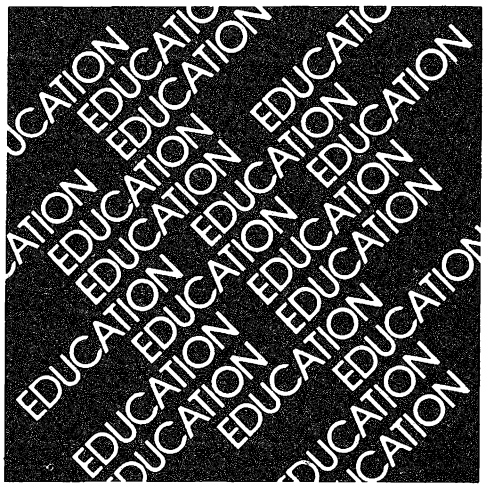
digital

PDP-15
SYSTEM SOFTWARE
PROGRAM SOLUTIONS

PDP-15
SYSTEM SOFTWARE
PROGRAM SOLUTIONS



EDUCATIONAL SERVICES

digital equipment corporation • maynard. massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

The software described in this document is furnished to the purchaser under a license for use on a single computer system and can be copied (with inclusion of DIGITAL's copyright notice) only for use in such system, except as may otherwise be provided in writing by DIGITAL.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.

The following are trademarks of Digital Equipment Corporation:

CDP	DIGITAL	INDAC	PS/8
COMPUTER LAB	DNC	KA10	QUICKPOINT
COMSYST	EDGRIN	LAB-8	RAD-8
COMTEX	EDUSYSTEM	LAB-8/e	RSTS
DDT	FLIP CHIP	LAB-K	RSX
DEC	FOCAL	OMNIBUS	RTM
DECCOMM	GLC-8	OS/8	RT-11
DECTAPE	IDAC	PDP	SABR
DIBOL	IDACS	PHA	TYPESET 8
			UNIBUS

```
                        .TITLE PROGRAM (#1) COUNTS # OF BITS SET IN DSW
                       /PROGRAM COUNTS THE NUMBER OF BITS SET TO 1 IN THE DATA
                       /SWITCH REGISTER,  BY ROTATING THE VALUE THROUGH THE
                       /AC AND CHECKING TO SEE IF BIT 17 IS SET.  THE RESULT
                       /IS PLACED IN THE ACCUMULATOR SO THAT IT IS DISPLAYED
                       /ON THE CONSOLE LIGHTS WHEN THE PROGRAM HALTS.
                       /
                       /RELOCATABLE PROGRAM
                       /
00000 R 740040 A       START   HLT             /HOW FAST ARE YOU?
00001 R 750004 A               LAS             /READ DATA SWITCHES INTO AC.
00002 R 040022 R               DAC TEST        /SAVE VALUE IN TEST.
                       /
00003 R 777756 A               LAW -22         /PUT -18 DECIMAL INTO AC
00004 R 040024 R               DAC COUNT       /AND STORE IT IN COUNT.
00005 R 140023 R               DZM BITS        /CLEAR LOCATION BITS TO ZERO.
                       /
00006 R 200022 R       NEXT    LAC TEST        /LOAD AC WITH TEST.
00007 R 500025 R               AND (1          /GET RID OF ALL BUT BIT 17.
                       /
00010 R 740200 A               SZA             /SKIP IF AC=0.
00011 R 440023 R               ISZ BITS        /BIT 17=1, SO ADD 1 TO BITS.
00012 R 200022 R               LAC TEST        /GET TEST
00013 R 740020 A               RAR             /ROTATE 1 PLACE TO THE RIGHT.
00014 R 040022 R               DAC TEST        /REPLACE VALUE OF TEST.
                       /
00015 R 440024 R               ISZ COUNT       /SEE IF DONE 18 TIMES YET.
00016 R 600006 R               JMP NEXT        /NOT DONE, GO BACK FOR NEXT BIT.
                       /
00017 R 200023 R               LAC BITS        /DISPLAY BIT COUNT IN AC.
00020 R 740040 A               HLT             /AND HALT. HIT CONTINUE TO GO
00021 R 600000 R               JMP START       /BACK AND GET ANOTHER VALUE.
                       /
00022 R 000000 A       TEST    0
00023 R 000000 A       BITS    0
00024 R 000000 A       COUNT   0
                       /         ****** POINTS TO BE NOTED ******
                       /
                       /*1. WHY DO WE NEED THE HALT INSTRUCTION IN LOCATION 0?
                       /
                       /*2. WHY IS THERE A "LAW -22" IN LOCATION 3? WHAT ARE TH
                       /ADVANTAGES OF USING A "LAW -22" INSTRUCTION INSTEAD OF
                       /"LAC (-22" ?
                       /
                       /*3. WHEN THIS PROGRAM IS LOADED, LOCATION "BITS"
                       /LOCATION 23) IS SET TO 0.  BUT THE PROGRAM ZEROES IT
                       /ANYWAY WITH A "DZM BITS" INSTRUCTION IN LOCATION 5.WHY?
                       /WHAT WOULD HAPPEN ON A SECONG RUN OF THE PROGRAM
                       /(WITH A NON-ZERO VALUE IN THE DSW) WITHOUT LOADING
                       /A FRESH COPY OF IT INTO CORE?
                       /
       000000 R                .END START
00025 R 000001 A *L
            SIZE=00026      NO ERROR LINES
```

```
                                .TITLE COUNT1 EXCEPT CHECKS FOR 0 AC INSTEAD OF
                        /
00000 R 740040 A    START   HLT
00001 R 750004 A            LAS
00002 R 040021 R            DAC TEST
                        /
00003 R 140022 R            DZM BITS
                        /
00004 R 200021 R    NEXT    LAC TEST
                        /
00005 R 741200 A            SNA             /IS THE AC=0? IF SO, DONE
00006 R 600016 R            JMP DONE        /YES
00007 R 500023 R            AND (1          /SOME BIT SET. IS IT BIT 17?
                        /
00010 R 740200 A            SZA             /SKIP IF 0,I.E. IF NOT BIT 17.
                        /
00011 R 440022 R            ISZ BITS        /OTHERWISE, ADD 1 TO BITS.
                        /
00012 R 200021 R            LAC TEST        /GET TEST
00013 R 744020 A            RCR             /CLEAR LINK BEFORE ROTATING.
                                            /CHECKING FOR AN ALL ZERO AC
                                            /DEPENDS UPON ROTATING 0S
                                            /INTO THE AC.
00014 R 040021 R            DAC TEST
                        /
00015 R 600004 R            JMP NEXT
                        /
00016 R 200022 R    DONE    LAC BITS
00017 R 740040 A            HLT
00020 R 600000 R            JMP START
                        /
00021 R 000000 A    TEST    0
00022 R 000000 A    BITS    0
                        /
      000000 R              .END START
00023 R 000001 A *L
          SIZE=00024        NO ERROR LINES
```

```
                        .TITLE PROGRAM (#2) COUNTS # OF BITS SET IN DSW
                    /PROGRAM COUNTS THE NUMBER OF BITS SET TO 1 IN THE DATA
                    /SWITCH REGISTER, MAKING USE OF THE MQ REGISTER.  THE
                    /RESULT IS PLACED IN THE ACCUMULATOR SO THAT IT IS
                    /DISPLAYED ON THE CONSOLE LIGHTS WHEN PROGRAM HALTS.
                    /
                        .ABS
50000                   .LOC 50000
                    /
50000   740040      START   HLT             /HALT AND ENTER VALUE FROM
50001   750004              LAS             /DATA SWITCHES.
50002   652000              LMQ             /PUT VALUE INTO MQ.
                    /
50003   150017              DZM BITS        /CLEAR BITS TO ZERO.
                    /
50004   777756              LAW -22         /LOAD AC WITH -18 DECIMAL
50005   050020              DAC COUNT       /AND STORE IN COUNT.
                    /
50006   750000              CLA             /CLEAR AC FOR FIRST TIME THRU.
50007   640601      NXT     LLS 1           /GET 1 BIT FROM THE MQ.
                    /
50010   750200              SZA!CLA         /SKIP IF AC=0, CLEAR AC.
50011   450017              ISZ BITS        /AC NOT 0 SO ADD 1 TO BITS.
50012   450020              ISZ COUNT       /TEST TO SEE IF DONE 18 TIMES.
                    /
                    /
50013   610007              JMP NXT         /NOT DONE, GO BACK FOR NEXT BIT.
                    /
50014   210017              LAC BITS        /DISPLAY RESULT IN AC.
50015   740040              HLT             /AND GO BACK FOR ANOTHER VALUE.
50016   610000              JMP START       /DEPRESS CONTINUE KEY.
                    /
50017   000000      BITS    0
50020   000000      COUNT   0
                    /         ****** POINTS TO BE NOTED ******
                    /
                    /*1. NOTE THAT THE LMQ INSTRUCTION IN 50002 MAKES USE
                    /OF THE MQ REGISTER EVEN THOUGH NO MULTIPLICATION NOR
                    /DIVISION WILL BE PERFORMED.  THE MQ REGISTER IS VERY
                    /CONVENIENT FOR TEMPORARY STORAGE (WHY USE  CORE
                    /LOCATION WHEN THE MQ IS ALWAYS THERE?)AND ALLOWS FOR
                    /LONG SHIFTS INTO THE AC.  YOU'LL FIND THE USE OF THE
                    /MQ REGISTER HANDY FOR CERTAIN TYPES OF CONVERSIONS.
                    /
                    /*2. NOTE THAT WHETHER OR NOT A SKIP IS PERFORMED WHEN
                    /LOCATION 50010 IS EXECUTED, THE AC IS CLEARED.
                    /    WHAT IS IT ABOUT THE CONTENTS OF LOCATION 50010
                    /MAKES THIS SO?  ALSO--WHY IS THIS NECESSARY? (WHAT
                    /CAN HAPPEN ON THE SECOND TIME THRU AFTER LOOPING BACK
                    /TO NXT IF THE AC IS NOT FIRST CLEARED?)
                    /
050000                  .END START
    SIZE=50021      NO ERROR LINES
```

```
                .TITLE COMMENTARY ON MOVE1
/
/                  **************************
/                  *                        *
/                  *  POINTS TO BE NOTED    *
/                  *                        *
/                  **************************
/
/*1. WHEN THE SUBROUTINE IS CALLED--
/       THE ACCUMULATOR SHOULD CONTAIN THE NUMBER OF
/LOCATIONS TO BE MOVED,I.E. THE SIZE OF THE TABLE.
/       THE LOCATION FOLLOWING THE "JMS" CALL SHOULD
/CONTAIN THE FIRST LOCATION TO BE MOVED (THE FIRST
/ADDRESS OF THE ORIGINAL TABLE),I.E. THE SOURCE ADDRESS.
/       THE NEXT LOCATION SHOULD CONTAIN THE ADDRESS TO
/WHICH THE TABLE IS TO BE MOVED (THE FIRST ADDRESS OF
/OF THE NEW TABLE), I.E. THE DESTINATION ADDRESS.
/
/CALLING SEQUENCE:      LAC (TABLE SIZE)
/                       JMS MOVE1
/                       (SOURCE ADDRESS)
/                       (DESTINATION ADDRESS)
/
/FOR EXAMPLE--  LAC (200            /STATES THAT WE WANT
/               JMS MOVE1           /TO MOVE CONTENTS
/               1450    )TRAILING   /OF LOCS 1450-1647
/               3120    )ARGUMENTS  /TO LOCS 3120-3317.
/
/THAT IS, A TABLE 200 LOCATIONS LONG: THE ORIGINAL TABLE
/STARTS AT LOCATION 1450, THE NEW TABLE STARTS AT 3120.
/
/*2. THE RETURN "PC" (THE ADDRESS OF THE LOCATION
/FOLLOWING THE JMS INSTRUCTION) IS STORED BY THE CPU
/IN THE FIRST LOCATION OF THE SUBROUTINE (MOVE1).
/
/*3. THE SUBROUTINE USES THIS POINTER TO PICK UP VALUES
/WHICH FOLLOW THE JMS CALL (KNOWN AS TRAILING ARGUMENTS)
/
/*4. THE "ISZ" INSTRUCTION MAKES IT VERY NICE TO USE
/NEGATIVE COUNTERS FOR LOOPING (SET COUNTER TO 2'S
/COMPLEMENT OF # OF TIMES A PROCESS IS TO BE PERFORMED).
/EACH TIME A LOOP OF PROCESSING IS COMPLETED, INCREMENT
/THE COUNTER BY 1.  WHEN THE COUNTER GETS INCREMENTED TO
/0, THE "ISZ" ALSO SKIPS AN INSTRUCTION ALLOWING THE
/PROGRAM TO GET OUT OF THE LOOP--THUS DETERMINING
/WHEN THE ENTIRE PROCESS IS COMPLETED.
/
/*5. IT IS THE SUBROUTINE'S RESPONSABILITY TO UPDATE THE
/POINTER,MOVE1, ENOUGH TIMES TO POINT PAST THE TRAILING
/ARGUMENTS BACK IN THE CALLING ROUTINE (HERE VIA "ISZ").
/
/*6. THE SUBROUTINE RETURNS TO THE CALLING ROUTINE VIA A
/"JMP*" THRU THE FIRST LOCATION (MOVE1).
```

```
                                .TITLE SUBROUTINE (#1) MOVES BLOCKS OF CORE
                        /
                        /
                        /EXAMPLE OF MOVE SUBROUTINE WHICH MOVES
                        /BLOCKS OF CORE LOCATIONS (TABLES).
                        /
                        /CALLING SEQUENCE:      LAC (TABLE SIZE)
                        /                       JMS MOVE1
                        /                       (SOURCE ADDRESS)
                        /                       (DESTINATION ADDRESS)
                        /
                        /
                        /
00000 R 000000 A        MOVE1   0
                        /
00001 R 740031 A                TCA             /DEVELOP NEGATIVE SIZE
00002 R 040020 R                DAC TABLSZ      /FOR A LOOP.
                        /
00003 R 220000 R                LAC* MOVE1      /GET SOURCE ADDRESS.
00004 R 040021 R                DAC SOURCE      /STORE IN A POINTER.
                        /
00005 R 440000 R                ISZ MOVE1       /ADD 1 TO RETURN "PC" SO THAT
                                                /IT NOW POINTS TO NEXT ARGUMENT
00006 R 220000 R                LAC* MOVE1      /GET DESTINATION ADDRESS.
00007 R 040022 R                DAC DESTIN      /STORE IN POINTER.
                        /
00010 R 220021 R        MORE    LAC* SOURCE     /MOVE1 TABLES!!!
00011 R 060022 R                DAC* DESTIN
                                                /ADD 1 TO LOCATIONS SOURCE AND
00012 R 440021 R                ISZ SOURCE      /DESTIN SO THEY POINT TO NEXT
00013 R 440022 R                ISZ DESTIN      /LOCATIONS IN OLD AND NEW TABLES
                        /
00014 R 440020 R                ISZ TABLSZ      /ADD 1 TO NEGATIVE COUNT
                                                /ARE WE DONE?
00015 R 600010 R                JMP MORE        /NO--GO BACK FOR MORE
                        /
00016 R 440000 R                ISZ MOVE1       /DONE! MOVE1 SHOULD POINT
                                                /TO LOCATION AFTER ARGUMENTS
                        /
00017 R 620000 R                JMP* MOVE1      /EXIT--RETURN TO CALLING ROUTINE
                                                /VIA THE ADDRESS STORED IN MOVE1
                        /
00020 R 000000 A        TABLSZ  0
00021 R 000000 A        SOURCE  0
00022 R 000000 A        DESTIN  0
                        /
        000000 A                .END
            SIZE=00023      NO ERROR LINES
```

```
                .TITLE COMMENTARY ON MOVE2
/
/
/                  **************************
/                  *                        *
/                  *  POINTS TO BE NOTED    *
/                  *                        *
/                  **************************
/
/
/*1. THERE ARE TWO MAJOR DIFFERENCES BETWEEN
/SUBROUTINES MOVE1 AND MOVE2:
/
/                           A)TYPE OF PROGRAM--
/                               RELOCATABLE VS. ABSOLUTE.
/
/                           B)TYPE OF ADDRESSING USED--
/                               INDIRECT VS. AUTO-INCREMENT.
/
/*2. MOVE1 IS  RELOCATABLE PROGRAM (RECALL THE ABSENCE
/OF .ABS AND .LOC STATEMENTS).  LOCATIONS USED STARTED
/WITH RELOCATABLE 0.
/     MOVE2 IS AN ABSOLUTE PROGRAM (NOTE THE USE OF
/.ABS AND .LOC STATEMENTS).  LOCATIONS USED START WITH
/ABSOLUTE 100.
/
/*3. MOVE1 MAKES USE OF INDIRECT ADDRESSING.  IT SETS UP
/LOCATIONS SOURCE AND DESTIN WITHIN ITSELF.  ALSO, TO
/UPDATE THESE POINTERS IT MAKES USE OF "ISZ SOURCE" AND
/"ISZ DESTIN" INSTRUCTIONS.
/     MOVE2 MAKES USE OF AUTO-INCREMENT ADDRESSING.  IT
/SETS UP ABSOLUTE LOCATIONS 10 AND 11 AS SOURCE AND
/DESTIN RESPRECTIVELY.  THIS MAKES FOR A SMALLER PROGRAM
/BECAUSE LOCATIONS 10-17 ARE "RESERVED" FOR AUTO-
/INCREMENT ADDRESSING ANYWAY (I.E. THEY ARE NOT NORMALLY
/USED TO HOLD EXECUTABLE INSTRUCTIONS), SO WHY NOT MAKE
/USE OF THEM TO HOLD VALUES (RATHER THAN TAKING UP SPACE
/WITHIN THE SUBROUTINE).
/ALSO,  BECAUSE THEY ARE AUTO-INCREMENT LOCATIONS, MOVE2
/DOESN'T NEED "ISZ SOURCE" AND "ISZ DESTIN" INSTRUCTIONS
/LOCATIONS SOURCE & DESTIN ARE AUTOMATICALLY INCREMENTED
/BY 1 BEFORE BEING USED AS POINTERS (SUCH IS THE FATE
/OF AUTO-INCREMENT REGISTERS WHEN USED WITH INDIRECT
/ADDRESSING).  MAKE SURE YOU UNDERSTAND THE PURPOSE OF
/THE "LAW -1" INSTRUCTION USED IN LOCATIONS 103 AND 107.
```

```
                        .TITLE SUBROUTINE (#2) MOVES BLOCKS OF CORE
                    /
                    /EXAMPLE OF MOVE SUBROUTINE WHICH MOVES
                    /BLOCKS OF CORE LOCATIONS (TABLES).
                    /
                    /CALLING SEQUENCE:      LAC (TABLE SIZE)
                    /                       JMS MOVE2
                    /                       (SOURCE ADDRESS)
                    /                       (DESTINATION ADDRESS)
                    /
                    /!!!!!ALSO SEE EXPLANATORY NOTES ON PREVIOUS PAGES
                    /!!!!!FOR SUBROUTINE MOVE1.
                    /
                            .ABS
00100                       .LOC 100
                    /
00100   000000      MOVE2   0
                    /
00101   740031              TCA             /DEVELOP NEGATIVE SIZE
00102   040120              DAC TABLSZ      /FOR LOOP.
                    /
00103   777777              LAW -1
00104   360100              TAD* MOVE2      /GET SOURCE ADDRESS
00105   040010              DAC SOURCE      /STORE IN A POINTER.
                    /
00106   440100              ISZ MOVE2       /ADD 1 TO RETURN "PC" SO THAT
                                            /IT POINTS TO NEXT ARGUMENT.
                    /
00107   777777              LAW -1
00110   360100              TAD* MOVE2      /GET DESTINATION ADDRESS
00111   040011              DAC DESTIN      /STORE IN POINTER.
                    /
00112   220010      MORE    LAC* SOURCE     /MOVE TABLES!!!
00113   060011              DAC* DESTIN
                    /
00114   440120              ISZ TABLSZ      /ADD 1 TO NEGATIVE COUNT
                                            /ARE WE DONE?
00115   600112              JMP MORE        /NO--GO BACK FOR MORE
                    /
00116   440100              ISZ MOVE2       /DONE--MOVE2 SHOULD POINT
                                            /TO LOCATIONAFTER ARGUMENTS
                    /
00117   620100              JMP* MOVE2      /EXIT--RETURN TO CALLING ROUTINE
                                            /VIA THE ADDRESS STORED IN MOVE2
00120   000000      TABLSZ  0
                    /
                    /
00010                       .LOC 10
00010   000000      SOURCE  0
00011   000000      DESTIN  0
                    /
        000000              .END
            SIZE=00121      NO ERROR LINES
```

```
                                .TITLE PROGRAM (#1) TO READ AND REVERSE OCTAL DI
                        /
                        / GET A SIX PLACE OCTAL NUMBER FROM THE DATA SWITCHES
                        / AND REVERSE THE OCTAL NUMBER.
                        / EXAMPLE: VALUE FROM SWITCHES = 123456
                        /          RESULTANT VALUE     = 654321
                        /
                                .ABS
00100                           .LOC 100
                        /
00100   740040          START   HLT
                        /
00101   750004                  LAS             / GET VALUE FROM SWITCHES
                                                / ASSUME VALUE IS 123456
00102   742030                  SWHA            / 456123
00103   040122                  DAC TEMP#       / 456123
                        /
00104   500123                  AND (70070      / 050020
00105   040121                  DAC FINAL#      / 050020
                        /
00106   200122                  LAC TEMP        / 456123
00107   640506                  LRS 6           / 004561
00110   500124                  AND (7007       / 004001
00111   340121                  TAD FINAL       / 054021
00112   040121                  DAC FINAL       / 054021
                        /
00113   200122                  LAC TEMP        / 456123
00114   640606                  LLS 6           / 612323
00115   500125                  AND (700700     / 600300
00116   340121                  TAD FINAL       / 654321
                        /
00117   740040                  HLT             /DISPLAY RESULT IN AC
                        /
00120   600100                  JMP START       /HIT CONTINUE TO GET BACK TO
                                                /BEGINNING FOR ANOTHER VALUE.
                        /
        000100                  .END START
00123   070070    *L
00124   007007    *L
00125   700700    *L
            SIZE=00126      NO ERROR LINES
```

```
                                .TITLE PROGRAM (#2) TO READ AND REVERSE OCTAL DI
                        /
                        /
                        /
                        /
                        /
00000 R 740040 A    REVRC2  HLT             /ALLOW SETTING OF DATA SWITCHES
00001 R 777772 A            LAW -6          /SET 6 DIGITS
00002 R 040016 R            DAC DIGITS#
00003 R 750004 A            LAS             /READ IN DATA SWITCHES
                        /
00004 R 742020 A            RTR             /MOVE 2 POSITIONS RIGHT SO THAT
                                            /FIRST OCTAL DIGIT IS 6 BITS OFF
00005 R 742010 A    LOOP    RTL             /ROTATE 6 LEFT TO BRING
00006 R 742010 A            RTL             /MOST SIGNIF. CHAR TO LEAST
00007 R 742010 A            RTL             /SIGNIFICANT END OF AC
00010 R 640503 A            LRS 3           /PLACE IN MQ
                        /
00011 R 440016 R            ISZ DIGITS      /TEST END
00012 R 600005 R            JMP LOOP        /NOT DONE YET--GO BACK FOR MORE
00013 R 641002 A            LACQ            /DISPLAY
                        /
00014 R 740040 A            HLT             /ON HALT
                        /
00015 R 600000 R            JMP REVRC2      /START AGAIN
                        /
        000000 R            .END REVRC2
            SIZE=00017      NO ERROR LINES
```

```
                                .TITLE UNSIGNED MULTIPLY ROUTINE
                        /
                                .ABS
                        /
00000   740040          START   HLT             /HALT TO ALLOW ENTRY OF
00001   750004                  LAS             /VALUE FOR A ON SWITCHES.
00002   040047                  DAC   A         /STORE IN LOCATION A.
00003   740040                  HLT             /HALT TO ALLOW ENTRY OF
00004   750004                  LAS             /VALUE FOR B ON SWITCHES.
00005   040050                  DAC   B         /STORE IN LOCATION B.
00006   140053                  DZM   PROD#     /CLEAR RESULT AREA.
00007   200047                  LAC   A#
00010   040051                  DAC   MCAND#    /ASSUME A>B, HENCE
00011   200050                  LAC   B#        /SMALLER NUMBER BECOMES
00012   040052                  DAC   MPLIER#   /MULTIPLIER
00013   740031                  TCA             /NEGATE B AND ADD A.
00014   340047                  TAD   A
00015   740100                  SMA             /IF SUM MINUS, REVERSE A,B
00016   600023                  JMP   ABOK      /A>B, OK AS IS
00017   200050                  LAC   B         /SINCE B>A REVERSE FUNCTION
00020   040051                  DAC   MCAND     /OF A AND B AS MPLIER AND MCAND.
00021   200047                  LAC   A         /
00022   040052                  DAC   MPLIER
                        /
00023   200052          ABOK    LAC   MPLIER    /GET MULTIPLIER
00024   741200                  SNA             /IF MPLIER=0, ALL DONE
00025   600044                  JMP   DONE
00026   500054                  AND   (1        /GET BIT 17 OF MPLIER
00027   740200                  SZA             /AND SEE IF IT IS 0.
00030   600040                  JMP   ADD       /GO ADD IN MCAND TO PROD
                        /
00031   200051          SHIFT   LAC   MCAND     /GET MCAND AND SHIFT ONE PLACE
00032   744010                  RCL             /LEFT FOR NEXT TIME THRU
00033   040051                  DAC   MCAND
00034   200052                  LAC   MPLIER    /GET MPLIER AND SHIFT ONE PLACE
00035   744020                  RCR             /RIGHT FOR NEXT TIME THRU.
00036   040052                  DAC   MPLIER    /
00037   600023                  JMP   ABOK      /GO BACK FOR NEXT BIT IN MPLIER.
                        /
00040   200053          ADD     LAC   PROD      /BIT 17 OF MPLIER WAS NOT 0,
00041   340051                  TAD   MCAND     /ADD MCAND AND STORE IN PROD.
00042   040053                  DAC   PROD
00043   600031                  JMP   SHIFT     /GO SHIFT MPLIER AND MCAND.
                        /
00044   200053          DONE    LAC   PROD
00045   740040                  HLT             /STOP TO EXAMINE RESULT IN AC.
00046   600000                  JMP   START     /CHANGE A AND B AND DO AGAIN.
                        /
        000000                  .END START
00054   000001  *L
            SIZE=00055      NO ERROR LINES
```

```
                                .TITLE ODD SEARCH PROBLEM
                        /
                        /A PROGRAM TO SEARCH 120-150 FOR
                        /ODD NUMBERS IN THE RANGE OF
                        /XXXX51 TO XXXX57, TOTAL STORED IN AC
                        /ADDRESS OF EACH STORED IN A TABLE
                        /STARTING AT 200
                        /
                                .ABS
                        /
00400                           .LOC 400
                        /
00400   100416          START   JMS SETUP       /SET UP VARIABLES
        000010          PTR1=10
        000011          PTR2=11
00401   220010                  LAC* PTR1       /GET A NUMBER
00402   500431                  AND (71         /LOOK FOR ODD BIT
00403   540432                  SAD (51         /AND THE NUMBER "50"
00404   100411                  JMS FOUND
00405   440427                  ISZ COUNT       /DONE WITH SEARCH?
00406   600401                  JMP .-5         /NO
00407   200430                  LAC HITS#       /YES
00410   740040                  HLT
                        /
                        /SUBROUTINES
                        /
00411   000000          FOUND   0
00412   200010                  LAC PTR1        /GET ADDRESS
00413   060011                  DAC* PTR2       /STORE IN TABLE
00414   440430                  ISZ HITS        /UPDATE HITS
00415   620411                  JMP* FOUND
                        /
00416   000000          SETUP   0
00417   140430                  DZM HITS        /PRESET HITS TO ZERO
00420   200433                  LAC (117
00421   040010                  DAC PTR1        /SETUP SOURCE TABLE
00422   200434                  LAC (177
00423   040011                  DAC PTR2        /SET UP TABLE OF RESULTS
00424   777747                  LAW -31
00425   040427                  DAC COUNT       /SET UP TABLE SIZE
00426   620416                  JMP* SETUP
00427   777747          COUNT   -31
        000400                  .END START
00431   000071  *L
00432   000051  *L
00433   000117  *L
00434   000177  *L
            SIZE=00435      NO ERROR LINES
```

```
                                .TITLE TABLE SEARCH PROBLEM
                        /A PROGRAM TO ACCEPT A NUMBER
                        /FROM THE DATA SWITCHES,
                        /SEARCH FOR THE 1ST WORD<O.S.
                        /IN LOCATIONS 400-477
                        /
                                .ABS
                        /
00100                           .LOC 100
                        /
00100   740040          STOP    HLT             /ALLOW OPERATOR TIME TO
00101   750004                  LAS             /SET THE DATA SWITCHES
00102   740031                  TCA             /SET UP ALPHA [-A]
00103   040121                  DAC ALPHA#      /EXAMPLE OF ASSIGNED VARIABLE "#
00104   200123                  LAC (377        /EXAMPLE OF A LITERAL "("
00105   040010                  DAC PTR         /SET UP AUTO-INCREMENT
        000010          PTR=10                  /EXAMPLE OF A DIRECT ASSIGNMENT
00106   777700                  LAW -100
00107   040122                  DAC TALLY#      /SET UP TABLE SIZE TALLY
00110   220010          ANG     LAC* PTR        /GET ANUMBER FROM TABLE
00111   340121                  TAD ALPHA       /(TABLE)+-(A)=RESULT
00112   751100                  SPA!CLA         /RESULT=(+) A<OR=(TABLE)
00113   600117                  JMP FOUND       /RESULT=(-) A>(TABLE)
00114   440122                  ISZ TALLY       /SEARCHED WHOLE TABLE?
00115   600110                  JMP ANG         /NO
00116   740040                  HLT             /YES
00117   200010          FOUND   LAC PTR
00120   740040                  HLT
        000100                  .END STOP
00123   000377    *L
            SIZE=00124      NO ERROR LINES
```

```
                .TITLE COMMENTARY ON SQUARE
/
/                  **************************
/                  *                        *
/                  *  POINTS TO BE NOTED    *
/                  *                        *
/                  **************************
/
/
/*1. NOTE THAT THE PROGRAM IS LOADED AT 20000.  THIS IS
/SO THAT IS DOES NOT GET LOADED OVER (HENCE,OVERLAY,--
/COLLOQUALLY KNOWN AS CLOBBERING) THE MONITOR.  AT THE
/END OF THE PROGRAM WE JUST RETURN TO THE MONITOR USING
/"CAL CODE 15" (THE MONITOR HAS A ROUTINE TO DETERMINE
/WHICH "CAL ROUTINE" IS BEING REQUESTED.  IN THIS CASE,
/MONITOR ROUTINE 15 IS AN EXIT ROUTINE).
/
/*2.THIS PROGRAM MAKES USE OF AUTO-INCREMENT REGISTER 10
/ONCE A SQUARE IS COMPUTED, IT IS STORED IN A TABLE VIA
/INDIRECT ADDRESSING ON LOCATION 10 (NOTE "DAC* 10" IN
/LOCATION 20012).  THIS TABLE STARTS AT LOCATION 20200.
/BUT WHEN AUTO-INCREMENT REGISTERS ARE USED INDIRECTLY,
/THEY ARE PRE-INCREMENTED.  HENCE, WE MUST SET UP
/ABSOLUTE LOC 10 TO CONTAIN "20177".  FIRST, WE LOAD
/"20177" INTO THE ACCUMULATOR (THE "LAC (20177"  IN
/LOCATION 20004).
/
/     NOW THE PROBLEM IS TO STORE THIS VALUE IN ABSOLUTE
/LOCATION 10.  WHERE WOULD THE VALUE "20177" BE STORED
/IF THE INSTRUCTION IN LOCATION 20005 READ "DAC 10"?
/(HINT: WHAT BANK WOULD IT BE IN?)
/
/     NOW LET'S LOOK AT THE "DAC* (10" INSTRUCTION IN
/LOCATION 20005.  THE ASSEMBLER SETS UP LOCATION 20304
/TO CONTAIN A "10" (WHY?).  THE INSTRUCTION "DAC* (10"
/IS ASSEMBLED AS 060304 (000 110 000 011 000 100):
/ DAC INTO THE LOCATION POINTED TO BY LOCATION
/20304, I.E. ABSOLUTE LOCATION 10.
/
/000 1XX XXX XXX XXX XXX          = DAC
/XXX X1X XXX XXX XXX XXX          = INDIRECT ADDRESSING
/XXX XX0 000 011 000 100          = VIA LOC 304 IN THIS
/                                    BANK.
/
/
```

```
                                .TITLE BUILDING A TABLE OF SQUARES
                        /
                        /THIS PROGRAM WILL BUILD A TABLE OF SQUARES
                        /FROM 1 TO 100 (OCTAL).
                        /THE TABLE STARTS AT LOCATION 20200.
                        /
                        /THE BUILDING OF THE TABLE OF SQUARES IS BASED UPON THE
                        /FACT THAT:
                        /
                        /THE NTH SQUARE = THE (N-1)TH SQUARE + THE NTH ODD #
                        /        2
                        /       1        =       0       +       1       =1
                        /        2
                        /       2        =       1       +       3       =4
                        /        2
                        /       3        =       4       +       5       =9
                        /                                .
                        /                                .
                        /                               ETC
                                .ABS
20000                           .LOC 20000
                        /
20000   750000          START   CLA
20001   040301                  DAC NXT#
                        /
20002   740030                  IAC
20003   040302                  DAC ODD#        /ODD STARTS AT 1.
                        /
20004   200303                  LAC (20177      /STORE TABLE AT LOC 20200
20005   060304                  DAC* (10        /VIA AUTO INC REGISTER 10
                        /
20006   777700                  LAW -100        /SET UP COUNTER
20007   040300                  DAC COUNT#      /COUNT 100 TIMES THRU LOOP.
                        /
20010   200301          SQR     LAC NXT         /         0          1          4
20011   340302                  TAD ODD         /         1     1+3=4     4+5=9
20012   060010                  DAC* 10         /(20200)=1 (20201)=4 (20202)=9
                        /
20013   040301                  DAC NXT         /    NXT=1      NXT=4      NXT=9
                        /
20014   200302                  LAC ODD         /         1          3          5
20015   340305                  TAD (2          /         3          5          7
20016   040302                  DAC ODD         /    ODD=3      ODD=5      ODD=7
                        /
20017   440300                  ISZ COUNT       /ARE WE DONE?
20020   600010                  JMP SQR         /NO--GO BACK FOR MORE.
20021   740040                  HLT             /YES--CHECK TABLE VIA CONSOLE.
20022   000000                  CAL             /GO BACK TO THE MONITOR BY
20023   000015                  15              /HITTING CONTINUE SWITCH
                        /
20200                           .LOC 20200
20200                           .BLOCK 100      /RESERVE ROOM FOR THE TABLE.
        020000                  .END START
20303   020177  *L
20304   000010  *L
20305   000002  *L
            SIZE=20306      NO ERROR LINES
```

```
 1      SORT   SRC      SORT ROUTINE

                                .TITLE SORT ROUTINE
                        /
                        /**SUBROUTINE SORTS VALUES INTO ASCENDING ORDER
                        /**AT TIME OF ENTRY, (AC)= THE NUMBER OF WORDS TO BE SOR
                        /                   (MQ)= ADDRESS OF BUFFER TO BE SORTE
                        /
                                .GLOBL SORT
                        /
00000 R 030000 A        SORT    0
00001 R 340047 R                TAD (-1         /HAVE (NUMBER OF WDS)-1 CHECKS
00002 R 040041 R                DAC COUNT#
00003 R 200041 R        AGAIN   LAC COUNT
00004 R 740031 A                TCA
00005 R 040046 R                DAC WDCNT#
00006 R 140042 R                DZM FLAG
00007 R 641002 A                LACQ            /PICK UP BUFFER ADDRESS FROM MQ
00010 R 040043 R                DAC PT1#        /PLACE IN PT1 FOR LATER REFERENC
00011 R 740030 A                IAC
00012 R 040044 R                DAC PT2#        /USE TO POINT TO NEXT WORD
                        /
00013 R 220043 R        LOOP    LAC* PT1
00014 R 740031 A                TCA             /-A
00015 R 350044 R                TAD* PT2        /B-A
00016 R 741100 A                SPA             /B-A>=0 OK B>A SO ORDER OK
00017 R 100030 R                JMS SWITCH      /B-A<0 MEANS A>B SO SWITCH ORDER
                        /
00020 R 440043 R                ISZ PT1         /PT1 NOW POINTS TO NEXT WORD
00021 R 440044 R                ISZ PT2         /PT2 NOW POINTS TO NEXT WORD
00022 R 440046 R                ISZ WDCNT       /ARE WE DONE?
00023 R 600013 R                JMP LOOP        /NO--GO BACK FOR MORE CHECKS
00024 R 200042 R                LAC FLAG#
00025 R 740200 A                SZA             /WERE CHANGES MADE ON LAST TIME
00026 R 600003 R                JMP AGAIN
00027 R 620000 R                JMP* SORT       /NO --WE'RE COMPLETELY DONE
                        /
00030 R 000000 A        SWITCH  0
00031 R 220043 R                LAC* PT1        /GET LARGER VALUE
00032 R 040045 R                DAC TEMP#       /HOLD TEMPORARILY
00033 R 220044 R                LAC* PT2        /GET SMALLER VALUE
00034 R 050043 R                DAC* PT1        /PLACE IN EARLIER ENTRY
00035 R 200045 R                LAC TEMP
00036 R 050044 R                DAC* PT2        /PLACE LARGER IN 2ND ENTRY
00037 R 440042 R                ISZ FLAG        /SET FLAG TO SHOW
                                                /SWITCH TOOK PLACE
                        /
00040 R 620030 R                JMP* SWITCH
                        /
                        /
      000000 A                  .END
00047 R 777777 A *L
          SIZE=00050      NO ERROR LINES
```

```
                                .TITLE LINE EDITOR
                            /
                            /
                                .ABSP
                            /
                    700401  TSF=700401
                    700406  TLS=700406
                    700301  KSF=700301
                    700312  KRB=700312
                            /
00100                       .LOC 100
                            /
00100   700416      START   TLS+10
00101   707762              DBA
00102   100160              JMS CRLF
00103   100141              JMS SETUP       /INITIALIZE XR AND LR
00104   100146      MORE    JMS LISN        /WAIT FOR CHAR. FROM KBRD
00105   540276              SAD (225        /CHECK FOR SPECIAL CHARACTERS
00106   600130              JMP LOLT        /AU:  ERASE ENTIRE LINE
00107   540277              SAD (377
00110   600133              JMP RUBOUT      /RUBOUT LAST CHAR.
00111   050166              DAC BUFF,X      /STORE CHARACTER IN BUFFER
00112   540300              SAD (215        /IS CHAR. A C.R.?
00113   600116              JMP OUTPUT      /YES - END OF INPUT LINE
00114   725001              AXS 1           /NO - INCREMENT XR, IF <
00115   600104              JMP MORE        /72 CHARS, GET NEXT CHAR.
00116   100160      OUTPUT  JMS CRLF        /ECHO A CR, LF
00117   100141              JMS SETUP       /RE-INITIALIZE
00120   210166      PRINT   LAC BUFF,X      /ECHO ONE CHAR. AT A TIME
00121   100153              JMS TYPE
00122   540300              SAD (215
00123   600126              JMP .+3
00124   725001              AXS 1
00125   600120              JMP PRINT
00126   100160              JMS CRLF        /DONE WITH ENTIRE LINE
00127   600103              JMP MORE-1      /READ IN NEXT LINE
                            /
00130   760300      LOLT    LAW 300         /ECHO "@"
00131   100153              JMS TYPE
00132   600103              JMP MORE-1
                            /
00133   760334      RUBOUT  LAW 334         /ECHO A REVERSE SLASH
00134   100153              JMS TYPE
00135   724000              PXA             /IF RUBOUT IS FIRST CHAR.
00136   740200              SZA             /DONT BACKUP BUFFER POINTER
00137   737777              AXR -1
00140   600104              JMP MORE
                                .EJECT
```

```
                        /
                        /SUBROUTINES
                        /
00141   000000  SETUP   0
00142   200301          LAC (110
00143   722000          PAL
00144   735000          CLX
00145   620141          JMP* SETUP
                        /
00146   000000  LISN    0
00147   700301          KSF
00150   600147          JMP .-1
00151   700312          KRB
00152   620146          JMP* LISN
                        /
00153   000000  TYPE    0
00154   700401          TSF
00155   600154          JMP .-1
00156   700406          TLS
00157   620153          JMP* TYPE
                        /
00160   000000  CRLF    0
00161   760215          LAW 215
00162   100153          JMS TYPE
00163   760212          LAW 212
00164   100153          JMS TYPE
00165   620160          JMP* CRLF
                        /
00166           BUFF    .BLOCK 110
        000100          .END START
00276   000225  *L
00277   000377  *L
00300   000215  *L
00301   000110  *L
            SIZE=00302      NO ERROR LINES
```

```
                                .TITLE PROGRAM TO COPY PAPER TAPE
                        /
                                .ABSP
00100                           .LOC 100
                        /
            700204      PSA=700204
            700104      RSA=700104
            700201      PSF=700201
            700112      RRB=700112
            700101      RSF=700101
                        /
00100   740040      A.1     HLT             /HALT TO INSERT TAPE TO COPY
00101   707762              DBA             /ENTER PAGE MODE
00102   200661              LAC (500        /PUT 500 IN LIMIT REGISTER
00103   722000              PAL
                        /
00104   700214              PSA+10          /PREPARE TO SET PUNCH FLAG
                        /
00105   140657      A.2     DZM RDFLAG#
00106   140660              DZM RDINDX#
00107   140656              DZM PCINDX#
                        /
00110   200660      A.3     LAC RDINDX      /GET READ INDEX REGISTER VALUE
00111   721000              PAX             /PUT INTO XR
00112   700104              RSA             /READ A CHARACTER
00113   700101              RSF             /WAIT FOR READ FLAG
00114   600113              JMP .-1         /TO GO UP
                        /
00115   700314      A.4     IORS            /GET I/O STATUS REGISTER
00116   500662              AND (1000       /SEE IF READER OUT OF TAPE.
00117   740200              SZA             /IF 0, NOT OUT
00120   600152              JMP A.17        /READER OUT OF TAPE
                        /
00121   700112      A.5     RRB             /GET CHAR INTO AC
00122   050156              DAC BUFF,X      /STORE IT IN BUFF,X
                        /
00123   725001      A.6     AXS 1           /UPDATE INDEX
00124   600127              JMP A.8 /BUFF NOT FULL
                        /
                        /
00125   200663      A.7     LAC (1          /BUFF FULL SO SET RDFLAG=1
00126   040657              DAC RDFLAG
                        /
00127   724000      A.8     PXA             /SAVE READ INDEX
00130   040660              DAC RDINDX      /IN RDINDX
                        /
00131   200656      A.85    LAC PCINDX      /GET PUNCH INDEX
00132   721000              PAX             /AND STORE IN XR
                        /
00133   700201      A.9     PSF             /SEE IF PUNCH READY
00134   600144              JMP A.15        /NOT READY, SEE IF CAN READ
                        /
```

```
00135  210156        A.10    LAC BUFF,X      /GET CHAR FROM BUFF
00136  741100        A.11    SPA             /AND SEE IF IT IS NEGATIVE
00137  740040        A.12    HLT             /IT WAS, SO HALT
00140  700204        A.13    PSA             /START PUNCHING NEXT CHARACTER
00141  725001        A.14    AXS 1           /IF SKIP, PUNCH BUFF EMPTY
00142  741000                SKP
00143  600105                JMP A.2         /BUFF EMPTY, GO READ MORE
00144  200657        A.15    LAC RDFLAG      /SEE IF READ BUFF FULL
00145  740200                SZA             /IF 0, NOT FULL, GO READ
00146  600133                JMP A.9         /READ FULL, SO LOOP IN PUNCH
00147  724000        A.16    PXA
00150  040656                DAC PCINDX      /SAVE PUNCH INDEX
00151  600110                JMP A.3         /READ BUFF NOT FULL--GO READ
00152  777777        A.17    LAW -1
00153  050156                DAC BUFF,X      /PUT TERMINAL CHARACTER IN BUFFE
00154  040657                DAC RDFLAG      /SET RDFLAG TO NON 0
00155  600131                JMP A.85
                       /
00156                  BUFF   .BLOCK 500
                       /
       000100                 .END A.1
00661  000500    *L
00662  001000    *L
00663  000001    *L
           SIZE=00664      NO ERROR LINES
```

```
                .TITLE COMMENTARY ON DUMP
/
/                  **************************
/                  *                        *
/                  *  POINTS TO BE NOTED    *
/                  *                        *
/                  **************************
/
/*1. NOTE THAT MOST "IOT" INSTRUCTIONS ARE NOT DEFINED
/WITHIN MACRO'S SYMBOL TABLES.  THEREFORE, THEY MUST
/BE DEFINED BY THE USER.  USUALLY ONE DOES THIS BY
/DIRECT ASSIGNMENT STATEMENTS AT THE BEGINNING OF THE
/PROGRAM--AS IS DONE IN THIS PROGRAM FOR "TSF","TLS",
/"KSF", AND "KRB".
/
/*2. NOTE THAT THE TELEPRINTER IS AN OUTPUT DEVICE AND
/AS SUCH, WE WANT TO CHECK ITS "READY FLAG" BEFORE DOING
/ANY OUTPUT,BUT ALL FLAGS ARE CLEARED BEFORE THE PROGRAM
/IS STARTED (THE ABSOLUTE LOADER CLEARS ALL FLAGS AND SO
/DOES THE RESET SWITCH ON THE CONSOLE).  THEREFORE, WE
/MUST SET THE FLAG OURSELVES.  CONSIDER HOW LONG WE
/WILL STAY IN THE LOOP:  TSF
/                        JMP .-1
/
/IF WE DO NOT TAKE CARE OF SETTING THE FLAG.  THIS IS
/DONE WITH THE "TLS+10" INSTRUCTION IN LOCATION 2.
/GO OVER THE IOT FORMAT TO MAKE SURE YOU UNDERSTAND WHAT
/THIS DOES FOR US.  TLS+10 = 700416.  BIT 14 IS SET.
/THIS CLEARS THE ACCUMULATOR BEFORE THE OUTPUT
/OPERATION IS PERFORMED.  ALL ZEROES IS THE ASCII CODE
/FOR THE NULL CHARACTER...AND THAT'S EXACTLY
/WHAT IS OUTPUT TO THE TELEPRINTER (LISTEN CAREFULLY).
/THE IMPORTANT THING IS THAT 100 MILLISECONDS AFTER THE
/PRINTING OF THIS CHARACTER IS INITIATED, THE "READY
/FLAG" IS RAISED.  FROM HERE ON IN THEN, WE'RE ALL SET
/TO DO OUTPUT.
/
/*3. RECALL THAT WHEN A CHARACTER COMES IN FROM THE KEY-
/BOARD, THE CODE STORED IN THE KEYBOARD BUFFER TO
/REPRESENT THE CHARACTER = THE 7-BIT ASCII CODE + 200.
/WHEN CHECKING FOR A SPECIFIC CHARACTER, MAKE SURE YOU
/TAKE THIS INTO ACCOUNT.  FOR EXAMPLE, WHEN CHECKING
/FOR A FORWARD SLASH "/" WHICH HAS A 7-BIT ASCII CODE
/OF 57, WE MUST NOTE THAT THE KEYBOARD BUFFER CONTAINS
/A "257" (57+200).  WE READ THE CONTENT OF THE KEYBOARD
/BUFFER INTO THE ACCUMULATOR WITH A "KRB" INSTRUCTION.
/THEN WE CAN COMPARE THE ACCUMULATOR WITH THE VALUE 257,
/<SAD (257>, OR WE CAN MASK OUT EVERYTHING EXCEPT
/FOR THE LEAST SIGNIFICANT 7 BITS IN THE AC,
/<AND (177>, AND LOOK FOR A 57.
/
```

```
                                .TITLE OCTAL DUMP PROBLEM
                        /
                        /PROGRAM ACCEPTS A LEGAL OCTAL ADDRESS FROM
                        /THE KEYBOARD, AND TYPES OUT ITS CONTENTS.
                        /
                        /THIS IS A RELOCATABLE PROGRAM WHICH WILL BE LOADED
                        /BY THE LINKING LOADER.  THE LINKING LOADER WORKS
                        /UNDER THE DOS SYSTEM AND, THEREFORE, INTERRUPT
                        /SYSTEMS WILL BE ON WHEN THE PROGRAM IS LOADED AND
                        /STARTED.  IF WE DO NOT TAKE CARE OF THIS, THE
                        /PROGRAM WON'T WORK BECAUSE IT USES DEDICATED I/O.
                        /THE INTERRUPT SYSTEM WILL PICK UP THE DEVICE FLAGS
                        /RATHER THAN THE "KSF" AND "TSF" INSTRUCTIONS IN THE
                        /PROGRAM.  THE FIRST THREE INSTRUCTIONS IN THE
                        /PROGRAM TURN OFF THE INTERRUPT SYSTEMS (PI & API).
                        /
          700401 A      TSF=700401
          700406 A      TLS=700406
          700301 A      KSF=700301
          700312 A      KRB=700312
                        /
00000 R 700002 A        START   IOF             /TURN OFF PI
00001 R 705514 A                ISA+10          /AND API INTERRUPT SYSTEMS
00002 R 700416 A                TLS+10
                        /
00003 R 100065 R                JMS CRLF
00004 R 100047 R                JMS SETUP
                        /
00005 R 100060 R        MORE    JMS LISTEN
                        /
00006 R 440074 R                ISZ DIGIT       /IS THIS THE 6TH DIGIT?
00007 R 540077 R                SAD (257        /IF NOT, IS IT A FORWARD SLASH?
00010 R 600021 R                JMP DONE        /YES IN EITHER CASE!
                        /
                        /GET HERE IF NOT  "/" OR 6TH CHARACTER
                        /
00011 R 500100 R                AND (7          /GET DIGIT
00012 R 040075 R                DAC SAVE#
                        /
00013 R 200073 R                LAC ADDSS       /GET ADDRESS
00014 R 744010 A                RAL!CLL         /ROTATE 3 POSITIONS LEFT TO
00015 R 742010 A                RTL             /MAKE ROOM FOR NEW DIGIT.
00016 R 340075 R                TAD SAVE        /UPDATE ADDRESS
00017 R 040073 R                DAC ADDSS
00020 R 600005 R                JMP MORE
                        /
00021 R 100047 R        DONE    JMS SETUP
00022 R 220073 R                LAC* ADDSS      /GET CONTENT
00023 R 744010 A                RAL!CLL         /ROTATE 4 POSITIONS FIRST TIME
00024 R 740010 A        LOOP    RAL     /3 POSITIONS AFTER THAT
00025 R 742010 A                RTL
00026 R 040073 R                DAC ADDSS       /SAVE CURRENT POSITIONS
```

```
00027 R 500100 R            AND (7          /GET OCTAL DIGIT
00030 R 340101 R            TAD (260        /MANUFACTURE ASCII CODE
00031 R 100053 R            JMS TYPE
                      /
00032 R 200073 R            LAC ADDSS
00033 R 440074 R            ISZ DIGIT       /ARE WE DONE??
00034 R 600024 R            JMP LOOP        /NO!--BACK FOR MORE
00035 R 140073 R            DZM ADDSS       /YES!-CLEAR ADDSS FOR NEXT TIME
00036 R 750004 A            LAS             /WANT TO CONTINUE?
00037 R 740200 A            SZA             /0 DSW SAYS GO BACK TO MONITOR
00040 R 600004 R            JMP START+4     /DON'T NEED TLS+10 ANYMORE.
00041 R 200102 R            LAC (1
00042 R 040076 R            DAC TIMER#
00043 R 440076 R            ISZ TIMER
00044 R 600043 R            JMP .-1
00045 R 000000 A            CAL
00046 R 000015 A            15
                      /
                      /SUBROUTINES
                      /
00047 R 000000 A      SETUP 0
00050 R 777772 A            LAW -6
00051 R 040074 R            DAC DIGIT
00052 R 620047 R            JMP* SETUP
                      /
00053 R 000000 A      TYPE  0
00054 R 700401 A            TSF
00055 R 600054 R            JMP .-1
00056 R 700406 A            TLS
00057 R 620053 R            JMP* TYPE
                      /
00060 R 000000 A      LISTEN 0
00061 R 700301 A            KSF
00062 R 600061 R            JMP .-1
00063 R 700312 A            KRB
00064 R 620060 R            JMP* LISTEN
                      /
00065 R 000000 A      CRLF  0
00066 R 760215 A            LAW 215
00067 R 100053 R            JMS TYPE
00070 R 760212 A            LAW 212
00071 R 100053 R            JMS TYPE
00072 R 620065 R            JMP* CRLF
                      /
00073 R 000000 A      ADDSS 0
00074 R 777772 A      DIGIT -6
                      /
      000000 R              .END START
00077 R 000257 A *L
00100 R 000007 A *L
00101 R 000260 A *L
00102 R 000001 A *L
          SIZE=00103      NO ERROR LINES
```

```
                                .TITLE PROGRAM TO AVERAGE DECIMAL VALUES
                        /
                        /000000000000000000000000000000000000000000000000000000000000
                        /
                        / PROGRAM ACCEPTS DECIMAL VALUES FROM THE KEYBOARD,
                        / SUMS THEM AND PRINTS OUT THE DECIMAL AVERAGE (GIVEN
                        / TO THE TENTHS PLACE) ON THE TELEPRINTER.
                        / USER SHOULD FOLLOW EACH VALUE WITH A COMMA; TERMINATE
                        / THE LINE WITH CR.  FOR EXAMPLE:  3,18,29,4,(CR)
                        /
                        /000000000000000000000000000000000000000000000000000000000000
                        /
          700301  A     KSF=700301
          700406  A     TLS=700406
          700401  A     TSF=700401
          700312  A     KRB=700312
                        /
00000  R 707762  A     BEGIN   DBA             /RUN IN PAGE MODE
00001  R 700002  A             IOF             /TURN OFF PI
00002  R 705514  A             ISA+10          /AND API INTERRUPT SYSTEMS
                        /
00003  R 700416  A             TLS+10          /INITIATE PRINT--GET FLAG
00004  R 140126  R             DZM COUNT#      /COUNT= # OF VALUES
00005  R 140127  R             DZM FINAL#      /FINAL= SUM OF VALUES
00006  R 100120  R             JMS CRLF
00007  R 140130  R     NXT     DZM NUMB#       /TEMPORARY LOCATION
                        /
00010  R 700301  A     NEXT    KSF
00011  R 600010  R             JMP .-1
00012  R 700312  A             KRB
00013  R 540134  R             SAD (215
00014  R 600035  R             JMP ALLDUN      /CR MEANS LAST VALUE
00015  R 540135  R             SAD (254        /"," SEPARATES VALUES
00016  R 600030  R             JMP DUN
00017  R 500136  R             AND (17         /GET OCTAL NUMBER
00020  R 040133  R             DAC TEMP#
                        /
00021  R 200130  R             LAC NUMB
00022  R 653122  A             MUL
00023  R 000012  A             12
00024  R 641002  A             LACQ            /RESULT SMALL..IN MQ
00025  R 340133  R             TAD TEMP        /ADD ON LAST DIGIT
00026  R 040130  R             DAC NUMB        /SAVE IN NUMB
00027  R 600010  R             JMP NEXT
                        /
00030  R 200130  R     DUN     LAC NUMB        /GET THIS VALUE
00031  R 340127  R             TAD FINAL       /ADD IT TO THE SUM
00032  R 040127  R             DAC FINAL
                        /
00033  R 440126  R             ISZ COUNT       /KEEP TRACK OF HOW MANY VALUES
00034  R 600007  R             JMP NXT
                        /
```

```
                        /
00035 R 200126 R    ALLDUN  LAC COUNT       /GET # OF VALUES
00036 R 040041 R            DAC .+3         /STORE FOR DIVISION
00037 R 200127 R            LAC FINAL       /GET SUM
00040 R 653323 A            IDIV
00041 R 000000 A            0
00042 R 040132 R            DAC REMAIN#     /REMAINDER RETURNED IN AC
00043 R 641002 A            LACQ            /GET QUOTIENT IN MQ
00044 R 040131 R            DAC QUOT#
                        /
00045 R 735000 A            CLX
00046 R 200131 R            LAC QUOT
00047 R 653323 A    NEXXT   IDIV
00050 R 000012 A            12
00051 R 050106 R            DAC DIGIT,X     /STORE IN TABLE
00052 R 641002 A            LACQ            /PICK UP QUOTIENT
00053 R 741200 A            SNA             /CONTINUE IF NOT 0
00054 R 600057 R            JMP DUNN        /STOP WHEN QUOTIENT = 0
00055 R 737001 A            AXR 1           /PUT IN TABLE FOR LATER OUTPUT
00056 R 600047 R            JMP NEXXT
                        /
                        /
00057 R 100120 R    DUNN    JMS CRLF        /ISSUE CR AND LF
                        /
00060 R 210106 R    DUNNN   LAC DIGIT,X
00061 R 340137 R            TAD (260        /TO MAKE ASCII
00062 R 100113 P            JMS PRINT
00063 R 737777 A            AXR -1          /GET NEXT CHARACTER--WORK
                                            /WAY BACK UP TALBE
00064 R 724000 A            PXA             /ARE WE DONE--WHEN XR=0, YES
00065 R 740100 A            SMA
00066 R 600060 R            JMP DUNNN
                        /
00067 R 200140 R            LAC (256        /ISSUE DECIMAL POINT "."
00070 R 100113 R            JMS PRINT
                        /
00071 R 200126 R            LAC COUNT
00072 R 040100 R            DAC REM
00073 R 200132 R            LAC REMAIN
00074 R 653122 A            MUL
00075 R 000012 A            12
00076 R 641002 A            LACQ
00077 R 653323 A            IDIV
00100 R 000000 A    REM     0
00101 R 641002 A            LACQ
00102 R 340137 R            TAD (260
00103 R 100113 R            JMS PRINT
00104 R 100120 R            JMS CRLF
00105 R 600000 R            JMP BEGIN
                        /
00106 R         A   DIGIT   .BLOCK 5
                        /
```

```
                                /
00113 R 000000 A        PRINT   0
00114 R 700401 A                TSF
00115 R 600114 R                JMP .-1
00116 R 700406 A                TLS
00117 R 620113 R                JMP* PRINT
                                /
00120 R 000000 A        CRLF    0
00121 R 760215 A                LAW 215
00122 R 100113 R                JMS PRINT
00123 R 760212 A                LAW 212
00124 R 100113 R                JMS PRINT
00125 R 620120 R                JMP* CRLF
                                /
        000000 R                .END BEGIN
00134 R 000215 A *L
00135 R 000254 A *L
00136 R 000017 A *L
00137 R 000260 A *L
00140 R 000256 A *L
            SIZE=000141     NO ERROR LINES
```

```
                                .TITLE OTOASC-OCTAL TO ASCII TRANSLATOR
                        /
                        /PROGRAM TRANSLATES ONE 18 BIT WORD TO SIX
                        /OCTAL ASCII CHARACTERS.  THE CHARACTERS ARE
                        /STORED IN A BUFFER POINTED TO BY THE WORD
                        /FOLLOWING THE JMS TO OTOASC.
                        /
                        /CALLING SEQUENCE:
                        /
                        /               LAC (WORD TO BE TRANSLATED)
                        /               JMS* OTOASC
                        /               BUFFER ADDRESS  /WHERE FIRST ASCII CHAR
                        /                               /IS TO BE STORED.
                        /
                        /
                                .GLOBL OTOASC
                        /
00000 R 000000 A        OTOASC  0
00001 R 652000 A                LMQ             /STORE 18 BITS TO BE TRANSLATED
00002 R 220000 R                LAC* OTOASC     /PICK UP STORAGE BUFFER ADDRESS
00003 R 040020 R                DAC PTR#
                        /
00004 R 440000 R                ISZ OTOASC      /SET UP FOR EXIT
00005 R 777772 A                LAW -6          /KEEP TRACK OF 6 CHARACTERS
00006 R 040017 R                DAC COUNT#
                        /
                        /DO CONVERSION
                        /
00007 R 750000 A        MORE    CLA             /CLEAR AC
00010 R 640603 A                LLS 3           /MOVE 3 BITS FROM MQ
00011 R 340021 R                TAD (60         /ADD 60 TO MAKE ASCII
00012 R 060020 R                DAC* PTR        /STORE IT AWAY
                        /
00013 R 440020 R                ISZ PTR         /POINT TO NEXT BUFFER LOCATION
00014 R 440017 R                ISZ COUNT       /ARE WE DONE?
00015 R 600007 R                JMP MORE        /NO--GO BACK FOR NEXT 3 BITS
00016 R 620000 R                JMP* OTOASC     /YES--RETURN TO CALLING ROUTINE
                        /
      000000 A                  .END
00021 R 000060 A *L
            SIZE=00022      NO ERROR LINES
```

```
PAGE   1     IA26B  SRC     TRANSLATES IMAGE ALPHA TO SIX-BIT ASCII

                                .TITLE TRANSLATES IMAGE ALPHA TO SIX-BIT ASCII
                            /
                            /PROGRAM TRANSLATES IMAGE ALPHA CHARACTERS TO SIX-BIT
                            /ASCII CHARACTERS.
                            /
                            /CALLING SEQUENCE:
                            /
                            /               LAC (ADDRESS OF FIRST IMAGE ALPHA CHAR)
                            /               JMS* IA26B
                            /               2                   /# OF SIX-BIT WORDS
                            /               BUF                 /SIXBT BUFFER ADDRESS
                            /
          642000 A          OAC=642000        /OR AC TO MQ...RESULT LEFT IN MQ
                            /
                                .GLOBL IA26B
                            /
     00000 R 000000 A       IA26B   0
     00001 R 040035 R               DAC IMGPTR#       /POINTER TO IMAGE CHARS TO BE CO
     00002 R 220000 R               LAC* IA26B        /PICK UP # OF SIXBT WORDS
     00003 R 740031 A               TCA               /MAKE IT NEGATIVE (2'S COMPLEMEN
     00004 R 040034 R               DAC COUNT#
     00005 R 440000 R               ISZ IA26B
     00006 R 220000 R               LAC* IA26B        /PICK UP ADDRESS OF WHERE TO STO
     00007 R 040036 R               DAC SXBPTR#       /6 BIT CHARS AND  SAVE IT
     00010 R 440000 R               ISZ IA26B         /SET UP FOR EXIT
                            /
     00011 R 744000 A       AGAIN   CLL               /CLEAR LINK
     00012 R 650000 A               CLQ               /CLEAR MQ
     00013 R 777775 A               LAW -3            /SET UP COUNTER OF 3
     00014 R 040033 R               DAC CNT3#         /BECAUSE 3 SIXBT CHARS PER WORD
                            /
     00015 R 640606 A       MORE    LLS 6             /GET MQ IN CORRECT POSITION
     00016 R 220035 R               LAC* IMGPTR       /PICK UP CHAR TO BE CONVERTED
     00017 R 500037 R               AND (77           /EXTRACT LOWER 6 BITS
     00020 R 642000 A               OAC               /OR THE MQ AND AC...RESULT LEFT
     00021 R 440035 R               ISZ IMGPTR        /POINT TO NEXT IMAGE CHARACTER
     00022 R 440033 R               ISZ CNT3#         /HAVE WE CONVERTED 3 IMAGE CHARS
     00023 R 600015 R               JMP MORE          /NO--GO BACK FOR MORE
                            /
     00024 R 641002 A               LACQ              /PICK UP SIXBT VALUE IN MQ
     00025 R 060036 R               DAC* SXBPTR       /STORE IT AWAY
     00026 R 440034 R               ISZ COUNT         /ARE WE DONE?
     00027 R 741000 A               SKP               /NO--
     00030 R 620000 R               JMP* IA26B        /YES--RETURN TO CALLING PROGRAM
     00031 R 440036 R               ISZ SXBPTR        /INCREMENT SIXBT BUFFER POINTER
     00032 R 600011 R               JMP AGAIN
                            /
          000000 A                  .END
     00037 R 000077 A *L
                SIZE=00040      NO ERROR LINES
```

```
                                .TITLE CLOCK DIAGNOSTIC--SKIP ON FLAG
                        /
                        / LIGHT IN AC AND LINK WILL MOVE 2 POSITIONS LEFT
                        / EVERY SECOND WHEN CLOCK IS RUNNING PROPERLY.
                        / AC=1 AT OUTSET OF PROGRAM.
                        / AC=1 IF CLOCK NOT OPERATIVE WHEN PROGRAM STARTS.
                        / IF CLOCK DISABLED FROM CONSOLE WITH CLOCK SWITCH,
                        / AC AND LINK ARE FROZEN.  WHEN CLOCK REENABLED
                        / FROM CONSOLE SWITCH, LIGHT STARTS MOVING AGAIN.
                        /
                        /
          700044 A      CLON=700044
          700001 A      CLSF=700001
                        /
    10100 A                     .LOC 10100
                        /
    10100 A 700002 A    START   IOF                 /TURN OFF PI AND
    10101 A 705514 A            ISA+10              /API SYSTEMS
    10102 A 754030 A            CLA!CLL!IAC         /AC=1
    10103 A 170117 A            DZM* (7
                        /
    10104 A 050116 A    AGAIN   DAC ACSAVE#         /SAVE AC WHILE TENDING CLOCK
    10105 A 777704 A            LAW -74             /60 TICKS PER SECOND(74 BASE 8)
    10106 A 370117 A            TAD* (7             /ADD IN ANY RUNOVER
    10107 A 070117 A            DAC* (7             /SET CLOCK REGISTER--LOC. 000007
    10110 A 210116 A            LAC ACSAVE          /GET AC BACK
    10111 A 700044 A            CLON                /TURN CLOCK ON AND CLEAR FLAG
    10112 A 700001 A            CLSF                /CHECK FOR CLOCK FLAG SET
    10113 A 610112 A            JMP .-1             /IF NOT, WE WAIT
    10114 A 742010 A            RTL                 /WHEN SET...ROTATE AC
    10115 A 610104 A            JMP AGAIN           /GO BACK FOR MORE
                        /
          010100 A              .END START
    10117 A 000007 A *L
                SIZE=10120      NO ERROR LINES
```

```
                                .TITLE PRINT "1" AFTER 5 SECONDS-DEDICATED I/O
                        /
                        /
                        / PROGRAM USES DEDICATED I/O WITH THE REAL TIME CLOCK
                        / TO PRINT A "1" ON THE TELEPRINTER EVERY 5 SECONDS.
                        / AS LONG AS THE DATA SWITCHES ARE NOT 0,
                        / THE PROGRAM CONTINUES.  WHEN THE DSW ARE 0, CONTROL
                        / IS RETURNED TO THE MONITOR.
                        /
          700001 A      CLSF=700001
          700044 A      CLON=700044
          700406 A      TLS=700406
          700402 A      TCF=700402
                        /
  10500 A                       .LOC 10500          /HIGH ENOUGH TO AVOID MONITOR
                        /
  10500 A 700002 A      START   IOF                 /LOADED BY DOS--THAT MEANS
  10501 A 705514 A              ISA+10              /INTERRUPTS WERE ON;SO TURN OFF
                        /
  10502 A 170524 A              DZM* (7             /ZERO LOC 7
                        /
  10503 A 777324 A      MORE    LAW -454            /ADD -5 SECONDS IN TICKS
  10504 A 370524 A              TAD* (7             /TO CLOCK COUNTER
  10505 A 070524 A              DAC* (7             /RESTORE TO CLOCK COUNTER
                        /
  10506 A 700044 A              CLON                /TURN CLOCK ON
  10507 A 700001 A      CLKTST  CLSF                /CHECK IF 5 SECONDS HAVE PASSED
  10510 A 610513 A              JMP DSWTST
  10511 A 110520 A              JMS PRINT           /YES--PRINT A "1"
  10512 A 610503 A              JMP MORE            /GO BACK FOR MORE TIMING
                        /
  10513 A 750004 A      DSWTST  LAS                 /CHECK DATA SWITCHES FOR 0
  10514 A 740200 A              SZA
  10515 A 610507 A              JMP CLKTST          /NOT 0, KEEP TESTING CLOCK FLAG
                                .EXIT               /0, GO BACK TO MONITOR
  10516 A 000000 A *G           CAL
  10517 A 000015 A *G           15
                        /
  10520 A 000000 A      PRINT   0
  10521 A 760061 A              LAW 61              /ASCII CODE FOR "1"
  10522 A 700406 A              TLS                 /PRINT IT
  10523 A 630520 A              JMP* PRINT          /RETURN
                        /
          010500 A              .END START
  10524 A 000007 A *L
              SIZE=10525      NO ERROR LINES
```

```
                        .TITLE PRINT "1" AFTER 5 SECONDS--USING PI
                    /
                    /
                    / PROGRAM USES PI WITH THE REAL TIME CLOCK
                    / TO PRINT A "1" ON THE TELEPRINTER EVERY 5 SECONDS.
                    / WHILE CLOCK IS TICKING, THE LINK IS BLINKED AS A
                    / BACKGROUND PROGRAM.
                    /
                    /
          700001    CLSF=700001
          700044    CLON=700044
          700406    TLS=700406
          700402    TCF=700402
                    /
                        .ABS
                    /
    00000               .LOC 0
                    /
    00000   000000      0
    00001   620002      JMP* .+1
    00002   010512      INT
                    /
    10500               .LOC 10500
                    /
    10500   700042  START   ION             /TURN PI ON
    10501   170535          DZM* (7         /CLEAR CLOCK COUNTER
                    /
    10502   777324  MORE    LAW -454        /ADD -5 SECONDS IN TICKS
    10503   370535          TAD* (7         /TO CLOCK COUNTER
    10504   070535          DAC* (7         /RESTORE TO CLOCK COUNTER
                    /
    10505   700044          CLON            /START CLOCK AND CLEAR FLAG
    10506   450533          ISZ COUNT#      /START BLINKING THE LINK
    10507   610506          JMP .-1         /WHEN COUNT OVERFLOWS.
    10510   740002          CML             /COMPLEMENT LINK
    10511   610506          JMP .-3
                    /
    10512   230536  INT     LAC* (0         /PICK UP RETURN PC
    10513   050534          DAC RET#        /SET UP FOR RETURN
    10514   700001          CLSF            /CHECK IF INTERRUPT FROM CLOCK
    10515   610527          JMP CLR         /IF NOT CLOCK--THEN JUST RETURN
    10516   760061          LAW 61          /IT IS, SO GET CODE FOR 1
    10517   700406          TLS             /AND PRINT
                    /
                        .EJECT
```

```
                        /
10520   777324                  LAW -454        /RESTORE CLOCK COUNTER AS BEFORE
10521   370535                  TAD* (7
10522   070535                  DAC* (7
                        /
10523   700044                  CLON            /CLEAR CLOCK FLAG
10524   700042                  ION             /TURN PI BACK ON
10525   707742                  RES             /SET UP TO RESTORE LINK,PAGE/BAN
10526   630534                  JMP* RET        /RETURN TO INTERRUPTED PROGRAM
10527   703302          CLR     CAF             /CLEAR ALL FLAGS--WAS NOT CLOCK
10530   700042                  ION
10531   707742                  RES             /SET UP TO RESTORE LINK,PAGE/BAN
10532   630534                  JMP* RET                /RETURN
                        /
        010500                  .END START
10535   000007    *L
10536   000000    *L
            SIZE=10537      NO ERROR LINES
```

```
                                        .TITLE CLOCK DIAGNOSTIC WITH PI ENABLED
                              /
                              /ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
                              /
                              /PROGRAM PERFORMS ROTATION OF AC WHEN CLOCK AND PI ARE
                              /ENABLED. WHILE THIS IS GOING ON, THE LINK CONTINUALLY
                              /BLINKS (WHETHER OR NOT THE CLOCK AND PI ARE ENABLED).
                              /
                              /IF WHEN PROGRAM STARTS, CLOCK IS NOT WORKING OR PI IS
                              /DISABLED THEN AC=1.
                              /IF CLOCK AND PI ARE ENABLED AND WORKING, THEN THE AC
                              /IS ROTATED ONE BIT POSITION TO THE LEFT ONCE A SECOND
                              /UNTIL THE AC=400000.  THEN THE AC IS ROTATED ONE BIT
                              /POSITION TO THE RIGHT EVERY SECOND UNTIL THE AC=000001.
                              /IF AT ANY POINT WHILE THIS IS GOING ON, THE CLOCK OR PI
                              /ARE DISABLED FROM THE CONSOLE, THE DISPLAYED CONTENT
                              /OF THE AC IS FROZEN UNTIL THEY ARE ENABLED.
                              /
                              /USER MAY RETURN TO THE MONITOR BY SETTING DATA SWITCHES
                              /TO ZERO.
                              /
                              /ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
                              /
          700044 A            CLON=700044
          700001 A            CLSF=700001
                              /
00000 A                                 .LOC 0
00000 A 000000 A              PI        0
00001 A 600500 A                        JMP SKPCHN        /JUMP TO SKIP CHAIN
                              /
00500 A                                 .LOC 500
00500 A 700001 A              SKPCHN    CLSF              /WAS IT THE CLOCK THAT INTERRUPT
00501 A 600524 A                        JMP CLR           /IF NOT CLOCK, THEN ILLEGAL
00502 A 040615 A                        DAC ACSAVE        /SAVE THE AC FOR ROTATION
00503 A 540617 A                        SAD (400000       /CHECK IF AC NOW AT EXTREME LEFT
00504 A 600530 A                        JMP CHANGE        /YES--CHANGE TO ROTATING RIGHT
00505 A 540620 A                        SAD (000001       /NO--IS IT AT EXTREME RIGHT?
00506 A 600530 A                        JMP CHANGE        /YES--CHANGE TO ROTATING LEFT
                              /
                              /*******************************************************
                              /BEFORE ROTATING THE AC, CHECK TO SEE IF USER WANTS TO
                              /TO RETURN TO MONITOR.  IF  DATA SWITCHES ARE ZERO, GO
                              /BACK TO MONITOR. IF DATA SWITCHES ARE NON 0, CONTINUE
                              /ON WITH THE PROGRAM.
                              /*******************************************************
                              /
00507 A 750004 A              CHECK     LAS               /READ THE DATA SWITCHES
00510 A 741200 A                        SNA               /IF NOT 0 CONTINUE ON
00511 A 600613 A                        JMP MONTOR        /IF 0, GO TO .EXIT
                              /
                                        .EJECT
```

```
                        /
                        /*************************************************************
                        /GET AC VALUE BACK AND ROTATE RIGHT OR LEFT ACCORDING
                        /TO THE INSTRUCTION IN LOCATION ROTATE (THIS MAY BE
                        /MODIFIED IN THE CHANGE SECTION).
                        /THEN SET THE CLOCK TO TICK FOR ANOTHER SECOND BEFORE
                        /INTERRUPTING....THEN RETURN TO THE INTERRUPTED PROGRAM.
                        /*************************************************************
                        /
00512 A 200615 A        GETAC   LAC ACSAVE      /GET AC VALUE TO BE ROTATED
00513 A 744020 A        ROTATE  RCR
00514 A 040615 A                DAC ACSAVE#
00515 A 777704 A                LAW -74
00516 A 040007 A                DAC 7
00517 A 200615 A                LAC ACSAVE#
00520 A 700044 A                CLON
00521 A 707742 A                RES
00522 A 700042 A                ION
00523 A 620000 A                JMP* 0
                        /
                        /*************************************************************
                        /GET HERE FROM LOCATION 501 WHEN  INTERRUPT IS NOT FROM
                        /THE CLOCK.  ALL WE DO IS CLEAR ALL FLAGS AND RETURN TO
                        /THE INTERRUPTED PROGRAM.
                        /*************************************************************
                        /
00524 A 703302 A        CLR     CAF
00525 A 707742 A                RES
00526 A 700042 A                ION
00527 A 620000 A                JMP* 0
                        /
                        /*************************************************************
                        /GET HERE FROM 504 OR 506 WHEN WE WANT TO CHANGE THE
                        /DIRECTION OF ROTATION.  THE CHANGE IS SIMPLY DONE:
                        /RCR=744020 WHILE RCL=744010. NOTICE THAT IF YOU XOR ONE
                        /INSTRUCTION WITH THE VALUE 000030, YOU GET THE OTHER
                        /INSTRUCTION.
                        /*************************************************************
                        /
00530 A 200513 A        CHANGE  LAC ROTATE
00531 A 240621 A                XOR (000030
00532 A 040513 A                DAC ROTATE
00533 A 600507 A                JMP CHECK
                        /
                                .EJECT
```

```
                        /
                        /**************************************************
                        /MAIN SECTION OF PROGRAM
                        /SET UP THE CLOCK INITIALLY
                        /PLACE 000001 IN AC
                        /BLINK THE LINK
                        /**************************************************
                        /
00600 A                         .LOC 600
                        /
00600 A 750000 A        START   CLA
00601 A 705504 A                ISA
00602 A 777704 A                LAW -74
00603 A 040007 A                DAC 7
00604 A 754030 A                CLA!CLL!IAC
00605 A 700042 A                ION
00606 A 700044 A                CLON
00607 A 440616 A        COMPL   ISZ COUNT#
00610 A 600607 A                JMP .-1
00611 A 740002 A                CML
00612 A 600607 A                JMP COMPL
                        /
                        /**************************************************
                        /GO BACK TO THE MONITOR WITH .EXIT
                        /**************************************************
                        /
00613 A                 MONTOR  .EXIT
00613 A 000000 A *G             CAL
00614 A 000015 A *G             15
                        /
                        /
                        /**************************************************
                        /NOTE THAT IN ORDER FOR THIS PROGRAM TO BE LOADED
                        /BY THE LINKING LOADER, API MUST BE ENABLED.
                        /THIS IS BECAUSE AS SOON AS LOCATION 1 IS LOADED,
                        /THE SYSTEM'S USE OF PI FOR LOADING IS CORRUPTED.
                        /
                        /$API ON
                        /
                        /$GLOAD
                        /LOADER VXX
                        /R←PICLK
                        /
                        /**************************************************
                        /
        000600 A                .END START
00617 A 400000 A *L
00620 A 000001 A *L
00621 A 000030 A *L
            SIZE=00622      NO ERROR LINES
```

```
                                .TITLE INTERRUPT LINE EDITOR PROBLEM
                        /
                        /A PROGRAM TO ECHO AN EDITED LINE
                        /USING THE PROGRAM INTERRUPT CONTROL
                        /
                        / EMPLOY'S TWO SOFTWARE SWITCHES:
                        /
                        /  "CRLF" = CARRIAGE RETURN LINE FEED
                        /       (-1=CR,0=LF)
                        /  "MODE" = INPUT/OUTPUT CONTROL
                        /       (-1=OUTPUT,0=INPUT,+1=CRLF ONLY)
                        /
                                .ABSP
                        /
            700301      KSF=700301
            700312      KRS=700312
            700402      TCF=700402
            700401      TSF=700401
            700406      TLS=700406
                        /
    00000                       .LOC 0
    00000   000000      PIC     0
    00001   600112              JMP SEVINT
                        /
    00100                       .LOC 100
    00100   700042      START   ION             /TURN ON "PIC"
    00101   700416              TLS+10          /SET UP 1ST INTERRUPT
    00102   100217              JMS SETUP       /SET UP VARIABLES
    00103   750001              CLA!CMA         /SET -1 IN "AC"
    00104   040337              DAC CRLF#       /SET UP "CRLF"
    00105   440340              ISZ MODE#       /CRLF ONLY!
    00106   440342              ISZ TIME#
    00107   600106              JMP .-1
    00110   740002              CML
    00111   600106              JMP .-3
                        /
                        / INTERRUPT SKIP CHAIN
                        /
    00112   040341      SEVINT  DAC SAVAC#      /SAVE INTERRUPTED "AC"
    00113   700301              KSF             /KEYBOARD?
    00114   741000              SKP
    00115   600166              JMP KB          /YES
    00116   700401              TSF             /PRINTER?
    00117   741000              SKP
    00120   600126              JMP TP          /YES
    00121   740040              HLT             /ILLEGAL INTERRUPT!
                        /
    00122   200341      EXIT    LAC SAVAC       /RESTORE SAVED "AC"
    00123   700042              ION             /TURN INTERRUPT BACK ON
    00124   703344              DBR             /RESET LINK
    00125   620000              JMP* PIC        /RETURN TO INTERRUPTED PROGRAM
                        /
```

```
                        /
                        / TELETYPE SERVICE ROUTINE
                        /
00126   700402  TP      TCF                     /CLEAR INTERRUPTING FLAG
00127   200340          LAC MODE                /INPUT,OUTPUT OR CRLF?
00130   751200          SNA!CLA
00131   600122          JMP EXIT                /INPUT!..GET OUT!
00132   200337          LAC CRLF
00133   741200          SNA
00134   600156          JMP LF                  /0=LINE FEED
00135   751100          SPA!CLA
00136   600152          JMP CR                  /-1=CARRIAGE RETURN
00137   230226          LAC* BUFPTR,X
00140   700406          TLS                     /PRINT CHARACTER
00141   725001          AXS +1                  /ADVANCE XR,72 CHARACTERS?
00142   540343          SAD (215                /CARRIAGE RETURN?
00143   741000          SKP
00144   600122          JMP EXIT                /MORE TO GO!
00145   100217          JMS SETUP               /TERMINATE OUTPUT
00146   440340          ISZ MODE                /SWITCH TO INPUT
00147   750001          CLA!CMA
00150   040337          DAC CRLF
00151   600122          JMP EXIT
                        /
00152   760215  CR      LAW 215
00153   700406          TLS
00154   140337          DZM CRLF                /SET TO LINE FEED
00155   600122          JMP EXIT
                        /
00156   760212  LF      LAW 212
00157   700406          TLS
00160   200340          LAC MODE
00161   750100          SMA!CLA                 /+1=CRLF ONLY
00162   140340          DZM MODE                /SHUT OFF OUTPUT
00163   440337          ISZ CRLF                /SHUT OFF CRLF
00164   735000          CLx                     /RESET TABLE FOR OUTPUT OR INPUT
00165   600122          JMP EXIT
                        /
                        .EJECT
```

```
                        /
                        / KEYBOARD SERVICE ROUTINE
                        /
00166   700312  KB      KRB
00167   540344          SAD (225
00170   600205          JMP LNDLT       /YES
00171   540345          SAD (377        /RUBOUT?
00172   600211          JMP RUBDLT      /YES
00173   070226          DAC* BUFPTR,X   /STORE CHARACTER
00174   725001          AXS +1          /ADVANCE XR,72 CHARACTERS?
00175   540343          SAD (215        /CARRIAGE RETURN?
00176   741000          SKP             /TERMINATE!
00177   600122          JMP EXIT        /MORE TO COME!
00200   750001          CLA!CMA         /SET UP FOR OUTPUT
00201   040340          DAC MODE
00202   040337          DAC CRLF
00203   700416          TLS+10          /GENERATE INTERRUPT!
00204   600122          JMP EXIT
                        /
00205   735000  LNDLT   CLX
00206   760300          LAW 300
00207   700406          TLS
00210   600122          JMP EXIT
                        /
00211   724000  RUBDLT  PXA
00212   740200          SZA             /BEGINING OF LINE?
00213   737777          AXR -1          /NO!
00214   760334          LAW 334
00215   700406          TLS
00216   600122          JMP EXIT
                        /
00217   000000  SETUP   0
00220   140340          DZM MODE
00221   140337          DZM CRLF
00222   735000          CLX
00223   200346          LAC (110
00224   722000          PAL
00225   620217          JMP* SETUP
                        /
00226   000227  BUFPTR  BUFF
00227           BUFF    .BLOCK 110
        000100          .END START
00343   000215  *L
00344   000225  *L
00345   000377  *L
00346   000110  *L
            SIZE=00347      NO ERROR LINES
```

```
                                .TITLE INTERRUPT TEST ROUTINE
                        /
                                .ABSP
                        /
00000                           .LOC 0
                        /
00000   000000          INT     0
00001   041055                  DAC SAVE#
00002   700314                  IORS
00003   600100                  JMP CHECK
                        /
00100                           .LOC 100
                        /
                        /
00100   041056          CHECK   DAC SAVSTA#     /SAVE IORS WORD
00101   501060                  AND (300000     /MASK UNEXPECTED FLAGS
00102   541060                  SAD (300000     /CHECK FOR SIMULTANEOUS FLAGS
00103   600300                  JMP READ        /HANDLE READER FIRST
00104   541061                  SAD (200000     /CHECK FOR READER FLAG
00105   600300                  JMP READ
00106   541062                  SAD (100000     /CHECK FOR PUNCH FLAG
00107   600327                  JMP PUNCH
00110   201063                  LAC (7
00111   700406                  TLS             /RING BELL ON TTY
00112   201056                  LAC SAVSTA      /HALT WITH IORS DISPLAYED
00113   740040                  HLT             /IN AC ON UNEXPECTED INTERRUPT.
                        /
00200                           .LOC 200
                        /
00200   201064          INIT    LAC (500
00201   722000                  PAL             /SETUP LR
00202   141053                  DZM PINDX#
00203   141054                  DZM RINDX#
00204   141057                  DZM TEMP#
00205   707762                  DBA             /SETUP FOR INDEX MODE
00206   735000                  CLX
00207   700042                  ION             /TURN ON INTERRUPT
00210   750000                  CLA
00211   700204                  PSA
00212   700104                  RSA             /MOVE TAPE TO GET FIRST INTERRUP
00213   754030          MAIN    CLA!CLL!IAC
00214   740010                  RAL
00215   441052                  ISZ CNTR#
00216   600215                  JMP .-1
00217   740100                  SMA
00220   600214                  JMP MAIN+1
00221   740020          RIGHT   RAR
00222   441052                  ISZ CNTR
00223   600222                  JMP .-1
00224   541065                  SAD (1
00225   600214                  JMP MAIN+1
00226   600221                  JMP RIGHT
```

```
                        /
00300                   .LOC 300
                        /
00300   201054  READ    LAC RINDX       /GET READER INDEX
00301   721000          PAX             /SETUP XR
00302   700112          RRB             /READ READER BUFFER
00303   050352          DAC BUFF,X      /STORE IT!
00304   201056          LAC SAVSTA      /GET IORS WORD
00305   501066          AND (1000
00306   041057          DAC TEMP        /SAVE FOR CHECK WHEN PUNCH DONE
00307   541066          SAD (1000       /CHECK FOR READER NO TAPE
00310   600322          JMP SETLIM      /SETUP LR FOR PUNCH
00311   725001          AXS+1           /INCREMENT XR,SKIP IF = LR
00312   741000          SKP
00313   600325          JMP RESET
00314   724000          PXA             /GET READER INDEX
00315   041054          DAC RINDX       /SAVE IT
00316   700104          RSA             /MOVE TAPE YOU'RE NOT DONE YET
00317   201055  RET     LAC SAVE
00320   700042          ION
00321   620000          JMP* INT        /RETURN TO MAIN PROGRAM
                        /
00322   725777  SETLIM  AXS -1          /SETUP FINAL LIMIT FOR PUNCH
00323   726000          PXL
00324   600317          JMP RET
00325   141054  RESET   DZM RINDX       /REINITIALIZE READER INDEX
00326   600317          JMP RET
                        /
00327   201053  PUNCH   LAC PINDX
00330   721000          PAX
00331   210352          LAC BUFF,X
00332   700206          PSA +2          /PUNCH A CHARACTER
00333   725001          AXS +1          /INCREMENT XR,SKIP IF = LR
00334   741000          SKP
00335   600341          JMP RNT         /XR=LR,CHECK FOR RDR NO TAPE
00336   724000          PXA             /GET PUNCH INDEX
00337   041053          DAC PINDX       /SAVE IT!
00340   600317          JMP RET
00341   201057  RNT     LAC TEMP        /GET MASKED IORS WORD
00342   541066          SAD (1000       /CHECK FOR RDR NO TAPE
00343   600347          JMP AHEAD       /GET OUT OF INTERRUPT ROUTINES
                                        /YOU'RE DONE!
00344   141053          DZM PINDX       /REINITIALIZE PUNCH INDEX
00345   700104          RSA             /START THE READER
00346   600317          JMP RET
00347   703302  AHEAD   CAF             /CLEAR THE WORLD!
00350   400317          XCT RET         /RESTORE THE AC FOR MAIN
                                        /DON'T TURN INTERRUPT ON.
00351   620000          JMP* INT
                        /
00352           BUFF    .BLOCK 500
                        /
```

```
                        /
            700406      TLS=700406
            700204      PSA=700204
            700104      RSA=700104
            700112      RRB=700112
            000000              .END
  01060     300000    *L
  01061     200000    *L
  01062     100000    *L
  01063     000007    *L
  01064     000500    *L
  01065     000001    *L
  01066     001000    *L
          SIZE=01067      NO ERROR LINES
```

```
                                .TITLE PROGRAM TO CONVERT IOPS ASCII TO IMAGE AL
                        /
                        /PROGRAM INPUTS IN IOPS ASCII FROM THE TTY,
                        /UNPACKS AND STORES AS IMAGE ALPHA,
                        /SORTS THE CHARACTERS IN ASCENDING ORDER
                        /AND OUTPUTS ON WHATEVER DEVICE IS ATTACHED
                        /TO DAT SLOT 3--USUALLY THE TT OR LP.
                        /
                                .IODEV =2,3
                        /
                                .GLOBL SORT
                        /
00000 R                 GO      .INIT 3,1,RST   /TT OR LP FOR OUTPUT
                                .INIT =2,0,RST  /TT ONLY FOR INPUT
                        /
00010 R                 MORE    .READ =2,2,BUFFI,34
                                .WAIT =2
                        /
00010 R 100225 R                JMS UNPACK
00017 R 000041 R                BUFFI           /ADDRESS OF INPUT BUFFER
00020 R 000103 R                BUFFO           /ADDRESS OF OUTPUT BUFFER
                        /
00021 R 200336 R                LAC (BUFFO+2    /GET ADDRESS OF DATA NOT HEADER
00022 R 652000 A                LMQ             /SUB REQUIRES IT IN MQ REGISTER
00023 R 200331 R                LAC HDCNT       /GET WORD COUNT IN AC
00024 R 120335 E                JMS* SORT       /GO OFF TO SORTING ROUTINE
                        /
                        /
                                .WRITE 3,3,BUFFO,82
                                .WAIT 3
00033 R 600010 R                JMP MORE
                        /
00034 R                 RST     .CLOSE =2
                                .CLOSE 3
00040 R 600000 R                JMP GO
                        /
00041 R           A     BUFFI   .BLOCK 42
00103 R 000000 A      BUFFO   0
00104 R 000000 A              0
00105 R           A     BUFDT   .BLOCK 120
                        /
                        /
                        /       UNPACK ROUTINE--IOPS ASCII TO IMAGE ALPHA
                        /
                        /
00225 R 000000 A      UNPACK  0
00226 R 220225 R                LAC* UNPACK     /PICK UP BUFFI ADDRESS
00227 R 040330 R                DAC     FROM#
00230 R 440225 R                ISZ UNPACK
00231 R 220225 R                LAC* UNPACK     /PICK UP BUFFO ADDRESS
00232 R 040334 R                DAC TOP#
00233 R 723002 A                AAC +2          /GET AROUND HEADER WORDS
```

```
00234 R 040333 R            DAC TOW          /POINTER TO DATA AREA
00235 R 440225 R            ISZ UNPACK       /SET UP FOR EXIT
00236 R 220330 R            LAC* FROM        /GET HEADER WORD 0 OF INPUT
00237 R 500337 R            AND (377000      /GET WORD PAIR COUNT (BITS 1-8)
00240 R 340340 R            TAD (-1000       /SUBTRACT ONE (TO EXCLUDE
                                             /HEADER WORD PAIR
00241 R 742030 A            SWHA             /PLACE WD PR COUNT IN BITS 9-17
00242 R 740031 A            TCA              /-NWORD PAIR COUNT
00243 R 040327 R            DAC COUNT#       /SAVE FOR LATER IN UNPACK
                                             /AND FOR USE IN SORT ROUTINE.
00244 R 220330 R            LAC FROM         /PICK UP HW0 OF INPUT BUFFER
00245 R 723002 A            AAC +2           /GET AROUND HEADER WORDS
00246 R 040330 R            DAC FROM         /FROM NOW POINTS TO FIRST DATA
                    /
00247 R 140331 R            DZM HOCNT#       /CLEAR IMAGE ALPHA CHAR. COUNTER
                    /
                    /DO THE ACTUAL UNPACKING
                    /
00250 R 220330 R    DO      LAC* FROM        /GET THE 1ST OF A WORD PAIR
00251 R 742030 A            SWHA
00252 R 742020 A            RTR
00253 R 100321 R            JMS STORE        /GET THE FIRST OF FIVE
00254 R 220330 R            LAC* FROM        /PICK UP FIRST WORD AGAIN
00255 R 742020 A            RTR      /MOVE 2ND CHARACTER INTO BITS 11-17
00256 R 742020 A            RTR
00257 R 100321 R            JMS STORE        /STORE THE 2ND OF FIVE
                    /
00260 R 220330 R            LAC* FROM
00261 R 742010 A            RTL
00262 R 740010 A            RAL
00263 R 500341 R            AND (170
00264 R 040332 R            DAC TEMP#
00265 R 440330 R            ISZ FROM
00266 R 220330 R            LAC* FROM        /PICK UP 2ND WORD OF PAIR
00267 R 742010 A            RTL              /MOVE 2ND HALF OF 3RD CHAR
00270 R 742010 A            RTL              /TO BITS 15-17
00271 R 500342 R            AND (7
00272 R 340332 R            TAD TEMP
00273 R 100321 R            JMS STORE        /FINALLY--STORE THE 3RD OF FIVE
                    /
00274 R 220330 R            LAC* FROM        /PICK UP 2ND WORD AGAIN TO GET
00275 R 740010 A            RAL              /AT THE 4TH OF FIVE--MOVE TO BIT
00276 R 742030 A            SWHA             /NOW TO BITS 11-17
00277 R 100321 R            JMS STORE        /STORE 4TH OF FIVE
                    /
00300 R 220330 R            LAC* FROM        /PICK UP WORD 2 FOR CHARACTER 5
00301 R 740020 A            RAR              /PLACE IN BITS 11-17
00302 R 100321 R            JMS STORE        /STORE LAST OF FIVE
                    /
                            .EJECT
```

```
00303 R 440330 R            ISZ FROM          /SETUP TO DO IT WITH NEXT 2 WDS
00304 R 440327 R            ISZ COUNT         /ARE WE DONE WITH ALL 5/7 WR
00305 R 000250 R            JMP DO            /NO--GO BACK FOR MORE
                      /
00306 R 200331 R            LAC HDCNT         /YES,DONE-INCLUDE HEADER WORD PA
00307 R 723003 A            AAC +3            /AND ONE FOR GOOD LUCK
00310 R 742030 A            SWHA              /DIVIDE BY 2 AND POSITION IN
00311 R 740020 A            RAR               /BITS 1-8
00312 R 500337 R            AND (377000
00313 R 723003 A            AAC +3            /SET DATA MODE TO IMAGE
00314 R 050334 R            DAC* TOP
00315 R 440334 R            ISZ TOP
00316 R 160334 R            DZM* TOP
00317 R 160333 R            DZM* TO
00320 R 620225 R            JMP* UNPACK
                      /
                      /
00321 R 000000 A     STORE  0
00322 R 500343 R            AND (177
00323 R 050333 R            DAC* TO
00324 R 440333 R            ISZ TO
00325 R 440331 R            ISZ HDCNT
00326 R 620321 R            JMP* STORE
                      /
       000000 R             .END GO
00335 R 000335 E *E
00336 R 000105 R *L
00337 R 377000 A *L
00340 R 777000 A *L
00341 R 000170 A *L
00342 R 000007 A *L
00343 R 000177 A *L
          SIZE=00344     NO ERROR LINES
```

```
                                .TITLE SOURCE FILE LISTER
                        /
                        /oooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
                        /
                        /LAB PROBLEM - DEVISE A PROGRAM TO LIST A "SRC" FILE OF
                        /ANY LENGTH ON A GIVEN OUTPUT DEVICE
                        /
                        /EXAMPLE OF A GLOBAL CALL
                        /
                        /INPUT DEVICE IS .DAT -15 (DISK OR DT)
                        /OUTPUT DEVICE IS .DAT 3 (TT OR LP)
                        /COMMAND STRING IS .DAT -2 (MUST BE TTY)
                        /
                        /NAME IS AN INTERNAL GLOBAL
                        /FILNAM IS AN EXTERNAL GLOBAL
                        /
                        /oooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
                        /
                                .GLOBL NAME,FILNAM
                        /
                                .IODEV -2,3,-15
                        /
00000 R                 START   .INIT -2,0,RST
                                .INIT 3,1,0
                                .INIT -15,0,0
                        /
                                .WRITE -3,2,MESS1,34
                                .WRITE -3,2,MESS2,34
                                .WRITE -3,2,MESS3,34
                                .WAIT -3
                        /
                                .READ -2,2,ANS,6
                                .WAIT -2
                                .CLOSE -2
                        /
00042 R 200167 R                LAC ANS+2               /FIRST PART OF 5/7 ASCII
00043 R 040050 R                DAC AUG1
00044 R 200170 R                LAC ANS+3               /SECOND PART OF 5/7 ASCI
00045 R 040051 R                DAC AUG2
00046 R 200171 R                LAC ANS+4               /LAST PART OF 5/7 ASCII
00047 R 120263 E                JMS* FILNAM             /GO CONVERT 5/7 TO SIXBT
00050 R 000000 A        AUG1    0
00051 R 000000 A        AUG2    0
                        /
                                .FSTAT -15,NAME         /FILENAME ON DEVICE?
00055 R 741200 A                SNA                     /FILE PRESENT?
00056 R 600111 R                JMP RST                 /NO GO RESTART
                                .SEEK -15,NAME          /YES OPEN AND READ FIRST
                        /
00062 R                 MORE    .READ -15,2,BUFF,255    /GET ONE LINE OF TEXT
                                .WAIT -15
                        /
```

```
GE   2     FILLST SRC      SOURCE FILE LISTER

                           /
00070 R 200120 R                   LAC BUFF
00071 R 500264 R                   AND (7
00072 R 540265 R                   SAD (5                          /DONE?
00073 R 600103 R                   JMP DONE                        /YES
                                   .WRITE 3,2,BUFF,34                          /NO - CO
                                   .WAIT 3                         /AND OUTPUT THE LINE
00102 R 600062 R                   JMP MORE                        /GO BACK FOR ANOTHER LIN
                           /
00103 R                    DONE    .WRITE -3,2,MESS4,34
                                   .WAIT -3
00111 R                    RST     .CLOSE -3
                                   .CLOSE -2
                                   .CLOSE -15
00117 R 600000 R                   JMP START
00120 R          A         BUFF    .BLOCK 42
                           /
                           /
00162 R 000000 A           NAME    .SIXBT "@@@@@@SRC"
00163 R 000000 A
00164 R 232203 A
                           /
00165 R          A         ANS     .BLOCK 10
                           /
00175 R 010002 A           MESS1   MESS2-MESS1/2*1000+2
00176 R 000000 A                   0
00177 R 445012 A                   .ASCII "I WILL RETRIEVE ANY 'SRC' FILE"<15>
00200 R 744630 A
00201 R 461012 A
00202 R 242650 A
00203 R 512230 A
00204 R 553212 A
00205 R 202031 A
00206 R 654500 A
00207 R 236472 A
00210 R 241516 A
00211 R 202151 A
00212 R 146212 A
00213 R 064000 A
00214 R 000000 A
00215 R 012002 A           MESS2   MESS3-MESS2/2*1000+2
00216 R 000000 A                   0
00217 R 522632 A                   .ASCII "TYPE IN A FULL SIX CHARACTER "
00220 R 042500 A
00221 R 446344 A
00222 R 340500 A
00223 R 432531 A
00224 R 446100 A
00225 R 516233 A
00226 R 020206 A
00227 R 442032 A
00230 R 240606 A
```

```
00231 R 522132 A
00232 R 220000 A
00233 R 432231 A              .ASCII "FILE NAME."<015>
00234 R 442500 A
00235 R 472031 A
00236 R 542534 A
00237 R 064000 A
00240 R 000000 A
00241 R 006002 A     MESS3    MESS4-MESS3/2*1000+2
00242 R 000000 A              0
00243 R 426610 A              .ASCII "EXAMPLE** ECHO@@<CR>"<015>
00244 R 146640 A
00245 R 462125 A
00246 R 225100 A
00247 R 426071 A
00250 R 047600 A
00251 R 401710 A
00252 R 351174 A
00253 R 064000 A
00254 R 000000 A
00255 R 003002 A     MESS4    ENDM=MESS4/2*1000+2
00256 R 000000 A              0
00257 R 422371 A              .ASCII "DONE!"<015>
00260 R 642502 A
00261 R 064000 A
00262 R 000000 A
        000263 R     ENDM=.
        000000 R              .END START
00263 R 000263 E *E
00264 R 000007 A *L
00265 R 000005 A *L
            SIZE=00266      NO ERROR LINES
```

```
                        .TITLE ROUTINE TO CONVERT 5/7 ASCII TO SIXBIT AS
                    /
                    /ROUTINE TO CONVERT 5/7 PACKED ASCII NAME TO SIXBIT NAME
                    /FILE NAME IS "ABCDEF" FOR COMMENTS
                    /"X"IS A DON'T CARE POSITION
                    /"0" IS A POSITION KNOWN TO BE ZERO
                    /BIT POSITIONS ARE GIVEN BY CHARACTER LETTERS A-F OF FIL
                    /
                    /
                    /       CALL SEQUENCE
                    /       JMS* FILNAM
                    /       AUG1                    /XAAAAAAXBBBBBBXCCC
                    /       AUG2                    /CCCXDDDDDDXEEEEEE0
                    /       AC=AUG3                 /XFFFFFF000000000000
                    /
                    /
                    /"NAME" IS AN EXTERNAL GLOBL
                    /"FILNAM" IS AN INTERNAL GLOBL
                    /
                        .GLOBL NAME,FILNAM
00000 R 000000 A    FILNAM  0
00001 R 040046 R            DAC SAVAC               /SAVE AUG3
00002 R 200050 E            LAC NAME
00003 R 723001 A            AAC 1
00004 R 040047 R            DAC NAME1
00005 R 775767 A            LAW 15767               / "B"&"C" 7TH BITS TO 0
00006 R 520000 R            AND* FILNAM             /AC=XAAAAAA0BBBBBB0CCC
00007 R 744010 A            CLLIRAL                 /AC=AAAAAA0BBBBBB0CCC0
00010 R 744000 A            CLL
00011 R 100037 R            JMS LFTPK               /MOVE "B"&"C" ONE LEFT
00012 R 003756 A            3756                    /AC=AAAAAABBBBBB0CCC00
00013 R 100037 R            JMS LFTPK               /MOVE "C" ONE  LEFT
00014 R 003734 A            34                      /AC=AAAAAABBBBBBCCC000
00015 R 060050 E            DAC* NAME
00016 R 440000 R            ISZ FILNAM
00017 R 220000 R            LAC* FILNAM             /AC=CCCXDDDDDDXEEEEEE0
00020 R 640517 A            LRS 17                  /AC=000000000000000CCC
00021 R 360050 E            TAD* NAME               /AC=AAAAAABBBBBBCCCCCC
00022 R 060050 E            DAC* NAME
00023 R 640623 A            LLS 23                  /AC=DDDDDDXEEEEEE000000
00024 R 500051 R            AND (773777             /FORCE AC 6 TO 0
00025 R 100037 R            JMS LFTPK               /MOVE "E" ONE  LEFT
00026 R 003740 A            3740                    /AC=DDDDDDEEEEEE000000
00027 R 060047 R            DAC* NAME1
00030 R 200046 R            LAC SAVAC               /AC=XFFFFFF000000000000
00031 R 640513 A            LRS 13                  /AC=000000000000XFFFFFF
00032 R 500052 R            AND (77                 /FORCE AC 11 TO 0
00033 R 360047 R            TAD* NAME1              /AC=DDDDDDEEEEEEFFFFFF
00034 R 060047 R            DAC* NAME1
00035 R 440000 R            ISZ FILNAM
00036 R 620000 R            JMP* FILNAM
                            .EJECT
```

```
00037 R 000000 A     LFTPK    0
00040 R 040045 R              DAC T1
00041 R 520037 R              AND* LFTPK
00042 R 340045 R              TAD T1
00043 R 440037 R              ISZ LFTPK
00044 R 620037 R              JMP* LFTPK
                      /
00045 R 000000 A     T1       0
00046 R 000000 A     SAVAC    0
00047 R 000000 A     NAME1    0
        000000 A              .END
00050 R 000050 E *E
00051 R 773777 A *L
00052 R 000077 A *L
            SIZE=00053      NO ERROR LINES
```

```
                        .TITLE "A SUBROUTINE TO CONVERT 5/7 ASCII TO 6 B
                    /
                        .GLOBL NAME,FILNAM
                    /
00000 R 000000 A    FILNAM  0
00001 R 040061 R            DAC SAVAC#      /SAVE LAST ARGUMENT
00002 R 220000 R            LAC* FILNAM     /GET FIRST
                    /
                    /GET 2 AND 1/2 SIXBIT CHARACTERS
                    /
00003 R 740010 A            RAL
00004 R 040062 R            DAC TEMP#
00005 R 500064 R            AND (770000
00006 R 060063 E            DAC* NAME
00007 R 200062 R            LAC TEMP
00010 R 740010 A            RAL
00011 R 040062 R            DAC TEMP
00012 R 500065 R            AND (007700
00013 R 360063 E            TAD* NAME
00014 R 060063 E            DAC* NAME
00015 R 200062 R            LAC TEMP
00016 R 740010 A            RAL
00017 R 500066 R            AND (000070
00020 R 040062 R            DAC TEMP
00021 R 440000 R            ISZ FILNAM
00022 R 220000 R            LAC* FILNAM     /GET SECOND
                    /
                    /COMPLETE 5 SIX BIT CHARACTERS
                    /
00023 R 500067 R            AND (700000
00024 R 744000 A            CLL
00025 R 742010 A            RTL
00026 R 742010 A            RTL
00027 R 340062 R            TAD TEMP
00030 R 360063 E            TAD* NAME
00031 R 060063 E            DAC* NAME
00032 R 220000 R            LAC* FILNAM
00033 R 742010 A            RTL
00034 R 742010 A            RTL
00035 R 040062 R            DAC TEMP
00036 R 500064 R            AND (770000
00037 R 440063 E            ISZ NAME
00040 R 060063 E            DAC* NAME
00041 R 200062 R            LAC TEMP
00042 R 740010 A            RAL
00043 R 500065 R            AND (007700
00044 R 360063 E            TAD* NAME
00045 R 060063 E            DAC* NAME
00046 R 200061 R            LAC SAVAC       /ASSEMBLE LAST SIX BIT
00047 R 742030 A            SWHA            /CHARACTER
00050 R 742020 A            RTR
00051 R 500070 R            AND (000077
```

```
GE   2      FILENM SRC      "A SUBROUTINE TO CONVERT 5/7 ASCII TO 6 BIT"

 00052 R 360063 E             TAD* NAME
 00053 R 060063 E             DAC* NAME
 00054 R 777777 A             LAW -1
 00055 R 340063 E             TAD NAME          /RESTORE CONTENT OF NAME
 00056 R 040063 E             DAC NAME
 00057 R 440000 R             ISZ FILNAM
 00060 R 620000 R             JMP* FILNAM
                      /
         000000 A             .END
 00063 R 000063 F *E
 00064 R 770000 A *L
 00065 R 007700 A *L
 00066 R 000070 A *L
 00067 R 700000 A *L
 00070 R 000077 A *L
             SIZE=000071      NO ERROR LINES
```

```
PAGE    1       FILALO SRC      PROGRAM TO ALLOCATE SPACE FOR RANDOM ACCESS FILE

                                        .TITLE PROGRAM TO ALLOCATE SPACE FOR RANDOM ACCE
                                /
                                /000000000000000000000000000000000000000000000000000000000000
                                /
                                /WHEN STARTED, THE PROGRAM EXPECTS TO FIND THE NAME
                                /AND DESIRED LENGTH OF THE FILE IN THE DATA SWITCH
                                /REGISTER.  THE PROGRAM HALTS 3 TIMES
                                /       1)PLACE FIRST 3 CHARS (6 BIT) OF NAME IN DSR
                                /       2)PLACE SECOND 3 CHARS (6 BIT) OF NAME IN DSR
                                /       3)PLACE 3 CHARS (6 BIT) OF EXTENSION IN DSR
                                /       4)PLACE DESIRED LENGTH OF FILE IN BLOCKS IN DSR
                                /
                                /WHEN OPERATOR HITS THIRD CONTINUE, PROGRAM CREATES
                                /THE DUMMY FILE 1 BLOCK LONGER THAN SPECIFIED IN THE DSR
                                /NOTE THAT THE EXTRA BLOCK IS USED TO HOLD THE "RIB",
                                /RETRIEVAL INFORMATION BLOCK, WHICH IS NECESSARY FOR
                                /RANDOM ACCESS OPERATIONS. DON'T TRY TO FOOL THE PROGRAM
                                /BY PLACING A VALUE 1 LESS THAN THE DESIRED LENGTH IN
                                /THE DATA SWITCH REGISTER.
                                /
                                /000000000000000000000000000000000000000000000000000000000000
                                /
                                        .IODEV 4
                                /
        000001  A               OUT=1
                                /
00000 R                         START   .INIT 4,OUT,0           /INITIALIZE OUTPUT DEVIC
                                /
 00004 R 740040  A                      HLT             /HALT TO
00005 R 750004  A                       LAS             /PICK UP FIRST 3 CHARACTERS
00006 R 040042  R                       DAC NAME        /OF NAME AND SAVE
                                /
00007 R 740040  A                       HLT             /HALT TO
00010 R 750004  A                       LAS             /PICK UP SECOND 3 CHARACTERS
00011 R 040043  R                       DAC NAME+1      /OF NAME AND SAVE
                                /
00012 R 740040  A                       HLT             /HALT TO
00013 R 750004  A                       LAS             /PICK UP 3 CHARACTER EXTENSION
00014 R 040044  R                       DAC EXT         /AND SAVE
                                /
00015 R 740040  A                       HLT             /HALT TO
00016 R 750004  A                       LAS             /PICK UP DESIRED FILE LENGTH
00017 R 741300  A                       SPA!SNA         /LEGAL ONLY IF STRICTLY POSITIVE
00020 R 600040  R                       JMP EXIT        /OTHERWISE EXIT
00021 R 740031  A                       TCA             /MAKE IT A NEGATIVE COUNT
00022 R 040443  R                       DAC COUNT#
                                /
                                        .EJECT
```

```
                        /
                        /
                                .ENTER 4,NAME           /GET DIRECTORY ENTRY FOR
00026 R                 OUTPUT  .WRITE 4,3,BUFF,254     /WRITE IT OUT--ONE BLOCK
                                .WAIT 4
00034 R 440443 R                ISZ COUNT               /ARE WE DONE?
00035 R 600026 R                JMP OUTPUT              /NO--BACK FOR MORE
                                .CLOSE 4                /YES--CLOSE FILE
00040 R                 EXIT    .EXIT                   /BYE....
                        /
                        /
                        /
00042 R 000000 A        NAME    0
00043 R 000000 A                0
00044 R 000000 A        EXT     0
                        /
00045 R 177000 A        BUFF    177000
00046 R 000000 A                0
                                .REPT 374
00047 R 000005 A                5
                        /
        000000 R                .END START
            SIZE=00444      NO ERROR LINES
```

```
                                .TITLE PROGRAM TO LIST BLOCKS USED BY A FILE
                        /
                        /oooooooooooooooooooooooooooooooooooooooooooooooooooooooo
                        /
                        /PROGRAM ASKS FOR THE NAME OF A FILE WHICH IS STORED
                        /ON THE DEVICE ASSOCIATED WITH DAT 3 (DISK OR DECTAPE).
                        /IF THE FILE EXISTS, THE BLOCKS USED BY THE FILE ARE
                        /LISTED ON THE DEVICE ASSOCIATED WITH DAT 12 (TT OR LP).
                        /IF THE FILE DOES NOT EXIST ON THE DEVICE, A MESSAGE
                        /TO THAT EFFECT IS PRINTED ON THE TT.
                        /
                        /oooooooooooooooooooooooooooooooooooooooooooooooooooooooo
                        /
                                .GLOBL IA26B,OTOASC
                        /
                                .IODEV 3,12
            000000 A    IN=0
            000001 A    OUT=1
                        /
    00000 R             START   .INIT 3,IN,0    /SETUP 3 FOR INPUT (FILE)
                                .INIT =2,OUT,0

                                .INIT 12,OUT,0  / SETUP 12 FOR OUTPUT (TT OR LP)
                                .INIT =3,IN,OVER
                        /
                        /**************************************************
                        /  GET FILE NAME AND EXTENSION
                        /**************************************************
                        /
                                .WRITE =2,2,MESS1,34    /TYPE
                                .WRITE =2,2,MESS1A,34
                                .WRITE =2,2,MESS2,34    /EXAMPLE
    00034 R             LIST    .WRITE =2,2,MESS3,34    /NAME=
                                .WAIT =2
                        /       /READ IN NAME
                                .READ =3,3,BUFFN,8
                                .WAIT =3
                                .WRITE =2,2,MESS4,34    /EXT=
                                .WAIT =2
                                .READ =3,3,BUFFE,5      /READ IN EXTENSION
                        /
    00062 R 201107 R            LAC (BUFFN+2            /WHILE READING EXTENSION
    00063 R 121105 E            JMS* IA26B              /TRANSLATE NAME
    00064 R 000002 A            2               /NUMBER OF SIX= BIT WORDS

    00065 R 000647 R            NAME
                        /
                                .WAIT =3                /WAIT FOR EXTENSION READ
                        /
    00070 R 201110 R            LAC (BUFFE+2
    00071 R 121105 E            JMS* IA26B
    00072 R 000001 A            1               /NUMBER OF SIX= BIT WORDS
    00073 R 000651 R            EXT
                        /
```

```
                        /************************************************************
                        /  FIND OUT IF FILE EXISTS
                        /************************************************************
                        /
00074 R 100174 R                JMS WRNAME
                        /
                                .FSTAT 3,NAME
00100 R 741200 A                SNA                     /DID IT GIVE START BLOCK #?
00101 R 600227 R                JMP NEXIST              /NO--SO DOESN'T EXIST
00102 R 040123 R        BLKNUM  DAC TRAN+2
                        /
                        /************************************************************
                        /
                        /  GET BLOCK #S AND PRINT OUT
                        /
                        /************************************************************
                        /
00103 R 121106 F                JMS* OTOASC
00104 R 000240 R                BUFASC
                                .WRITE 12,2,CRLF,4
                                .WAIT 12
                                .WRITE 12,3,BLKHD,8
                                .WAIT 12
00121 R                 TRAN    .TRAN 3,IN,0,BUFF,256
                                .WAIT 3
00130 R 200646 R                LAC BUFF+377
00131 R 541111 R                SAD (777777
00132 R 600134 R                JMP ENDFIL
00133 R 600102 R                JMP BLKNUM
                        /
                        /************************************************************
                        /
                        /  GET HERE WHEN HAVE PRINTED ALL BLOCK #S
                        /
                        /************************************************************
                        /
                        /
00134 R                 ENDFIL  .WRITE 12,2,CRLF,4
                                .WRITE 12,2,MESS6,34
00144 R                 ANOTHER .WRITE -2,2,CRLF,4
                                .WRITE -2,2,MESS7,34    /WANT ANOTHER FILE?
                                .READ -2,3,BUFMR,3
                                .WAIT -2
00162 R 200654 R                LAC BUFMR+2
00163 R 541112 R                SAD (131                /131=YES
00164 R 600167 R                JMP OVER                /START OVER
                                .EXIT
                        /
00167 R                 OVER    .WRITE -2,2,CRLF,4
00173 R 600034 R                JMP LIST
                        /
                        /
```

```
                        /
                        /
00174 R 000000 A    WRNAME  0                                                   1
                            .WRITE 12,2,CRLF,4
                            .WAIT 12
00203 R 200671 R            LAC BUFFE
00204 R 341113 R            TAD (1000
00205 R 040667 R            DAC BUFEHD
00206 R 201114 R            LAC (20
00207 R 040671 R            DAC BUFFE
00210 R 201115 R            LAC (11
00211 R 040672 R            DAC BUFFE+1
                            .WRITE 12,3,BUFFN,10
                            .WAIT 12
                            .WRITE 12,3,BUFEHD,8
                            .WAIT 12
00226 R 520174 R            JMP* WRNAME
                        /
00227 R             NEXIST  .WRITE 12,2,MESS8,34
                            .WAIT 12
00235 R 600144 R            JMP ANOTHER
                        /
                        /
                        /*****************************************************
                        /
                        /  BUFFERS
                        /
                        /*****************************************************
                        /
00236 R 004003 A    BLKHD   004003
00237 R 000000 A            0
00240 R         A    BUFASC  .BLOCK 6
00246 R 000015 A            15
                        /
00247 R         A    BUFF    .BLOCK 400      /SET UP FOR TRAN OF BLOCK
                        /
00647 R 000000 A    NAME    0
00650 R 000000 A            0
00651 R 000000 A    EXT     0
                        /
00652 R         A    BUFMR   .BLOCK 4
                        /
00656 R 000000 A    BUFFN   0
00657 R 000000 A            0
00660 R         A            .BLOCK 6
00666 R 000175 A            175
00667 R 000003 A    BUFEHD  000003
00670 R 000000 A            0
00671 R 000000 A    BUFFE   0
00672 R 000000 A            0
00673 R         A            .BLOCK 3
00676 R 000015 A            15
```

```
                                .TITLE MESSAGE PAGE
                        /
                        /
00677 R 017000 A        MESS1   MESS1A-MESS1/2*1000
00700 R 000000 A                0
00701 R 522632 A                .ASCII /TYPE IN A FULL 6 CHARACTER NAME /
00702 R 042500 A
00703 R 446344 A
00704 R 040500 A
00705 R 432531 A
00706 R 446100 A
00707 R 331010 A
00710 R 344202 A
00711 R 512030 A
00712 R 352212 A
00713 R 511011 A
00714 R 640632 A
00715 R 425000 A
00716 R 000000 A
00717 R 406350 A                .ASCII /AND THREE CHARACTER EXTENSION./<15>
00720 R 420250 A
00721 R 442450 A
00722 R 542500 A
00723 R 416210 A
00724 R 151202 A
00725 R 416510 A
00726 R 551100 A
00727 R 426612 A
00730 R 442634 A
00731 R 516231 A
00732 R 747134 A
00733 R 064000 A
00734 R 000000 A
                        /
00735 R 006000 A        MESS1A  MESS2-MESS1A/2*1000
00736 R 000000 A                0
00737 R 432231 A                .ASCII /FILL IN NULLS WITH "0"/<15>
00740 R 446100 A
00741 R 446344 A
00742 R 047252 A
00743 R 462312 A
00744 R 320256 A
00745 R 446511 A
00746 R 020104 A
00747 R 401041 A
00750 R 500000 A
                        /
00751 R 010000 A        MESS2   MESS3-MESS2/2*1000
00752 R 000000 A                0
00753 R 432372 A                .ASCII /FOR EXAMPLE:/
00754 R 220212 A
00755 R 542031 A
```

```
00756 R 550230 A
00757 R 425640 A
00760 R 000000 A
00761 R 472031 A                .ASCII /NAME=SORT@@/<11>/EXT=SRC/<15>
00762 R 542572 A
00763 R 516372 A
00764 R 252200 A
00765 R 400230 A
00766 R 554250 A
00767 R 366472 A
00770 R 241432 A
                        /
00771 R 003000 A        MESS3   MESS4=MESS3/2*1000
00772 R 000000 A                0
00773 R 372350 A                .ASCII />NAME=/<175>
00774 R 146612 A
00775 R 367720 A
00776 R 000000 A
                        /
00777 R 003000 A        MESS4   MESS5=MESS4/2*1000
01000 R 000000 A                0
01001 R 046133 A                .ASCII <11>/EXT=/<175>
01002 R 052172 A
01003 R 764000 A
01004 R 000000 A
                        /
01005 R 002000 A        MESS5   MESS6=MESS5/2*1000
01006 R 000000 A                0
01007 R 201004 A                .ASCII /    /<15>
01010 R 006400 A
                        /
01011 R 006000 A        MESS6   MESS7=MESS6/2*1000
01012 R 000000 A                0
01013 R 522210 A                .ASCII /THAT'S ALL THERE IS!/<15>
01014 R 152116 A
01015 R 515010 A
01016 R 146230 A
01017 R 202511 A
01020 R 042644 A
01021 R 425011 A
01022 R 151502 A
01023 R 064000 A
01024 R 000000 A
                        /
01025 R 017000 A        MESS7   MESS8=MESS7/2*1000
01026 R 000000 A                0
01027 R 422364 A                .ASCII /DO YOU WANT ANOTHER FILE?/
01030 R 054636 A
01031 R 525012 A
01032 R 740634 A
01033 R 521010 A
01034 R 147236 A
```

```
01035 R 522210 A
01036 R 551100 A
01037 R 432231 A
01040 R 442576 A
01041 R 406352 A                .ASCII /ANSWER Y FOR YES./
01042 R 353612 A
01043 R 511013 A
01044 R 120214 A
01045 R 476444 A
01046 R 054612 A
01047 R 515340 A
01050 R 000000 A
01051 R 265330 A                .ASCII /--ANYTHING ELSE FOR NO/<15>
01052 R 147262 A
01053 R 522211 A
01054 R 147216 A
01055 R 202131 A
01056 R 451612 A
01057 R 202151 A
01060 R 751100 A
01061 R 472361 A
01062 R 500000 A
                        /
01063 R 007000 A        MESS8   CRLF-MESS8/2*1000
01064 R 000000 A                0
01065 R 202111 A                .ASCII / DOES NOT EXIST ON DEVICE/<15>
01066 R 742646 A
01067 R 202351 A
01070 R 752100 A
01071 R 426611 A
01072 R 151650 A
01073 R 202371 A
01074 R 620210 A
01075 R 426551 A
01076 R 141612 A
01077 R 064000 A
01100 R 000000 A
                        /
01101 R 002000 A        CRLF    002000
01102 R 000000 A                0
01103 R 064000 A                .ASCII <15>
01104 R 000000 A
      000000 R                  .END START
01105 R 001105 E *E
01106 R 001106 E *E
01107 R 000660 R *L
01110 R 000673 R *L
01111 R 777777 A *L
01112 R 000131 A *L
01113 R 001000 A *L
01114 R 000020 A *L
01115 R 000011 A *L
            SIZE=01116      NO ERROR LINES
```

```
$L
     EXAMPLE OF FILBLK USE----


     TO THE TELETYPE

$A TT 12

$GLOAD

LOADER V3A000
>←FILBLK,OTOASC,IA26B


TYPE IN A FULL 6 CHARACTER NAME AND THREE CHARACTER EXTENSION.
FILL IN NULLS WITH "@"
FOR EXAMPLE: NAME=SORT@@ EXT=SRC
>NAME=MAIN1@     EXT=SRC
MAIN1@  SRC
000520
000522
THAT'S ALL THERE IS!

DO YOU WANT ANOTHER FILE?  ANSWER Y FOR YES--ANYTHING ELSE FOR NO
N

DOS-15 V3A000
$L
     TO THE LINEPRINTER

$A LP 12

$GLOAD

LOADER V3A000
>←FILBLK,OTOASC,IA26B


TYPE IN A FULL 6 CHARACTER NAME AND THREE CHARACTER EXTENSION.
FILL IN NULLS WITH "@"
FOR EXAMPLE: NAME=SORT@@ EXT=SRC
>NAME=MAIN1@     EXT=SRC
DO YOU WANT ANOTHER FILE?  ANSWER Y FOR YES--ANYTHING ELSE FOR NO
N

DOS-15 V3A000
$
```

```
MAIN1@  SRC

000520

000522

THAT'S ALL THERE IS!
```

digital

DIGITAL EQUIPMENT CORPORATION, Maynard, Massachusetts, Telephone: (617) 897-5111 • ARIZONA, Phoenix • CALIFORNIA Sunnyvale, Santa Ana, Los Angeles, San Diego and San Francisco (Mountain View) • COLORADO, Engelwood • CONNECTICUT, Meriden • DISTRICT OF COLUMBIA, Washington (Riverdale, Md.) • FLORIDA, Orlando • GEORGIA, Atlanta • ILLINOIS, Northbrook • INDIANA, Indianapolis • LOUISIANA, Metairie • MARYLAND, Riverdale • MASSACHUSETTS, Cambridge and Waltham • MICHIGAN, Ann Arbor and Detroit (Southfield) • MINNESOTA, Minneapolis • MISSOURI, Kansas City and Maryland Heights • NEW JERSEY, Fairfield, Metuchen and Princeton • NEW MEXICO, Albuquerque • NEW YORK, Huntington Station, Manhattan, New York, Syracuse and Rochester • NORTH CAROLINA, Durham/Chapel Hill • OHIO, Cleveland, Dayton and Euclid • OKLAHOMA, Tulsa • OREGON, Portland • PENNSYLVANIA, Bluebell, Paoli and Pittsburgh • TENNESSEE, Knoxville • TEXAS, Dallas and Houston • UTAH, Salt Lake City • WASHINGTON, Bellevue • WISCONSIN, Milwaukee • ARGENTINA, Buenos Aires • AUSTRALIA, Adelaide, Brisbane, Crows Nest, Melbourne, Norwood, Perth and Sydney • AUSTRIA, Vienna • BELGIUM, Brussels • BRAZIL, Rio de Janeiro, Sao Paulo and Porto Alegre • CANADA, Alberta, Vancouver, British Columbia; Hamilton, Mississauga and Ottawa, Ontario; and Quebec • CHILE, Santiago • DENMARK, Copenhagen and Hellerup • FINLAND, Helsinki • FRANCE, Grenoble and Rungis • GERMANY, Cologne, Hannover, Frankfurt, Munich and Stuttgart • INDIA, Bombay • ISRAEL, Tel Aviv • ITALY, Milano • JAPAN, Osaka and Tokyo • MEXICO, Mexico City • NETHERLANDS, The Hague • NEW ZEALAND, Auckland • NORWAY, Oslo • PHILIPPINES, Manila • PUERTO RICO, Miramar and Santurce • REPUBLIC OF CHINA, Taiwan • SCOTLAND, West Lothian • SPAIN, Barcelona and Madrid • SWEDEN, Solna and Stockholm • SWITZERLAND, Geneva and Zurich • UNITED KINGDOM, Birmingham, Bristol, Edinburgh, London, Manchester, Reading and Warwickshire • VENEZUELA, Caracas

PRINTED IN USA 0302 00174 3534/E 14 1.2