers to utilize as much core as is made available.
Equipment Required	Two input devices, one for the master file and one for the transaction file; one output device for the updated file. The input device(s) and output device can be the same device (DSK:). The two input devices can be the same DECtape.

use this space for text

INITIALIZATION

.R FUDGE2↵	Loads the File Update Generator program.
*	FUDGE2 is ready to receive a command.

COMMANDS

General Command Format

```
new-dev:filename.ext←master-dev:filename.ext<prognamel,progname2,..prognamen>,
    transaction-dev:filename.ext<prognamea,prognameb,...prognamez >(commands) ALTMODE
```

new-dev:

The destination device, on which the updated file is written.

DTAn:	(DECtape)
DSK:	(disk)
MTAn:	(magnetic tape)
PTP:	(paper tape punch)

master-dev:

The device containing the file to be updated.

DTAn:	(DECtape)
DSK:	(disk)
MTAn:	(magnetic tape)
PTR:	(paper tape reader)

NOTE: If more than one file is to be transferred from a magnetic tape or paper tape reader, dev: is followed by a colon (:) for each file after the first.

transaction-dev:

The device containing the file of programs to be used in the updating process.

DTAn:	(DECtape)
DSK:	(disk)
MTAn:	(magnetic tape)
PTR:	(paper tape reader)

NOTE: If more than one file is to be transferred from a magnetic tape or paper tape reader, dev: is followed by a colon (:) for each file after the first.

More than one transaction device, with its associated filenames and program names, can be specified in certain instances (see "Switches").

filename.ext (DSK: and DTAn: only)

The filename.ext of the new, updated version of the program file.

The filename.ext of the program file containing the programs to be deleted, replaced, or augmented.

The filename.ext of the program file containing the programs to be used in performing additions or replacements to the master file.

If no .ext is given, .REL is assumed.

<progname,.....>(DSK: and DTAn: only)

Program names must be specified in the same relative order in which they appear in the file.

Program names are grouped together within angle brackets < > and are separated by commas.

If it is desired to append, replace, insert, or extract all programs within a file, only the filename.ext need be specified.

Program names cannot be specified for the output file.

←

The new output file is separated from the master and transaction files by the left arrow symbol (←).

Command Codes

The function to be performed by FUDGE2 is selected by including one of the following command codes at the end of the command string. Command codes are enclosed within parentheses (or preceded by a slash) and one (and only one) must appear in every command string.

Table FUDGE-1 FUDGE2 Command Codes

Command	Meaning
A	Append one or more programs from the transaction file(s) to the master file and write out the new file. The command string is as follows: new-file ← master-file, transaction-file, ... (A) ALTMODE

Table FUDGE-1 (Cont) FUDGE2 Command Codes

Command	Meaning
D	Delete one or more programs from the master file and write out the new file. The files (and programs) to be deleted are listed after master-dev:. The command string is as follows: new-file ← master-file<file(s) to be deleted>(D) ALTMODE
E	Extract the specified files (and programs) from one or more input files and create a new output file. If program names are not specified for a file, the entire file is extracted. The command string is as follows: new-file ← masterfile<file(s) to be extracted>(E) ALTMODE
I	Insert programs from one or more transaction files onto the master file and write out the new file. The programs from the transaction file(s) are inserted immediately before the specified programs on the master file. The command string is as follows: new-file ← master-file<file(s)to be inserted before>, transaction-file(s) (I) ALTMODE
L	List all relocatable programs within a file and print the listing on the output device, which must be either TTY: or LPT: The command string is as follows: listing-device ← file(L) ALTMODE
R	Replace the named program(s) on the master file with the named program(s) from the transaction file and write out the new file. The command string is as follows: new-file ← master-file<file(s) to be replaced>,transaction-file<replacement file(s)> (R) ALTMODE

NOTE: Only one operation can be specified per command string. Thus, to delete a file and replace some other one, two command strings are required.

EXAMPLES

.R FUDGE2↵	
*LPT:←DTA1:LIB40(L)(ALTMODE)↵	List all relocatable programs (.REL) from the file LIB40, located on DTA1 on the line printer.
*DTA2:LIB4AA←DTA1:LIB40<EXP.2>(D)(ALTMODE)↵	Delete the program EXP.2 from the file LIB40 on DTA1; write the new file on DTA2 and call it LIB4AA.REL.
*DSK:LIB4BB~~←A~~←DTA2:LIB4AA<EXP.3>,DTA1:F1↵ <EXP.3A,EXP.3B>,DTA3:EXPNT(R)(ALTMODE)↵ S.M.V.	Replace program EXP.3, located in file LIB4AA on DTA2, with programs EXP.3A and EXP.3B in file F1 on DTA1 and with all the programs in file EXPNT on DTA3; write out the new LIB4AA file on disk and call it LIB4BB.
*PTP:←DSK:LIB4BB,DTA4:SCIENC<COSRTE>/A(ALTMODE)↵	Append the program COSRTE, located in file SCIENC on DTA4, to the file LIB4BB on disk; write out the updated LIB4BB file on the paper tape punch.
*↑C↵	Return to the Monitor.
.KJOB↵	Kill the job, deassign all devices, and release core.
.	

SWITCHES

Switches are used to manipulate file directories and to position magnetic tape. They are either preceded by a slash or enclosed in parentheses and can appear anywhere in the command string.

Table FUDGE-2 FUDGE2 Switch Options

Switch	Meaning
B	Backspace magnetic tape one file.
K	Advance magnetic tape one file.
W	Rewind magnetic tape.
Z	Clear directory of destination device (DTAn: only).

EXAMPLES

```
.R FUDGE2↵
*DTA2:TESTA←MTA1:(WK),MTA2: :(ZA)ALTMODE↵     Clear the directory of DTA2; rewind
                                               MTA1 and advance the tape one file;
                                               append the first two program files
                                               from MTA2 to the second file on MTA1
                                               and write out the resultant file on
                                               disk, calling it TESTA.
*↑C↵                                           Return to the Monitor.
.KJOB↵                                         Kill the job, deassign all devices,
                                               and release core.
.
```

DIAGNOSTIC MESSAGES

Table FUDGE-3 FUDGE2 Diagnostic Messages

Message	Meaning
?CANNOT DO I/O AS REQUESTED	Input cannot be performed on one of the devices specified for input (it is an output only device) or output cannot be performed on the device specified for output.
?DEVICE ERROR ON OUTPUT DEVICE	A write error has occurred on the output file.
?DIRECTORY FULL ON OUTPUT DEVICE	No more files can be added to the file directory on the output device (the directory is full).
?ENTRY BLOCK TOO LARGE, PROGRAM xxxxxx	The entry block of program xxxxxx is too large for the FUDGE2 entry table, which allows for 32 entry names. FUDGE2 can be reassembled with a larger table.
?FUDGE SYNTAX ERROR	The command string is illegal (e.g., the left arrow was omitted, a program name was specified for the output file, or some meaningless command was entered).
?x IS AN ILLEGAL CHARACTER	An illegal character has been encountered in the command string.
?x IS AN ILLEGAL SWITCH	An illegal or otherwise meaningless switch has been encountered in the command string.
?dev NOT AVAILABLE	The device either does not exist or has been assigned to another job.
?NOT ENOUGH ARGUMENTS	An insufficient number of files of one type or another has been specified.
?dev filename.ext progname NOT FOUND	Either the filename.ext or the program name was not found on the device (or in the file) specified. If a program name is printed, this may indicate that the program names in the command string appear in a sequence different from their sequence within the file; thus, the program may actually exist in the named file but was missed because of the incorrectly entered sequence in the command string.
?PROGRAM ERROR WHILE RESETTING MASTER DEVICE	Either FUDGE2 cannot find the master device or cannot find the program name on the master device.

Table FUDGE-3 (cont.) FUDGE2 Diagnostic Messages

Message	Meaning
?TOO MANY FILE NAMES OR PROGRAM NAMES	More than 40 program names or file names were given in a command string. Break the job into several segments and rerun.
?TRANSMISSION ERROR ON INPUT DEVICE dev	A transmission error has occurred while reading data from device dev.
?UNEQUAL NUMBER OF MASTER AND TRANSACTION PROGRAMS	An unequal number of master and transaction programs (or files) has been specified with a Replace request.

DYNAMIC DEBUGGING TECHNIQUE (DDT)

FUNCTION

To provide a powerful, easy-to-use on-line debugging system.

- Monitors the status of a running program
- Enables the user to modify program instructions at any point during run time
- User may stop program at predetermined points (breakpoints)
- Can be used to create and execute a program
- Performs "effective address" searches to obtain all references to a given core location·
- Input/output can be in symbolic, numeric, and ~~Test~~ modes

text S,M,O.

ENVIRONMENT

Monitor	All
Minimum Core	2K
Additional Core	Not used.
Equipment Required	Only the equipment required by the program being debugged, plus a Teletype.

INITIALIZATION

DDT is loaded with the user's program to be debugged during the Linking Loader phase. Loading of DDT is requested by use of the /D switch. Control is transferred to DDT by typing

```
.DDT↵
```

at the monitor level. Thus, a typical initialization sequence might be

```
              .R LOADER 5↵
S.M.U. line * */D dev:filename.ext,......          ?
              *~~DTA5:TEST↵~~ dev:filename.ext
              *(ALTMODE)
              LOADER↵                              Program to be debugged and DDT (along
                                                   with the symbol table) is loaded in core.

              EXIT↵
              ↑C ↵
              .DDT↵                                Transfer control to DDT.
              ↵                                    Two carriage returns indicate that DDT is
              ↵                                    ready to accept commands.
              {look at memory, alter locations, etc.}
              {set breakpoints                      }
              $G↵                                  Start execution of the user's program under
                                                   control of DDT. If the user desires to
                                                   start execution at other than the previously
                                                   defined starting address (JOBSA), he can
                                                   type adr$G↵.
```

NOTE: DDT is initialized to

1. Interpret numbers in octal radix, and
2. Output addresses in the symbolic mode, relative to symbolic labels.

COMMONLY USED COMMANDS

NOTE: In general, commands can be entered before the program execution is started or during a breakpoint halt.

To Set a Breakpoint

Up to eight breakpoints can be set at any one time.

adr$B

Set a breakpoint (the number of the breakpoint is assigned, usually in a sequential manner, by DDT) at adr (symbolic or absolute.) When the breakpoint is reached, the program halts and types out

```
$nB>>adr
```

adr$nB

Set breakpoint #n at adr (symbolic or absolute). When the breakpoint is reached, the program halts and types out

```
$nB>>adr
```

adr (loc)$nB

Set breakpoint #n at adr (symbolic or absolute). When the breakpoint is reached, the program halts and types out

```
$nB>>adr
```

followed by

```
loc/  contents of loc
```

The address loc is now open for modification by the user.

adr (loc)$$nB

Same as above, except that only the contents of loc are typed out and the program proceeds automatically.

NOTES:

1. The $ symbol can be entered with the ALTMODE key.
2. Breakpoints cannot be set on instructions which are: (1) modified by the program; (2) used as data or literals; (3) used as part of an indirect addressing chain; or (4) the user-mode Monitor command INIT.

To Proceed From a Breakpoint Halt

$P — Resume program execution. Halt again next time the breakpoint is reached.

n$P — Resume program execution. Do not halt until the nth time this breakpoint is encountered.

To Remove a Breakpoint

0$nB — Remove breakpoint #n. It can then be assigned to another address, if desired.

$B — Remove all breakpoints.

To Use Symbols (Tags)

progname$:

Before the user can type DDT commands which reference symbols in his program Symbol Table, he must type this entry to select the appropriate Symbol Table.

To Assign a Symbol (Tag) to an Address

adr/ contents symbol:

Assigns the name "symbol" to adr (the last location opened by DDT) and places the appropriate entry in the Symbol Table.

To Assign a Value to a Symbol

value<symbol:

Assigns "value" to the name "symbol" and places the appropriate entry in the Symbol Table. This function is similar to that of the Direct Assignment statement in Macro-10.

To Examine and Modify the Contents of a Program Storage Location

adr/ contents

Type out the contents of adr (symbolic or absolute) and open the location. If adr is a point (./), the contents of the currently open location are typed.

Following the typeout, the contents can be changed by typing the new contents:

adr/ contents new-contents

Several returns can be made following these operations. All returns place the new contents (if any have been typed) into the currently open word and then close the word.

↵

Executes a carriage return, line feed, and waits for the next command.

LINE FEED

Executes a carriage return, line feed; increments the pointer by 1 to the next program storage location; types out the address of the new location and its contents and opens the location.

↑

Executes a carriage return, line feed; decrements the pointer by 1 to the preceding program storage location; types out the address of the new location and its contents and opens the location.

=

Retype the contents of the location last typed out, this time numerically in the current radix mode (normally octal). Used when the symbolic typeout is meaningless, as in the case where numeric data is stored in the location.

adr/ contents (symbolic) = contents (in current radix)

EXAMPLE

.DDT↵	
TEST2$: ↵	Select the Symbol Table of the program named TEST2.[1]
LOOP$1B↵	Set breakpoint #1 at the location LOOP.
START$G↵	Start the program execution at symbolic location START.
$1B>>LOOP⇥ COUNT/⇥ MOVE 1,CTR↵ $P↵	Breakpoint #1 has been encountered. User requests typeout of contents of location COUNT. User decides to proceed until breakpoint is reached again.
$1B>> LOOP⇥ CTR/⇥ (garbage)⇥ = 1734563210 ↵	Breakpoint #1 has again been encountered. User requests typeout of contents of location CTR. Since the contents of CTR as a symbolic instruction are meaningless, the user types = to request retyping of contents in the current radix (octal).
TST/⇥ ADD 4,@NUM(17)⇥ ADD 4,@NUM(16)LINE FEED↵ TST+1/⇥ JRST 4,TEMP ⇥ ↵	User requests typeout of location TST and changes its contents. LINE-FEED typein causes typeout of address and contents of next sequential location.
0$1B⇥ SUBRTE$G↵	Remove breakpoint #1; resume execution at location SUBRTE.
EXIT↵	Program execution is finished; automatic return to the Monitor.
↑C ↵	
.	

[1] The Teletype keys ALTMODE (ALT), PREFIX, or ESCAPE (ESC) are all equivalent to the $ in DDT commands (except when typing in text strings).

SUMMARY OF DDT COMMANDS

Type-Out Modes

To set the type-out mode to:	Type this	Sample Output(s)
Symbolic instructions	$S	ADD 4, TAG+1 ADD 4, 4002
Numeric, in current radix	$C	69. 105
Floating point	$F	0.125E-3
7-bit ASCII text	$T	PQRST
SIXBIT text	$6T	TSRQPO
RADIX50	$5T	4 DDTEND
Halfwords, two addresses	$H	(4002) 4005 4002,, 4ØØ5 (X+1) X+4 X+1,, X+4
Bytes (of n bits each)	$nO	$8O could yield 0,14,237,123,0

Address Modes

To set the address mode for typeout of symbolic instructions and halfwords (see examples above) to

Relative to symbolic address	$R	TAG+1
Absolute numeric address	$A	4005

Radix Change

To change the radix of numeric typeouts to n (for n ≥2), type

$nR	$2R could yield 1101011000000100000000000011100101100

Permanent vs Temporary Modes

To set a temporary type-out or address mode or a temporary radix as shown in the commands above, type	$	$C $10R
To instead set a permanent type-out or address mode or a permanent radix, in the commands above, substitute	$$	$$C $$10R

Permanent vs Temporary Modes (cont)	Type this	Sample Output(s)
To terminate temporary modes and revert to permanent modes, or re-enter DDT, type a carriage return.	↵	
Initial permanent (and temporary) modes are	$$S $$R $$8R	
Examining Storage Words		
To open and examine the contents of any address in current type-out mode	adr/	LOC/ (254020)DDTEND 254020,,DDTEND
To open a word, but inhibit the type out of contents	adr!	LOC!
To open and examine a word as a number in the current radix	adr[	LOC[254020003454 254020,,003454
To open and examine a word as a symbolic instruction	adr]	LOC] JRST @DDTEND
To retype the last quantity typed (particularly used after changing the current type-out mode)	;	$F; #5.4999646E+11 $6T; 5%0 <L
Examining A Related Storage Word		
To close the current open word (making any modification typed in) and to open the following related words, examining them in the current type-out mode:		
To examine adr +1	↓ (line feed)	
To examine adr -1	↑ (or backspace, on the Teletype Model 37)	
To examine the contents of the location specified by the address of the last quantity typed, and to set the location pointer to this address	→| (TAB)	
To examine the contents of address of last quantity typed, but not change the location pointer	\ (backslash)	
To close the currently open word, without opening a new word, and revert to permanent type-out modes.	↵ (carriage return)	
One-Time Only Typeouts		
To repeat the last typeout as a number in the current radix	=	

One-Time Only Typeouts (cont)	Type this	
To repeat the last typeout as a symbolic instruction (the address part is determined by $A or $R)	←	
To type out, in the current type-out mode, the contents of the location specified by the address in the open instruction word, and to open that location, but not move the location pointer.	/	
To type out, as a number, the contents of the location specified by the open instruction word and to open that location, but not move the location pointer.	[	
To type out, as a symbolic instruction, the contents of the location specified by the open instruction word, and to open that word, but not move the location pointer.	]	
Typing In		
Current type-out modes do not affect typing in, instead		
To type in a symbolic instruction	ADD AC1,@DATE(17)	
To type in half words, enclose the left half in parentheses.	(402)403	402,,403
To type in octal values	1234	
To type in a fixed-point decimal integer	99.	
To type in a floating-point number	101.11 77.0E+2	
To type in up to five 7-bit PDP-10 ASCII characters, left justified, delimited by any printing character	"/ABCDE/	(/is delimiter)
To type in one PDP-10 ASCII character, right justified	"A$	($ must be ALTMODE)
To type in up to six SIXBIT characters, left justified, delimited by any printing character	$"ABCDEFGA	(A is delimiter)
To type in one SIXBIT character, right justified	$"Q$	($ must be ALTMODE)
Symbols		Example
To permit reference to local symbols within a program titled "name"	name$:	MAIN.$:
To insert or redefine a symbol in the symbol table and give it the value n	n<symbol:	14<TABL3:

Symbols (cont)	Type This	Example
To insert or redefine a symbol in the symbol table, and give it a value equal to the location pointer (.)	symbol:	SYM:
To delete a symbol from the symbol table	symbol$$K	LPCT$$K
To kill a symbol for typeouts (but still permit it to be used for typing in)	symbol$K	TBIT$$K
To perform $K on the last symbol typed out and then to retype the last quantity	$D	
To declare a symbol whose value is to be defined later	symbol#	JRST AJAX#
To type out a list of all undefined symbols (which were created by #)	?	
Special DDT Symbols		
To represent the address of the location pointer	. (point)	
To represent the last quantity typed	$Q	
To represent the indirect address bit	@	
To represent the address of the search mask	$M	
To represent the address of the saved flags, etc.	$I	
To represent the pointers associated with the nth breakpoint	$nB	
Arithmetic Operators Permitted in Forming Expressions		
Two's complement integer addition	+	
Two's complement integer subtraction	-	
Integer multiplication	*	
Integer division (remainder discarded)	' (apostrophe)	
Hierarchical parentheses	(( ))	A*((B+C))
Field Delimiters in Symbolic Typeins		
To delimit op-code name, type one or more spaces		
To delimit accumulator field	,	
To delimit index register	( )	
To indicate indirect addressing	@	

left,, right half

Breakpoints (cont)	Type this	Example
To set a specific breakpoint n (1≤n≤8)	adr $nB	CAR$8B
To set the next unused breakpoint	adr$B	303$B
To set a breakpoint with automatic proceed	adr$$nB adr$$B	CAR$$8B 303$$B
To set a breakpoint which will automatically open and examine a specified address ,x	adr,, x$nB adr(x)$nB adr(x)$B adr(x)$$nB adr(x)$$B	Z+6,, AC3$5B Z+6(AC3)$5B etc ABLE(AC4)$B Z+6(AC3)$$5B ABLE(AC4)$$B
To remove a specific breakpoint	0$nB	0$8B
To remove all breakpoints	$B	$B
To check the status of breakpoint n	$nB/	
To proceed from a breakpoint	$P	$P
To set the proceed count and proceed	n$P	25$P
To proceed from a breakpoint and thereafter proceed automatically	$$P n$$P	$$P 25$$P
Conditional Breakpoints		
To insert a conditional instruction (inst), or call a conditional routine, when breakpoint n is reached	$nB+1/ 0 For example, $2B+1/ 0	inst CAIE 3,100
If the conditional instruction does not cause a skip, the proceed counter is decremented and checked. If the proceed count ≤0, a break occurs.		
If the conditional instruction or subroutine causes one skip, a break occurs.		
If the conditional instruction or subroutine causes two skips, execution of the program proceeds.		
Starting the Program		
To start at the starting address in JOBSA	$G	$G
To start, or continue, at a specified address	adr $G	LOC$G
To execute an instruction	inst $X	JRST 2,@JOBOPC$X returns to program after ↑C and .DDT command sequence

Searching	Type this	Example
To set a lower limit (a), an upper limit (b), a word to be searched for (c), and search for that word	a<b>$W	200<250>0$W
To set limits and search for a not-word	a<b>c$N	351<721>0$N
To set limits and search for an effective address	a<b>c$E	401<471>LOC+6$E
To examine the mask used in searches (initially contains all ones)	$M/	$M/ -1
To insert another quantity n in the mask	n$M	777000777777$M

Zeroing Memory	
To zero memory, except DDT, locations 20-137, and the symbol table	$$Z
To zero memory locations FIRST through LAST inclusive	FIRST<LAST $$Z

Special Characters Used in DDT Typeouts	Typeout	
Breakpoint stops		
Break caused by conditional break instruction	>	
Break because proceed counter≤0	>>	
Undefined symbol cannot be assembled	U	
Half-word type-outs Left hand is enclosed in parentheses	(401)402	401,,402
Unnormalized floating-point number	#1.234E+27	
To indicate an integer is decimal, the decimal point is printed	$10R	77= 63.
Illegal command	?	
If all eight breakpoints have been assigned	?	
RUBOUT echo	XXX	

Paper Tape Commands (Available only in EDDT	Type this
To punch a RIM10B loader	$L
To punch checksummed data blocks where ADR1 is the first, and ADR2 is the last location of the data (TAPE is ↑R)	ADR1<ADR2 TAPE

Paper Tape Commands (cont)

To punch a one-word block to cause a transfer to adr after the preceding program has been loaded from paper tape	adr$J	
To read (Yank) a tape into core starting at ADR1, up to ADR2	adr1<adr2$Y	feature removed
To verify a tape with core, starting at ADR1, up to ADR2	adr1<adr2$V	" "

EXAMPLES

.DDT↵ ↵ ↵	
PROG1$:↵	Select the appropriate Symbol Table.
4505$3B ↵	Set a conditional breakpoint (breakpoint #3 at location 4505).
$3B+1/⇥ 0⇥ CAIE AC1,100↵	1. Insert a conditional instruction at $3B+1; if the execution of this instruction does not cause a program counter skip, decrement and test the proceed counter ($3B+2); if a skip occurs, halt at breakpoint.
$3B+2/⇥ 0⇥ 200↵ or 200$P↵	2. Set the proceed counter at 200. If execution of instruction at $3B+1 does not cause a program skip, decrement the proceed counter and test for less than or equal to 0; if test succeeds, halt at breakpoint; if test fails, continue execution.
$G↵	Begin execution of program under DDT control.
$3B>>4505↵	Breakpoint #3 occurs; the >> indicates that the instruction at $3B+1 did not cause a skip, but that the proceed counter did reach 0.
CTR$E↵	Search for and type out all instruction words which reference the location CTR.
4517/⇥ SETZM X↵ 4721/⇥ MOVEM 2,CTR↵ 5000/⇥ MOVE 3,@4721↵	 (indirectly addresses CTR through address 4721)

4003,,4502 ~~000000,,000000~~ 0

```
$A$HBUFF1/→| (4003) 4502[ →| (0000) 0000↲
```

Type out the contents of location BUFF1 in half-word, current radix mode; type out the contents of the last address typed (4502).

```
4003<4502>$$Z↲
```

Zero out core from location 4003 through 4502, inclusive.

```
4003/→| 0→| 000067777777LINE-FEED↲
```

Enter the octal value 000067777777 into location 4003: open next location.

```
4004/→| 0 →| -9980.LINE-FEED↲
```

Enter the fixed decimal value -9980 into location 4004; open the next location.

```
4005/→| 0 →|12.34E5 LINE-FEED↲
```

Enter the floating point value 12.34 with an exponent of 5 into location 4005; open the next location.

```
4006/→| 0 →|"%TEXT%↲
```

Place the ASCII characters "TEXT" into location 4006, left justified.

```
0$3B$P↲
```

Remove breakpoint #3 and resume execution.

```
EXIT↲
```

Program execution is finished; automatic return to the Monitor.

```
↑C↲
.
```

use this space & save whole page

DIAGNOSTIC MESSAGES

Table DDT-2 DDT Diagnostic Messages

Message	Meaning
?	An illegal command has been entered or an attempt has been made to select more than eight breakpoints.
U	An undefined symbol which cannot be assembled has been entered.

CROSS-REFERENCE LISTING (CREF)

FUNCTION

To produce a sequence-numbered assembly listing followed by one to three tables, one showing cross references for all operand-type symbols (labels, assignments, etc.), another showing cross references for all user-defined operators (macro calls, OPDEF's etc.), and another (if the proper switch is specified) showing the cross references for all op codes and pseudo-op codes (MOVE,XALL, etc.). S.M.U.

The '#' appears on the definition lines of all symbols

The input to CREF is a modified assembly listing file created during a Macro-10 or Fortran IV S.M.U. assembly when the /C switch is specified in the command string.

- Provides an invaluable aid for program debugging and modification

ENVIRONMENT

Monitor	All
Minimum Core	2K
Additional Core	Takes advantage of any additional core available, as necessary.
Equipment Required	One input device (normally disk) which contains the modified assembly listing file; one output device (normally the line printer) for the listing.

INITIALIZATION

.R CREF↲

Loads the Cross-Reference Listing program into core.

*

The program is ready to receive a command.

COMMANDS

General Command Format

output-dev:← input-dev:filename.ext

output-dev:

The device on which the assembly listing and cross-reference tables are to be printed (LPT: is assumed if device is not specified).

input-dev:

The device on which the modified assembly listing was written during Macro-10 assembly (DSK: is assumed if device is not specified).

filename.ext (DSK: or DTAn: only)

The filename and filename extension of the modified assembly listing file (CREF.~~TMP~~ is assumed if filename.ext is not specified). LST S.M.U.

←

The output device and the input device are separated by the left arrow symbol.

Disk File Command Format

DSK:filename.ext[proj,prog]

[proj,prog]

Project-programmer number assigned to the disk area to be searched for the source file if other than the user's project-programmer number.

EXAMPLES

.R MACRO↲	Loads the Macro-10 Assembler into core.
*PTP:,/C←DTA1:TXCALC↲ THERE ARE NO ERRORS↲ PROGRAM BREAK IS 003771↲ 6K CORE USED↲	Assembles the program S,M,U, TAXCALC from DTA1; writes the object program coding on the paper tape punch; writes a modified assembly listing on DSK: (assumed) and assigns it the filename CREF.~~TMP~~ LST
*↑C↲	Return to the Monitor.
.R CREF↲	Loads CREF into core.
*↲	Selects the default assumptions of: output-dev: LPT: input-dev: DSK: filename.ext CREF.~~TMP~~ LST
*↑C↲	Return to the Monitor.
.KJOB↲	Kill the job, deassign all devices, release core.
.	

SWITCHES

Switches are used to specify such options as:

1. Magnetic tape control, and
2. List selection.

All switches are preceded by a slash (/).

Table CREF-1 CREF Switch Options

Switch	Meaning
A	Advance magnetic tape reel by one file., may be repeated S.M.O.
B	Backspace magnetic tape reel by one file. may be repeated S.Mills
K	Suppress listing of references to basic symbols (labels, assignments, etc.).
M	Suppress listing of references to user-defined operators (macro calls, OPDEF's, etc.).
O	Allow listing of references to machine and pseudo-operation codes (MOVE, XALL, etc.)
S	Suppress program listing (list only the selected tables).
T	Skip to logical end of magnetic tape.
W	Rewind magnetic tape.
Z	Zero the DECtape directory (DECtape must be output only).

R Requests (by typing out RESTART LISTING AT LINE: ...) the line number at which the listing is to restart. (LPT paper jams etc)

EXAMPLES

.R CREF↵	Loads CREF into core.
*/M←MTA1:/W↵	Rewind MTA1 and process the first file, listing only the cross references for operand-type symbols (labels, assignments, etc.).
*DTA5:SAVE1/Z← ↵	Process the file named CREF.~~TMP~~ LST in the user's area of disk; write the program listing and operand-type cross references on DTA5 and call the file SAVE1.
*↑C↵	Return to Monitor.
.KJOB↵	Kill the job, deassign all devices, and release core.
.	

DIAGNOSTIC MESSAGES

Table CREF-2 CREF Diagnostic Messages

Message	Meaning
?dev NOT AVAILABLE	Device is assigned to another job.
?CANNOT ENTER FILE fnme.ext	DTA or DSK directory is full; file cannot be entered.
?CANNOT FIND FILE fnme.ext	The file cannot be found on the device specified.
?COMMAND ERROR	Error in last command string entered.
?DATA ERROR DEVICE dev:	Read or write error.
?IMPROPER INPUT DATA	Input data not in CREF format.
? INPUT ERROR ON DEVICE dev:	Read error has occurred on the device.
? INSUFFICIENT CORE	Additional core is required for execution but none is available from Monitor.

?ERROR READING COMMAND FILE

Disk data error while reading the ###CRE#.TMP file S.M.O.

GLOBAL SYMBOL CROSS REFERENCE LIST (GLOB)

FUNCTION

To read multiple binary program files produced by Macro ~~and~~ F40 and generate an alphabetic cross-referenced list of all global symbols encountered.

S.M.U. work!

ENVIRONMENT

Monitor	All
Minimum Core	2K
Additional Core	Requests additional core from the monitor as required.
Equipment Required	An input device for each binary file to be scanned for global symbols and one or more listing devices for output.

INITIALIZATION

```
.R GLOB↵
```

Loads the Global Symbol Cross-Reference Listing program.

```
*
```

The program is ready to receive a command.

COMMANDS

Input Command

```
dev:filename.ext,....filename.ext,dev:filename.ext,....filename.ext,..↵
```

dev:

The device(s) containing the binary program files to be scanned.

MTAn:	(magnetic tape)
DTAn:	(DECtape)
DSK:	(disk)
PTR:	(paper tape reader)

filename.ext (DSK: and DTAn: only)

The filename and filename extension of each binary program which resides on either disk or DECtape.

Output Command

```
dev:← (ALTMODE)
```

dev:

The device on which the global symbol listing is to be printed.

LPT:	(line printer)
TTY:	(Teletype)

Other output devices can be specified if desired.

More than one output command can be given if it is desired to produce several types of listings on several different devices. Each new output command is typed after the previous request has been completed.

EXAMPLES

```
.R GLOB↵
*DSK:F1,F2,DTA3:CALC1,CALC5↵
```

The binary program files to be scanned are F1 and F2 on DSK, and CALC1 and CALC5 on DTA3.

```
*LPT:←ALTMODE↵
```

All global symbols in these programs are to be listed on the printer. Printed with each symbol are its value, the name of the program in which it was defined, and the names of all the programs in which it was referenced (i.e., declared external).

```
*↑C↵
```

Return to the Monitor.

```
.KJOB↵
```

Kill the job, deassign all devices, and release core.

SWITCHES

The switches available in Glob are used to determine the types of global symbols to be listed on each of the specified output devices. If no switches are typed, all global symbols are listed.

All switches are either preceded by a slash or enclosed in parentheses and can appear anywhere in the output command string. However, only the most recently specified switch is in effect at any given time.

Table GLOB-1 GLOB Switch Options

Switch	Meaning
A	All global symbols are to be listed (assumed if no switch is given).
E	List erroneous (multiply defined or undefined) symbols only.
F	List fixed (nonrelocatable) symbols only.
N	List only those symbols which are never referred to.
R	List relocatable symbols only.
S	List multiply specified (i.e., symbols defined in more than one program, but with non-conflicting values) only.
X	Omit printing of listing title when output is other than TTY. Include printing of listing title when output is TTY. NOTE: Normally, the title is printed on all devices except the Teletype.

EXAMPLES

```
.R GLOB↵
*DTA1:TEST1.REL,SUBRTE,DSK:ARITH1,↵
*SCIENC,RETEST↵
```

The binary programs to be scanned are files TEST1.REL and SUBRTE on DTA1, and ARITH1, SCIENC, and RETEST on disk.

```
*LPT:←/RALTMODE↵
```

List only relocatable symbols on the printer.

```
*TTY:←/EALTMODE↵
```

Printer listing is completed. Enter command to print all erroneous symbols on the Teletype.

```
U EXTSYM                                SUBRTE
```

("U" = Undefined; "EXTSYM" is the undefined symbol; "SUBRTE" is the program in which EXTSYM appears.)

```
*↑C↵
```

Return to the Monitor.

```
.KJOB↵
```

Kill the job, deassign all devices, and release core.

```
.
```

DIAGNOSTIC MESSAGES

Table GLOB-2 GLOB Diagnostic Messages

Message	Meaning
?COMMAND SYNTAX ERROR	An illegal command string has been entered.
?DESTINATION DEVICE ERROR	An I/O error has occurred on the output device.
?DIRECTORY FULL	No more files can be added to the directory of the output device.
?dev NOT AVAILABLE	The device either does not exist or has been assigned to another job.
?filename.ext NOT FOUND	The filename.ext cannot be found in the directory of the device specified.
?TABLE OVERFLOW - CORE UUO FAILED TRYING TO EXPAND TO xxx	GLOB requested additional core from the Monitor, but none was available.

Table GLOB-3 GLOB Error Flags

Flag	Meaning
M	Multiply defined symbol (all values are shown).
N	Never referred to (i.e., was not declared external in any of the binary programs).
S	Multiply specified symbol (i.e., defined in more than one program, but with non-conflicting values). In the listing, the name of the first program in which the symbol was found is followed by a plus sign.
U	Undefined symbol.

#10-97
Vol #1#8

Systems Users Guide (DEC-1Ø-NGCA-D)

PIP-6 In the switch column add note 2 to the Y switch (Y^2) and at the bottom of the page add the following:

2. This is an optional switch obtained by setting RIMSW=1 at assembly time (see CUSP=3).

CUSP-3 Replace section a. with the following:

a. PIP: Assemble source file PIP.MAC.

This file has 5 conditional assembly switches (WCH,DISK3Ø,BLOCØ,RIMSW,CCLSW) and 1 conditional parameter (SYSPP).

WCH=Ø New (I.E., PDP-10) Format DECtapes are assumed.
WCH=1 Old (I.E., PDP-6) Format DECtapes are assumed.

DISK3Ø=Ø PIP will run with 1Ø/4Ø and 1Ø/5Ø Monitors.
DISK3Ø=1 PIP will run with 1Ø/3Ø Disk Monitor.

BLOCØ=Ø BLOCK Ø (DECtape) can be copied.
BLOCØ=1 BLOCK Ø can not be copied.

RIMSW=Ø /Y switch option unavailable.
RIMSW=1 /Y switch available.

CCLSW=Ø PIP will not process CCL commands
CCLSW=1 PIP will execute CCL commands from Disk.

SYSPP:XWD PROJ#,PROG# If defined as shown, PIP will use this P,P#(I.E., 5,6) as the system P,P#. If undefined PIP uses 1,1 as the System P,P#.

Due to the number of conditional assembly switches, a "normal" configuration is setup automatically at assembly time. It is as follows:

WCH=Ø
DISK3Ø=Ø
BLOCØ=Ø
RIMSW=Ø
CCLSW=1

(Continued on following page)

#10-97
Vol #1#8

For a "normal" configuration do a standard assembly then load and save on your CUSP. (See below for example of loading and saving.)
For other than a "normal" configuration follow the example below.
To set a conditional switch, assemble as follows:

```
.R MACRO↵
*DTAn:PIP,←TTY:,DTAm:PIP.MAC↵
WCH=1↵
↑Z
END OF PASS ONE↵
WCH=1↵
↑Z
NO ERRORS DETECTED↵
*↑C
```

Then load and save as follows:

```
.R LOADER↵
*DTAn:PIP (ALTMODE)↵
LOADER↵
EXIT↵
↑C↵
.SAVE DTAn:PIP↵
```

PERIPHERAL INTERCHANGE PROGRAM (PIP)

FUNCTION

To transfer data files from any standard I/O device to any other standard I/O device; additionally, to perform simple editing and magnetic tape control functions. PIP1, a compact version of PIP, performs a subset of PIP functions.

- Handles all data formats
- Eliminates the need for a satellite computer to handle off-line data conversions

ENVIRONMENT

	PIP	PIP1
Monitor	All	All
Minimum Core	3K	1K
Additional Core	1K if disk is one of the I/O devices; any core above that required is used for extra I/O buffers.	Any core greater than 1K is used for extra input buffers.
Equipment Handled	DECtape, disk, magnetic tape, paper tape reader, paper tape punch, card reader, line printer, and Teletype.	

INITIALIZATION

.R PIP↵ or .R PIP1↵ — Loads PIP (or PIP1) into core.

* — PIP is ready to receive a command; an asterisk is typed after each requested action has been completed.

COMMANDS

General Command Format

destination-dev:filename.ext← source-dev:filename.ext, . . . source-n↵

destination-dev:
source-dev:

The destination device, to which the data is to be transferred; the source device(s), from which the data is to be read

DTAn:	(DECtape)[1]
PTR:	(paper tape reader)
PTP:	(paper tape punch)
DSK:	(disk)
CDR:	(card reader)
MTAn:	(magnetic tape)
LPT:	(line printer)
TTY: or TTYn:	(Teletype)

If more than one file is to be transferred from a magnetic tape, card reader, Teletype, or paper tape reader, dev: is followed by a comma for each file after the first; these devices can also be followed by * or *.* to indicate all files are to be transferred.

filename.ext (DSK: and DTAn: only)

The filename and filename extension to be assigned to the file on the destination device; the filename and filename extension of the file(s) to be read from the source device.

An asterisk can be used for source files as follows.

filename.* – Transfer all files having the specified filename.

*.ext – Transfer all files having the specified extension.

[1] If logical device SYS (the CUSP device) is a DECtape, it must not be modified using the /R or /D switches or any other request requiring it to be initialized for input and output at the same time.

. - Transfer all files.

- Transfer all files with null extensions.

← The destination descriptors and the source descriptors are separated by the left arrow symbol (←).

Disk File Command Format

DSK:filename.ext[proj,prog] <protection>

[proj,prog] Project-programmer number assigned to the disk area to be used, if other than the user's project-programmer number.

<protection> Protection value to be assigned to the destination file. If omitted, the standard protection is assigned.[1]

Standard Assumptions

Unless otherwise changed by switches, all files which are on directory devices and which have a filename extension of .REL, .SAV, .DMP, or .CHN are copied in binary; all other files are assumed to be in ASCII line mode. Magnetic tape files, unless otherwise changed by switches, are read in odd parity and written in odd parity at 556 bpi.

- installation standard?

[1] Standard protection (055) designates that the owner is permitted to read or write, or change the protection of, the file while others are permitted only to read the file.

EXAMPLES

```
.R PIP↵
*LPT:←DTA1:FILE1↵
```

Transfer the file named FILE1 from DTA1 to the line printer.

```
*LPT:← DTA1:*↵
```

Transfer all files with null extensions from DTA1 to the line printer.

```
*DTA2:FILE2 ← DTA1:FILE1.TMP↵
```

Transfer the file named FILE1.TMP to DTA2 and give it the name FILE2.

```
*DTA2:FILE3 ← DTA1:FILE1,FILE2↵
```

Transfer the files named FILE1 and FILE2 from DTA1 to DTA2, combining them as one file under the name FILE3.

```
*DTA2:FILE3 ←DTA1:FILE1,DTA3:FILE2↵
```

Transfer the file named FILE1 from DTA1 and the file named FILE2 from DTA3 to DTA2, combining them as one file under the name FILE3.

```
*DSK:FILE1 ← MTA1:↵
```

Transfer the next file from the present position of MTA1 to the user's area on the disk, call it FILE1, and assign the standard protection of 055.

```
*DSK:FILE1<177>←MTA1:*↵
```

Transfer all files from MTA1 (starting at the current position of the read head) to the user's area on the disk, combining them into one file called FILE1, and assign protection 177.

```
*DSK:FILE1[1,3]← MTA1:,↵
```

Transfer the next two files from the present position of MTA1 to area 1,3 on the disk, combining them into one file called FILE1, and assign the standard protection (055).

```
*PTP: ← PTR:,,,,↵
```

Transfer five files from the paper tape reader to one file on the paper tape punch.

```
*↑C↵
```

Return to Monitor.

```
.KJOB↵
```

Kill the job, deassign all devices, and release core.

```
.
```

SWITCHES

Nonmagnetic-tape switches, when used, are preceded by a slash (if more than one is specified, they may be enclosed by parentheses instead) and may appear anywhere in the command string; however, if the command string contains commas, the switches must be specified prior to the first comma.

Magnetic tape switches are enclosed by parentheses and must appear immediately following the device or file to which they refer.

Switches are used to specify

1. Particular files for transferral or deletion;
2. Editing;
3. Mode of transfer;
4. Directory manipulation (DECtape and DSK); and
5. Magnetic tape control.

A listing of PIP switches can be obtained by typing

```
*output-dev:/Q←↵
```

where output-dev: may be either LPT: or TTY:

Table PIP-1 PIP Switch Options

Switch	Meaning
A[1]	Line blocking
B[1]	Process file in binary mode.
C	Suppress trailing spaces and convert multiple spaces to tabs.
D	Delete the file.
E	Treat (card) columns 73 through 80 as spaces.
F	List disk directory (filenames and extensions only).
G	Ignore I/O errors.
H	Process file in image binary mode.
I	Process file in image mode.
L	List the directory (DSK: or DTAn: only)
NOTE: 1. Available for use in PIP1.	

Table PIP-1 PIP Switch Options (Cont)

Switch	Meaning
M	Magnetic tape switches. A string of one or more magnetic tape switches begins with an M and is enclosed in parentheses. #nA Advance the tape n files. #nD Advance n Records E Even parity. #nB Backspace the tape n Files. #nP Backsp n Records F Mark EOF. 2 200 bpi density. T Skip to logical end of tape. 5 556 bpi density. 8 800 bpi density. U Rewind tape and unload. A Advance tape one file. W[1] Rewind the tape. B Backspace tape one file. NOTE: MTA switches always apply to the device or file immediately preceding the switches.
N[1]	Delete the sequence numbers from a file.
O	Same as /S switch except sequence numbers are incremented by 1.
P	FORTRAN output file format assumed as input. Convert format control characters for line printer listing.
Q	Print this list of switches.
R	Rename the file.
S[1]	Add sequence numbers to the file or resequence a file already containing sequence numbers; sequence numbers are incremented by 10.
T	Suppress trailing spaces.
U	Copy blocks 0, 1 and 2 of a DTA file. Commonly used to transfer TENDMP.
V	Count unmatched angle brackets < >.
X	Copy specified files only.
Y	Perform a RIM DTA to PTP conversion. Source extension: Destination format: .RTB RIM Loader, RIM10B File, XFERWD .SAV RIM10B File only. .RMT RIM10
Z[1]	Zero out the directory (DTAn: or DSK: only).

NOTE: 1. Available for use in PIP1.

EXAMPLES

.R PIP↲	
DSK:/X ←DTA1:.*↲	Transfer all files from DTA1 to DSK, keeping them separate and retaining their filenames.
DSK:(DX) ← DTA1:FILE1,.REL↲	Transfer all files, except FILE1 and any files with the extension. .REL, from DTA1 to DSK, keeping them separate and retaining their filenames.
*MTA2:/S← CDR:↲	Transfer a file from the card reader to MTA2 and add sequence numbers.
*LPT:/P ←DTA1:FILE1↲	Take FILE1 (a FORTRAN output print file), interpret the carriage control characters, and print it.
*DTA2:FILE1/I← PTR:↲	Initialize both DTA2 and the paper tape reader in image mode and transfer one file from the paper tape reader to DTA2, calling it FILE1.
*TTY:/L ←DTA1:↲	
nnnn FREE BLOCKS LEFT↲ filename.ext creation date↲ . . .	List the directory of DTA1 on the Teletype.
*DTA1:/Z←↲	Zero the directory of DTA1.
*MTA2:(M8E) ←MTA1:(ME8)↲	Transfer a file from MTA1 to MTA2 in 800 bpi, even parity mode.
*MTA2:(MW) ←↲	Rewind MTA2.
*LPT:← MTA1:(M2W),(MA),,↲	Set MTA1 to 200 bpi, odd parity, rewind the tape, and transfer the first, third, fourth, and fifth files to the printer.
*MTA1:(M#4A)← CDR:↲	Advance MTA1 four files before transferring a file from the card reader.
*↑C↲	Return to the Monitor.
.KJOB↲	Kill the job.
.	

DIAGNOSTIC MESSAGES

Table PIP-2 PIP Diagnostic Messages

Message[1]	Type[2]	Meaning
?4K NEEDED	S	4K is not currently available but is needed when a disk is present in the system.
?DECTAPE I/O ONLY	S	I/O device for copy block 0 (/U) must be a DECtape.
?DEVICE dev:DOES NOT EXIST	I/O	Either device name has been misspelled or there is no such device.
?DEVICE dev:NOT AVAILABLE	I/O	The device has been assigned to another job.
?DIRECTORY FULL	FR	There is no room for an entry in a DEC-tape directory.
DISK DIRECTORY READ[3]	I/O	Followed by a second message (see Table PIP-3).
?DISK OR DECTAPE INPUT REQUIRED	I/O	This command requires a directory device for input.
?DTA TO PTP ONLY	RIM	DTA input and PTP output must be specified for /Y.
?FILE filename.ext ILLEGAL EXTENSION	RIM	Extension for /Y request must be .RMT, .RTB, or .SAV.
?FILE filename.ext ILLEGAL FORMAT	RIM	1. Zero-length file; or 2. Requisite job data info not available; or 3. Block overlaps previous block (RIM 10) or 4. EOF found when data was expected, or 5. A pointer word was expected but not found in the source file.
FILE PROTECTION FAILURE DURING (/X, /Z, /D) REQUEST	S	Each file requested does exist, but one or more was unavailable for processing. This message is never fatal.

NOTES: 1. All fatal diagnostic messages are preceded by a question mark (?).

2. Type of message:

C	Command string error
FR	File reference error
I/O	I/O error
RIM	Read-In-Mode specification error.
S	Other types of errors

3. These messages are nonfatal if the /G switch is used; otherwise, they are fatal and are preceded by a ?.

Table PIP-2 (Cont) PIP Diagnostic Messages

Message[1]	Type[2]	Meaning
?filename.ext FILE WAS BEING MODIFIED	FR	Disk file named is currently being processed by another job.
?filename.ext FILE WAS NOT FOUND	FR	Filename.ext not found during LOOKUP.
?filename.ext ILLEGAL FILE NAME	FR	Indicates that 1. No filename was specified for DTA output file; or 2. A reject occurred on a /R request for disk file; or 3. Illegal filename was specified for a /R request on DTA.
?filename.ext INCORRECT PROJECT-PROGRAMMER NUMBER	FR	The project-programmer number specified for a DSK file is incorrect.
INPUT DEVICE dev: FILE filename.ext[3]	I/O	Followed by a second message (see Table PIP-3).
?LINE TOO LONG	S	A line >140 characters was detected in the source file
?LOAD POINT BEFORE END OF (MB) or (MP) REQUEST	S	Load point on a magnetic tape file has been reached before the tape has been backspaced the number of files specified in (M$^{\#}$nB). or ~~#~~ (M#nP)
?NO BLOCK 0 COPY	C	/U given but PIP assembled without provision for this.
?NO FILE NAMED filename.ext	FR	No such file found during PIP directory search.
?NO FILE NAMED QPIP	S	The data file for the /Q switch is not available.
OUTPUT DEVICE dev: FILE filename.ext	I/O	Followed by a second message (see Table PIP-3.

NOTES:

1. All fatal diagnostic messages are preceded by a question mark (?).
2. Type of message:

C	Command string error
FR	File reference error
I/O	I/O error
RIM	Read-In-Mode specification error.
S	Other types of errors

3. These messages are nonfatal if the /G switch is used; otherwise, they are fatal and are preceded by a ?.

Table PIP-2 (Cont) PIP Diagnostic Messages

Message[1]	Type[2]	Meaning
?PIP COMMAND ERROR	C	1. Illegal format for command string; or 2. Nonexistent switch requested; or 3. Filename.ext other than * (or *.*) requested for a nondirectory device; or 4. The illegal switch combination RX.
?filename.ext PROTECTION FAILURE	FR	Same as "FILE PROTECTION FAILURE DURING ..." message except that the processing halts.
?TERMINATE /X MAX OF 999 FILES PROCESSED	S	The /X switch specified for nondirectory device source files has processed the maximum number of files (999).
?TOO MANY REQUESTS FOR dev:	C	Conflicting parity and/or density requests have been given for a magnetic tape.
?TRY PIP	C	During a PIP1 run, a switch or function which is not present in PIP1 has been requested.

NOTES:

1. All fatal diagnostic messages are preceded by a question mark (?).
2. Type of message:

C	Command string error
FR	File reference error
I/O	I/O error
RIM	Read-In-Mode specification error.
S	Other types of errors

3. These messages are nonfatal if the /G switch is used; otherwise, they are fatal and are preceded by a ?.

Table PIP-3 Secondary PIP I/O Diagnostic Messages

Message	Device	Meaning
BINARY DATA INCOMPLETE[1]	PTR	Length of block disagrees with word count.
BLOCK TOO LARGE	DTA	DTA link number>1101_8.
CHECKSUM OR PARITY ERROR[1]	All	Read or write error.
INPUT BUFFER OVERFLOW[1]	All except DTA	Block too large for buffer.
DEVICE ERROR[1]	All	The data control unit has detected the loss of data.

NOTE: 1. These error conditions are nonfatal if the /G switch has been specified.

Table PIP-3 (Cont)   Secondary PIP I/O Diagnostic Messages

Message	Device	Meaning
PHYSICAL EOT[1]	MTA	The end of tape has been reached.
WRITE (LOCK) ERROR	DTA, DSK, MTA,	Attempt has been made to write on a write-locked file.
7-9 PUNCH MISSING[1]	CDR	Binary card lacks 7-9 punch.

NOTE: 1 These error conditions are nonfatal if the /G switch has been specified.

Foo

1K Version of PIP (PIP1)

Limitations

1. Z and MW requests ignore all source devices.
2. B switch included since REL, SAV, DMP, and CHN files are not automatically copied in 36-bit bytes.
3. Error messages assume all I/O devices are DECtape.
4. Neither project-programmer numbers nor protection can be specified for disk files.
5. The * cannot be used for filenames or extensions.
6. SAV files cannot be successfully copied with PIP1.

DECTAPE FORMAT CONVERTER (CONVRT)

FUNCTION

To convert DECtapes from the old format to the new format (or vice versa).[1]

The new format

- Increases DECtape processing speed by allocating blocks in a nonconsecutive manner so that a continuous read is possible
- Allows for file deletion to free assigned blocks for use by subsequently written files

ENVIRONMENT

Monitor	All
Minimum Core	3K
Additional Core	Not used.
Equipment Required	One or more DECtape units for input: one DECtape unit for output.

[1] Note: CONVRT runs on the PDP-6 or PDP-10. However, it does not use the LOOKUP or ENTER programmed operators or do standard I/O. As a result, the Monitor is not aware of the directory of the output tape (or the input tape) and the tape must be assigned appropriately if it is to be used on the machine on which it is converted.

INITIALIZATION

.R CONVRT↲ — Loads the DECtape Format Converter program into core.

* — The program is ready for the first command.

COMMANDS

General Command Format

```
DTAn:←DTAx:,DTAy:.....DTAz: ALTMODE
```

DTAn: — The DECtape on which the converted output is to be written. Output is assumed to be in new format.

DTAx:....DTAz: — The input tape(s) containing the files to be converted; input is assumed to be in old format.

← — The output DECtape is separated from the input DECtape(s) by the left arrow symbol.

EXAMPLES

```
.R CONVRT↵
*DTA2:←DTA1: ALTMODE ↵                Convert the files on DTA1 (old format)
                                      to new format and add them to DTA2.
DTA1 FINISHED . . .↵
*DTA3:← DTA4:,DTA5: ALTMODE ↵         Convert the files on DTA4 and DTA5
DTA4 FINISHED. . . WAIT↵              (both old format) to new format and
                                      add them to DTA3.
DTA5 FINISHED . . .↵
*↑C ↵                                 Return to the Monitor.
.KJOB↵                                Kill the job, deassign all devices,
                                      and release core.

.
```

SWITCHES

Switches are used to specify such options as:

1. The format of a tape,
2. The specific files to be copied,
3. Directory handling, and
4. Processing continuation.

not otherwise mentioned?? format

All switches can either be preceded by a slash or enclosed within parentheses. ~~Switches~~ normally apply only to the device with which they appear and must be specified before any filename.ext given for that device.

Table CONVRT-1. CONVRT Switch Options

Switch	Meaning
C	Copy the named files only.
D	Copy other than the named files.
G	Continue processing. Used only after the operator has mounted a new output reel following the MOUNT NEW OUTPUT TAPE message.
L	List the directory (source tapes only). See format of list shown below.
N	Tape is in new format (new format is assumed for the output tape unless otherwise specified).
O	Tape is in old format (old format is assumed for all source tapes unless otherwise specified).
Z	Zero out the tape directory (output tape only).

Format of DECtape Directory Listing (/L Switch)

If DECtape is in old format:

filename	ext	date	1st block	Dump mode files command list word
CONBSV	DMP	2-SEP-67	2	(-n,y in decimal) -5060,59
CONB		2-SEP-67	108	(last block in use) 207

If DECtape is in <u>new</u> format:

				Dump mode files
				# of 1K blocks
filename	ext	date	# of blocks in file	used
CONBSV	SAV	2-SEP-67	106	5
CONB		2-SEP-67	100	

NOTE: When requesting a listing of a directory from a new format DECtape, the user must include the /N switch on the source side; otherwise, an old format DECtape is assumed.

```
*LPT:/L←DTA1:/N  (ALTMODE)
```

EXAMPLES

```
.R CONVRT↵
*DTA5:/Z←DTA1:(C)JOB1,JOB4,JOB6,DTA2:(D)↵
JOB.REL ALTMODE ↵
```

Perform the following steps:

1. Zero out the directory of DTA5.
2. From DTA1, convert files JOB1, JOB4, and JOB6 (all in old format) and write them on DTA5.
3. From DTA2, convert all files <u>except</u> JOB.REL and write them on DTA5.

```
DTA1 FINISHED . . . WAIT↵

MOUNT NEW OUTPUT TAPE↵
/Z/G ALTMODE ↵
```

The reel on DTA5 is full; mount new output reel and type /Z/G to zero the directory of the new output tape and to continue processing.

```
DTA2 FINISHED . . .↵
*↑C ↵
```

Return to the Monitor.

```
.KJOB↵
```

Kill the job, deassign all devices, release core.

```
.
```

DIAGNOSTIC MESSAGES

Table CONVRT-2 CONVRT Diagnostic Messages

Message	Meaning
?filename.ext DIRECTORY ENTRY INCONSISTENT	1. An old format file has word four of the directory entry greater than zero and the extension is neither CHN nor DMP. 2. An old format file has the extension SAV and word four of the directory entry is zero.
?DTAn DATA ERROR	An error has occurred while reading or writing a file. The file is deleted from the directory; the directory is written on the output tape before this message is typed.
?DTAn DEVICE NOT AVAILABLE	The DTAn specified has been assigned to another job.
?DTAn DIRECTORY WRITE DATA ERROR	A parity or data error has occurred while writing the directory on the output tape.
DTAn FINISHED . . . (WAIT)	Conversion has been completed for all files on the specified input tape. If the word "WAIT" follows the message, more input files are to be processed; if it does not follow, the entire conversion process is complete and the output reel can be dismounted.
?DTAn NO DIR	A directory block is not recognizable.
?DTAn WRITE (LOCK) ERROR	A write error has occurred on the output tape; the tape is probably write locked.
?filename.ext FILE NOT FOUND	A file specified after a /C or /D switch cannot be found on the input tape. No conversion has been performed for this tape and only those files mentioned prior to this one have been checked. The run is terminated.
?ILLEGAL COMMAND	An unrecognizable, incomplete, or illegal command string has been typed. The run is terminated.
?ILLEGAL SWITCH	An unrecognizable or illegal switch has been typed.
?IMPOSS. SAVED FILE FORMAT	This message should never occur. However, during a new-to-old format conversion, this typeout indicates one of the following conditions. 1. Overlapping blocks occurred. 2. There are more words in the file than are indicated by the number of 1K blocks needed to load the file.

Table CONVRT-2 (Cont) CONVRT Diagnostic Messages

Message	Meaning
?IMPOSSIBLE CONDITION FOUND	This typeout should never occur. If it does appear, it indicates one of the following conditions. 1. The fourth word in the directory of a saved file on an old format DECtape is not negative. 2. There are no blocks associated with a filename.ext in a new format tape directory. 3. The word count in a block of a new format saved file is greater than 128.
?MORE CORE NEEDED	Insufficient core is available for processing the last command string.
BELL MOUNT NEW OUTPUT TAPE	The current output tape if full. Dismount the tape, mount a new tape, and type /G to continue.
BELL ? MOUNT NEW OUTPUT TAPE	An incorrect response to the previous typeout of this message was entered.
?dev: NOT A DECTAPE	All selected devices must be DECtapes.

CODE TRANSLATOR (CODE)

This section was not available at publication time, but will soon be available as an insert.

If you do not receive this insert, please write to:

DEC Program Library
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

SOURCE COMPARE (SRCCOM)

(Version 010)

FUNCTION

To compare, line by line, two versions of a source file coded as lines of ASCII characters and to output any differences.

ENVIRONMENT

Monitor	All
Minimum Core	2K
Additional Core	The minimum core allows for comparing files with minimal differences. SRCCOM automatically requests more core from the Monitor when it needs it. Major differences can usually be handled in 3K, but for comparing two completely different files, enough core is required to store all of both files simultaneously.
Equipment Required	User Teletype for control: Two input devices for the two files to be compared; one output device for listing the differences.

INITIALIZATION

.R SRCCOM ↲	Loads the Source Compare routine.
*	Source Compare is ready to receive a command.

COMMANDS

General Command Format

```
list-dev:filename.ext←input1-dev:filename.ext,input2-dev:filename.ext ↲
```

list-dev: The device on which the differences are to be listed.

LPT:	(line printer)	(Any device that can output ASCII characters)
TTY:	(Teletype)	
MTAn:	(magnetic tape)	
DTAn:	(DECtape)	
DSK:	(disk)	

input1-dev:
input2-dev:

The devices on which the two source files to be compared are located.

MTAn:	(magnetic tape)	(Any device
DTAn:	(DECtape)	that can input
DSK:	(disk)	ASCII characters)
PTR:	(paper tape reader)	

filename.ext (DSK: and DTAn: only)

The filename and extension of either of the input source files.

The filename and extension to be assigned to the output list file. (SRCCOM.LST is assumed if no filename is specified.)

←

The output device is separated from the input source file devices by the left arrow symbol.

Default Conditions

TTY: is assumed as the output device if no other device is specified.

DSK: is assumed as input device #1 if no other device is specified.

Input device #1 is assumed as input device #2 if no other device is specified.

SWITCHES

Switches are used to specify the manner in which the comparison is to be done.

All switches consist of a single character preceded by a slash (/), anywhere in the command string.

Table SRCCOM-1
Source Compare Switches

Switch	Meaning
/B	Enables the comparing of Blank lines. Normally blank lines are completely ignored.
/C	Comments (all text on a line beginning with a semicolon) are ignored. /C will not cause a line consisting entirely of a comment to become a blank line which will be ignored. /S is also implied.
/S	Spacing (spaces and tabs) is ignored.
/n	(n = 1, 2,...,9) A "match" consists of n lines. (n is normally 3.) n successive lines must be found identical in the two input files for a "match" in the two files to be found. When a match is found, all differences between the current match and the previous match are listed. The first line of the match is also listed in order to make the location in the file easier to find.

EXAMPLES

.R SRCCOM ↲	
*LPT:←DTA2:SOURCE.001,DTA3:SOURCE.002 ↲	Compare the source file SOURCE.001 on DTA2 with the source file SOURCE.002 on DTA3 and list all differences on the line printer.
*LPT:←DSK:TRY1,DSK:TRY2 ↲	Compare the two files, TRY1 and TRY2, both of which are on the disk, and list the differences on the printer.
*↑C ↲	Return to the Monitor.
.KJOB ↲	Kill the job, deassign all devices, and release core.

Table SRCCOM-2
Example of Source Compare Output

SRCCOM output	File 1 input	File 2 input
FILE 1) FILE #1 FILE 2) FILE #2 1)1↙ FILE #1 * 1) A **** 2)1↙ FILE #2 * 2) A *************** 1)1↙ D 1) E 1) F 1) G **** 2)1↙ G *************** 1)1↙ K 1) L 1) M 1)2↙ N **** 2)1↙ 1 2) 2 2) 3 2)2↙ N *************** 1)2↙ W **** 2)2↙ 4 2) 5 2) W ***************	page 1 FILE #1 A B C D E F G H I J K L M page 2 N O P Q R S T U V W X Y Z	page 1 FILE #2 A B C G H I J 1 2 3 page 2 N O P Q R S T U V 4 5 W X Y Z

The numbers with the ↙ arrows in the SRCCOM listing are page numbers referring to the input files.

*A line identical to both input files is listed to help find the location of the differences within the two files.

DIAGNOSTIC MESSAGES

Table SRCCOM-3 Source Compare Diagnostic Messages

Message	Meaning
?2K CORE NEEDED AND NOT AVAILABLE	SRCCOM needs 2K in order to initialize IO devices and the core is not available from the Monitor
?BUFFER CAPACITY EXCEEDED AND NO CORE AVAILABLE	The buffer is not large enough to handle the number of lines required for looking ahead and no more core is available from the Monitor.
?COMMAND ERROR	Error in last command string entered.
?DEVICE device NOT AVAILABLE	One of the input devices cannot be initialized; generally, the device either does not exist or has been assigned to another job.
?FILE 1 READ ERROR	An error has occurred on the first input device specified in the command.
?FILE 2 READ ERROR	An error has occurred on the second input device specified in the command.
?INPUT ERROR - filename.ext FILE NOT FOUND	The specified filename can't be found.
?NO DIFFERENCES ENCOUNTERED	No differences were found between the two source files.
?OUTPUT DEVICE ERROR	An error has occurred on the output device.
?OUTPUT INITIALIZATION ERROR	The output device cannot be initialized; the device either does not exist or has been assigned to another job, the device is not an output device or the filename could not be entered on the device.

BINARY COMPARE (BINCOM)

FUNCTION

To compare, word by word, two versions of a binary (.REL) program file and to output any differences.

ENVIRONMENT

Monitor	All
Minimum Core	1K if output device is other than DTAn:, MTAn:, or DSK:; otherwise 2K.
Additional Core	See "Minimum Core."
Equipment Required	Two input devices for the two files to be compared; one output device for listing the differences. Both input files can be on disk.

INITIALIZATION

```
.R BINCOM↵
*
```

.R BINCOM↵ — Loads the Binary Compare routine.

* — Binary Compare is ready to receive a command.

COMMANDS

General Command Format

```
list-dev:filename.ext← input1-dev:filename.ext,input2-dev:filename.ext↵
```

list-dev:

The device on which the differences are to be listed.

LPT:	(line printer)
TTY:	(Teletype)
MTAn:	(magnetic tape)
DTAn:	(DECtape)
DSK:	(disk)

If list-dev:filename.ext← is omitted, TTY: is assumed.

input1-dev:
input2-dev:

The devices on which the two binary files to be compared are located

DTAn:	(DECtape)
DSK:	(disk)
CDR:	(card reader)
PTR:	(paper tape reader)
MTAn:	(magnetic tape)

filename.ext (DSK: and DTAn: only)

The filename and extension of either of the input binary files.

The filename and extension to be assigned to the output list file.

NOTE: If .ext is omitted, .REL is assumed.

←

The output device is separated from the input binary file devices by the left arrow symbol.

EXAMPLES

```
.R BINCOM↵
*LPT:←DSK:PROG1.REL,DTA1:PROG1.REL↵
```

Compare the binary program file PROG1.REL in the user's area of the disk with a binary program file, PROG1.REL, on DTA1, and list all differences on the line printer.

```
NO ERRORS ENCOUNTERED↵
```

No differences were found between the two files.

```
*DTA1:BINA,DTA2:BINB↵
file1-word file2-word   XOR
     :          :         :
```

Compare the binary program file BINA on DTA1 with the binary program file BINB on DTA2 and list all differences on the Teletype.

NOTE: .REL is assumed as the extension name for both BINA and BINB.

```
*↑C↵
```

Return to the Monitor.

```
.KJOB↵
.
```

Kill the job, deassign all devices, and release core.

DIAGNOSTIC MESSAGES

Table B C-1 Binary Compare Diagnostic Messages

Message	Meaning
?COMMAND ERROR	Error in last command string entered.
?END OF FILE PHASE ERROR	One input file is longer than the other.
?INPUT INITIALIZATION ERROR	One of the input devices cannot be initialized; generally, either the device does not exist or it is assigned to another job.
NO ERRORS ENCOUNTERED	No differences were found between the two binary program files.
?OUTPUT INITIALIZATION ERROR	The output device cannot be initialized; generally, either the device does not exist or it is assigned to another job.

Error Differences

Whenever a difference is encountered between the two files being compared, a line is printed on the listing device in the following format.

file1-word file2-word XOR of both words

S.M.U.

binary loc

Should be octal

Replacement for Table BC-1, page BINCOM-4.

Table BINCOM-1 Binary Compare Diagnostic Messages

Message	Meaning
?COMMAND ERROR	Error in last command string entered.
?DEVICE dev: NOT AVAILABLE	Device has been assigned to another job or does not exist.
?END OF FILE PHASE ERROR	One input file is longer than the other.
?FILES BEING COMPARED ON SAME INPUT DEVICE	Files cannot be compared from the same input device unless that device is DSK:.
?INPUT ERROR filename.ext NOT FOUND	The file specified could not be found on the input device.
NO ERRORS ENCOUNTERED	No differences were found betweent he two binary program files.
?OUTPUT INITIALIZATION ERROR	The file cannot be entered.

ALGEBRAIC INTERPRETIVE DIALOGUE (AID)

FUNCTION

To provide users with a personal computing service for solving complex numerical problems

ENVIRONMENT

Monitor	All
Minimum Core	11K (with 1K of user data area)[1]
Additional Core	Up to 3K additional (providing a total of 4K of user data area)
Equipment Required	May use DECtape, disk, etc., for filing and subsequent recalling of data

Note that AID will not run in 256 ~~262~~K of core if Monitor occupies more than 245K of that core

[1] Note that AID will not run in 16K of core if Monitor occupies more than 5K of that core.

INITIALIZATION

.R AID↵	Loads the AID program into core
AID (revision date) AT YOUR SERVICE . . . ↵	AID responds with message when loaded.
*	AID is ready to receive a command.

COMMANDS

Format Rules

1. Only one step (command) can be typed per line, and only one line can be used for each step.

2. The text of each step begins with a verb and terminates with a period followed by a carriage return.

3. Words, variables (1-character identifiers), and numerals can neither abut each other nor contain imbedded spaces; spaces cannot appear between an identifier (when it appears in an array, a formula, or a function) and its associated grouped operators and arguments. Otherwise, spaces can be used freely.

4. Characters can be erased by striking the RUBOUT key once for each erasure; an entire line can be deleted by typing an asterisk followed by a carriage return.

5. A direct step is interpreted and executed by AID immediately (i.e., following the terminating carriage return typed by the user). No step number precedes a direct step; the line begins with the verb of the command.

6. An indirect step is entered by preceding the step with a numeric label containing both an integer and a decimal portion (e.g., 1.23); maximum number of significant digits in a step number is nine. Indirect steps are organized into parts according to the integer portion of their step number; all indirect steps having the same value in the integral portion of their step number belong to the same part. Indirect steps are not executed immediately but are stored for later execution (e.g., by a DO or TO command).

Identifiers — Single alphabetic characters (A through Z, a through z)

Defined by a value

SET x = value. (Set A = 2*5.7.)

DEMAND x. (Demand A.)

Defined by an expression

Arithmetic Formulas:
LET x = arithmetic formula — Let A = sqrt (b)+c/d.

Boolean Expressions (Propositions):
LET x = proposition

b = true
c = false
Let A = b and c.

User-Defined Function:

LET x(arguments) = expression — Let A(b,c) = b↑2 + c↑2.

Indexed identifiers (arrays)

x(i1,i2,i3,...)

One to ten indices (subscripts) allowed.
Each subscript can be in the range -250 through +250.

SPARSE items in an array:

LET x BE SPARSE. — Sets all undefined items in array "x" to 0 and saves the core which would be occupied by those items.

Arithmetic Operators (listed in order of precedence)

Standard Designation	AID Symbology	Meaning
$\|x\|$	!x!	Absolute value of x
[ ]	[ ]	1st level grouping
( )	( )	2nd level grouping
x^e	x↑e	x raised to the power of e
a·b, a(b), or a x b	a*b	a multiplied by b
a/b or $\frac{a}{b}$	a/b	a divided by b
a + b	a + b	a added to b
a - b	a - b	b subtracted from a

NOTE: Within nested pairs of brackets (or parentheses), the order of evaluation is from the innermost pair outward.

AID functions

arg (x,y) Computes angle between +x axis of x,y plane and line joining point 0,0 and point x,y. Answer is in radians.

cos(x) Computes cosine of x (x must be in radians and less than 100).

dp(x) Returns the digit part of x.

exp(x) Exponential function: e^x, where e is Euler's number (2.178281828) (e^x must be less than 10^{100})

first(i=iterative expression: i proposition) Finds first value in an array to satisfy the proposition, using i as an index. Result = value of i

fp(x) Returns the fractional part of x.

ip(x) Returns the integer part of x.

log(x) Computes natural log of x (x must be greater than zero).

max(i=iterative clause: i expression) Computes expression iteratively for each value of i and returns largest value.

or

max(series) Returns the largest value in the series.

min(iterative clause: i expression) Computes expression iteratively for each value of i and returns smallest value.

or

min(series) Returns the smallest value in the series.

prod(i=iterative clause: i expression) Computes expression iteratively for each value of i and returns the product of all the computed values.

or

prod(series) Returns the product of the series of values.

sgn(x) Signum function:

$x > 0$, signum = +1;
$x = 0$, signum = 0;
$x < 0$, signum = -1

sin(x) Computes sine of x (assumed to be in radians; x must be > 0).

sqrt(x) Computes the square root of x (x must be > 0).

sum(i=iterative clause: i expression) Computes expression iteratively for each value of i and returns the sum of all the computed values.

or

sum(series) Returns the sum of the series of values.

tv(proposition) Returns the truth value of the proposition:

true = 1
false = 0

xp(x) Returns the exponent value of x.

User-Defined Functions

f(a,b,c, . . .) = expression

f — function identifier (any single alphabetic character)

(a,b,c, . . .) — dummy arguments; the use of a character as a dummy argument in no way affects its use as an identifier.

expression — the arithmetic expression representing the user function.

Propositions (Boolean Expressions)

Relational Operators:

= (equal)   #(not equal)   >(greater than)   <(less than)

>= (greater than or equal to)   <= (less than or equal to)

Logical Operators: and, or

Negation: not

```
*x = true↵
*y = false↵
*Let z = x and y or x and (100 sqrt(959) ).↵
*Type z.↵
          z =          true↵
```

Order of Execution:

(1) Evaluation of expressions
(2) Parentheses (from innermost pair outward)
(3) Relational operations
(4) not
(5) and
(6) or

A series of relational operators (a = b>c<d) is assumed to be an and chain (a = b and b>c and c<d).

Conditional Expressions

Allows an expression to have different values depending upon which one of a number of conditions is true.

A conditional expression is a series of expressions separated by semicolons, with each expression preceded by a proposition and a colon. The entire conditional expression is enclosed in parentheses (or brackets).

(proposition: expression; proposition: expression;......)
Let C(x) = (x>0:x ↑ 2; x=0:0; x<0:x).

states that:

if x> 0, then $C(x) = x^2$
if x =0, then C(x) = 0
if x< 0, then C(x) = x

If the last expression is to be true for all cases which do not satisfy any of the other stated conditions, the expression can be typed without a preceding proposition.

Let C(x) = (x> 0:x ↑ 2; x =0:0;x).

Every possible combination of the variable must be provided for, either by explicitly stating a proposition and an expression for it, or by simply specifying a terminating expression to be executed for all remaining cases.

Table AID-1 AID Command Summary

Command Format	Type	Description
CANCEL.	D,O	Cancels a currently stopped process when the user does not desire to resume execution.
(CANCEL.)	D,O	Cancels a currently stopped process which was initiated by a parenthetical DO.
DELETE { L S S(m,n) form m step m.n part m formula f all steps all parts all formulas all forms all values all } .	O	Erases the specified item from immediate storage and frees the space occupied by it for some other use. Several DELETE commands can be combined into one.
DEMAND { L S(m,n) L as "any text" S(m,n) as "any text" } .	I,O	Causes AID to type out a message requesting the user to supply a value for the specified item. Only one variable can be specified in each DEMAND command.
DISCARD ITEM m(code).	F	Deletes item #m from the external storage file currently in use. (Code) is optional.
DO { step m.n step m.n, p times step m.n for L = range part m part m, p times part m for L = range } .	O	Executes an indirect step or part. If the DO command is a direct step, control returns to the user at the completion of the DO; if an indirect step, control returns to the step following the DO.
(DO.... same as above...)		Initiates a new execution without cancelling the currently stopped process.

Table AID-1 (Cont) AID Command Summary

Command Format	Type	Description
DONE.	I,O	Skips execution of the remaining steps of a part during the current iteration.
FILE { L S S(m,n) form m step m.n part m formula f all steps all parts all formulas all forms all values all } { AS ITEM n (code) }.	F	Stores the specified item in the external storage file currently open. Immediate storage is not affected in any way. (Code) is optional.
FORM m: ....◁◁◁◁ text	O	Defines a format to be used in editing typeouts for purposes of readability. ◁◁◁◁◁.◁◁◁◁ fixed point notation (up to nine digit positions plus the decimal point) .... scientific notation (minimum of seven positions, maximum of fourteen) text any text to be included in the line; not enclosed in quotation marks unless they are part of the text.
GO.	D,O	Continues execution of a currently stopped process; opposite of the CANCEL command.
IF Clause Verb....IF proposition.	M	Can be appended to any command (except the abbreviated SET command) to make the command conditional; the command is executed only if the proposition is true.
LET { L = m L = formula F(L) = m F(L) = proposition } :	O	Defines arithmetic formulas, Boolean expressions (propositions), and user functions and associates them with identifiers. The formula, expression, or function with which an identifier is associated is re-evaluated each time the identifier appears during an execution.

Table AID-1 (Cont) AID Command Summary

Command Format	Type	Description
LET S be sparse .	S	Sets undefined array elements to zero.
LINE .	O	Advances the Teletype paper form one line.
PAGE .	O	Advances the Teletype paper form to the top of the next page.
QUIT .	O	Skips execution of the remaining steps of a part and satisfies the DO command for that part by cancelling any further iterations. Usually given conditionally.
RECALL ITEM m (code) .	F	Reads an item, previously stored by a FILE command, from the currently open external storage file into immediate storage. (Code) is optional and is for documentation only.
RESET TIMER.	S	Resets TIMER to zero.
SET $\left\{\begin{array}{l} \text{L} = \text{m} \\ \text{L} = \text{proposition} \\ \text{S(m, n)} = \text{m} \\ \text{S(m, n)} = \text{proposition} \end{array}\right\}$ ∸	O	Defines an identifier as equivalent to a fixed value, which is calculated once and then used whenever the identifier appears. A short form of the SET command, where the word SET and the period are omitted, can be used if the command is direct.
STOP.	I, O	Temporarily halts the current process at the point where the STOP command appears and returns control to the user. The stopped process can be resumed by typing GO.
TO $\left\{\begin{array}{l} \text{part m} \\ \text{step m.n} \end{array}\right\}$ ∸	I, O	Discontinues the sequential execution of the part currently being executed and transfers control to another step or part; when the new part is finished, the direct command which initiated the execution is satisfied.

Table AID-1 (Cont) AID Command Summary

Command Format	Type	Description
TYPE { m S S(m,n) proposition "any text" ← form m step m.n part m formula f F(x) F(proposition) all steps all parts all formulas all forms all values all time timer size item-list } ⨪	O S S S F	Types out the specified information on the user's console. Several individual TYPE commands may be combined into one (except for TYPE "any text" or TYPE ITEM-LIST). The command Type ... in form n. causes the listed items to be typed out in the format specified by form n. n can be a numeric value (e.g., Form 3) or it can be a numeric formula (e.g., Form (2*x-y)).
USE FILE filename (device) ⨪	F	Makes an external storage file available for use. The external file thus addressed remains open for use (by DISCARD, FILE, RECALL, and TYPE ITEM-LIST commands) until another USE command is given or the AID program is terminated.

Command Format Symbology

L = letter
S = subscripted letter
m, n, p = numeric values
f = formula
F = function
range = an iterative sequence or series of values

Type Symbology

D = Can be given directly only
I = Can be given indirectly only
O = Operational command
F = File command
S = Special command

Table AID-2 File Command Subset

Command Format	Section Reference [1]	Description
DISCARD ITEM m (code)	5.4	Deletes item #m from the external storage file currently in use. (Code) is optional.
FILE $\left\{\begin{array}{l}\text{L}\\ \text{S}\\ \text{S(m,n)}\\ \text{form m}\\ \text{step m.n}\\ \text{part m}\\ \text{formula f}\\ \text{all steps}\\ \text{all parts}\\ \text{all formulas}\\ \text{all forms}\\ \text{all values}\\ \text{all}\end{array}\right\}\begin{array}{c}\text{AS}\\ \text{ITEM}\\ \text{n (code) .}\end{array}$	5.7	Stores the specified item in the external storage file currently open. Immediate storage is not affected in any way. (Code) is optional.
RECALL ITEM m (code).	5.15	Reads item #m, previously stored by a FILE command, from the currently open external storage file into immediate storage. (Code) is optional and is for documentation only.
TYPE ITEM-LIST.	5.20	Obtains a typeout of the directory of the currently open external storage file.
USE FILE filename (device)	5.21	Makes an external storage file available for use. The external storage file thus addressed remains open for use (by DISCARD, FILE, RECALL, and TYPE ITEM-LIST commands) until another USE command is given or the AID program is terminated.

[1] Section references refer to sections within the PDP-10 AID (Algebraic Interpretative Dialogue) Programmer's Reference Manual, DEC-10-AJA0-D.

Table AID-3 AID Character Set

Standard Math Symbol	AID Symbol	Typing Method		JOSS Symbol	Notes
		Model 37[1]	Models 33 and 35		
	A through Z	Strike appropriate key with SHIFT.	Strike appropriate key; no SHIFT.	A through Z	
	a through z	Strike appropriate key without SHIFT	Not available; use upper-case letters.	a through z	
	0,1 through 9	Strike appropriate key; no SHIFT.	Strike appropriate key; no SHIFT	0,1 through 9	
Operators: | | (absolute)	! !	Strike the !,1 key with SHIFT.	Strike the !,1 key with SHIFT.	| |	
[ ] (brackets)	[ ]	Strike appropriate keys.	[Strike K with SHIFT] Strike M with SHIFT	[ ]	
( ) (parentheses)	( )		(Strike the (,8 key with SHIFT.) Strike the) ,9 key with SHIFT.	( )	
x^e (exponent)	x↑e		Strike the N,↑ key with SHIFT	*	
/ (divide)	/		Strike the ?,/ key; no SHIFT.	/	
. (multiplication)	*		Strike the *,: key with SHIFT.	.	
+ (addition)	+		Strike the +,; key with SHIFT.	+	
- (subtraction)	-		Strike the =,- key; no SHIFT.	-	
Boolean Expressions: = (equal)	=		Strike the =, - key with SHIFT.	=	
≠ (not equal)	#		Strike the #,3 key with SHIFT.	≠	

Table AID-3 (Cont) AID Character Set

Standard Math Symbol	AID Symbol	Typing Method		JOSS Symbol	Notes
		Model 37[1]	Models 33 and 35		
≤ (equal to or less ≥ than)	< = (2 characters)		Strike the<, .key with SHIFT; then strike the =,-key with SHIFT.	≤	
≥ (equal to or greater than)	> = (2 characters)		Strike the >, . key with SHIFT; then strike the =, - key with SHIFT.	≥	
	RUBOUT (types back as \)	Strike DELETE key to erase each preceding character in error; then type correctly. Example: TPE \\YPE PART 1.	Strike RUBOUT key to erase each preceding character in error; then type correctly.	BACK-SPACE and type over	Used to correct typing errors.
null item	←		Strike O key with SHIFT.	_(underscore)	
	$ (current line number)	Strike the $,4 key with SHIFT	Strike the $,4 key with SHIFT.	$	
	* (cancel entire line)		Strike the *,: key with SHIFT.	*	

[1] Blank boxes in this column represent characters whose keyboard positions have not yet been determined. This information will be supplied at a later date.

EXAMPLES

1. Estimation of the square root (x_i) of a positive number (A) by the iterative formula:

$$x_{i+1} = \tfrac{1}{2}(x_i + \frac{A}{x_i}) \text{ until } A - (x_i)^2 \leq \text{ some "permitted deviation" value}$$

```
. R AID↵
AID (1-30-68) AT YOUR SERVICE . . .↵
*1.1 Demand A.↵                          "A" is the positive value for which the square
                                         root is to be calculated.
*1.15 Type "GIVE ESTIMATION".↵
*1.2 Demand X.↵                          "X" is an estimated square root.
*1.3 Set N = ( ½ )*(X + A/X).↵           This is the given formula (above) expressed in
                                         the AID language.
*1.4 To part 2 if !A-(X↑2) !>.00001.↵    Go to part 2 if the difference between the square
                                         root of the estimated square root and the actual
                                         number is greater than .00001.
*1.5 Type A,X in form 1.↵                The difference is equal to or less than .00001;
                                         type the two values, A and X.
*1.6 Type X, X↑2 in form 2.↵             Type the "proof" statement.
*2.1 Set X = N.↵                         Set the value of X for the next iteration.
*2.2 To step 1.3.↵                       Return to recalculate.
*Form 1:↵                                Set up forms for typing out results.
*THE SQUARE ROOT OF ◁◁◁◁.◁◁ IS ◁◁◁.◁◁◁◁◁◁↵
*Form 2:↵
*PROOF: ◁◁◁.◁◁◁◁◁◁ SQUARED IS ◁◁◁◁.◁◁◁◁◁◁↵
*Do part 1, 4 times.↵
                    A =   *102↵
GIVE ESTIMATION↵
                    X =   *12↵
THE SQUARE ROOT OF     102.00 IS 10.099505↵
PROOF:    10.099505 SQUARED IS   102.000001↵
                    A = *133.39↵
GIVE ESTIMATION↵
                    X =  *11.1↵
THE SQUARE ROOT OF 133.39 IS     11.549459↵
PROOF:    11.549459 SQUARED IS   133.390001↵
```

```
                  A = *1234.55 ↵
GIVE ESTIMATION
                  X = *35 ↵
THE SQUARE ROOT OF   1234.55 IS    35.136164 ↵
PROOF:      35.136164 SQUARED IS  1234.550000 ↵
                  A =   *5 ↵
GIVE ESTIMATION ↵
                  X =   *2 ↵
THE SQUARE ROOT OF        5.00 IS      2.236068 ↵
PROOF:       2.236068 SQUARED IS       5.000000 ↵
*
```

2 . Generation of Sine Wave

```
        *1.1     Do part 2 for I = 1 (6*3) 3 60*1.5. ↵

        *2.1     Set X = I. ↵
        *2.2     Do part 3. ↵
        *2.3     Do part 4. ↵

        *3.01    Set X = X/57.2957795. ↵
        *3.02    Set A = 1. ↵
        *3.03    Set C = 3. ↵
        *3.04    Set S = 0. ↵
        *3.05    Set G = 1. ↵
        *3.06    Set Y = X. ↵
        *3.07    To step 3.50 if (X-2*3.14159) <0. ↵
        *3.08    Set X = X - 6.283191. ↵
        *3.09    To step 3.06 ↵
        *3.10    Set Y = Y*X*X. ↵
        *3.20    Set A = A*C*(C-1). ↵
        *3.30    Set C = C +2. ↵
        *3.40    Set G = -G. ↵
        *3.50    Set T = S+ G * Y/A. ↵
        *3.60    To step 3.90 if (T-S)= 0. ↵
        *3.70    Set S = T. ↵
        *3.80    To step 3.10. ↵
        *3.90    Done. ↵

        *4.1     Type M in form IP ( (12 + 10*S) +1). ↵

        *Form 1: ↵
        *← ↵
        *Form 2: ↵
        *    ← ↵
        *Form 3: ↵
        *        ← ↵
```

Generation of Sine Wave (cont)

```
*Form 4:↵
*        ← ↵
*Form 5:↵
*            ← ↵
*Form 6:↵
*              ← ↵
*Form 7:↵
*                 ← ↵
*Form 8:↵
*                   ← ↵
*Form 9:↵
*                      ← ↵
*Form 10:↵
*                        ← ↵
*Form 11:↵
*                          ← ↵
*Form 12:↵
*                             ← ↵
*Form 13:↵
*                               ← ↵
*Form 14:↵
*                                  ← ↵
*Form 15:↵
*                                     ← ↵
*Form 16:↵
*                                       ← ↵
*Form 17:↵
*                                          ← ↵
*Form 18:↵
*                                            ← ↵
*Form 19:↵
*                                              ← ↵
*Form 20:↵
*                                                ← ↵
*Form 21:↵
*                                                   ← ↵
*Form 22:↵
*                                                     ← ↵
*Form 23:↵
*                                                       ← ↵
```

Generation of Sine Wave (cont)

```
*Form 24:↵
*                                                              ←↵

*Form 25:↵
*                                                                 ←↵

*Form 26:↵
*                                                                   ←↵

*M = 8↵
*Do part 1.↵




                                              8
                                                     8
                                                             8
                                                                8
                                                                   8
                                                                    8
                                                                    8
                                                              8
                                                          8
                                                  8
                                           8
                                     8
                               8
                          8
                         8
                        8
                         8
                             8
                                  8
                                        8
                                               8
                                                      8
                                                             8
                                                                 8
                                                                    8
                                                                    8
                                                                    8
                                                               8
                                                           8
                                                      8
                                                8



*
```

Table AID-4 AID Diagnostic Messages

Message	Meaning
x = ???	A value has not been supplied by the user for variable x.
DONE.	Signals completion of a File command (DISCARD, FILE, RECALL).
DONE. I'M READY TO GO $\left\{\begin{array}{l}\text{AT}\\ \text{FROM}\\ \text{IN}\end{array}\right\}$ STEP m.n.	...AT STEP m.n...Task was suspended by an interruption or error during the interpretation of an indirect step. ...FROM STEP m.n...Task was suspended by a stopping command. ...IN STEP m.n...Task was suspended during an indirectly initiated DO command. AID resumes execution whenever the user types GO.
DONE. I'M READY TO GO $\left\{\begin{array}{l}\text{AT}\\ \text{FROM}\\ \text{IN}\end{array}\right\}$ STEP m.n, ALTHO I CAN'T FIND IT.	Same as above, except that the step at which AID is prepared to resume can no longer be found in immediate storage. Possibly, a direct command (or a routine initiated by a parenthetical DO) has deleted the step in the interim. Upon receipt of a GO command from the user, AID will attempt to resume at the step following the missing step.
DON'T GIVE THIS COMMAND $\left\{\begin{array}{l}\text{DIRECTLY}\\ \\ \text{INDIRECTLY}\end{array}\right\}$	This command can be given only indirectly (TO, DONE, STOP, DEMAND) or only directly (CANCEL, GO).
EH ?	The previously entered line is incorrect. Indirect commands: The step number was incorrectly typed. Direct LET commands: LET x portion is incorrect. Other direct commands: A space was omitted. The terminating period was omitted. The command is not legitimate. An expression is incorrectly written. To continue, retype the command correctly.

Table AID-4 (Cont) AID Diagnostic Messages

Message	Meaning
ERROR AT STEP m.n: EH?	The step number is correct, but the command is incorrect. a. Request a typeout of the step in error. b. Check for the errors listed under "Eh ?". c. Retype the command correctly. d. Type GO. to continue.
ERROR AT STEP m.n: I CAN'T FIND THE REQUIRED $\left\{\begin{matrix}\text{STEP}\\ \text{FORM}\\ \text{PART}\\ \text{FORMULA}\end{matrix}\right\}$.	The step in error refers to a nonexistent step or part. Correct the error and type GO. to continue.

ERROR AT STEP m.n: (IN FORMULA x):
z = ???
	The variable z has not been assigned a value by the user.
	Check for any other errors, define variable z correctly, and type GO. to continue.

ERROR IN FORMULA x: EH?	(Following a direct command in which x was used) The form of the expression for x is in error. a. Request a typeout of formula x. b. Check for the errors listed under "Eh ?". c. Formula x may be correctly written, but the definition of one or more identifiers is not consistent with their use in formula x.
FILE NUMBER MUST BE POSITIVE INTEGER< = 2750	The filename of a USE command must not be greater than the value 2750.
FORM NUMBER MUST BE INTEGER AND 1< = FORM< 10↑9.	Form numbers must be integers in the range 1 through 10^9- 1.
I CAN'T EXPRESS THE VALUE IN YOUR FORM.	A value cannot be expressed in the format specified by the FORM (e.g., the value is too large to specify in fixed point notation). To correct, follow the steps given under "I HAVE TOO MANY VALUES FOR THE FORM."
I CAN'T FIND THE REQUIRED $\left\{\begin{matrix}\text{FORM}\\ \text{ITEM}\\ \text{PART}\\ \text{STEP}\end{matrix}\right\}$.	Either the element has never been defined or has been deleted.

Table AID-4 (Cont) AID Diagnostic Messages

Message	Meaning
I CAN'T MAKE OUT YOUR FIELDS IN THE FORM.	The fields in the form specified were typed in such a way that AID cannot distinguish their beginning or ending. Possibly, there are either no fields in the form or two or more are run together with no intervening space.
I HAVE AN ARGUMENT< = 0 FOR LOG.	The argument for the LOG function must be greater than 0.
I HAVE A NEGATIVE ARGUMENT FOR SQRT.	Square root arguments must be positive.
I HAVE A NEGATIVE BASE TO A FRACTIONAL POWER.	An attempt was made to raise a negative value to a fractional power. For example, Type (-y) ↑ (1/2).
I HAVE AN OVERFLOW.	Some number has exceeded 9.99999999.10 ↑ 99 in magnitude.
I HAVE A ZERO DIVISOR.	An attempt was made to divide by 0.
I HAVE NOTHING TO DO.	The user has typed GO., but there is no currently stopped process which can be continued.
I HAVE TOO FEW VALUES.	An insufficient number of arguments have been supplied for a function.
I HAVE TOO MANY VALUES FOR THE FORM.	There are not enough fields in the form to receive all the values to be typed. a. Type the form and the values. b. Check for errors. c. Change either the TYPE command or the FORM to make them compatible and then type GO. to continue.
I HAVE ZERO TO A NEGATIVE POWER.	An attempt was made to raise 0 to a negative power.
ILLEGAL SET OF VALUES FOR ITERATION.	An error has been detected in a range clause of a function or a DO command, such that the ending value can never be reached (e.g., the increment is 0).
I'M AT STEP m.n.	When the user responds to a DEMAND-produced request (x=*) with a carriage return only, AID types back this message.

Table AID-4 (Cont) AID Diagnostic Messages

Message	Meaning
INDEX VALUE MUST BE INTEGER AND !INDEX! < 250	All index values (subscripts) must be integral and must have an absolute value of < 250.
I NEED INDIVIDUAL VALUES FOR A FORM .	A command was given to type a subscripted variable in a form (e.g., Type B in form 1, where B is a subscripted variable). Individual values only can be specified for TYPE ....IN FORM n commands.
I RAN OUT OF SPACE.	User's immediate memory is filled due to one of the following errors. a. Endless loops because of DO commands or because DO was typed instead of TO. b. Unlimited recursive definition. c. Variable x defined in terms of y, and variable y defined in terms of x via LET command. d. Program is too large for available memory; use TYPE SIZE command to determine how much immediate storage has been used. File commands can be used to store parts of the routine and execute them one at a time.
I RAN OUT OF FILE SPACE.	DECtape directory is full (limit = 22 items).
ITEM NUMBER MUST BE = 25.	The item number in file commands (DISCARD, FILE, RECALL) must be less than or equal to 25 (22 for DECtapes).
NUMBER-OF-TIMES MUST BE INTEGER AND >= 0.	The value specified in the TIMES clause of a DO command must be a positive integer.
PART NUMBER MUST BE INTEGER AND 1< = PART 10 ↑ 9.	Part numbers must be integers and in the range 1 through 10^9 -1.
PLEASE DELETE THE ITEM OR USE A NEW ITEM NUMBER.	The user has attempted to FILE information into an item which already exists on the currently open external storage file. The user must either DISCARD the item prior to filing the new information or use a different item number in the FILE command.
PLEASE KEEP !X!< 100 FOR SIN(X) AND COS(X).	Arguments for the SINE and COSINE functions must be less than 100.
PLEASE LIMIT ID'S TO 5 LETTERS AND/OR DIGITS.	Filename in a USE file command or code in a DISCARD, FILE, or RECALL command exceeds five characters in length or contains special characters.

Table AID-4 (Cont) AID Diagnostic Messages

Message	Meaning
PLEASE LIMIT LINES TO 78 UNITS (CHECK MARGIN STOPS) SAY AGAIN:	User typeins are limited to single-line, 78-character strings.
PLEASE LIMIT NUMBERS TO 9 SIGNIFICANT DIGITS.	Numeric values are limited to nine significant digits.
PLEASE LIMIT NUMBER OF INDICES TO 10.	The number of subscripts following on identifier cannot exceed 10.
PLEASE LIMIT NUMBER OF PARAMETERS TO TEN	The number of arguments for a function is limited to 10.
PLEASE LIMIT STEP LABELS TO 9 SIGNFICANT DIGITS.	Step numbers can be up to nine digits in length.
REVOKED. I RAN OUT OF SPACE.	See "I RAN OUT OF SPACE."
ROGER.	Signals successful completion of a USE file command.
SOMETHING'S WRONG. I CAN'T ACCESS THE FILES.	A system I/O error (or other type of AID error) has occurred. Begin again.
SOMETHING'S WRONG. TRY AGAIN.	AID has found something unusual in its internal records or has received contradictory signals from its I/O routine. Begin again.
SORRY. SAY AGAIN:	A transmission error occurred on the previous typein. This message is preceded by the erroneous line with # symbols typed where the failure occurred. Retype the line.
STEP NUMBER MUST SATISFY 1< =STEP<10 ↑ 9.	Step numbers must be in the range 1 through 10^9-1.
STOPPED BY STEP m.n.	Process has been temporarily halted by a STOP command at step m.n.
YOU HAVEN'T TOLD ME WHAT FILE TO USE.	The user has issued a DISCARD, FILE, RECALL, or TYPE ITEM-LIST command before he has given a USE file command.

DESK CALCULATOR (DESK)

FUNCTION

To give the user direct access to arithmetic functions for problem solving.

- A powerful conversational problem-solving system
- No complex programming rules to follow; problems are entered by use of familiar arithmetic operators and functions
- Permits the user to define new macro operations for those functions not contained in the basic repertoire

ENVIRONMENT

Monitor	All
Minimum Core	4K (no macros) or 5K[1]
Additional Core	None
Equipment Required	Console Teletype; other devices are optional.

said above

[1] DESK can be run in 4K only if no macro definitions ($D) are used; otherwise 5K is required.

INITIALIZATION

.R DESK↵	Loads the Desk Calculator program into core.
↵	The Desk Calculator is ready to receive the first problem statement.

COMMANDS

Table DESK-1. Desk Pseudo-Commands

Command	Operation
$C	Clear the accumulator.
$Dfunctionname (2 or 3 char.)	Define macro routine.
$Kfunctionname	Delete the function specified by the user. If no function is specified, delete all user functions and set all variables to 1.0.
$G a	Get the value assigned to the symbol "a." If none is assigned, the value 0 is assumed.
$L	Output line (carriage return followed by line feed). Used in macro operations.
$P a	Place a value into "a."
$T	Output tab. Used in macro operations.
$Fn (0≤ n ≤ 7)	Round output to n significant digits.
$I	Round output to nearest integer (same as $F0).
$R	Secondary exit from a macro operation.

Table DESK-2 Desk Functions

Function	Description
ATN	arc tangent in radians
COS	cosine in radians
CSD	cosine in degrees
EXP	exponential (n^e)
LOG	natural logarithm ($LOG_e X$)
SIN	sine in radians
SND	sine in degrees
XNT	integer part (i.e., truncate floating point number so that only a floating point integer remains)

Table DESK-3 Desk Command String Symbols

Symbol	Function
←	Directs Desk to output the value of the expression.
;	Delimits the macro mnemonic and also delimits the instruction string which constitutes the macro function.
!	Directs Desk to output the expression value followed by a tab.
ALTMODE key	Directs Desk to evaluate the expression and type the output on the same line.
Arguments	A maximum of nine arguments can be specified within a macro function in the form macro-mnemonic $(A_1, A_2, \ldots\ldots A_n)$ Arguments are written in the form #n $(1 \leq n \leq 9)$
Arithmetic Operators	Add + Subtract - Multiply * Divide / Raise to the power ↑ Parentheses for grouping subexpressions ( ) Replacement; store the expression on the left in the variable on the right =
Comments	Comments are enclosed within quotation marks (").
Labels	Labels within macro functions can consist of two or three digits followed by a colon (:).
Macro Mnemonics	Macro mnemonics can consist of two or three alphabetic characters.

Table DESK-3 (cont) Desk Command String Symbols

Symbol	Function	
Transfers Within Macro Statements	STP[1]	Transfer if result of subtracting one from the first argument is greater than zero.
	TA[2]	Transfer unconditional.
	TE	Transfer if equal to zero.
	TG	Transfer if greater than zero.
	TGE	Transfer if greater than or equal to zero.
	TL	Transfer if less than zero.
	TLE	Transfer if less than or equal to zero.
	TN	Transfer if not equal to zero.
Variables	Single alphabetic characters are used to represent variables. Variables are actually single memory locations, the contents of which may vary.	

NOTES: 1. The first argument is a variable and the second is a label.

2. The only argument is a label.

For all others, the first argument is an expression and the second is a label.

EXAMPLES

.R DESK↵ ↵	
$F5↵	Round all results to the first five significant decimal positions.
SIN(10)← ALTMODE −0.54402↵ minus sign S.M.V.	Compute the sine (in radians) of 10 and type the result on the same line.
$F3↵	Round all results to the first three significant decimal positions.
COS (10)←↵ −0.839↵ S.M.V.	Compute the cosine (in radians) of 10 and type the result on the next line.
$DCMP;↵	Define a macro operation called CMP.
#1=X ↵	The first argument entered whenever CMP is used is represented in the formula by X.
#2=Y↵	The second argument entered whenever CMP is used is represented in the formula by Y.
$F1X!Y!X+Y!X↑2+Y↑2← ;↵	Each time CMP is used, print the following values rounded to the first significant decimal position: X Y X+Y X^2+Y^2
CMP (4,6)↵	To use macro routine, type the macro mnemonic followed by the arguments within parentheses.
4.0 6.0 10.0 52.0↵	Values are typed out. X Y X+Y X^2+Y^2
$DTYP;↵	Define a macro operation called TYP.
"X SINX COSX"↵	Step 1. Print a columnar heading.
$L ↵	2. Perform a line feed.
#1=X↵	3. Argument 1 is represented in the formula as X.
#2=Y↵	4. Argument 2 is represented in the formula as Y.

```
1:$F1X!$F5SND(X)!CSD(X)!$L
```

5. tag1: Print out the value of X (accurate to the first significant decimal position) followed by a tab;
Print out the value of SND(X) - accurate to the first five significant decimal positions - followed by a tab;
Print out the value of CSD (X) - accurate to the first five significant decimal positions - followed by a tab;
Perform a line feed.

```
X+#3=X↵
```

6. Increment the current value of X by the third argument to produce a new value of X.

```
TGE(Y-X,1)←— ; ↵
```

7. If the second argument minus the new value of X is equal to or greater than zero, go back to tag 1: - otherwise, exit from the routine.

```
TYP (10,60,10)↵
```

To use the routine, type the macro mnemonic, followed by the three arguments in parentheses.

```
X            SINX           COSX↵
10.0         0.17365        0.98481↵
20.0         0.34202        0.93969↵
30.0         0.50000        0.86603↵
40.0         0.64279        0.76604↵
50.0         0.76604        0.64279↵
60.0         0.86603        0.50000↵
↑C↵
```

Return to Monitor.

```
.KJOB↵

.
```

Kill the job, release core.

DESK INPUT/OUTPUT COMMANDS

Table DESK-4 Desk Input/Output Commands

Command	Meaning
$Ooutput-dev:filename.ext	Assign output-dev: and filename.ext (if DTAn or DSK) for all output (results and macro definitions) from Desk.
$Sinput-dev: filename.ext	Assign input-dev: and filename.ext (if DTAn: or DSK:) as source of all macro definitions not presently existing in core. Example: If user types in a command in the form of a macro call, core is first searched for the macro routine definition; if not found, the assigned input device is searched.
$W	Write out on the selected output device all macro definitions currently active (i.e., not killed) in core.
$Y	Cancel the current output device and reassign the Teletype for all output.

Example

.R DESK↵	
$ODSK:SAVE↵	Assign a file called SAVE on disk for all Desk output.
$DCMP;↵ #1=X↵ #2=Y↵ $F1X!Y!X+Y!X↑2+Y↑2 ← ;↵	Define the macro CMP.
$W↵	Write all macro routines currently in memory (i.e., CMP) on the assigned output device.
CMP (4,6)↵	Execute CMP. Results are written on the assigned output device instead of being typed.
$KCMP↵	Delete the CMP routine from core. NOTE: It still exists on the output device.
$SDSK:SAVE↵	Select the file called SAVE on the disk as the source of all macro routine definitions. NOTE: All macro definitions written on an output device remain there until the file is deleted or otherwise destroyed and can be used any time Desk is run without the necessity of redefining the macro routine.

`CMP(5,8)↵`	Core is searched for the CMP routine (it has been killed and no longer exists in core); the selected file on the input device (DSK:SAVE) is then searched, CMP is found and brought into core for execution. Results are written on the selected output device.
`$Y↵`	Cancel the selected output device.
`CMP (3,9)↵`	Search is executed as explained in the previous example. Since the output device has been cancelled, the results are printed on the Teletype.
`3.0   9.0   12.0   90.0↵`	
`↑C↵`	Return control to the Monitor.
`.KJOB↵`	Kill the job, deassign all devices, and release core.
`.`	

DIAGNOSTIC MESSAGES

Table DESK-5 Desk Diagnostic Messages

Message	Meaning
?DEVICE NOT AVAILABLE	Either device does not exist or it has been assigned to another job.
?DIRECTORY CAPACITY EXCEEDED	DECtape or disk directory is full.
EVALUATOR UNABLE TO DECIPHER THIS	An illegal formula (macro definition, etc.) or an illegal character has been entered.
?FILE NOT FOUND	The filename.ext specified as the input source cannot be found on the device.
FUNCTION EXPECTS ARGUMENT LIST	Required arguments are missing for the last function specified.
ILLEGAL FUNCTION DEFINITION	An illegal format has been entered.
IMPROPER DUMMY SYMBOL	A symbol has not been assigned or used properly.
IMPROPERLY FORMED ARGUMENT LIST	Desk expected a list of arguments but found constants mixed in.
IMPROPER NAME IN LEFT HALF	Desk expected an argument name in the left half of a word but found something else.
IMPROPER PARENTHESES COUNT	The number of left parentheses does not match the number of right parentheses.
?INTERNAL ERROR	An error has been made by Desk. Restart from beginning.
entry IS AN ILLEGAL CHARACTER	A character cannot be interpreted as a variable, operator, or pseudo-op.
entry IS AN ILLEGAL FIELD SIZE	Size of field exceeds expected number of decimal positions.
entry IS AN ILLEGAL FUNCTION NAME	Entry has not previously been defined as a function.
entry IS AN ILLEGAL PSEUDO-OP	A $x entry, where x is other than C, D, Fn, G, I, K, L, O, P, R, S, T, W, or Y, is unrecognizable.
?PUSHDOWN OVERFLOW	The pushdown list in Desk has overflowed.
UNDEFINED LABEL IN A TRANSFER	A label appearing in a transfer statement (STP or Txx) is not defined elsewhere in the routine.
VARIABLE NAME IN RIGHT HALF INCORRECT	Desk expects a variable in the right half of an expression but instead finds a constant.

FUNCTION

To supervise the sequential execution of a series of jobs with a minimum of operator attention.

- Operates as one of the "users" of the system in a time-sharing evnironment
- Maintains constant communication with the operator and allows him at any time to interrupt, skip, repeat, or prematurely terminate one or more of the jobs in the series
- Runs concurrently with the Batch-controlled object jobs

ENVIRONMENT

Monitor	All
Minimum Core	3K
Additional Core	Sufficient core to run the Batch-controlled jobs.
Equipment Required	One Batch input device, one Batch output device, one scratch device, and the shared SYS device. If on disk, all of these may be the same device.

INITIALIZATION

.ASSIGN dev BPTEMP↵	The logical name BPTEMP (scratch device) must be assigned to a retrievable device (DSK, DTAn, MTAn) before beginning Batch.
.R BATCH↵	Loads the Batch Processor into core.
*	Indicates that Batch is ready to receive a command. NOTES: 1. Batch is always ready to receive a command, whether or not an asterisk has previously been typed. However, when an asterisk does appear, Batch is specifically requesting a command and will not proceed without one. 2. ↑C must never be typed except immediately following an asterisk typeout.

COMMANDS

all commands can be abbreviated to the two or three letters to make them unique S. M. U.

Batch Input, Batch Output Device Assignments

INPUT dev:.ext

Specifies the Batch input device (e.g., the device containing the job control file produced by Stack).

dev: Can be

CDR:	(card reader)
MTAn:	(magnetic tape)
DTAn:	(DECtape)
DSK:	(disk)
PTR:	(paper tape reader)
TTYn:	(Teletype - NOTE: Operator's line number cannot be "n.")

.ext (DSK: and DTAn: only)

The extension of the job file.
Each user's job is assumed to be a separate file with the name IJOBxy.ext, where xy is in the range of 01 through 99. If Stack was used to create the files, filename.ext's of this format were automatically assigned to the files in a sequential manner.

OUTPUT dev:.ext

Specifies the Batch output device (i.e., the listing device).

dev: Can be

LPT:	(line printer)
DTAn:	(DECtape)
MTAn:	(magnetic tape)
DSK:	(disk)
PTP:	(paper tape punch)
TTYn:	(Teletype - NOTE: Operator's line number cannot be "n.")

.ext (DSK: and DTAn: only)

The extension to be assigned to the output files. Batch automatically assigns the name OJOBxy.ext, where xy in the range 01 through 99, to each output file in a sequential manner. In any case, .ext is optional.

NOTE: If either or both of the above commands are typed while a user's job is being processed by Batch, Batch stores them until the user's current job is completed. Batch then takes its next input or writes its next output on the most recently specified devices.

Job Control Commands

SKIP n↵

Execute the nth job on the Batch input device.

The first job control command to Batch is normally

SKIP 0

(i.e., no jobs are skipped and execution begins with the first job).

Once Batch processing has begun, the SKIP commnad has the following effects.

SKIP 0 Restart the current job.

SKIP n After the current job is finished, execute the nth job after the current job.

SKIP -n After the current job is finished, execute the nth job prior to the current one (i.e., repeat the previous n job executions).

NOTE: The following commands can be issued at any point after Batch job execution has been started by the first SKIP command. Once Batch has been started, automatic sequential execution of the jobs on the Batch input device is performed unless the operator intervenes.

BREAK↵
*

Finish the current job and halt before continuing on to the next job. The operator must then enter a command before Batch continues processing.

CONTINUE↵

Resume processing at point where Batch was interrupted. If Batch was doing nothing when interrupted, Batch responds with an asterisk; the operator must then type a SKIP command to continue. CONTINUE is used to resume processing after the user has responded to one of the Batch operational messages (see Table BATCH-1).

DUMP↵

Output a core dump of the current job on the Batch output device. This command can be issued at any time. Batch automatically continues at the next job.

EFOUT↵

Write an end-of-file on the Batch output device. Batch responds by typing "SPECIFY NEW OUTPUT DEVICE" followed by an asterisk. The operator must then type an "OUTPUT dev:.ext" command followed by a CONTINUE command.

END↵

Terminate the current job immediately.

EXAMPLES

```
.ASSIGN DSK BPTEMP↲                          Assign the Batch scratch file, BPTEMP,
DSK ASSIGNED↲                                to the disk.

.R BATCH↲                                    Load Batch.
*INPUT PTR:↲                                 Assign the paper tape reader as the
                                             input device.

*OUTPUT LPT:↲                                Assign the line printer as the output
                                             device.

*SKIP 0↲                                     Initialize Batch execution; begin with
                                             the first job found on the input device.

$JOB TEST01,12 200  16,11 NANCY↲             Contents of the first $JOB card are
                                             typed out.

MOUNT TAPE D2621 WRITE ENABLED↲              Message to operator to mount a DEC-
                                             tape, serial number 2621, on a DEC-
                                             tape drive (write enabled).

DTA6:↲                                       Operator mounts the tape on an avail-
                                             able DECtape unit, either before or after
                                             typing the physical name and number
                                             of the drive.

OK↲                                          Batch responds with message "OK."
*CONTINUE↲                                   Operator types "CONTINUE" to re-
                                             sume execution.

BREAK↲                                       Complete the current job, but halt be-
                                             fore continuing to the next job.

RUN TIME  -  23    SECS.↲                    Run time typed out.
*SKIP 2 ↲                                    Skip the next two jobs and continue at
                                             the third job.

$JOB  PASS1  BEAVER, 12 200  16,11 NANCY↲    Typeout of $JOB card.
DUMP↲                                        Perform a core dump on the current job
RUN TIME  -  12    SECS.↲                    and terminate the job; resume processing
                                             at the next job.

$JOB  PASS2  BEAVER, 12 200  16,11 NANCY↲    Typeout of $JOB card.
RUN TIME  -  32    SECS.↲
END OF BATCH ↲                               All jobs on the Batch input device have
                                             been processed.

*↑C↲                                         Return to the Monitor.
.KJOB ↲                                      Kill the job, deassign the BPTEMP de-
                                             vice, release core.

.
```

Table BATCH-1 Batch Diagnostic Messages

Message	Meaning
xxx xxx ? dev ?	The previous operator typein, xxx xxx has no meaning to Batch. An illegal device was specified in an INPUT or OUTPUT command.
END OF BATCH	The end of file has been reached on an input stack (nondirectory devices). If the Batch input stack is on a directory-oriented device, the message can mean that either (1) end of the input stack has been reached, or (2) no such file exists on the device.
FILE BEING MODIFIED	A selected disk file is currently being accessed by another user.
MAX. TIME EXCEEDED	The time limit specified on the $JOB card has been exceeded. The operator can respond with a typein of "TIME" to extend the time limit indefinitely.
MOUNT TAPE label WRITE $\begin{Bmatrix}\text{ENABLED}\\\text{LOCKED}\end{Bmatrix}$	Operator looks for available tape unit (if label begins with a "D," a DECtape unit is being requested; if label begins with an "M," a magnetic tape unit is being requested) and types in the physical name and number of the device in the form DTAn:↲ or MTAn:↲ If the device is available, Batch responds with "OK." The operator then mounts the requested tape (identified by the serial number "label") on the drive, sets the WRITE switch in the proper position, and types "CONTINUE" to resume processing.
NO PTY'S AVAILABLE	No pseudo-Teletypes are available. The user must wait until one of the current Batch jobs is finished. NOTE: The user can increase the number of pseudo-Teletypes available by recreating his Monitor via Build. There are no hardware limitations (except core - each additional pseudo-Teletype requires about 20_8 words of core and each possible time-shared "job" requires about 50_8 words).
NO SUCH JOB	The operator typed a SKIP command with an argument which resulted in a reference to a nonexistant job number on a directory device.
NO SUCH UFD	Reference has been made to a nonexistant UFD (User File Directory).

Table BATCH-1 (cont) Batch Diagnostic Messages

Message	Meaning
dev NOT AVAILABLE	The device either does not exist or has been assigned to another job. The "MOUNT......" message is repeated.
OK *	Typed following a correct operator response. Type "CONTINUE" to resume processing.
PLEASE ASSIGN dev TO BATCH PROCESSOR *	Operator must: 1. Return to Monitor level (↑C); 2. Assign the requested device (ASSIGN dev); 3. Type "CONTINUE"[1] to Monitor; and 4. Type "CONTINUE" to Batch.
PLEASE ASSIGN DEVICE BPTEMP *	The operator has failed to assign a device to the Batch scratch file, BPTEMP. He must: 1. Return to Monitor level (↑C); 2. Assign the device (ASSIGN dev BPTEMP); 3. Type "CONTINUE"[1] to Monitor; and 4. Type "CONTINUE" to Batch.
PLEASE MOUNT A SCRATCH TAPE	Operator responds by typing in the name of the drive on which the tape has been mounted (e.g., DTA3, DTA7 etc.).
PROTECTION FAILURE	The disk file referenced was protected.
RUN TIME - n SECS.	The total running time for each user job is typed at the end of the job.
SPECIFY NEW OUTPUT DEVICE *	This message appears following the operator's use of the EFOUT command. The operator must type OUTPUT dev:↵ before typing CONTINUE.
dev : TRANSMISSION ERROR	An I/O data or device error has occurred.
TRIED RENAME TO EXISTING NAME	This is the result of attempting to rename an existing file to a name that already exists on the disk (applies to disk files only).
USER NEEDS DEVICE dev *	When device is available, operator types "CONTINUE" to resume processing.

NOTE: 1. Typing "REENTER" (instead of "CONTINUE") to Monitor causes Batch to respond with an asterisk, following which the operator types "CONTINUE" to Batch. (If "CONTINUE" is typed to Monitor, Batch responds only with a carriage return.)

This section was not available at publication time, but will soon be available as an insert.

If you do not receive this insert, please write to:

DEC Program Library
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

SYSTEM BUILDER (BUILD)
(FOR TIME-SHARING MONITORS)

FUNCTION

To construct a time-sharing monitor specialized for the user's particular machine configuration.

- Permits the distribution of Monitor as a set of modular subprograms, the selection of which is determined by the user's requirements
- Employs an easy-to-use dialogue technique for requesting information from the operator

ENVIRONMENT

Monitor	10/40, 10/50, or Minimal Monitor
Minimum Core	3K plus core needed for monitor being built.
Additional Core	Uses as much core as given in R(UN) command.
Equipment Required	One DECtape unit for the input Build tape, containing the following files: SYS50 - for 10/50 systems SYS40D - for 10/40 systems with a disk SYS40N - for 10/40 systems without a disk

irrelevant

NOTE: An up-to-date set of instructions for constructing your time-sharing system and "getting on the air" upon delivery of your PDP-10 is included in the software kit accompanying your machine.

COMMANDS

.AS DTA3 DTAI ↵ DTA3 ASSIGNED↵	DTAI is the logical device name assigned to the Monitor library file.
.R BUILD core↵	Core available can be specified as any number of 1 K modules.

The user has the option of using an existing command list file as input to Build instead of conducting the dialogue described below (the first question asked is whether the user has such a file on a device). If the user has no such file (this would, of course, be the case when a given configuration is to be built for the first time), he may have one created as a by-product on some retrievable device as he conducts his Teletype dialogue; whether or not such a file is created is determined by the user's response to the second question asked. Thus, for all subsequent building operations where the same configuration is desired, the user can utilize the command list file created during the first building process.

The file created is automatically given the extension LST, and any command list file read by Build is expected to have the same extension (thereby ensuring compatibility); the user may not type an extension. If the user types the filename only, the device is presumed to be DTAI, the Build input device.

a. TYPE "DEVICE: NAME<CR>" OF FILE WHERE ANSWERS ARE PRESTORED↵ TYPE <CR> ONLY FOR TELETYPE DIALOGUE↵	If a file is specified, only question j will be asked out of those listed below.
b. TYPE "DEVICE: NAME<CR>" TO CREATE A COMMAND FILE; <CR> FOR NONE↵ DSK: SYSABC↵	
c. IS EITHER A 10, 20, OR 30 SYSTEM TO BE BUILT (TYPE Y OR N)?↵	If Y, the single-user Build code is written over the current time-sharing Build code and begins executing.
N↵	A 10/40 or 10/50 system is to be built.
d. IS A 10/40 SYSTEM TO BE BUILT (Y OR N)?↵	Type Y if 10/40; type N if 10/50.
Y↵ or N↵	
e. DO YOU HAVE A DISK?↵	Type Y if you have a disk; type N if you do not.
Y↵ or N↵	
f. DO YOU HAVE AN RD10 SWAPPING DISK?↵	
Y↵ or N↵	
g. DO YOU HAVE A PDP-10 PROCESSOR↵	
Y↵ or N↵	If response is N, a PDP-6 processor is assumed.
h. DO YOU WISH TO HAVE EXEC DDT LOADED (Y OR N)?↵	Type Y if you wish DDT loaded for Monitor testing; type N if you do not.
Y↵ or N↵	
i. DO YOU WISH TO HAVE LOCAL SYMBOLS LOADED (Y OR N)?↵	Needed only if EXEC DDT is loaded. Type Y if yes; type N if no.
Y↵ or N↵	NOTE: EXEC DDT and local symbols need be loaded only for Monitor debugging.

j. TYPE NAME OF THIS SYSTEM (10 CHARACTERS OR LESS)↵

VERSION10↵

Type in the name you wish assigned to this monitor version. This name is printed on the Teletype whenever this version is loaded.

k. TYPE NAME OF SYSTEM DEVICE↵

DSK↵

Type the name of the device to which the logical name SYS is to be assigned.

l. WHICH OF THE FOLLOWING LINE SCANNERS DO YOU HAVE?↵

DCS (DATA COMMUNICATION SYSTEM 630)↵
DLS (DATA LINE SCANNER DC10)↵
CCI (COMPUTER-COMPUTER INTERFACE DA10 WITH PDP-8 AND 680 SYSTEM)↵
TYPE DEVICE MNEMONIC↵

DLS↵ (or either of the other two)

m. DO YOU HAVE ANY OF THE FOLLOWING?↵
TYPE Y OR N↵

Type Y if you have the device or feature; type N if you do not.

PT READER?↵

Y↵ or N↵

PT PUNCH?↵

Y↵ or N↵

PLOTTER?↵

Y↵ or N↵

LINE PRINTER?↵

Y↵ or N↵

CARD READER?↵

Y↵ or N↵

IS YOURS A MODEL CR-10 PDP-10 CARD READER?↵

Y↵ or N↵

DISPLAY?↵

Y↵ or N↵

HOW MANY (DECIMAL) OF EACH OF THE FOLLOWING DO YOU HAVE↵

Type a decimal number signifying the number of each type of device.

DEC TAPES?↵

n↵

n = 0 through 8 (Type 0, not N, if none).

MAG TAPES?↵

n↵

n = 0 through 8 (Type 0, not N, if none).

DO YOU HAVE A TM-10 MAG TAPE CONTROL (Y OR N)?↵

Y↵ or N↵

n. JOBS TO RUN AT ONE TIME (BOTH ATTACHED AND DETACHED)?↵

n↵

o. PSEUDO TELETYPES?↵

n↵ .

p. TYPE "DEVICE-MNEMONIC: CHANNEL" FOR ANY NONSTANDARD DEVICES
TYPE ALT-MODE WHEN THROUGH↵

Enter the device mnemonics and priority channels for any of the user's own device service routines.

*MET:6

* (ALTMODE)

q. TYPE "SYMBOL = VALUE" FOR ANY CHANGES (VALUE IN DECIMAL)
TYPE ALT-MODE WHEN THROUGH↵

Enter symbol and value for any global symbol to be changed, in the form
symbol = value

JIFSEC = nn↵

Power source frequency. Standard = 60.

NSPMEM = nnnn↵

Memory cycle μs X 1000. Standard = 2000.

DTTRY = n↵

Number of retries on DECtape errors. Standard = 4.

MTSIZ = nnn↵

Size of magnetic tape records (i.e., number of words in buffer). Standard = 128.

LPTSIZ = nn↵

Size of printer buffer (in number of words). Standard = 24.

DETDDB = n↵

Maximum number of detached jobs. Standard = 0.

NOTE: (Number of jobs run at one time)-(number of Teletypes) = n

STDENS = n↵

Magnetic tape density and parity

STDENS = D + P

D	P
1 (200 bpi)	0 (odd)
2 (556 bpi)	4 (even)
3 (800 bpi)	

Standard = 2 (~~200~~ 556 bpi, odd)

Global = PI-number

User can also change standard priority interrupt assignments.

Device	Global	Standard 10/40N	 10/40D, 10/50
Paper tape rdr	PRTRCHN	4	5
Paper tape pnch	PTPCHN	5	6
Line printer	LPTCHN	4	5
Card reader	CDRCHN	3	4
Display	DISCHN	6	6
Scanner	SCHCHN	4	5
C. Teletype	CTYCHN	4	5
Arith. Proc.	APRCHN	3	4
Light pen	PENCHN	5	6
Clock	CLKCHN	7	7
D/Ctl - DEC/mag	DCTCHN	2	3
D/Ctl - Disk	DCBCHN	-	1
DECtape	DTCCHN	5	5
Magnetic tape	MTCCHN	4	5
Disk	DSKCHN	-	6

PTRCHN = 3↵

If on 10/40N system, changes PI assignment of paper tape reader from 4 to 3.

ALTMODE↵

r. TYPE"DEVICE:NAME"FOR ANY SPECIAL ROUTINES TO BE LOADED.
TYPE ALT-MODE WHEN THROUGH↵

DTA1:XYZSER↵

Enter device and routine name for any routine to be included in the Monitor but not linked with the rest of Monitor (i.e., not linked to either the priority interrupt chain or with the other data (blocks); also for any device specified in response to the query "TYPE..... FOR ANY NONSTANDARD DEVICES" whose binary code is not in the appropriate Build file. All files must be in Macro-output, Loader-input (relocatable binary) format.

ALTMODE↵

Delay occurs after ALTMODE until loading of routines is completed.

s. TYPE "DEVICE:NAME <CR>" FOR STORAGE MAP:
<CR> FOR NONE↵

LPT:↵ or ↵

LPT NOT AVAILABLE↵
TYPE "DEVICE:NAME<CR>" FOR STORAGE MAP: <CR> FOR NONE↵

DSK: SYSABC↵

File SYSABC.MAP will be written on the disk from which it can be copied at some later point in time.

t.

```
EXIT↵
↑C↵
```

BUILD overlays itself with the constructed Monitor and exits.

u.

```
.SAVE DTA1 filename.ext↵
JOB SAVED↵
↑C↵
.
```

User must now save the constructed Monitor on a DECtape.

DIAGNOSTIC MESSAGES

Table BUILD-1 Build Diagnostic Messages

Message	Meaning
CANNOT ENTER FILE	The file-structured device for the command list file or the storage map is unable to receive an entry into its directory. Try: a. DECtape - Directory full b. DSK: - User file directory is write protected, or the file is being modified.
ERROR IN LOADER COMMAND	A system or hardware malfunction has occurred.
ILL. FORMAT LIBRARY TAPE BAD	Build input tape is in error. Try to rerun. If second attempt fails, recreate the Build input tape and run again.
INPUT ERROR LIBRARY TAPE BAD	A read error has occurred on the Build input tape. Try to rerun. If second attempt fails, recreate the Build input tape and run again.
name.LST NOT FOUND ON DEVICE dev (Preceding message is retyped)	User specified a command list filename that did not exist on the particular device.
symbol old-val new-val MUL.DEF GLOBAL	A global symbol has been multiply defined. The old value is accepted, the new value is ignored, and processing continues.
dev NOT AVAILABLE	Device is not available. Request another device.
NOT ENOUGH CORE	Occurs only when Build is being run in a time-sharing environment. Assign more core and rerun Build. If this method fails, try one or more of the following. a. Run Build under a monitor containing only DECtape and line printer service routines. b. Do not load either EXEC DDT or local symbols.
PROGRAM MUST BE RESTARTED WITH "RUN" COMMAND	User typed "START", but part of the START code has been overwritten by Build. User must use GET or RUN command to bring new copy of Build into core.
filename.REL NOT FOUND or SYS....REL NOT FOUND	a. One of the user's programs requested for inclusion in the Monitor cannot be found on the device specified. Correct and re-enter the request. b. User has wrong library input file for configuration desired; check answers to questions c, d, and e of dialogue.
symbol NOT FOUND TYPE "DEVICE:NAME<CR>" FOR STORAGE MAP; <CR> FOR NONE	A device data block or device interrupt service routine cannot be found. Try the following:

Table BUILD-1 Build Diagnostic Messages (Cont)

Message	Meaning
LPT: [Prints map on line printer.] EXIT .	a. If the user is including his own routines in the Monitor, he should check that the tags on his device data blocks and interrupt service routines are spelled correctly and are globals. b. Check if a device service routine is missing from the input library file. c. Check storage map for symbols. d. Restart Build from the beginning.
UNDEFINED GLOBALS	This message may appear at the end of the requested storage map printout on either the line printer or the Teletype. If all input to Build is supplied by Digital, this message should not occur.
WHAT ?	User has made a keyin error. Correct and re-enter.

SYSTEM BUILDER (BUILD) (FOR SINGLE-USER MONITORS)

FUNCTION

To construct a single-user Monitor specialized for the user's particular machine configuration.

- Permits the same modular distribution of Monitor components and user specialization and utilizes the same easy-to-use dialogue technique as the time-sharing version.

ENVIRONMENT

Monitor	10/20, 10/30, or Minimal Monitor
Minimum Core	8K
Additional Core	Not used
Equipment Required	One DECtape unit for the input Build tape, containing the following files: SYSPAR SYSDEV

NOTE: An up-to-date set of instructions for constructing your single-user system and "getting on the air" upon delivery of your PDP-10 is included in the software kit accompanying your machine.

COMMANDS

```
·INIT↵
·AS DTA3 DTAI↵
```

DTAI is the logical device name assigned to the Monitor library file.

```
DTA3 ASSIGNED↵
·RUN DTAI BUILD↵
DO YOU HAVE FLOATING POINT AND BYTE
HARDWARE?↵
Y↵  or  N↵
```

If answer is N(o), simulators for these functions will be loaded as part of the Monitor.

```
DO YOU HAVE ANY OF THE FOLLOWING?
TYPE Y OR N↵
PT READER?↵
Y↵  or  N↵
PT PUNCH↵
Y↵  or  N↵
LINE PRINTER ?↵
Y↵  or  N↵
CARD READER?↵
Y↵  or  N↵
DISK?↵
Y↵  or  N↵
HOW MANY OF EACH OF THE FOLLOWING DO
YOU HAVE↵
DECTAPES?↵
n↵
```

Where n is in the range 0 through 8.

```
MAGTAPES?↵
n↵
```

Where n is in the range 0 through 8.

```
TYPE "DEVICE-MNEMONIC:CHANNEL" FOR
ANY NONSTANDARD DEVICES↵
TYPE ALT-MODE WHEN THROUGH↵
```

Enter the device mnemonics and priority channels for any of the user's own device service routines.

```
*MET:3  (ALTMODE)↵
```

```
TYPE "SYMBOL = VALUE" FOR ANY CORRECTIONS
DESIRED↲

TYPE ALT-MODE WHEN THROUGH↲

*PTRCHN = 3  ALTMODE ↲
```

The value of any global symbol can be changed by typing

symbol = value

Among the globals whose values can be changed (and their standard values in decimal) are:

DTTRY = 3 (Number of retries on DECtape errors)

MTSIZ = 128 (Size of magnetic tape records, i.e., number of data words in the buffer)

LPTSIZ = 24 (Size of line printer records, i.e., number of data words in the buffer)

STDENS = 2 (Magnetic tape density and parity - see page BUILD-4)

Priority channel interrupt assignments

Device	Global	Standard Assignment
Paper tape reader	PTRCHN	4
Paper tape punch	PTPCHN	5
Line printer	LPTCHN	5
Card reader	CDRCHN	4
Teletype	TTYCHN	7
Arithmetic Processor	APRCHN	
DECtape	DTACHN	3
Magnetic tape	MTACHN	2
Disk	DSKCHN	1

```
TYPE "DEVICE:NAME" FOR ANY SPECIAL
ROUTINES TO BE LOADED.↵
TYPE ALT-MODE WHEN THROUGH↵
*DTA2:XYZSER ALTMODE ↵
```

Enter device and routine name for any routine to be included in the Monitor but not linked with the rest of the Monitor (i.e., not linked to either the priority interrupt chain or with the other data blocks); also for any device specified in response to the query "TYPE......FOR ANY NONSTANDARD DEVICE" whose binary code is not in the appropriate Build file. All files must be in Macro-output, Loader-input (relocatable binary) format.

```
WHAT IS THE SIZE (IN K) OF CORE?↵
nn↵
```

Type the size of core in decimal; the value must be a multiple of 8.

```
DO YOU WANT A STORAGE MAP?↵
Y↵  or  N↵
```

If a storage map is requested, an INIT is performed by Build on the line printer. If a line printer is not available, the storage map is printed on the user's Teletype.

Build overlays the old system with the newly generated Monitor, initializes it, and exits to the new Monitor. The new Monitor responds with a dot. At this point the user can begin typing commands to the new Monitor, or first save the new Monitor on DECtape by typing

```
.AS DTA0 SYS↵
.REENTER↵
.
```

This causes the new Monitor to be saved on the SYS device (DTA0, in this example).

DIAGNOSTIC MESSAGES

The diagnostic messages for the single-user Monitor Build are the same as those for the time-sharing Monitor Build (see Table BUILD-1).

TIME-SHARING MONITORS

FUNCTION

To schedule multiple-user time sharing of the system, to allocate available facilities to user programs, to accept input from and direct output to all system I/O devices, and to relocate and protect user programs in core.

- Provide an advanced, third-generation, multiprogramming, and time-sharing environment
- Allow for a wide range of system facilities, from a minimum configuration of 16K of core and two DECtapes up through 262K of core and a variety of devices, such as magnetic tapes, disks, displays, plotters, real-time digitizers, and analog converters

246K

MONITOR 10/40

A proven multiprogramming time-sharing system which includes an I/O controller, run-time selection of I/O devices, job-to-job transition, job save and restore features, and memory dump facilities. All of these features are incorporated with concurrent real-time processing, batch processing, and time sharing.

MONITOR 10/50

A full-range, disk-swapping, multiprogramming time-sharing system incorporating all of the features of Monitor 10/40 with greatly extended capacity.

Both of these systems are custom tailored to the user's needs by use of the System Builder.

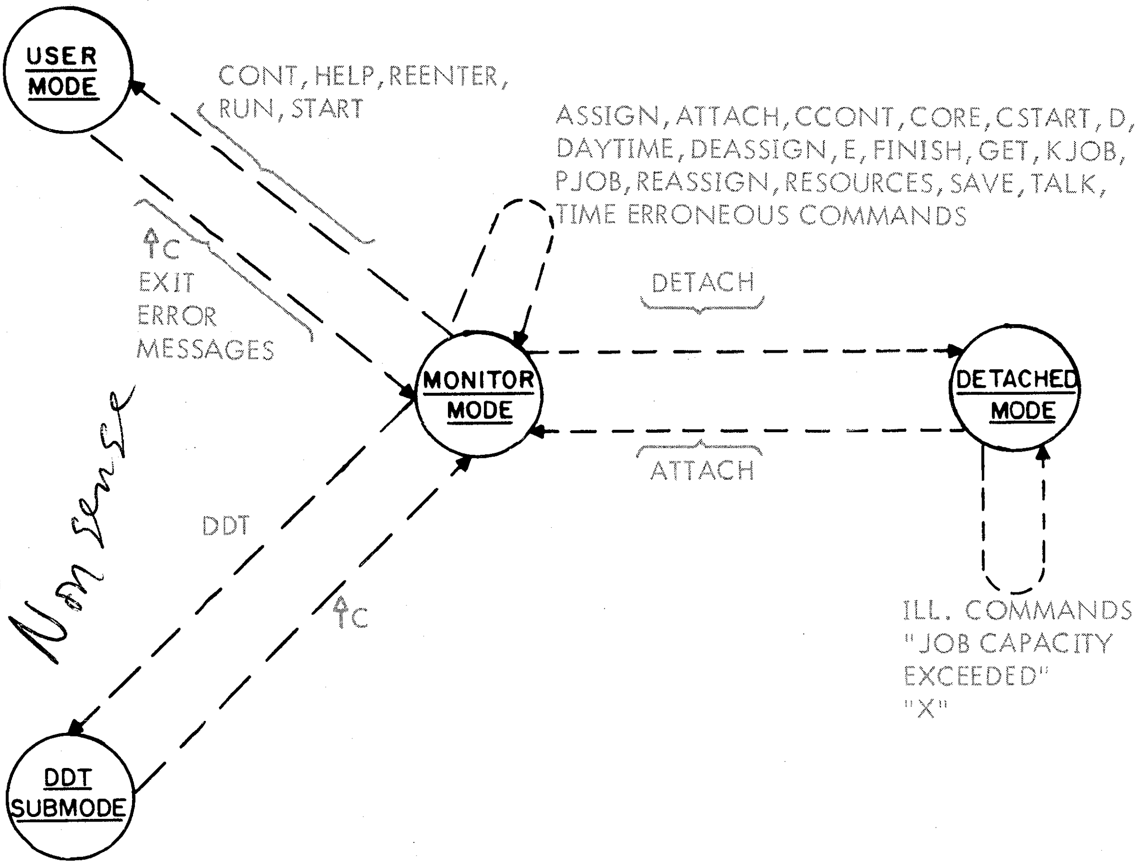
CONSOLE MODES

Monitor Mode — The console is in communication with the Monitor. All characters typed in are presented to and interpreted by the Monitor Command Interpreter.

User Mode — The console acts as an ordinary I/O device under control of the user's program.

DDT Submode — A special user Teletype I/O mode which does not interfere with the normal user I/O mode.

Detached Mode — The console is not in communication with either the Monitor Command Interpreter, DDT, or a user's job. Entered when the Monitor is first initialized or when DETACH is typed.


USER MODE
CONT, HELP, REENTER, RUN, START
ASSIGN, ATTACH, CCONT, CORE, CSTART, D, DAYTIME, DEASSIGN, E, FINISH, GET, KJOB, PJOB, REASSIGN, RESOURCES, SAVE, TALK, TIME ERRONEOUS COMMANDS
↑C
EXIT
ERROR
MESSAGES
DETACH
MONITOR MODE
DETACHED MODE
ATTACH
DDT
↑C
ILL. COMMANDS
"JOB CAPACITY EXCEEDED"
"X"
DDT SUBMODE

This is Nonsense

Figure MONITOR-1 Console Teletype Mode

Table MONITOR-1 Time-Sharing Monitor Commands

	Format	Action	Charac-teristics	Diagnostic Messages
TO GAIN ACCESS TO THE SYSTEM	HELP	The HELP command calls in a routine which carries on a dialogue with the user at the console to explain the use of the Monitor.	u	
	LOGIN (10/50 Monitor)	LOGIN initializes a Monitor routine to accept the user's LOGIN data. .LOGIN↵ JOB n↵ Job number assigned to user. xxx.xxx↵ Monitor version being used. proj,prog↵ User types in his project-programmer number (each number can be up to nine octal digits). [illegible] Monitor types out password mask; user types in password over mask. ↑C↵ . If user entries are correct, Monitor responds with↑C and a period, indicating readiness to accept a command.	u	LOGIN PLEASE ? Command typed requires that the user be logged in. ? SORRY --- WRONG NUMBER An illegal project-programmer number was entered. ?INCORRECT CODE --- TRY AGAIN An illegal password was entered.
TO ALLOCATE FACILITIES	ASSIGN phys-dev log-dev	To assign an I/O device to the user's job for the duration of the job or until a DEASSIGN command is given. phys-dev Any device listed in Table MONITOR-2.[1] This argument is required. log-dev A logical name assigned by the user (e.g., in writing a program, the user may use arbitrarily selected device names which he assigns to the most convenient physical devices at run time). This argument is optional When a device is assigned to a job, it is removed from the Monitor's pool of available devices.	m L, J	dev: ASSIGNED The device has been successfully assigned to the job. NO SUCH DEVICE Device name does not exist. ALREADY ASSIGNED TO JOB n The device has already been assigned to another user's job. LOGICAL NAME ALREADY IN USE DEVICE dev: ASSIGNED The user has previously assigned this logical name to another device

[1] If DTA or MTA is used, Monitor performs a search for an available drive and then types out DTAn (or MTAn) ASSIGNED

~~NOTE: On~~e command string only is typed per line, and ~~each comm~~and string must be typed on a single line.

This is meaningless, i.e. each command is terminated with

Table MONITOR-1 Time-Sharing Monitor Commands (Cont)

	Format	Action	Characteristics	Diagnostic Messages
TO ALLOCATE FACILITIES (cont.)	ASSIGN SYS:dev	To change the systems device to device "dev." The user must be logged in under either[1,1] or [1,2] .	m L	
	DEASSIGN dev	Returns one or more devices, currently assigned to the user's job to the Monitor's pool of available devices. dev If this argument is not specified, all devices assigned to the user's job are deassigned. If this argument is specified, it can be either the logical or physical device name.	m L	NO SUCH DEVICE Device name does not exist. DEVICE WASN'T ASSIGNED The device isn't currently assigned to this job.
	REASSIGN dev job	Allows one job to pass a device to a second job without going through the Monitor device pool. dev The physical name of the device to be reassigned. Cannot be a user console. job The number of the job to which the device is to be reassigned.	m L,J,I	DEVICE dev WASN'T ASSIGNED The device isn't currently assigned to this job. JOB NEVER WAS INITIATED The job number specified has not been initialized. NO SUCH DEVICE The device does not exist. DEVICE CAN'T BE REASSIGNED A user's console Teletype cannot be reassigned.
	FINISH dev	Terminates any input or output currently in progress on the device. dev The logical or physical name of the device on which I/O is to be terminated. If no name is specified, I/O is terminated on all devices assigned to the job.	m L	NO SUCH DEVICE Either the device does not exist or it was not assigned to this job.
	TALK dev	To allow the user to type directly on another user's console. dev Must be one of the following CTY - Console Teletype TTYn - Where n can be in the range of 0 through 77.	m	BUSY The console addressed is either (1) not in the detached mode or the Monitor mode, or (2) is not positioned at the left margin.

Table MONITOR-1 Time-Sharing Monitor Commands (Cont)

	Format	Action	Characteristics	Diagnostic Messages
TO ALLOCATE FACILITIES (cont.)	TALK (cont.)	OPR - Operator's console (the Teletype designated as such when the Monitor was initialized).		
	CORE n	To modify the amount of core assigned to the user's job. n - Total number of 1K blocks of core to be assigned to the job from this point on. If n is omitted, Monitor types out a value representing the number of 1K blocks of unallocated core in its pool.	m J,A,I	
	RESOURCES	To print out all the available devices (except TTY's) and the number of free blocks on the disk.	m	
	DETACH dev	To assign the device "dev" to JOB 0, thus making it unavailable. The user must be logged in under [1,1].	m L	
	ATTACH dev	To return a detached device to the Monitor pool of available devices. The user must be logged in under [1,1].	m L	
TO CALL, LOAD, AND CONTROL PROGRAMS	RUN dev filename.ext [proj,prog] core	To load a core image from a retrievable storage device and start it at the location specified within the file (JOBSA). dev The logical or physical name of the device containing the core image. Omitted if it is the systems (SYS:) device. filename.ext The name of the file containing the core image; if .ext is omitted, it is assumed to be .DMP. [proj,prog] Project-programmer number; required only if core image file is located in a disk area other than the user's.	u L	dev: NOT AVAILABLE The device has been assigned to another job. NO SUCH DEVICE The device does not exist. nK OF CORE NEEDED There is insufficient free core to load the file. NOT A DUMP FILE The file is not a core image file. TRANSMISSION ERROR A parity or device error occurred during loading.

Table MONITOR-1 Time-Sharing Monitor Commands (Cont)

TO CALL, LOAD, AND CONTROL PROGRAMS (cont.)

Format	Action	Charac-teristics	Diagnostic Messages
RUN (cont.)	core Amount of core to be assigned if different from minimum core needed to load the program or from the core argument of the SAVE command which saved the file. Required only if dev: is MTAn:		
GET dev filename.ext [proj,prog] core S.M.V.	Same as RUN command except that Monitor types out JOB SETUP and does not start execution; also, if the core argument is not included it is assigned a value equal to the user's current core allocation (the SAVE core argument, if it had been used, is ignored).	m L,J,A	Same as RUN.
START adr	Begins execution of a program previously loaded with the GET command. adr The address at which execution is to begin if other than the location specified within the file (JOBSA).	u L,J,C,A,I	NO CORE ASSIGNED No core was allocated to the user when the GET command was given and no core argument was specified in the GET.
HALT (↑C)	Places the console in Monitor mode and transmits a HALT command to the Monitor Command Interpreter. Stops the job and stores the program counter in the job data area (JOBPC).	m	
CONT	Starts the program at the saved program counter address stored in JOBPC by a HALT command (↑C) or a HALT instruction.	u L,J,C,I	CAN'T CONTINUE The job was halted due to a Monitor-detected error and cannot be continued.
DDT	Copies the saved program counter value from JOBPC into JOBOPC and starts the program at an alternate entry point specified in JOBDDT (beginning address of DDT as set by Linking Loader). To return to normal execution, type ↑C and START (or type prog-start-adr$G).	u L,J,C,I	
REENTER	Similar to the DDT command. Copies saved program counter value from JOBPC into JOBOPC and starts program at an alternate entry point specified in JOBREN (must be set by the user or his program). To return to the interrupted computation, type REENTER	u L,J,C,I	

Table MONITOR-1 Time-Sharing Monitor Commands (Cont)

	Format	Action	Charac-teristics	Diagnostic Messages
TO CALL, LOAD, AND CONTROL PROGRAMS (cont.)	E adr	To examine a core location in the user's area. adr — If this argument is specified, the contents of the location are typed out in half-word octal mode. Adr is required the first time the E command is used. If adr is not specified, the contents of the location following the previously specified adr are typed out.	m L, J, C, I	OUT OF BOUNDS The specified adr is not in the user's core area.
	D 1h rh adr	Deposit information in the user's core area. 1h — The octal value to be deposited in the left half of the location. rh — The octal value to be deposited in the right half of the location. adr — The address of the location into which the information is to be deposited. If adr is omitted, the data is deposited in the location following the last location examined or deposited.	m L, J, C, I	OUT OF BOUNDS The specified adr is not in the user's core area.
	SAVE dev filename.ext core	To write out a core image of the user's core area on the specified device. If DDT was loaded with the program, the entire core area is written; if not, the area starting from zero up through the program break (as specified by JOBFF) is written. After output is completed, the message JOB SAVED is typed. dev — The device on which the core image file is to be written. filename.ext — The name to be assigned to the core image file. If .ext is omitted it is assumed to be .DMP.	m L, J, C, A, I	n 1K BLOCKS OF CORE NEEDED The user's current core allocation is less than the contents of JOBFF. DEVICE NOT AVAILABLE Device dev is assigned to another user. TRANSMISSION ERROR An error was detected while reading or writing the core image file. DIRECTORY FULL The directory of device dev is full; no more files can be added.

S.M.V

Table MONITOR-1 Time-Sharing Monitor Commands (Cont)

	Format	Action	Charac-teristics	Diagnostic Messages
TO CALL, LOAD, AND CONTROL PROGRAMS (*cont.*)	SAVE (cont.)	core — Amount of core in which the program is to be run. This value is stored in the job's core area (JOBCOR) and is used by the RUN and GET commands. Specified as number of 1K blocks. If core is omitted, only the number of blocks required by the core image area (as explained above) is assumed.		
TO CONTROL BACK-GROUND JOBS	PJOB	Monitor responds by typing the job number to which the user's console is attached. If the console is not attached to a job, Monitor assigns a job number and types the job number and a line identifying the Monitor version.	m L, J	
	CSTART CCONT	Identical to the START and CONT commands, respectively, except that the console is left in the Monitor mode. To Use: 1. Begin the program with the console in user mode. 2. Type control information to the program, then type↑C to halt job with console in Monitor mode. 3. Type CCONT to allow job to continue running and leave console in Monitor mode. 4. Further Monitor commands can now be entered from the console.	m L, J, C, I	Same as START and CONT.
	DETACH	Disconnects the console from the user's job without affecting the status of the job. The user console is now free to control another job, either by initiating a new job or attaching to a currently running background job.	d L	
	ATTACH job [proj, prog]	To connect a console to a background job. job — The job number of the job to which the console is to be attached.	m	If an error message occurs, the console remains attached to its current job.

Table MONITOR-1 Time-Sharing Monitor Commands (Cont)

	Format	Action	Charac-teristics	Diagnostic Messages
TO CONTROL BACK-GROUND JOBS (cont.)	ATTACH (cont.)	[proj, prog] The project-programmer number of the originator of the desired job. Automatically detaches the console from any job to which it is currently attached. If job is running, typing CONT places the console in the user mode without affecting the operation of the job.		TTYn ALREADY ATTACHED Job number typed is erroneous and is attached to another console, or another user is attached to the job. JOB NEVER WAS INITIATED The job number is not assigned to any currently running job. NOT. JOB ORIGINATOR The project-programmer number entered is not that of the originator of the desired job.
JOB TERMIN-ATION	KJOB	Stops all allocated I/O devices and returns them to the Monitor pool. Returns all allocated core to the Monitor pool. Returns the job number to the pool. Leaves the console in the Monitor mode. Performs an automatic TIME command.	m A	
SYSTEM TIMING	DAYTIME	Types the date followed by the time of day. Time is typed in the format. hhmm:ss.ss where hh = hours mm = minutes ss.ss = seconds to the nearest hundredth.	m	
	TIME job	Types out the total running time used by the job since it was initialized. Interrupt level and job scheduling times are charged to the user who was running when the interrupt or rescheduling occurred. job The job number of the job whose timing is desired.	m	Huh

Table MONITOR-1 Monitor Commands (Cont)

	Format	Action	Charac-teristics	Diagnostic Messages
SYSTEM TIMING (cont.)	TIME (cont.)	If job is omitted, the job to which the console is attached is assumed. In this case, Monitor types out the incremental running time (running time since last TIME command) as well as the total running time since the job was initialized. If job = 0, an approximation of the time spent core shuffling is printed, followed by the running time of the null job, and the total system up time.		

Characteristics:

d =places console in detached mode	L =LOGIN required (10/50 Monitor)	J =requires a job number
m =places console in Monitor mode	A =no active device	
u =places console in user mode	C =core required	
	I =must be in core	

Table MONITOR-2 Time-Sharing Monitor Device Summary

Physical Name	Device	~~Program~~ ~~Operato~~rs UUO's	Data Modes
CTY	Console Teletype	INPUT, OUTPUT	A, AL
TTYn (n = 0 thru 7)	Teletype ϕ thru 2ϕϕ	INPUT, OUTPUT	A, AL
PTR	Paper tape reader	INPUT	A, AL, IB, B, I
PTP	Paper tape punch	OUTPUT	A, AL, IB, B, I
LPT	Line printer	OUTPUT	A, AL
CDR	Card reader	INPUT	A, AL, B, I
DTAn (n = 0 thru 7)	DECtape	INPUT, OUTPUT LOOKUP, ENTER, USETO, USETI, UGETF, CALL [SIXBIT/ UTPCLR/]	A, AL, IB, B, I, DR, D
MTAn (n = 0 thru 7)	Magnetic tape	INPUT, OUTPUT MTAPE	A, AL, IB, B, I, DR, D
DSK	Disk	INPUT, OUTPUT LOOKUP, ENTER	A, AL, IB, B, I, DR, D

Data Modes:

A = ASCII
AL = ASCII line
IB = Image binary
B = Binary

DR = Dump records ~~(same as D)~~
D = Dump

mode 1
2
3
.
.
14
15
16
17

OPR
PTY
DIS
PLT

DIAGNOSTIC MESSAGES

In addition to the diagnostic messages given for each command, Monitor also types out the following error diagnostics. Except for the DEVICE dev OK? and HALT AT USER adr messages, all messages are for errors so serious that Monitor stops the job and does not allow the user to continue execution of the job with a CONT command.

Table MONITOR-3 Time-Sharing Monitor Diagnostic Messages

Message	Meaning
ADDRESS CHECK FOR DEVICE dev AT USER adr	Monitor has checked a user address and has found it to be too large (> C(JOBREL)) or too small (≤ JOBPFI). Some user addresses can be the user's accumulators while others cannot. One of the following addresses may be wrong: buffer buffer header dump mode command list data specified by dump mode command list insufficient core available for setting up Monitor-generated buffers.
BAD DIRECTORY FOR DEVICE DTAn; UUO AT USER adr	The DECtape directory is not in proper format or had a parity error when read. Many times this error occurs when an attempt is made to use a virgin tape.
DEVICE dev OK?	Device dev is temporarily in an inoperable state, such as LPT offline. The user should correct the obvious condition and then type a CONT command.
ERROR IN JOB n	A fatal error has occurred in the user's job (or in Monitor while servicing the job). This typeout is normally followed by a 1-line description of the error.
HALT AT USER adr	The user program has executed a halt instruction at loc. adr. Typing CONT will resume execution at the effective address of the halt.
HUNG DEVICE dev; UUO AT USER adr	A device has not generated an interrupt for a timed period and, therefore, is in need of attention.
ILLEGAL DATA MODE FOR DEVICE dev AT USER adr	The data mode specified for a device in the user's program is illegal.

Table MONITOR-3 Time-Sharing Monitor Diagnostic Messages (Cont)

Message	Meaning
ILLEGAL UUO AT USER adr	An illegal UUO has been executed at user location adr.
ILL INST. AT USER adr	An illegal operation code has been encountered in the user's program.
ILL MEM REF AT USER adr	An illegal memory reference has been made by the user program at adr or adr+1.
INCORRECT RETRIEVAL INFORMATION: UUO AT USER adr	The retrieval pointers for a file are not in the correct format; the file is unreadable. If this typeout occurs, the user should report it on a Software Trouble Report.
INPUT DEVICE dev CANNOT DO OUTPUT; UUO AT USER adr	An illegal OUTPUT UUO has been executed at user location adr.
I/O TO UNASSIGNED CHANNEL AT USER adr	No OPEN or INIT was performed on the channel.
JOB CAPACITY EXCEEDED or X	The job capacity of the system (i.e., the capacity selected when this Monitor was generated by Build) has been exceeded. The command is ignored. The user must wait until another user has relinquished his job number.
LOGIN PLEASE ?	A Monitor command requiring that the user be logged in has been typed, but the user is not logged in. Perform LOGIN and repeat the command.
LOOKUP AND ENTER HAVE DIFFERENT NAMES: UUO AT USER adr	An attempt has been made to read and write a file on the disk. However, the LOOKUP and ENTER UUO's have specified different names on the same user channel. This message does not indicate a DECtape error.
MASS STORAGE DEVICE FULL; UUO AT USER adr	The storage disk is full. Users must delete unneeded files before the system can proceed.
NON-RECOVERABLE DISC READ ERROR; UUO AT USER adr NON-RECOVERABLE DISC WRITE ERROR; UUO AT USER adr	Monitor has encountered an error while reading or writing a critical block in the disk file structure (e.g., the MFD or the SAT table). If this condition persists, the disk must be reloaded using Failsafe after the standard location for the MFD and SAT table has been changed using the Monitor once-only dialogue.

Table MONITOR-3 Time-Sharing Monitor Diagnostic Messages (Cont)

Message	Meaning
NOT ENOUGH FREE CORE IN MONITOR: UUO AT USER adr	The Monitor has run out of free core for assigning disk data blocks and Monitor buffers. If this typeout occurs, the user should report it on a Software Trouble Report.
NOT FOUND	The program file requested cannot be found on the systems device (or on the specified device).
OUTPUT DEVICE dev CANNOT DO INPUT; UUO AT USER adr	An illegal INPUT UUO has been executed at user location adr.
PC EXCEEDS MEMORY BOUND AT USER adr	An illegal transfer has been made by the user program to user location adr.
SWAP READ ERROR	A consistent checksum error has been encountered when checksumming locations JOBDAC through JOBDAC+74 of the Job Data area during swapping.
SWAPPING DEVICE FULL	The swapping device is full of user core images. One or more users must eliminate their core images by typing KJOB before the user can proceed. This message is printed every 30 seconds until space is made available.
TOO FEW ARGUMENTS	A required argument was omitted in a Monitor command string.

FUNCTION

To perform job-to-job transitions, assign I/O devices at run time, and perform all I/O device service functions for standard devices.

- Provide Monitor control of jobs and ease of programming for I/O devices in a single-user environment
- Provide upward compatibility with the time-sharing Monitors 10/40, 10/50
- Allow for a wide range of system facilities, from a minimum configuration of 8K of core (16K for the 10/30 Monitor) and paper tape up through 262K of core and the same range of devices given for the time-sharing Monitors

MONITOR 10/10 A single-user Monitor for an 8K paper tape system. Includes an I/O Controller, run-time selection of I/O devices, job-to-job transition, job save and restore features, and memory dumps.

MONITOR 10/20 A single-user Monitor for an 8K DECtape system. Includes the same versatile features as the 10/10.

MONITOR 10/30 A single-user Monitor for 16K and larger systems. Includes the same versatile features as the 10/10.

All of these systems are custom-tailored to the user's needs by use of the System Builder.

CONSOLE MODES

Monitor Mode
The console is in communication with the Monitor. All characters typed in are presented to and interpreted by the Monitor Command Interpreter.

User Mode
The console acts as an ordinary I/O device under control of the user's program.

DDT Submode
A special user mode which does not interfere with the normal user mode.

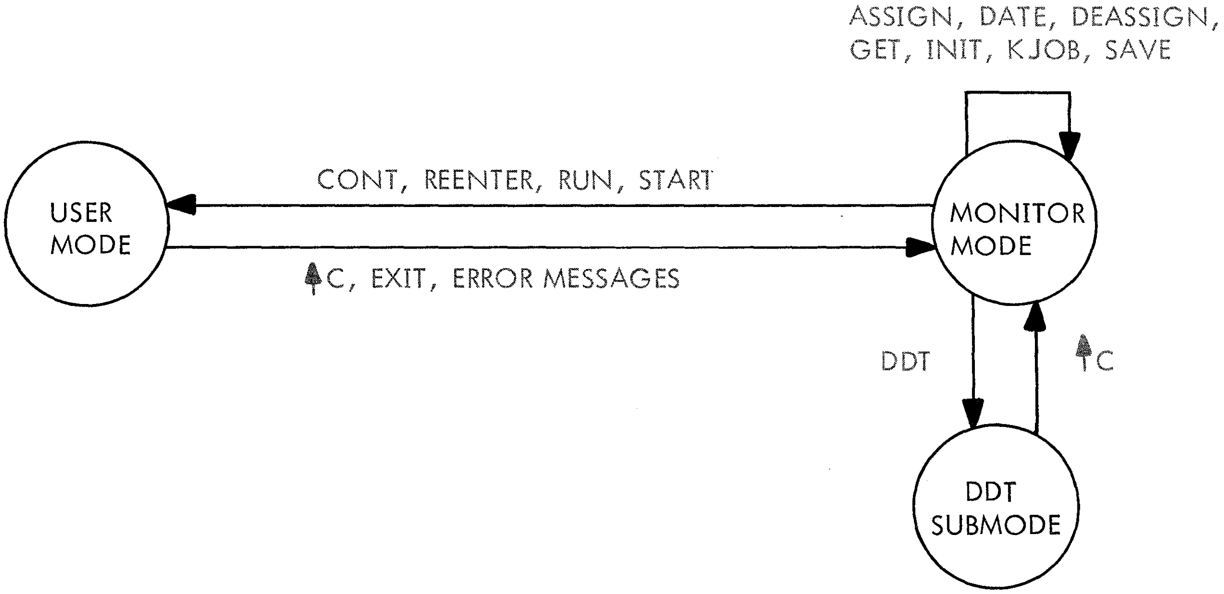


Figure MONITOR-2 Console Teletype Modes (Single-User)

Table MONITOR-4 Single-User Monitor Commands

	Format	Action	Charac-teristics	Possible Causes of "?" Diagnostic
TO INITIALIZE THE SYSTEM	INIT	Initializes the I/O package for the user as follows. 1. Releases the logical names (except SYS) assigned to every device. 2. Release the directory space of every device having a logical name. 3. Sets the contents of JOBREL equal to the address of the first location below the Monitor.	m	
TO ALLOCATE FACILITIES	ASSIGN phys-dev log-dev	To assign an I/O device to the user's job for the duration of the job or until a DEASSIGN command is given for the device. phys-dev Any device listed in Table MONITOR-5. This argument is required. log-dev A logical name assigned by the user (e.g., in writing his program, the user may use arbitrarily selected device names which he can then assign to the most convenient physical devices at run time). This argument is optional.	m	a) phys-dev could not be found. b) log-dev is already assigned.
	DEASSIGN dev	Releases the logical name from the device and core space allocated to its directory (if any); updates JOBREL dev This argument can be either the logical or physical device name.	m	dev could not be found.

Table MONITOR-4 Single-User Monitor Commands (Cont)

	Format	Action	Charac-teristics	Possible Causes of "?" Diagnostic
TO CALL, LOAD, AND CONTROL PROGRAMS	RUN dev progname	To load a core image from a retrievable storage device and start it at the location specified within the file (JOBSA). dev The logical or physical name of the device containing the core image. If this argument is omitted, the Systems (SYS) device is assumed. progname The name of the file containing the core image.	u	a) dev could not be found. b) progname could not be found. c) $(\text{JOBSA})_{\text{RH}} = 0$.
	GET dev progname	Same as the RUN command, except that the job is not started.	m	a) dev could not be found. b) progname could not be found.
	START	Gives control to a job at its starting address, as specified by the right half of JOBSA.	u	$(\text{JOBSA})_{\text{RH}} = 0$.
	↑C (HALT)	Places the console Teletype in Monitor mode and transmits a HALT command to the Monitor Command Interpreter. Stops the job and stores the program counter in the job data area JOBOPC.	m	
	CONT	Starts the program at the address stored in the saved program counter (JOBOPC) following a HALT command (↑C).	u	C(JOBOPC) = 0
	DDT	Starts execution of the program at the location specified in JOBDDT. This location is set at the starting address of DDT by Linking Loader.	u	$(\text{JOBDDT})_{\text{RH}} = 0$
	REENTER	Restarts an interrupted job at a point specified by the user in JOBREN; the program counter value is in JOBOPC. To return to the interrupt point, execute a JRST 2,@ JOBOPC.	u	$(\text{JOBREN})_{\text{RH}} = 0$

Table MONITOR-4 Single-User Monitor Commands (Cont)

	Format	Action	Characteristics	Possible Causes of "?" Diagnostic
TO CALL, LOAD, AND CONTROL PROGRAMS (cont.)	SAVE dev progname	Copies the contents of the user's core area (and part of the job data area) onto device "dev" and assigns it the filename "progname.SAV". If DDT is loaded, the area of core from JOBDDT through the address contained in JOBREL is saved: otherwise, the core area from JOBDDT through the job break (address in JOBFF) is saved.	m	a) dev could not be found. b) progname cannot be ENTERed on the device c) Transmission error
JOB TERMINATION	KJOB	Kills the job and releases every active device.	m	
SYSTEM TIMING	DATE mm-dd-yy	Converts the date typed by the user to the standard internal format and stores it in the system date location.	m	

Characteristics: m Command places user's console in Monitor mode.
u Command places user's console in user mode.

DIAGNOSTIC MESSAGES

Table MONITOR-5 Single-User Monitor Diagnostic Messages

Message	Meaning
?	Generally, this typeout means that a command was typed incorrectly (e.g., a space was omitted, a parameter was omitted or misspelled). Other reasons for this message are given under the heading "Possible Causes of the ? Diagnostic," Table MONITOR-4.

ASSEMBLY INSTRUCTIONS FOR CUSP SOURCES

The only source programs requiring special assembly or loading procedures are:

F4, F40, F4S, F40S
PIP and PIP1
DDT and Exec Mode DDT
GLOB
AID
DESK

Detailed instructions for assembling and loading the above programs follow. All other CUSP (Commonly Used System Programs) sources are assembled using Macro, and then simply loaded and saved.

F40

The PDP-10/PDP-6 FORTRAN IV Compiler consists of a single set of source files which, through conditional assemblies, produce the following:

1. F40/F4: Full-scale FORTRAN IV, which will compile programs on any PDP-10/PDP-6 having a minimum of 9K of user core available. The code it produces will run on the computer which compiled it.
2. F40S/F4S: An abbreviated version of F40/F4 which will operate in 5.5K of user core on the PDP-10/PDP-6.

The Compiler source consists of five files: EXEC, FX0, FX1, FX2, FX3.

F40 is the PDP-10 Compiler.

F4 is the PDP-6 Compiler.

F40.SAV is the name given to the F40 Compiler on the PDP-10 CUSP.

F4.DMP is the name given to the F4 Compiler on the PDP-6 CUSP.

To create F40.SAV or F4.DMP:

1. Assemble FX1, FX2, FX3 as one file (F40.REL or F4.REL).
2. Assemble EXEC as EXEC.REL.
3. Load EXEC, then F40.REL or F4.REL in 11K.
4. Save F40 or F4 in 9K.

To create F40.SAV or F4S.DMP:

1. Assemble FX0, FX1, FX2, FX3 as one file (F40S.REL or F4S.REL)
2. Load F40S.REL or F4S.REL in 8K.
3. Save F40S or F4S (in 6K for 10/40, 10/50 systems).

The executive (FX0 or EXEC) handles all input/output operations. Before assembling the files FX1, FX2, and FX3, the user may define some parameters to conserve space or give the Compiler additional features (see below). All the parameters except DEBUG have been defined in the small executive (FX0), enabling the abbreviated version to reside in approximately 5.5K of core instead of 9K.

HALFWD	Assemble interpretive instructions two per word.
$CCONS	The Compiler feature of collapsing constants will be removed.
$FAD	The Compiler will not use floating point instructions in its calculations. If $FAD is defined, $CCONS will be automatically defined.
$IMPL	The IMPLICIT statement will not be accepted.
$DATA	The DATA statement will not be accepted.
$CODE	The binary listing option will be removed.
$NAME	The NAMELIST statement will not be accepted.
$CREF	The cross-reference symbol table feature will be removed.
DEBUG	The debug package will be assembled.

ASSEMBLE/LOAD PROCEDURE

```
1.  F40:
    .R MACRO↲
    *DSK:EXEC, ← DTAn:EXEC↲
    *DSK:F40, ← DTAn:FX1.MAC, FX2.MAC, FX3.MAC↲
    *↑C↲
    .R LOADER↲
    *DSK:EXEC, F40 ALTMODE ↲
    LOADER↲
    EXIT↲
    ↑C↲
    .SAVE DSK F40 9↲

2.  F40S:
    .R MACRO↲
    *DSK:F40S, ← DTAn:FX0, FX1, FX2, FX3↲
    *↑C↲
    .R LOADER↲
    *DSK:F40S  ALTMODE ↲
    LOADER↲
    EXIT↲
    ↑C↲
    .SAVE DSK F40S 6↲
```

a. PIP: Assemble separately the files COMSN and PIP.

There are two assembly parameters in PIP: WCH and BLOC0.

WCH = 0	New (i.e., PDP-10) format DECtapes are assumed.
WCH = 1	Old (i.e., PDP-6) format DECtapes are assumed.

BLOC0 not defined	Block 0, 1, and 2 copying is allowed.
BLOC0 defined	No block 0, 1, and 2 copying is allowed.

Typical procedures for PIP assembly, load, and save are:

```
.R MACRO↵
*DTAn:PIP, ← TTY:,DTAm:PIP↵
WCH = 0↵
↑Z↵
END OF PASS ONE↵
WCH = 0↵
↑Z↵

*DTAn:COMSN, DTAm:COMSN↵
*↑C↵
```

Then load the two files as follows.

```
.R LOADER↵
*DTAn:COMSN,PIP ALTMODE ↵
LOADER↵
EXIT↵
↑C↵
.SAVE DTAn:PIP 3↵
```

b. PIP1: Assemble separately the files COMSCN.MAC and PIP1.MAC. Then load them as follows, noting the order.

```
.R LOADER↵
*DTAn:COMSCN,PIP1 ALTMODE ↵
LOADER↵
EXIT↵
↑C↵
```

Now save PIP1 on your CUSP

```
.SAVE DTAn:PIP1 1↵
```

c. QPIP is an ASCII file containing the switch information which PIP prints upon encountering the /Q switch in a command string. The QPIP file should simply be transferred to your CUSP using PIP.

a. User DDT is kept as a REL file and as a SAVED file on the CUSP tape.

Assemble the file DDT,MAC with Macro. DDT assembles with one argument 'A' error. This error may be ignored and does not affect the user at all. DDT.REL is transferred onto a CUSP via PIP, while the GET and SAVE Monitor commands are used for DDT.SAV.

How to Load and Save DDT.SAV

`·R LOADER 4↵`	Load in 4K of core[1]
`*DTA1:DDT,/140G ALTMODE↵`	
`LOADER↵`	
`EXIT↵`	
`↑C↵`	
`·START↵`	Enter DDT.
`$$H JOBSYM/ (777616)7616↵`	Type out, in halfword mode (JOBSYM).
`6! (7616)3616)↵`	Open register 6; put $(JOBSYM)_{RH}$ into $(6)_{LH}$; put $(JOBSYM)_{RH}-4000_8$ into $(6)_{RH}$.
`BLT 6, 3777$X↵`	Perform block transfer through 3777_8.
`JOBSYM! (777616)3616↵`	Open JOBSYM; leave left half as is; change right half to 4000_8 less than it was.
`$$Z↵`	Zero memory (except DDT).
`JOBSA/ (0)DDT (DDTEND)DDT↵`	Open JOBSA. If left half ≠ DDTEND, change it.
`$$S↵`	Change back to symbol type-out mode.
`↑C↵`	Return to Monitor level.
`·CORE 2↵`	Reduce core to 2K.
`·START↵`	Reenter DDT.
`JOBREL/ 3777↵`	Check JOBREL.
`↑C↵`	Return to Monitor level.
`·SAVE DTA1 DDT↵`	Save DDT on CUSP.

Explanation - The DDT saved file must be saved in 2K (minimum amount of core needed for it). Also, a starting address must be set up for DDT as location 140. To get DDT into 2K, the DDT symbol table must be moved down to the upper end of the first 2K of core. Any unused locations in DDT should be set to zero ($$Z) and JOBSYM should be set to the new location of the start of the DDT symbol table. Before saving the resulting file, a CORE 2 request should be given to the Monitor to ensure that DDT is saved as a 2K core image.

[1] In this example, the Loader ran in 4K of core. Even if the Loader is run in more than 4K of core, the same general principles apply.

b. EXEC DDT

The Exec mode version of DDT assembles from the file DDT.MAC by using a parameter file from the Teletype as shown below.

```
.R MACRO↵
*DTAn:EDDT,LPT: ← TTY:,DTAm:DDT↵
EDDT = 1↵
↑Z↵
END OF PASS ONE↵
EDDT = 1↵
↑Z↵
*
```

There are other versions of DDT that may be assembled. Please refer to the first page of the source file for explicit instructions.

GLOB

The three source files of the global cross-reference program (CROSSX, CROSS, and SCAN) are assembled separately by Macro and loaded as one file. The specific loading instructions are:

```
.R LOADER↵
*DTAn: CROSSX, CROSS, SCAN ALTMODE ↵
LOADER↵
EXIT↵
↑C↵
.SAVE DTAm: GLOB 2↵
```

AID

There are three source files for AID: INTERP, ARITH, and KMON. Assemble these files separately and load as follows.

```
.R LOADER↵
*DTAn:KMON,ARITH,INTERP ALTMODE ↵
LOADER↵
EXIT↵
↑C↵
.SAVE DTA1 AID 11↵
```

The three source files of the Desk Calculator are: FOP, NUMBER, and DECS. Assemble these separately and load them as one file. The specific loading instructions are:

```
.R LOADER↵
*DTAn:FOP, NUMBER, DECS (ALTMODE)↵
LOADER↵
EXIT↵
↑C↵
.SAVE DTAn:  DESK n↵                    n = 4 if macro capability is not desired;
                                            4 if it is desired.
```

digital

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Printed in U.S.A.