ATTENTION

This is a USER program. Other than requiring that it conform to submittal and review standards, no quality control has been imposed upon this program by DECUS.

The DECUS Program Library is a clearing house only; it does not generate or test programs. No warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related material, and no responsibility is assumed by these parties in connection therewith.
THE PDP8 COOKBOOK

BY

FLOOR ANTHONI

Medical Biological Laboratory TNO, Rijswijk 2100, The Netherlands

SUBJECT: Subroutine standardisation

INTRODUCTION:

By the beginning of 1972, nearly 14,000 computers of the PDP8 family have been produced and field-installed. All of them have to be programmed to fulfill the tasks, dedicated to them.

The small size of most PDP8 configurations has forced most programmers to program the machine in assembly language. Many programs have since then found their way to the DECUS PROGRAM LIBRARY. The typical application-oriented programs, however, were rarely submitted to the LIBRARY, because nobody would ever be likely to apply for them. The experience, accumulated elsewhere, was therefore not available to others.

In programming the PDP8 computer, I have experienced the usefulness of program modularity at the assembly level. The basic modules are, in effect, subroutines that perform a certain function, and that have been programmed in such a way, that they can be used as "recipies" in a cookbook. When these "recipies" are being sent to a central editor, and published regularly, they will accumulate experience into a common module library, THE PDP8 COOKBOOK, available to others.

This paper proposes a norm for modules, submitted to the library.
THE SUBROUTINE AND ITS USE

The subroutine jump certainly is the most powerful instruction of any computer. It enables the programmer to avoid duplication of code, and to build hierarchical structures of software intelligence, increasing the semantic power of each free location in core.

Subroutines in hierarchical structures will in general do the task expected from them, with a minimum of "directions" given from "above". They can, themselves, set lower level subroutines to work for them, also with a minimum of directions. These directions are in general, information, that has to be transferred down to the subroutine. The subroutine can, on the other hand, send information back. Subroutines that can be directed to do many tasks, will, in general need more "instructions" from above. The programmer has to consider this aspect with great care. The following remarks on the ways, information can be sent to and from subroutines may assist him in this respect.

When only one parameter needs to be transferred, use the ACCUMULATOR. The LINK can be used as additional YES or NO information, although it is, in general not frequently used. The use of other registers, like the MULTIPLIER-QUOTIENT register, must be strongly dissuaded, because the module will then not be able to run on many machine configurations.

More information can be transferred as arguments, following the JMS instruction. This is especially useful for parameters that can be set at assembly time, or that need not to change very often. Use the AC for frequently changing information. A common information area in page 0 can also be used. This is especially useful when those parameters need to be accessed by many modules.

(For example program- and buffer-limits, pointers, etc.). The main problem of the sharing of the same storage locations, by
different subroutines, is that extreme care must be exerted when calling subroutines within those subroutines.

All subroutine modules in the COOKBOOK will be provided with the storage locations they need, in order to avoid conflicting use of these locations.

Another way to circumvent such problems is to employ the techniques of reentrant and recursive programming, in which push-down list structures are being used. This aspect will not be within the scope of this paper. The concept of creating an information "vector", that is a limited area in core with all the information, in order that only the pointer to this "vector" needs to be transferred, is, however, very useful for transfers, both in and out of the subroutine.

**HOW TO PREVENT UNWANTED INTERFERENCE**

When using subroutines, that have been used before, the most likely assembly error is that illegal redefinitions will result from the duplicate use of symbols. Therefore care must be taken to label a location. The following conventions are proposed: use very few tags. Put all storage locations and other items in front of the subroutine entry, that needs to have more than 3 characters. All other tags need to share, at least the first 3 characters of the subroutine entry.

Those programmers that want to "pack" subroutines into the least possible space, will find it easy to modify the subroutines in this respect.

**DOCUMENTATION**

Simple subroutines need less documentation than the more sophisticated ones. Comments should be inserted, wherever additional
information is needed. Avoid trivial comments like CLA/CLEAR AC, but express the general concept and thoughts, as if it were a flow chart. The documentation must be adequate for the reader to easily understand how the subroutine works. For more sophisticated routines a flow chart is a must. Each subroutine must have a compact functional description of not more than one line (52 characters). Then follows a general description of the subroutine and an example of its use. All program lines and comment lines should not exceed 52 positions, as assembler output and cross-reference numbers must have room to be inserted.

The source tape should be submitted with the tabulations, not being converted to spaces.

The listing should preferably be made with a teletype printer (teletype type of character), printed with tabs converted to spaces. Use a clean typing head and a new black ribbon, as the listing will be offset-copied. Drawings and flow-charts should be drawn with black ink, or taped with special stickers.

For the use of symbols, the reader is referred to Appendix I.

PROGRAM SUBMISSION

Submit your program subroutine to

The Editorial Board of
The PDP8 COOK BOOK
c/o Floor Anthoni,
Medical Biological Laboratory TNO,
139, Lange Kleiweg,
RIJSWIJK (ZH), 2100,
The Netherlands.

NOTE! It is of vital importance that errors are reported back to the authors or the editorial board. Only by doing so one can achieve the highest reliability of the published subroutines.
COOKBOOK VOLUME 1 CATALOG LISTED BY NUMBER

001 Type the characters following the JMS instruction

002 Teletype type routine with overlap

003 Type a character chain

004 Binary to decimal conversion, single prec. no sign

005 Binary to octal conversion, no sign, fixed field

006 High speed reader subroutine

007 Tabulator routine

008 Move a block through core

009 Binary punch with field setting, checksum, leader

010 PAL message printer

011 General branch routine

012 Check AC if octal

013 Logical operators, AND, OR, NAND, NOR, EXCL.OR, etc.

014 PS8/OS8 option decoder

015 Print 2 digits in decimal

016 Print the PS8/OS8 date

017 Print the AC as a FOCAL linenumber

018 Print 4 decimal digits, using routine 015, no sign

019 Read a decimal number in core

020 Decimal print, leading blanks, no sign

021 Print double length decimal, no sign

022 Octal print, no sign, leading spaces

023 Double word octal print using 022

024 Translate TELEX code to ASCII

025 Translate TELEX code to ASCII

026 Translate ASCII code to TELEX

027 Interrupt ASCII output handler with rotating buffer

028 Device interrupt handler (part of 027)
029 Read or write DECtape in both directions

030 Subroutine to pack a fixed buffer in core (300 chars) into a fixed output buffer (200 chars) in TSS8 packed format

031 Pack characters into a buffer in TSS8 format, one by one

032 As 031, but with a fixed allocated buffer

033 Unpack TSS8 format packed buffer into an output buffer

034 Unpack TSS8 format packed buffer, one character at a time

035 Subroutine to read a 6 character name in core

036 Search a file name in DN blocks (Disk monitor)

037 Search for an unused block in SAM block, and reserve it for the current file

038 Search internal file number in SAM blocks (Disk Monitor)

039 Subroutine to read or write on disk (TSS8).
LIST OF CONTRIBUTORS

Floor Anthoni
Medisch Biologisch Laboratorium TNO,
Lange Kleiweg 139,
Rijswijk (ZH),
The Netherlands

Thierri den Dunnen
Dr. Neher Laboratorium,
St. Paulusstraat 4,
Leidschendam,
The Netherlands

Hans Mees,
Prins Maurits Laboratoria, C.D.,
Lange Kleiweg 137,
Rijswijk (ZH),
The Netherlands

Paul Lohman,
Medisch Biologisch Laboratorium TNO,
Lange Kleiweg 139,
Rijswijk (ZH),
The Netherlands

Contributions
1, 2, 3, 4, 5, 6, 7, 9, 9, 10, 11, 12, 13, 14, 15, 17, 18
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39
8
16
FLOW-CHART conventions
The flow-charts make use of relatively few symbols:

Entry, or exit of a program or sub-program, also used for the inter-connection of flow-charts on different pages.

A rectangle describes one or more program steps.

Decision, branching

Subroutine call. A subroutine may have more than one return (branching).

Comments appear behind a slash (/).

Used to name program ties in agreement with the listings.
/001 TYPE THE CHARACTERS FOLLOWING THE JMS INSTR.
/TERMINATOR IS A ZERO.
/
/     JMS TYPTEX    /TYPE "ABC"
/     301          /"A"
/     302          /"B"
/     303          /"C"
/     0            /TERMINATOR
/     RETURN       /AC=0

TYPTEX, 0
TAL I TYPTEX    /GET CHAR.
ISZ TYPTEX      /ZERO?
SNA
JMP I TYPTEX*   /YES, JMP TO NEXT LOC.
JMS TYPE        /NO, TYPE.
JMP TYPTEX+1
/ 02 TELETYP TYPE ROUTINE.
/ INITIALIZES WHEN ENTERED FOR FIRST TIME.
/ NOT RESTARTABLE !
/
/  TAD CHARACTER
/  JMS TYPE
/  RETURN /AC=0

NOP

TYPE= 0
JMP +3 /OVERLAYS BY "NOP"
TSP
JMP -1
TLS
CLA
TAD TYPE-1
LCA TYPE+1
JMP I TYPE
*/ TYPE A CHARACTER CHAIN
*/ TYPE THE CHARACTERS IN THE LIST, POINTED TO
*/ BY THE FIRST ARGUMENT. LIST TERMINATOR = 0
/
/  JMS TYPTEX  /TYPE "ABC"
/  LIST
/  RETURN  /AC=0
/
/
/LIST, 301
/
302
/
303
/
0

0  /USED AS POINTER
TYPTEX, 0  /TYPE TEXT STRING
TAD I TYPTEX  /GET ARG
DCA TYPTEX-1  /SAVE TO USE AS POINTER
ISZ TYPTEX  /FOR CORRECT RETURN
TAD I TYPTEX-1  /GET CHAR
SNA  /ZERO?
JMP I TYPTEX  /YES, RETURN
JMS TYPE  /NO
ISZ TYPTEX-1
JMP TYPTEX+4  /LOOK FOR NEXT
/* Routine to convert a binary word to decimal and type it. */
/* Valid for numbers 0-4995 - no sign. */
/* If used for 3 digits, delete 0030; -4 = 3 digit count. */

TAD WORD
JMS PRINTD
RETURN

AC=0

6030  /* 1000 conversion constants */
7634  /* 100 */
7766  /* 10 */
7777  /* 1 */
TAD *  /* Used for conv. constants */
0  /* Digit bcd to be typed */
0  /* Counter */
260  /* To make a char. */
0  /* Save area */
-4  /* Digits to be typed (-4,-3,-2) */

PRINTD, 0  /* Enter with word in AC */

DCA PRINTD-2
TAD PRINTD-1  /* Set up count */
DCA PRINTD-4
DCA PRINTD-5  /* Clear bcd */
TAD PRINTD-6  /* Fetch curr. conv. const. */
TAD PRINTD-4  /* By adding count to TAD */
DCA +4
HLT

TAD PRINTD-2  /* Value = constant */
SRL  /* Overflow? */
JMP +4  /* No, type it */
ISZ PRINTD-5  /* Yes, next try */
DCA PRINTD-2
JMP PRINTD+5
CLA
TAD PRINTD-5  /* Bcd */
TAD PRINTD-3  /* +260 */
JMS TYPE
ISZ PRINTD-4
JMP PRINTD+4  /* Next digit */
JMP I PRINTD

12
;005 binary to octal conversion and print
;routine prints the AC in octal, no sign.
/
TAD WORD
JMS PRINT8
RETURN /AC=0
/
260 /mask
7
0 /digit counter
-4 /# of digits
0 /temporary
PRINT8, 0
RAL CLL
LCA PRINT8-1
TAD PRINT8-2
LCA PRINT8-3 /set up count
TAD PRINT8-1
RAL
RTL
LCA PRINT8-1
TAD PRINT8-1
AND PRINT8-4 /mask
TAD PRINT8-5 /make ascii
JMS TYPE
ISZ PRINT8-3 /# done?
JMP PRINT8+5 /not yet
JMP I PRINT8
HIGH SPEED READER SUBROUTINE
ENTER WITH AC=0; ROUTINE INITIALIZES HSR EACH ENTRY
AFTER AN END-OF-TAPE CONDITION (TIME-OUT)
WHEN STOPPED IN TAPE MOTION IT SIGNALS TIME-OUT THE
NEXT ENTRY; THE ROUTINE HAS A BUILT-IN TIMING LOOP
THAT TIMES OUT IF THE READER IS NOT SWITCHED ON,
OR IF THE READER LOSES ITS FLAG BY RUNNING OUT OF TAPE.
/
JMS HSR READ
/
OUT OF TAPE RETURN
/
NORMAL RETURN, CHAR. IN AC
/

0           /USED AS TIME-OUT COUNT
HSREAD, 0    /ENTRY
DCA HSREAD-1 /SET UP COUNT
HSRFLG, 1    /FLAG SIGNALS TO INIT READER
TAD HSRFLG   /THOSE INSTR. CONTRIBUTE TO LOOP
SZA CLI
JMP' +3      /INIT READER
RSF
JMP +5       /NO. COUNT TIME-OUT
DCA HSRFLG   /CLEAR FLAG
6016
ISZ HSREAD   /RETURN, CHAR IN AC
JMP I HSREAD
ISZ HSREAD-1
JMP HSRFLG
ISZ HSRFLG   /SET FLAG TO SIGNAL TIMEOUT
JMP I HSREAD /EOT RETURN

14
/ 887 TABULATOR ROUTINE.
/ THE USER HAS TO TAKE CARE OF:
/ INCREMENTING TARCNT WITH EACH INCOMING CHARACTER, CLEARING
/ IT WHEN CARRIAGE RETURN. TAP-INTERVAL IS VARIABLE.
/ A JMS TO TAB WILL MOVE THE TYPING HEAD TO THE NEXT
/ TABULATOR POSITION.
/
/   CLA
/   JMS TAB
/   RETURN /AC=0
/
TARCNT, 0
-10 /TAB INTERVAL
240 /SPACE
TAB, 0 /ENTER WITH AC =0
TAD TARCNT /SUBTRACT N TIMES TO GIVE REMAINDER
TAD TAB-2
SMA
JMP -2 /USE AS NEGATIVE COUNTER
LCA TARCNT
TAD TAB-1
JMS TYPE
ISZ TARCNT /READY?
JMP -3
JMP I TAB /YES
SUBROUTINE TO MOVE A BLOCK THROUGH CORE

CALLING SEQUENCE
JMS MOVE
BEGINADDRESS
ENDADDRESS
DESTINATION OR FIRST WORD
RETURN /AC=6

IF BEGINADDRESS AND ENADDRESS ARE
THE SAME ADDRESS, OR BEGINADDRESS
IS GREATER THAN ENADDRESS,
NO MOVE IS PERFORMED

IF BEGINADDRESS AND DESTINATION ARE
THE SAME ADDRESS, A COMPLETE MOVE
IS PERFORMED: YOU SHOULD BE LESS STUPID!

56 (OCTAL) CORE LOCATIONS ARE USED
0
0
0

MOVE, 0
TAD MOVE /GET BEGINADDRESS
LCA MOVE-1
ISZ MOVE
TAD MOVE /GET ENADDRESS
LCA MOVE-2
ISZ MOVE
TAD MOVE-2
CMA
TAD MOVE-1 /CALCULATE WORDCOUNT
SNA /IS IT POSITIVE OR ZERO?
JMP MOVRET /YES, NO MOVE NEEDED
LCA MOVE-4 /SAVE WORDCOUNT
TAD MOVE-1
CIA CLL
TAD MOVE /CALCULATE MOVECOUNT
LCA MOVE-3 /AND SAVE
SZL /LINK IS ON IF MOVE TO HIGHER CORE
JMP +3 /SKIP NEXT INSTRUCTIONS
IAC CM.
TAD MOVE-4 /FIRST IN ADDRESS IS BEGINADDRESS
TAD MOVE-2
LCA MOVE-2 /SAVE INPUT POINTER
SZL /SKIP IF MOVE TO LOWER CORE
CMA CMA HAL /TO HIGHER CORE, INC = -1
IAC
DCA MOVE-1 /SAVE INCREMENT
TAD MOVE-2 /SET UP OUTPUT POINTER
TAD MOVE-3
DCA MOVE-3 /AND SAVE
MVLOOP, TAD MOVE-2 /GET A WORD
DCA I MOVE-3 /AND STORE IT IN DESTINATION BLOCK
TAD MOVE-2
TAD MOVE-1 /INCREMENT INPUT POINTER
TAD MOVE-3
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAD MOVE-1</td>
<td>Increment output pointer</td>
</tr>
<tr>
<td>DCA MOVE-3</td>
<td>Increment word count</td>
</tr>
<tr>
<td>ISZ MOVE-4</td>
<td>Again if not zero</td>
</tr>
<tr>
<td>JMP MULoop</td>
<td></td>
</tr>
<tr>
<td>MOVRET, ISZ MOVE</td>
<td>Set up return address</td>
</tr>
<tr>
<td>JMP I MOVE</td>
<td>Return</td>
</tr>
</tbody>
</table>
JOE BINARY PUNCH WITH FIELD SETTING

THREE SUBROUTINES TO PUNCH AN AREA OF CORE IN BINARY
LOADER FORMAT: FIELD SETTINGS AND ORIGIN SETTINGS
ARE BEING PUNCHED AT EACH ENTRY; CHECKSUM IS PUNCHED
WHEN PUNCH IS CALLED.

THE ROUTINE CAN TAKE DATA FROM A DIFFERENT FIELD.
IT CAN OPERATE IN ALL FIELDS; SEVERAL USES APPLY:
1) NORMAL USE: THE DATA IS LOCATED IN THE SAME FIELD OR
   STRANGE FIELD: ENTER WITH FIELD IN AC; LINK=0
2) THE CODE IS IN SAME FIELD AS BINPUN, ONLY A DIFFERENT
   FIELD SETTING NEEDS TO BE PUNCHED. ENTER BINPUN
   WITH FIELD IN AC AND LINK=1.
3) THE CODE HAS BEEN MOVED IN CORE. THE FIRST LOCATION
   IS NOT NECESSARILY THE ORIGIN. NOW ENTER BINPUN WITH
   AC=FIELD FOR SETTING: LINK=1; SET ORIGIN UNEQUAL TO
   FIRST LOCATION IF THIS IS TRUE.
EXAMPLE OF NORMAL USE:

JMS LEADER /PUNCH LEADER, CLEAR CHECKSUM
CLL
TAD 0010 /FIELD 1
JMS BINPUN
ORIGIN /IN NORMAL USE=FIRST LOC.
FIRST LOC.
LAST LOC.
JMS PUNCHK /PUNCH CHKSM AND TRAILER

BINEND, 0 /LAST LOC. TO PUNCH
6201
100
300 /FOR FIELD SETTING
0 /TEMP. STORAGE
BINPUN, 0
DCA BINPUN-1
SZL
JMP BIN3
TAD BINPUN-1
TAD BINPUN-4 /MAKE CDF
BIN3,
DCA BIN2
TAD BINPUN-1 /MAKE FIELD SETT. AND PUNCH
TAD BINPUN-2 /NOTE!! *FIELD SETT. NOT IN CHECKSUM!!
JMS TYPE
TAD I BINPUN
ISZ BINPUN
DCA BINPUN-1
TAD BINPUN-1 /PUNCH ORIGIN
JMS BINLH /LEFT HALF
TAD BINPUN-3 /+100 FOR ORIGIN
JMS BINCHK
TAD BINPUN-1 /RIGHT HALF AND PUNCH
AND BINLH-1
JMS BINCHK
TAD I BINPUN /SET UP POINTER
DCA BINPUN-1
ISZ BINPUN
TAD I BINPUN /GET END
DCA BINEND

BIN2,
CDF 0 /OVERLAI BY CDF STRANGE FIELD
TAD I BINPUN-1 /GET DATA
JMS BINLH /PUNCH LEFT HALF
JMS BINCHK
TAD I BINPUN-1 /PUNCH RIGHT HALF
AND BINLH-1
JMS PINCHK
TAD BINEND /END REACHED?
CIA
CLL
TAD BINPUN-1
ISZ BINPUN-1
SNL CLA
JMP BIN2+1 /NO, TAKE NEXT DATA
RIF /YES, RESTORE LF
TAD BINPUN-4
DCA •+1
CLF 0 /OVERLaid
ISZ BINPUN
JMP I BINPUN
SPA CLA

/GET LEFT HALF OF THE AC
77
BINLH, 0
RTR
RTR
RTR
AND BINLH-1
JMP I BINLH

/UPDATE THE CHECKSUM, AND PUNCH FRAME
0 /CHECKSUM
PINCHK, 0 /ENTER WITH 6 BIT FRAME IN AC
DCA BINLH /USE BINLH TEMPORARILY
TAD BINLH
TAD BINCHK-1 /UPD. CHKSM
DCA BINCHK-1
TAD BINLH
JMS TYPE
JMP I BINCHK

/PUNCH 100(8) LEADER OR TRAILER HOLES; CLEAR CHECKSUM
200
LEADER. 0
TAD BINPUN-3 /USE CHKSM AS NEG COUNT
CIA
DCA BINCHK-1
TAD LEADER-1
JMS TYPE
ISZ BINCHK-1 /READY?
JMP •-3 /NO
JMP I LEADER /YES

/PUNCH CHECKSUM; PUNCH TRAILER; CLEAR CHECKSUM

PUNCHK, 0
TAD BINCHK-1 /GET CHECKSUM
JMS BINLH
JMS TYPE
TAD BINCHK-1
AND BINLH-1
JMS TYPE
JMS LEADER /CLEARS CHECKSUM
JMP I PUNCHK

19
PAL MESSAGE PINTER
PRINTS A MESSAGE CODED WITH THE PAL PSEUDO-OP
'TEXT': PAL3 AND PAL5 COMPATIBLE
/
/  JMS PRMSG
/  MEC
/  RETURN /AC=0
/
/PRMSG, TEXT 'ABC82' /CODED AS 0102:0370:6200

77       /MASK
-40      /TO TEST
240      /TO MAKE ASCII
100      /TO MAKE ASCII
0        /PACKSWITCH 0=LEFT; 7777=R
0        /POINTER
PRMSG, 0
CMA
TAD I PRMSG
DCA PRMSG-1
ISZ PRMSG    /FOR RETURN
PRM1, CMA
DCA PRMSG-2   /PACKSW=RIGHT
ISZ PRMSG-1   /NEXT WORD
TAD I PRMSG-1 /FETCH AND ROTATE 6
RTR
RTR
PRM2, AND PRMSG-6 /MASK 6 BITS
SNA
JMP I PRMSG    /ZERO ENDS THE LIST
TAD PRMSG-5    /<40?
SPA
TAD PRMSG-3   /YES ASCII 301-337
TAD PRMSG-4   /NO, ASCII 240-277
JMS PRINT
ISZ PRMSG-2  /LEFT OR RIGHT?
JMP PRM1   /LEFT
TAD I PRMSG-1 /RIGHT
JMP PRM2
GENERAL BRANCH ROUTINE
BRANCH ROUTINE BRANCHES ACCORDING TO THE CONTENTS
OF THE AC, COMPARED TO EACH ITEM OF A LIST.
EXIT FROM BRANCH IS ALWAYS WITH AC=0

TAD AC
JMS BRANCH
LIST-1
RETURN IF NOT IN LIST (AC=0)

/List, 212
/IF "CHAR"=212, PROGRAM JUMPS TO "LF"
/215
/CH; ETC; ETC.........
/0 # IS LIST TERMINATOR!!!!!!
0 /AC
0 /BRANCH POINTER
BRANCH, 0 /ENTER WITH ARGUMENT IN "CHAR"
LCA BRANCH-2
TAD I BRANCH
ISZ BRANCH
LCA BRANCH-1 /INIT POINTER
BRANCH, ISZ BRANCH-1
TAD I BRANCH-1 /FETCH ELEMENT FROM LIST
SNA /END OF LIST?
JMP I BRANCH /YES
CIA
TAD BRANCH-2
ISZ BRANCH-1
SZA CLA
JMP BRANC /NO, TRY NEXT
TAD I BRANCH-1 /YES, GO TO IT
LCA BRANCH
JMP I BRANCH
/312 CHECK IF OCTAL
/ROUTINE CHECKS WHETHER THE AC IS AN OCTAL DIGIT.
/
/ TAD CHARACTER
/ JMS OCTCHK
/ NOT OCTAL RETURN /AC=0
/ OCTAL RETURN /AC=0

10
-270

OCTCHK, 0
TAD OCTCHK-1
SMA
JMP OCT2
TAD OCTCHK-2
SPA CLA
JMP I OCTCHK
ISZ OCTCHK

OCT2, CLA
JMP I OCTCHK
LOGICAL OPERATORS ON TWO NUMBERS
THE RESULT OF LOGICAL OPERATIONS IS IN THE AC.

/AND (MASKING)       A 1010
/                     B 1100
/                     = 1000
/                     TAD A
/                     AND B

/INCLUSIVE OR      A 1010
/SETS BITS B IN A  B 1100
/                     = 1110
/                     TAD A
/                     CMA
/                     AND B
/                     TAD A

/CLEAR BITS B IN A  A 1010
/                     B 1100
/                     = 0010
/                     TAD B
/                     CMA
/                     AND A

/NOR                A 1010
/                     B 1100
/                     = 0001
/                     TAD A
/                     CMA
/                     DCA TEM
/                     TAD B
/                     CMA
/                     AND TEM

/NAND               A 1010
/                     B 1100
/                     = 0111
/                     TAD A
/                     AND B
/                     CMA

/EXCLUSIVE OR       A 1010
/                     B 1100
/                     = 0110
/                     TAD A
/                     AND B
/                     CMA
/                     DCA TEM
/                     TAD A
/                     AND TEM
/                     TAD B
/                     AND TEM
/014 PSB-05/8 OPTION DECODER
/CHECKS THE OPTION, SPECIFIED IN THE AC AND CAUSES
/A RETURN, DEPENDING ON WHETHER THE OPTION HAS BEEN
/SET
/OPTIONS IN OS8 RESIDE IN FIELD 1 LOC 7643-7645:
/
/7643 A B C D E F G H I J K L ASCII 301-314
/7644 M N O P Q R S T U V W X ASCII 315-330
/7645 Y Z 0 1 2 3 4 5 6 7 8 9 ASCII 331, 332, 260-271
/
/ TAD (16) /CHECK OPTION 16 (N)
/ JMS OPTION
/ OPTION NOT SET RETURN/AC=0
/ OPTION SET RETURN /AC=0
/
/OPTION, 0
/DCA OPTION-2
/TAD OPTION-3 /RESTORE COUNTER
/DCA OPTION-1
/TAD OPTION-2 /SUBTRACT 12 TO FIND WORD
/TAD OPTION-4
/ISZ OPTION-1
/SMA SZA
/JMP -3
/TAD OPTM1 /FOR L AND X
/DCA OPTION-2 /SAVE REMAINDER MODULO 12
/CLL CML /AND ROTATE ONE BIT INTO POSITION
/RAL /ROTATE FURTHER
/ISZ OPTION-2
/JMP -2
/CDF 10 /AND WITH OPT WORD FIELD 1
/AND I OPTION-1
/CDF 0
/SZA CLA
/ISZ OPTION /IN CASE IT HAD BEEN SET
/JMP I OPTION

24
015 PRINT TWO DIGITS IN DECIMAL
THE VALUE OF THE AC IS PRINTED IN TWO DIGITS
CORRECTLY IF < 99 (DECIMAL).
/
/
TAD VALUE
JMS PRNT2
/
RETURN /AC=0
/
260 /TO MAKE ASCII
-12 /10 DECIMAL
0 /TEMP STORAGE
0 /COUNTER
PRNT2, 0
DCA PRNT2-2
TAD PRNT2-2 /TRY SUBTRACT 10 UNTIL OVFLO
TAD PRNT2-3
SPA
JMP +3
ISZ PRNT2-1 /SUBTRACT FURTHER
JMP PRNT2+1
CLA
TAD PRNT2-1 /PRINT HIGH ORDER DIGIT
TAD PRNT2-4
JMS PRINT
TAD PRNT2-2
TAD PRNT2-4
JMS PRINT
DCA PRNT2-1 /RESET COUNTER
JMP 1 PRNT2
/016 PRINT THE PS8-OS8 DATE
/ THE DATE IS PRINTED AS: 07/17/72
/ THE ROUTINE MAKES USE OF PRINT2, TO TYPE TWO
/ DECIMALS. REQUIRES ROUTINES PRINT2 AND PRINT.
/ DATE IN OS8 IS STORED IN LOC 7666, FIELD 1:
/
/7666 MMMMDDDDDDYYYY /M=MONTH, D=DAY, Y=YEAR
/
/ JMS DATE
/ RETURN /AC=0

DATM, 7 /MASKS
 17
 37
 257 /SLASH
 106 /70 YEARS
 0 /STORAGE
 7666 /DATE LOC. IN OS8

date, 0
CDF 10 /PICK THE DATE
TAD I DATE-1
CDF 0
DCA DATE-2
TAD DATE-2
CLL RTL /SHIFT MONTH OUT
RTL
RAL
AND DATM+1 /AND (17
JMS PRINT2
TAD DATE-4 /PRINT SLASH
JMS PRINT
TAD DATE-2
RTR /SHIFT MONTH OUT AND PRINT
RAR
AND DATM+2
JMS PRINT2
TAD DATE-4 /SLASH
JMS PRINT
TAD DATE-2 /NOW THE YEAR
AND DATM
TAD DATE-3 /+70
JMS PRINT2
JMP I DATE
/017 PRINT THE AC AS A FOCAL LINENUMBER
/THE VALUE OF THE AC IS PRINTED AS IN FOCAL: 11.35
/XX.YY STORED AS FOLLOWS: XXXXXXXXXX IN 1 WORD.
/IF YYYYYYY>99 STRANGE DIGITS OCCUR AS IN FOCAL.
/REQUIRES ROUTINES PRNT2 AND PRINT.
/
/ TAD VALUE
/ JMS PRNTF
/ RETURN /AC=0

PRNTFM, 37 /Masks
177
256 /Period.
0 /Storage

PRNTF, 0
DCA PRNTF-1 /Isolate and print high order
TAD PRNTF-1
CLL RTL
RTL
AND PRNTFM /AND (37
JMS PRNT2
TAD PRNTF-2
JMS PRINT
TAD PRNTF-1 /Now low order
AND PRNTFM+1
JMS PRNT2
JMP I PRNTF
/018 PRINT 4 DECIMAL DIGITS USING ROUTINE PRNT2
/ THE CONTENT OF THE AC IS DIVIDED BY 100(10)
/ GIVING TWO LOW ORDER DIGITS AND 2 HIGH ORDER.
/ THESE ARE PRINTED BY PRNT2.
/
/ TAD VALUE
/ JMS PRNT4
/ RETURN    /AC=0

7634     /=-100(10)
0          /STORAGE AND LOW ORDER
0         /HIGH ORDER COUNTER
PRNT4.  0
DCA PRNT4-2
CLL
TAD PRNT4-2    /TRY TO SUBTRACT 100 UNTIL OVERFLOW
TAD PRNT4-3
SNL
JMP +3
ISZ PRNT4-1
JMP PRNT4+1
CLA
TAD PRNT4-1    /PRINT HIGH ORDER DIGITS
JMS PRNT2
TAD PRNT4-2    /PRINT LOW ORDER DIGITS
JMS PRNT2
DCA PRNT4-1     /RESET COUNTER
JMP I PRNT4
SUBROUTINE READS A DECIMAL NUMBER FROM KEYBOARD.
RUBOUT REMOVES NUMBER COMPLETELY.

/ CALL JMS DECINP
/ RETURN WITH NUMBER BINARY IN AC
/

DECINP, 0

CLA
DCA DECNUM /CLEAR REGISTER
JMS READ /READ CHAR FROM KEYBOARD
TAD CHAR
JMS PRINT /PRINT THAT CHAR
TAD CHAR /GET CHARACTER
TAD M377 /IS IT RUBOUT?
SNA CLA
JMP DECINP+1 /YES READ ALL OVER AGAIN
TAD CHAR /NO
TAD M260
SPA /CHAR>=260?
JMP DECOUT /NO, CHARACTER IS DELIMITER
TAD M12 /YES
SMA CLA /CHAR<272?
JMP DECOUT /NO, CHARACTER IS DELIMITER
TAD DECNUM /YES, CHARACTER IS FIGURE
CLL RAL
DCA DECTMP /NUMBER*2
TAD DECTMP
RTL /NUMBER*8
TAD DECTMP /NUMBER*8+NUMBER*2=NUMBER*10
TAD CHAR /ADD LAST FIGURE
TAD M260
DCA DECNUM /DECIMAL NUMBER
JMP DECINP+3
/

DECOUT, CLA
TAD DECNUM
JMP I DECINP /EXIT
/

/ VARIABLES
/ DECNUM, 0
DECTMP, 0
/

/ GENERAL CONSTANTS
M12, -12
M260, -260
M377, -377
/ 020 DECIMAL PRINT ROUTINE,
/ PRINTS AC DECIMAL IN 4 DIGITS
/ MAX NUMBER = 4095 DECIMAL
/ SKIPS LEADING ZERO'S
/
LPHT, 0
DCA DPRREG /SAVE AC IN PRINTREG.
TAD DPRINS /GET INSTRUCTION
DCA DPRPTP /PUT INSTR. ON POINTER
TAD M4
DCA DPRFAC /4 DIGITS
DCA DPRFL /CLEAR PRINT 0 FLAG
DCA DPRFIG /CLEAR DIGIT
DPRSUB, CLL
TAD DPRREG /PICK UP SAVED AC
DPRPTP, TAD DprtE. /SUBTRACT POWER OF TEN
SNL /REMAINDER POSITIVE?
JMP +4 /NO, PRINT DIGIT
DCA DPRREG /YES, SAVE REMAINDER
ISZ DPRFIG /DIGIT:=DIGIT+1
JMP DPRSUB /REPEAT SUBTRACTION
CLA CLL
TAD DPRFIG /GET DIGIT
SNA /A ZERO?
JMP DPRZRO /YES
DPRIN, TAD C260 /NO, CONVERT TO ASCII
JMS PRINT
ISZ DPRFL /MAKE NOT EQUAL 0
DPRIN1, ISZ DPRPTP /MODIFY INSTR ON DPRPTP
ISZ DPRFAC /PRINTED 4 DIGITS?
JMP DPRSUB-1 /NO, PRINT NEXT DIGIT
JMP I DPHT /YES, RETURN
/
DPRZRO, TAD DPRFL
SZA CLA
JMP DPRIN
JMP DPRIN1
/
DPRREG, 0
DPRFL, 0
DPRINS, TAD DPTEN
DPRFAC, 0
DPRFIG, 0
DPTEN, 60 30 /-1000
7 634 /-100
7 766 /-10
7 777 /-1
/
/GENERAL CONSTANTS
M4, -4
C260, 260
SUBROUTINE TO PRINT DOUBLE LENGTH DECIMAL

CALL: JMS DDECPR

MOST SIGNIFICANT PART
LEAST SIGNIFICANT PART
NUMBER OF DIGITS TO BE PRINTED ( <=8 )
RETURN

DDECPR, 0

TAD I DDECPR /FETCH MOST SIGNIFICANT PART
DCA DDx /SAVE
DCA DDPD /CLK NUMBER OF PRINTED DIGITS
ISZ DDECPR
TAD I DDECPR /FETCH LEAST SIGNIFICANT PART
DCA DDx+1 /SAVE
ISZ DDECPR
TAD I DDECPR /FETCH FORMAT
DCA DUNDIG
ISZ DDECPR /CORRECT RETURN
TAD DDATPL /ADDRESS 10-POWER LOW
DCA DDPTPL /POINTER 10-POWER LOW
TAD DDATPH /ADDRESS 10-POWER HIGH
DCA DDPTPH /POINTER 10-POWER HIGH
TAD M10
DCA DNFAC /FACTORISE 8 DIGITS
DCA DDIGIT /CLEAR DIGIT

DDSUB, CLL

TAD DDx+1 /SIGNIFICANT PART OF NUMBER
TAD I DDP1PL /LOW PART FACTOR
DCA DDx+1 /STORE
RAL /OVERFLOW IN AC
TAD DDx /M SIGNIFICANT PART OF NUMBER
TAD I DDPTPH /HIGH FACTOR
SNL /RESULT NEGATIVE?
JMP *+4 /YES
DCA DDx /STORE RESULT OF SUBTRACTION
ISZ DDIGIT /NO, STEP UP DIGIT
JMP DDSUB /SUBTRACT 2-LENGTH AGAIN
CLA /CLEAR BEFORE CORRECTION
TAD I DDPTPL /10-POWER LOW
CIA /MINUS
TAD DDx+1 /CORRECT LAST SUBTRACTION
DCA DDx+1 /STORE
TAD DDIGIT /GET DIGIT
SZA /=0?
JMP DDPDIN /NO
TAD DDPD
SZA CLA /ALREADY PRINTED?
JMP DDPDIN /YES
IAC
TAD DNFAC
SMA CLA /ALL DI#0 = 0?
JMP DDPDIN /YES
TAD DNFAC /NEGATIVE VALUE
TAD DUNDIG /POSITIVE VALUE
SPA CLA /SPACE?
JMP DDPDIN
TAD C240 /YES
JMP DDPFR
DEPTIN, ISZ DDPTD
TAD C260
/CONVERT TO ASCII
DDPR, JMS PRINT
/PRINT DIGIT
DEPTIN, ISZ DDPTPL
ISZ DDPTPH
/STEP UP POINTER LOW
ISZ DNNFAC
/READY FACTORIZE?
JMP DDSUB-1
/NO, NEXT DIGIT
TAD DDPTD
CIA
TAD DNDIG
SPA SNA CLA
JMP +3
TAD DNDIG
DCA DDPTD
CLL
JMP I DDECPR
/EXIT, END PUNCHOUT ROUTINE
/
/CONSTANTS PUNCH OUT ROUTINE
DDATPL, DDATL
DDATPH, DDATPH
DDPTPL, 0
DDPTPH, 0
DLX, 0
DNNFAC, 0
DDPTD, 4600
6700
4540
4360
6030
7634
7766
7777
DDPTPH, 3166
7413
7747
7775
7777
7777
7777
7777
DDPTD, 0
DNDIG, 0
/
/GENERAL CONSTANTS
M10, -10
C240, 240
C260, 260
OCTAL PRINT ROUTINE
NONSIGNIFICANT ZEROS BECOME SPACES

CLA
DCA OCTFIG /CLEAR FLAG FIGURE PRINTED
DCA OCTSPC /CLEAR SPACE-COUNTER
TAD OCTNUM
CALL JS: OCTPRT /WITH NUMBER IN AC
RETURN AC=0 /IF NUMBER=0,
OCTSPC=4, = # OF SPACES TO PRINT
IF NUMBER IS ZERO, OCTSPC=4 IS # SPACES TO PRINT

OCTPRT, 0
RAL /ROTATE IN LINK
DCA OCTTMP /TEMP STORAGE
TAD M4 /4 OCTADES
DCA OCTCNT
OCTPR0, TAD OCTTMP
RAL
RTL
DCA OCTTMP /STORE RESULT
TAD OCTTMP /GET IT BACK
AND C7 /MASK OCTADE
SNA CLA /ZERO?
JMP OCTZER /YES
TAD OCTSPC /NO SPACES TO PRINT?
SNA
JMP OCTNUM /NO GO PRINT FIGURE
CLA
DCA OCTSPC
TAD C240
JMS PRINT /PRINT THE SPACES
ISZ OCTSPC
JMP -3
OCTNUM, CLA IAC /SET FLAG FIG. PRINTED
DCA OCTFIG
TAD OCTTMP
AND C7
OCTZER, TAD OCTPR0 /MAKE THE FIGURE
JMS PRINT
OCTPR1, ISZ OCTCNT /READY?
JMP OCTZER /NO
JMP I OCTPRT /YES, EXIT
OCTFIG, SZA CLA /FIGURES PRINTED?
JMP OCTZER /YES, PRINT THIS ZERO TOO
ISZ OCTSPC /NO COUNT AS SPACE
JMP OCTPR1
OCTTMP, OCTCNT, OCTSPC, OCTFIG, M4, -4
C7, 7
C240, 240
C260, 260
/023 DOUBLE WORD OCTAL PRINT ROUTINE
/USES ROUTINE OCTPR
/CALLING: JMS DOCTRPR
/HIGH ORDER NUMBER
/LOW ORDER NUMBER
/RETURN AC=0
/
DOCTRPR, 0

CLA
DCA OCTFIG /CLEAR FLAG FIGURE PRINTED
DCA OCTSPC /CLEAR SPACE-COUNTER
TAD I DOCTRPR /HIGH ORDER PART
ISZ DOCTRPR
JMS OCTPRRT /PRINT OCTAL
TAD I DOCTRPR /LOW ORDER PART
ISZ DOCTRPR
JMS OCTPRRT /PRINT OCTAL
TAD OCTSPC
CIA
SNA /SPACES TO PRINT?
JMP I DOCTRPR /NO, EXIT
IAC /YES, NUMBER IS ZERO
DCA OCTSPC /PRINT SPACES
TAD C240
JMS PRINT
ISZ OCTSPC
JMP * -3
TAD C260 /AND A "0"
JMS PRINT
JMP I DOCTRPR /EXIT
SUBROUTINE TRANSLATES TELX TO ASCII

CALL :JMS TLXAS WITH TELX CHARACTER IN AC
    RETURN CHARACTER IS SHIFT
    RETURN WITH ASCII CHARACTER IN AC

WHO IS TRANSLATED AS $
? IS TRANSLATED AS *
BELLS IS TRANSLATED AS ;

TLXAS, 0
    AND C37 /MASK 5 BITS
    DCA TLXTMP /TEMP STORAGE
    TAD TLXTMP SNA
    JMP TLXOUT /BLANK
    TAD M2 SNA
    JMP TLXCR /CARRIAGE RETURN
    TAD M6 SNA
    JMP TLXNL /NEW LINE
    TAD M23 SNA
    JMP TLXSW1 /FIGURESHEET
    TAD M4 SNA CLA
    JMP TLXSW0 /LETTERSHEET
    TAD TLXTMP /GET CHARACTER AGAIN
    TAD TLXLA /ADD LIST ADDRESS
    DCA TLXTMP /TEMP STORAGE
    TAD TLXSW SNA /WHICH SIDE?
    SZA CLA
    JMP TLXRGT /RIGHT SIDE
    TAD I TLXTMP RTR
    TAD TLXSW1 /GET ASCII 6 BIT
    RTR
    RTR
    TLXMS, AND C77 /MASK 6 BIT
    TAD M40 SPA
    TAD C100 /CHAR<40:300<=CHAR<=337
    TAD C240 /CHAR>40:240<=CHAR<=277
    TLXOUT, ISZ TLXAS /NORMAL RETURN
    JMP I TLXAS

TLXRGT, TAD I TLXTMP
    JMP TLXMS

TLXSW1, IAC
    TLXSW0, DCA TLXSW /REMEMBER WHICH SHIFT
    JMP I TLXAS /RETURN SHIFT

TLXCH, TAD C215
    JMP TLXOUT
TLXNL, TAD C212
    JMP TLXOUT

TLXLA, TLXLST
TLXLST, 0
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C37,</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>1771</td>
</tr>
<tr>
<td></td>
<td>4040</td>
</tr>
<tr>
<td></td>
<td>1036</td>
</tr>
<tr>
<td></td>
<td>1654</td>
</tr>
<tr>
<td></td>
<td>1556</td>
</tr>
<tr>
<td>M40,</td>
<td>-40</td>
</tr>
<tr>
<td></td>
<td>1451</td>
</tr>
<tr>
<td></td>
<td>2264</td>
</tr>
<tr>
<td></td>
<td>0735</td>
</tr>
<tr>
<td></td>
<td>1170</td>
</tr>
<tr>
<td></td>
<td>2060</td>
</tr>
<tr>
<td></td>
<td>0372</td>
</tr>
<tr>
<td></td>
<td>2675</td>
</tr>
<tr>
<td></td>
<td>0563</td>
</tr>
<tr>
<td></td>
<td>3253</td>
</tr>
<tr>
<td></td>
<td>0477</td>
</tr>
<tr>
<td></td>
<td>0252</td>
</tr>
<tr>
<td></td>
<td>2347</td>
</tr>
<tr>
<td></td>
<td>3166</td>
</tr>
<tr>
<td></td>
<td>0633</td>
</tr>
<tr>
<td></td>
<td>3057</td>
</tr>
<tr>
<td></td>
<td>0155</td>
</tr>
<tr>
<td></td>
<td>2762</td>
</tr>
<tr>
<td></td>
<td>1273</td>
</tr>
<tr>
<td>C100,</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>2567</td>
</tr>
<tr>
<td></td>
<td>2161</td>
</tr>
<tr>
<td></td>
<td>1350</td>
</tr>
</tbody>
</table>

/ VARIABLES
TLXTMP, 0
TLXSW, 0

/ GENERAL CONSTANTS
M2,   -2
M4,   -4
M6,   -6
M23,  -23
C71,  77
C212, 212
C215, 215
C240, 240
/CALL: JMS TLXASI 
/ RETURN IF SHIFT CHARACTER 
/ RETURN 
/ 
TLXASI, 0 
AND TLX37 
DCA TLXTMP /STORE 
TAD TLXTMP 
TAD TLX37 
SNA /LETTERSHIFT? 
JMP TLXLSH /YES, SET SHIFT 
TAD C4 
SNA CLA /FIGURESHIFT? 
JMP TLXFSH /YES, CLEAR SHIFT 
TAD TLXTMP 
TAD TLXSH 
TAD TLXLST 
DCA TLXTMP 
TAD I TLXTMP 
ISZ TLXASI1 
JMP I TLXASI1 
/

TLXLSH, TAD TLX40 
TLXFSH, DCA TLXSH 
JMP I TLXASI1 
/
/

TLXLST, +1 
0000 /BLANK 
"5 
0215 /CR 
"9 
0240 /SPACE 
0000 ">
"," 
0212 /LF 
") 
"4 
0000 
"8 
"0 
": 
"=" 
"3 
"+ 
0205 /WHU 
"? 
"=" 
"6 
0000 
"/ 
"- 
"2 
0237 /FELL 

TLXSH, 0 
"7 
"1
TLX43, 40
TLX37, 37
TLX37, -37

⍎/VARIABLES

⍎/GENERAL CONSTANTS

C4, 4
/026 ROUTINE TO TRANSLATE ASCII TO TELEX
/CALL :JMS ASTLX
/ RETURN
/
/BEFORE FIRST CALL INITIALIZE: ASTSET:=4 AND
/PRINT A LETTERSHIFT
/
/NOT EXISTING CHARACTERS ARE PRINTED AS BLANK
/ALTMODE IS TRANSLATED AS FIGURES.SHIFT
/FUBOUT IS TRANSLATED AS LETTERSHIFT
/
ASTLX, 0
DCA ASTTMP /TEMP STORAGE.
TAD ASTTMP
AND C77 /MAKE 6 BIT
SNA.
.
JMP ASTOUT+2 /BLANK=BLANK
TAD ASTLA /LISTADDRESS
DCA ASTHLP /LISTADDRESS + 6-BIT CHAR
TAD ASTTMP
TAD M300
SMA CLA
JMP ASTBIG /CHAR>=300:RIGHT HALF OF LIST
TAD I ASTHLP /CHAR<300:LEFT HALF OF LIST
RTR
RTR
RTR
SKP

ASTBIG, TAD I ASTHLP
DCA ASTTMP /TEMP STORAGE.
TAD ASTTMP
AND C77
SNA.
JMP ASTOUT+2 /NOT EXISTING IN TELEX: BLANK
AND C40 /GET SHIFT BIT
SZA CLA /WHICH SHIFT
JMP ASTSHIFT /MUST BE FIGURES
TAD ASTSFT /MUST BE LETTERS
SZA CLA /IS IT LETTERS?
JMP ASTOUT /YES, PRINT CHAR
CLA CLL IAC RTL /4: NO, MAKE AND PRINT

ASTPSH, DCA ASTSFT
TAD ASTSFT
TAD C33 /MAKE SHIFT
JMS PRINT /PRINT

ASTOUT, TAD ASTTMP
AND C37 /MASK 5 BITS
JMS PRINT /PRINT
JMP I ASTLX /EXIT
/

ASTSHIFT, TAD ASTSFT /MUST BE FIGURES
SNA CLA /IS IT FIGURES?
JMP ASTOUT /YES, PRINT CHAR
JMP ASTPSH /NO, MAKE AND PRINT
/
ASTLA, ASTLST
ASTLST, 0000 /*
0030 /*
0023 /*

39
5773 /*, ALTMOD=HIGHSHIFT
3000 */
6337 /*, SUBOUT=LETTERSHIFT

/* VARIABLES */
ASTSFT, 0
ASTTMP, 0
ASTHLP, 0

/* GENERAL CONSTANTS */
C33, 33
C37, 37
C40, 40
C77, 77
M300, -300
/027 INTERRUPT OUTPUT HANDLER
/WITH HEAD-TAIL COUPLED BUFFER
/
/INITIALIZE ONCE BUFFER BUFFER BUFFER BUFFER Buffer:=0
/
/
/CHARACTER HANDLER
/
/CALL :JMS BUFINP WITH CHAR IN AC
/RETURN AC:=0
/

BUFINP, 0
DCA BUFTMP /TEMP STORAGE
TAD BUFIBO /INPTR BEHIND OUTFTR?
SNA CLA
JMP BUFPUT /No, STORE CHARACTER
TAD BUFIPT /YES
CIA
TAD BUFOPT /INPTR = OUTFTR ?
SNA CLA
JMP BUFINP+2 /YES, WAIT FOR PLACE TO STORE

BUFPUT, TAD BUFTMP /NO, GET CHAR
DCA I BUFIPT
ISZ BUFIPT
TAD BUFBUS /PRINTER BUSY?
SNA CLA
6046 /No, INIT WITH AC:=0
IAC /YES, SET PRINTER BUSY
DCA BUFBUS
TAD BUFIPT
TAD BUFEND /END OF BUFFER?
SZA CLA
JMP I BUFINP /NO, EXIT
TAD BUFADR /YES, POINTER TO HEAD
DCA BUFIPT
IAC /AND SET INPTR BEHIND OUTFTR
LCA BUIFO
JMP I BUFINP /EXIT
/028 DEVICE INTERRUPT HANDLER
/
/CALL :JMP BUFOUT /DEVICE INTERRUPT DETECTED!
/ ROUTINE RETURNS TO INTERRUPT RESTORE "EXIT"
/
BUFOUT, CLA
  6042 /CLEAR DEVICE FLAG
  TAD BUFIPT
  CIA
  TAD BUFOPT /INPTR =OUTPTR?
  SZA CLA
  JMP BUFGET /NO, GET CHAR AND PRINT
  TAD BUFIBO /YES, INPTR BEHIND OUTPTR?
  SZA CLA
  JMP BUFGET /YES, GET AND PRINT
  DCA BUFBUS /NO, PRINTER READY
  JMP EXIT
/
BUFGET, TAD IBUFOPT /GET CHAR
  ISZ BUFOPT
  6044 /PRINT CHAR
  CLA
  TAD BUFOPT
  TAD BUFBND /END OF BUFFER?
  SZA CLA
  JMP EXIT /NO, END OF ROUTINE
  TAD BUFADR /YES, POINTER TO HEAD
  DCA BUFOPT
  DCA BUFBND /RESET INPTR BEHIND OUTPTR
  JMP EXIT /END OF HANDLING
/
/GENERAL INTERRUPT RETURN ROUTINE
/
EXIT, CLA CLL
/ TAD LINK
/ RAL /RESTORE LINK
/ TAD ACCU /RESTORE ACCU
/ ION /INTERRUPT ON
/
/VARIABLES
BUFTMP, 0
BUFIBO, 0
BUFPIPT, 0
BUFOPT, 0
BUFBUS, 0
BUFBND, -BUFEND
BUFADR, BUFFER
BUFFER, 0
/
*BUFFER+200
BUFEND, 0
IDEFINING BITS: BIT 0, 1, 2  UNIT NUMBER
/  3  0=FORWARD; 1=REVERSE
/  4, 5  0 (NOT USED)
/  6, 7, 8  MEMORY FIELD
/  9  0 (NOT USED)
/  10  0=DIRECT RETURN; 1=INDIRECT
/  11  0=READ; 1=WRITE

DTCA=  6762
DTXA=  6764
DTRL=  6774
DTRA=  6761
DTSF=  6771
DTRR=  6772

DCTAPE, 0
CLA
TAD I DCTAPE  /IDEFINING BITS
DCA DCTCOD  /SAVE
1SZ DCTAPE
TAD DCTCOD
AND C7400  /UNIT # & DIRECTION BIT
TAD C10  /SEARCH MODE
DTCA DTXA  /I/O
DTRL  /CLEAR FIELD REGISTER
TAD DCTWC  /WORD COUNT ADDRESS
DCA I DCTCA  /WORD COUNT=BLKNR ADDRESS
TAD C200  /GO BIT
DCTCNT, JMS DCTTEN  /TURN DETC AND WAIT FOR FLAG
TAD I DCTWC  /READ NUMBER
CIA  /NEG.
TAD I DCTAPE  /NUMBER TO FIND
SNA
JMP DCTMAY  /FOUND, CHECK DIRECTION
DCTSET, CLL RAL  /SAVE SIGN DIFFERENCE
CLA
DTHA
AND C400  /DIRECTION BIT
SNA CLA
CML  /IS FORWARD
SNL  /IS REVERSE
TAD C400  /CHANGE DIRECTION
JMP DCTCNT  /DIRECTION OK, NEXT NUMBER

DCTMAY, TAD DCTCOD  /UNIT # & DIRECTION
AND C400  /MASK DIRECTION
SNA CLA
JMP DCTRFW  /MUST BE FORWARD
DTRA  /MUST BE REVERSE
AND C400
SZA CLA
JMP DCTRDR /IS REVERSE, GO READ OR WRITE
JMP DCTCNT /IS FORWARD, CONT SEARCHING
DCTRFW, DTRA /MUST BE FORWARD
AND C400
SNA CLA
JMP DCTRDR /IS FORWARD, GO READ OR WRITE
JMP DCTCNT /IS REVERSE, CONT SEARCHING
/ DCTRDR, ISZ DCTAPE
TAD I DCTAPE /# WORDS
DCA I DCTWC /SET WORD COUNT
ISZ DCTAPE
TAD I DCTAPE /CORE ADDRESS-1
DCA I DCTCA /SET CURRENT ADDRESS
TAD DCTCOD
DTLB /LOAD FIELD BITS
TAD DCTCOD
RAR
SZL CLA /READ OR WRITE?
TAD C200 /WRITE
TAD C130 /WRITE
DTXA
DTSF DTRB
JMP -1
ISZ DCTAPE /ADVANCE TO ERROR RETURN
SMA CLA /SKIP IF ERROR
ISZ DCTAPE /NORMAL RETURN
TAD DCTCOD /DIRECT OR INDIRECT?
RTR
SML CLA
JMP +3 /DIRECT
TAD I DCTAPE /INDIRECT, PREPARE
DCA DCTAPE
LTRB
AND C200 /GO BIT
TAD C2 /PRESERVE ERROR FLAG
DTXA /STOP TAPE
JMP I DCTAPE /READY, EXIT /
DCTTERN, 0
DTXA
DTSF DTRB
JMP -1
SPA
JMP DCTERR
CLA
JMP I DCTTERN /
DCTERR, RAL
RAL
CLA CMN
SML
TAD C400
JMP DCTCNT-1

45
/VARIABLES
DCTCOD, 0
DCTWC, 7754
DCTCA, 7755

/GENERAL CONSTANTS
C2, 2
C10, 10
C20, 20
C130, 130
C200, 200
C400, 400
C7400, 7400
SUBROUTINE TO PACK CHARACTERS (TSS8)
THREE CHARACTERS IN TWO WORDS (TSS8 FORMAT)
PACKED: 111111112222
222233333333

CALL :JMS PACK
CALL ADDRESS INPUTBUFFER
CALL ADDRESS OUTPUTBUFFER
RETURN

ROUTINE USES AUTO INDEX 10 AND 11

FORMAT INPUTBUFFER=1 CHAR/WORD
LENGTH OUTPUTBUFFER=200
LENGTH INPUTBUFFER=300

PACK, 0
TAD PCKBFL
STL PCKBFL
DCA PCKCINT
CLA CMA
TAD I PACK
DCA 10
ISZ PACK
CMA
TAD I PACK
DCA 11
ISZ PACK
PCKLOP, TAD I 10
CLL RTL
RTL
DCA PCKTMP
TAD I 10
RTR
RTR
DCA PCKTP1
TAD PCKTP1
AND C17
TAD PCKTMP
DCA I 11
TAD PCKTP1
RAR
AND C7400
TAD I 10
DCA I 11
ISZ PCKCINT
JMP PCKLOP
JMP 1 PACK

/VARIABLES
PCKCINT,0
PCKTMP,0
PCKTP1,0
PCKBFL,-200

/GENERAL CONSTANTS
C17, 17
C7400, 7400
031 SUBROUTINE PACKS CHARACTERS ONE BY ONE (TSB)
/THREE CHARACTERS IN TWO WORDS (TSSB FORMAT)
/PACKED:111111112222
/
/CALL :JMS PCKSGL WITH CHAR IN AC
/ADDRESS OF OUTPUTBUFFER
/RETURN BUFFER FULL
/RETURN NORMAL AC=0
/
/INITIALIZE CE PCKSWT:=0
/
/
PCKSGL, 0
ISZ PCKSWT /INITIALIZE?
JMS PCKINI /YES
LCA I PCKRP /NO PUT CHAR IN TEMP BUF
ISZ PCKRP /INCREMENT POINTER
ISZ PCKECT /3 CHAR'S IN TEMP BUF?
JMP PCKNRM /NO, NORMAL EXIT
JMS PCKRES /YES,RESET POINTER TEMP BUF
TAD I PCKRP /GET FIRST CHAR
ISZ PCKRP
CLL RTL
RTL
DCA I PCKPTR /TEMP STORAGE
TAD I PCKRP /GET SECOND CHAR
ISZ PCKRP
RTR
RTR
LCA PCKSWT
TAD PCKSWT /TEMP STORAGE
AND C17 /MOST SIGN. 4 BITS
TAD I PCKPTR
DCA I PCKPTR /FIRST WORD
ISZ PCKPTR
TAD PCKSWT
FAR
AND C7400 /LEAST SIGNIFICANT 4 BITS
TAD I PCKRP /GET THIRD CHAR
DCA I PCKPTR /SECOND WORD
ISZ PCKPTR
JMS PCKRES /RESET POINTER TEMP BUF
ISZ PCKCNT /BUFFER FULL?
JMP PCKNRM /NO
DCA PCKSWT /YES SET SWITCH
JMP PCKEND
/
/
PCKNRM, CMA
DCA PCKSWT /SET SWITCH
ISZ PCKSGL
PCKEND, ISZ PCKSGL
JMP I PCKSGL
/
PCKINI, 0
DCA PCKSWT /TEMP STORAGE
48
JMS PCKRES
TAD I PCKSSL
DCA PCKPTR
TAD PCKBFL
STL BAR
DCA PCKCNT
TAD PCKSWT
JMP I PCKINI

/!
PCKRES, 0
TAD M3
DCA PCKRCT
TAD PCKERAA
DCA PCKRP
JMP I PCKRES

/!
/VARIABLES /
PCKSWT, 0
PCKPTR, 0
PCKRP, 0
PCKRCT, 0
PCKCNT, 0
PCKERAA, PCKRB
PCKBFL, -400
PCKRB, 0

/!
/GENERAL CONSTANTS 
M3, -3
C17, 17
C7400, 7400
SUBROUTINE TO PACK CHARACTERS ONE BY ONE (TSS8)
THREE CHARACTERS IN TWO WORDS (TSS8 FORMAT)
Packed: 111111112222
      222233333333

CALL: JMS DSOUT WITH CHAR IN AC
      RETURN BUFFER FULL
      RETURN NORMAL

INITIALIZE ONCE DSPTR TO BUFFER ADDRESS
AND DSCNT := DSBFL DIVIDED BY 2

BSW = 7002
DSBUF = 400

DSOUT, 0
DCA DSTMP /TEMP STORAG
RAR
DCA DSLNK /SAVE LINK
TAD DSCNTW /FIRST, SECOND OR THIRD CHAR
CLL RTR
SNL SMA CLA
JMP DSFRST /FIRST CHAR OF THREE
SNL
JMP DSSEC /SECOND CHAR OF THREE
TAD DSTMP /THIRD CHAR
TAD I DSPTR
DCA I DSPTR /PUT IN BUFFER
DCA DSCNTW /RESET CHAR COUNT
ISZ DSPTR
ISZ DSCNT /BUFFER FULL?
JMP DSEX3 /NO, EXIT
TAD DSBFA /YES, RESET POINTER
DCA DSPTR
TAD DSBFL /BUFFER LENGTH
STL RAR /DIVIDE BY 2
DCA DSCNT
TAD DSLNK /RESTORE LINK
CLL RAL
JMP I DSOUT /EXIT BUFFER FULL
DSSEC, TAD DSTMP
CLL RTL
BSW /BYTE SWAP
AND C77
TAD I DSPTR
DCA I DSPTR
ISZ DSPTR
TAD DSTMP
AND C17
BSW
CLL RTL
DCA I DSPTR
JMP DSEX2
DSFRST, TAD DSTMP
CLL RTL
RTL
DCA I DSPTR
DSEX2, ISZ DSCNTW
DSEX3, TAD DSLNK /RESTORE LINK
CLL FAL.
LSZ LSOUT
JMP I BSOUT /NORMAL EXIT

/VARIABLES
/DSBFL, -490
LSBUF, IBSUF /OUTPUT BUFFER ADDRESS
LSLNK, 0
DSTMP, 0
LSCNTW, 0
LSCNT, 0
DSPTR, 0

/GENERAL CONSTANTS
C17, 17
C77, 77
033 SUBROUTINE TO UNPACK CHARACTERS (TSS8)
Packed Three Characters In Two Words (TSS8 Format)

PACKED: 11111111112222
222233333333
CALL JMS UNPACK
ADDRESS OF INPUTBUFFER
ADDRESS OF OUTPUTBUFFER
RETURN

ROUTE USES AUTO-INDEX 16

UNPACK, 0
TAD UNPBF1L /*BUFFERLENGTH INPUTBUFFER
STL HARI /*DIVIDE BY 2
DCA UNPCNT
TAD I UNPACK /*ADDRESS INPUTBUFFER
DCA UNPPTH
ISZ UNPACK
CLA CMA /*-1
TAD I UNPACK /*ADDRESS OUTPUTBUFFER
DCA 10
ISZ UNPACK

UNPLOP, TAD I UNPPTR
RTR
RTR
AND C377 /*FIRST CHAR
DCA I 10 /*SECOND CHAR
TAD I UNPPTR /*PICK UP CHAR AGAIN
CLL R1L
RTL
AND C360 /*SECOND CHAR
DCA UNPTMP /*TEMP. STORAGE
ISZ UNPPTMP
TAD I UNPPTR
CLL RAL
RTL
AND C17 /*SECOND CHAR
TAD UNPTMP
DCA I 10 /*THIRD CHAR
TAD I UNPPTR
AND C377
DCA I 10 /*THIRD CHAR
ISZ UNPPTR
ISZ UNPCNT /*READY ?
JMP UNPLOP /*NO, CONTINUE
JMP I UNPACK /YES, EXIT

VARIABLES
UNPPTR, 0
UNPTMP, 0
UNPCNT, 0
UNPBF1L, -400

GENERAL CONSTANTS
C17, 17
C360, 360
C377, 377

52
SUBROUTINE UNPACKS CHARACTERS ONE BY ONE (1SS8)
PACKED THREE CHARACTERS IN TWO WORDS (1SS8 FORMAT)
PACKED: 111111112222
222233333333

CALL :JMS UNPSGL
ADDRESS INPUTBUFFER
RETURN BUFFER EMPTY AC=0
NORMAL RETURN AC=CHAR.

/INITIALIZE ONCE UNPREF=UNPREP=UNPCNT=0

UNPSGL, 0
CLA CALL
TAD UNPREF /ARE THERE CHAR'S IN
SZA CLA /TEMP. BUFFER ?
JMP UNPGET /YES, GET ONE
TAD UNPREF /NO, INPUTBUFFER EMPTY ?
SZA CLA
JMP UNPREF /YES, RETURN END OF BUFFER
TAD UNPCNT /NO OR YES, MUST I
SNA CLA /START UP POINTERS ?
JMS UNPINI /YES, PLEASE DO
TAD UNPREF /NO, JUST UNPACK NEXT WORDS
DCA UNPREF
TAD I UNPPT /NEXT WORD FROM INPUTBUF
RTH
RTR
AND C377
DCA I UNPREF /FIRST CHAR IN TEMP. BUF
ISZ UNPREF
TAD I UNPPT /GET WORD AGAIN
CLL RTL
RTL
AND C360
DCA I UNPREF /TEMP. STORAGE
ISZ UNPPT
TAD I UNPPT /NEXT WORD
CLL RA
RTL
RTL
AND C17
TAD I UNPREF
DCA I UNPREF /SECOND CHAR
ISZ UNPPT
TAD I UNPPT /THAT WORD AGAIN
ISZ UNPPT
AND C377
DCA I UNPREF /THIRD CHAR
TAD UNPREF /RESET POINTER TEMP. BUF
DCA UNPREF
CLA CLA CMA RTL /-3
DCA UNPREF /3 CHAR'S IN TEMP. BUF
ISZ UNPCNT /INPUTBUFFER EMPTY ?
JMP UNPGET /NO, GET CHAR NOW
IAC /YES, SET FLAG BUFFER EMPTY
DCA UNPREF /AND THAN GET CHAR
UNPGET, ISZ UNPREF /LAST FROM TEMP. BUF ?
IAC /NO, SET FLAG

53
DCA UNPRBF /YES RESET FLAG
TAD I UNPEP /GET CHAR
ISZ UNPRP
ISZ UNPSGL /NORMAL EXIT
UNPEMT, ISZ UNPSGL
JMP I UNPSGL
/
UNPEMP, DCA UNPRBF /RESET FLAG
JMP UNPEMT /AND EMPTY BUFFER RETURN
/
UNPINI, 0
DCA UNPRBF /RESET FLAG
TAD I UNPSGL. /ADDRESS INPUTBUFFER
DCA UNPPTR
TAD UNPRFL /
-LENGTH OF BUFFER
STL RAR /DIVIDE BY 2
DCA UNPCNT
JMP I UNPINI
/
/VARIABLES
UNPRFL, -400'
UNPCNT, 0
UNPRCT, 0
UNPRP, 0
UNPPT, 0
UNPRF, 0
UNPBF, 0
UNPBA, UNPRB
UNPRB, 0
0
0
/
/GENERAL CONSTANTS
C17, 17
C360, 360
C377, 377
SUBROUTINE TO READ A NAME FROM KEYBOARD

CALL :JMS RDNAME
WORD 1, 2 CHAR'S FROM NAME IN EXCESS-40 CODE
WORD 2, 2 CHAR'S FROM NAME
WORD 3, 2 CHAR'S FROM NAME

ERROR RETURN
NORMAL RETURN

ROUTINE USES AUTO INDEX 10, ROUTINES READ, PRINT
AND CRLF

BS=7002
BUFADR=400

RDNAME, 0
TAD RDNAMEF          /ADDRESS ASCII BUFFER
DCA RDPT1R
DCA RDCT1NT          /CHAR. COUNTER
RDIN, JMS READ      /READ CHAR FROM KEYB.
DCA RDCHAR
TAD RDCHAR
TAD RDINT0           /ROBOUT?
SNA
JMP RDROS            /YES. TO SERVICE
TAD RDMCHR          /NO. CARRIAGE RETURN?
SNA
JMP RDTWNR           /YES, TO SERVICE
TAD RDMLF D          /NO, LINE FEED
SNA
JMP RDTWNR           /YES, SAME SERVICE AS CR
TAD RDMSPE          /NO, CHAR>240?
SPA SNA CLA
JMP RDFTNM          /NO, ERROR RETURN
TAD RDCHAR          /YES, IN BUFFER
DCA I RDPT1R
ISZ RDCNT           /+ CHAR'S
ISZ RDP1RT
JMP RDIN            /NEXT CHAR
JMP RDFTNM          /4K BUFFER FULL, ERROR

RDROS, TAD RDCNT          /ALREADY SOMETHING IN BUFFER?
SNA CLA
JMP RDIN            /NO, STUPID RO-TYPER!
CMA
TAD RDCNT
DCA RDCNT
CMA
TAD RDPT1R
DCA RDPT1R
TAD I RDPT1R
JMS PRINT
JMP RDIN          /END RO-SERVICE

RDTWNR, JMS CR LF
TAD RDCNT
SNA
JMP RDFTNM        /NAME WITHOUT CHAR'S IS RUBBISH
TAD ME
SMA S4A /SIX OR LESS CHAR'S
CLA /MORE THAN MAKE IT SIX
TAD C6 
CLA DCA RDCTNT /*# CHAR'S
TAD RDNAME /BUFFER ADDRESS
DCA RDPR TR
TAD RDNAME /PACKED NAME ADDRESS
DCA RDTMP /PLACED UNDER CALLING
TAD RDTRMP
DCA 10
DCA 1 10
DCA 1 10 /*CLEAR BUFFER
RDNXT, TAD I RDPR TR /MAKE EXCESS-40 CODE
TAD C240
AND C77
BSW
DCA I RDTMP
ISZ RDPR TR
ISZ RDCTNT
SKP
JMP RDNMOK /READY READING NAME
TAD I RDPR TR /NOT READY NEXT CHAR
TAD C240
AND C77,
TAD I RDTMP
DCA I RDTMP,
ISZ RDTMP
ISZ RDPR TR
ISZ RDCTNT
SKP
JMP RDNXT /NEXT CHAR'S
RDNMOK, ISZ RDNAME /NORMAL RETURN
RDFTNM, ISZ RDNAME
ISZ RDNAME
ISZ RDNAME
JMP I RDNAME /EXIT
/
/
/VARIABLES
/
RDNMBF, BUFADR /ADDRESS BUFFER
RDPR TR, 0
RDCTNT, 0
RDCHAR, 2
RDTMP, 0
RDCHAR, -377
RDNAME, 377-215
RDLFD, 215-212
RDMSPE, 212-240
/
/GENERAL CONSTANTS
N6, -6
C6, 6
C77, 77
C240, 249
SUBROUTINE SEARCHES NAME IN DNS-BLOCKS (DISK MON.)
(DISK MONITOR SYSTEM)

CALL :MS DNSRC

NAME FIRST TWO CHAR'S IN EXCESS-40 6 BIT
ME LAST " " " " "
RETURN NAME NOT FOUND AC=0
RETURN NAME FOUND AC=INT. FILE NR

SUBROUTINE USES AUTO INDEX 11 AND MONITOR DISK HANDLER

BUFFER=400

DNSRC, 0

TAD C177
JMS DNSRBK
TAD I DNSRC
CLIA
DCA DNSMNA
ISZ DNSRC
TAD I DNSRC
CLIA
DCA DNSMME
ISZ DNSRC

DNSBLK, CLA CALL IAC RAL +2
TAD DNSBFA
DCA 11
TAD M31
DCA DNSCNT

DNSNXT, TAD I 11
TAD DNSMNA
TAD DNSMNA
SZA CLA
JMP DNSNOT
TAD I 11
TAD DNSMME
SZA CLA
JMP DNSNT1
ISZ 11
ISZ 11
TAD I 11
AND C7
ISZ DNSRC

DNSERR, JMP I DNSRC

DNSNOT, CLA IAC
DNSNT1, TAD C3
TAD 11
DCA 11
ISZ DNSCNT
JMP DNSNXT
TAD DNSLK
SNA
JMP DNSERR
JMS DNSRBK
JMP DNSBLK

DNSRBK, 0
DCA FSTBLK
TAD C3
LCA FUNCTION
TAD DNSRFA
LCA BUFADR
DCA DNSLNK
JMS I SYSIO  /MONITOR DISK HANDLER

FUNCTI, 0
FSTBLK, 0
BUFADR, 0
DNSLNK, 0
HLT  /ERROR RETURN
JMP I DNSRBA

/ VARIABLES

/ SYSIO, 7642
DNSMNA, 0
DNSMMN, 0
DNSCNT , 0
DNSRBA, BUFFER

/ GENERAL CONSTANTS
C3, 3
C7, 7
C177, 177
M31, -31
SUBROUTINE SEARCHES UNUSED BLOCK ON DISK (DISKMON)
AND RESERVES IT FOR FILE (DISK MONITOR SYSTEM)

CALLING: JMS SAMFIL WITH INT. FILE NR IN AC
RETURN DISK FULL
RETURN NORMAL WITH BLOCKNR IN AC

SAMFIL, 0
DCA SAMSAV /SAVE INT FILE #
JMS SAMSAV /SEARCH FOR EMPTY BLOCK
JMP I SAMFIL /NOT FOUND SO DISK FULL
CLA CMA /BLOCKNR STILL IN SAMRKN
TAD 10 /AUTO INDEX STILL ON SPOT
DCA 10
TAD SAMMSK /WHICH HALF IS MASK
TAD M77
SNA CLA /LEFT OR RIGHT?
JMP SAMRGT /MASK IS ON RIGHT HALF
TAD SAMSAV
CLL RTL /PUT INT FILE # ON LEFT HALF
RTL
RTL
DCA SAMSAV
JMP +3

SAMRGT, TAD I 10
TAD SAMSAV /ADD INT FILE #
DCA SAMSAV /TEMP. STORAGE
CMA
TAD 10
DCA 10
TAD SAMSAV /PUT IN BUFFER
DCA I 10
TAD C5
DCA FUNCTION
TAD SAMBAF
DCA BUFADR
JMS SAMRDB /RESTORE SAM ON DISK
ISZ SAMFIL
TAD SAMRKN /GET BLOCKNR
JMP I SAMFIL /RETURN

VARIABLES
SAMSAV, 0

GENERAL CONSTANTS
M77, -77
C5, 5
// SUBROUTINE SEARCHES INT. FILE # (DISKMON)  
// IN SAMBLOCKS (DISK MONITOR SYSTEM)  
//  
// CALL :JMS SAMSRC WITH INT. FILE # IN AC  
// RETURN NUMBER NOT FOUND: AC=0  
// RETURN NR FOUND, AC=# FIRST BLOCK FROM FILE  
//  
// SUBROUTINE USES AUTO INDEX 10 AND MONITOR DISK HANDLER  
//  
// BUFFER=400  
//  
// SAMSRC, @  
LCA SAMLIN /INT FILE # TO SEARCH FOR  
TAL SAMLIN /MAKE IT TWO IN ONE WORD  
CALL RTL  
RTL  
TAD SAMLIN  
LCA SAMLIN  
LCA SAMLIN /COUNTER FOR BLOCKNR  
TAD CR200 /# FIRST SAMBLOCK  
SAMSRC, LCA BLKNR  
TAD C3 /READ FUNCTION  
LCA FUNCTION  
TAD SAMPFA /BUFFER ADDRESS  
DCA BUFFER  
JMS SAMPFR /READ BLOCK  
SAMSRC, TAD C77  
DCA SAMSKR /SEARCH RIGHT HALF  
TAD C200  
LCA SAMCN /200 WORDS  
CXA  
TAD SAMPFA  
LCA 10  
SKP  
SANNXT, ISZ SAMLIN /COUNT BLOCKNR  
TAD I 10 /GET WORD  
AND SAMSKR /MASK  
CIA /NEGATIVE  
DCA SAMPBP /TEMP. STORAGE  
TAD SAMLIN /INT FILE # TO SEARCH FOR  
AND SAMSKR /EASLY CORRECT HALF  
TAD SAMPBP /SAMP #?  
SNA CLA  
JMP SAMPFL /YES, FOUND IT  
ISZ SAMCAT /NO, MORE IN THIS HALF?  
JMP SANNXT /YES, SEARCH  
ISZ SAMLIN /NO, UPDATE BLOCKNR  
TAD SAMSKR /WHERE WHERE WE SEARCHING?  
AND C7700  
SZA CLA /LEFT OR RIGHT HALF  
JMP +3 /LEFT HALF, BOTH SIDES DONE  
TAD C7700 /RIGHT HALF, SO DO LEFT NOW  
JMP SAMBP+1  
TAD SAMLNK /LAST SAMBLOCK?  
SNA  
JMP SANNOT /YES, SO NOT FOUND  
JMP SAMSRC /NO, READ NEXT BLOCK
/   
SAXNF, 1SZ SASLC
TAB SASLN
SAXNCO, JMP I SASLC
/   
SAXFLR, 0
JMS I SYSIO /MONITOR DISK HANDLER
FUNCTION 0 /READ=3, WRITE=5
FLKRN, 0 /BLOCKNR
BUFADD, 0 /BUFFERADDRESS
SAMLNK, 0 /NR NEXT BLOCK, 0=LAST BLOCK
HLT /ERROR RETURN, SYSTEM ERROR
JMP I SAXFLR
/ VARIABLES
/ SAMLMP, 0
SAMLF, 0
SAMFKN, 0
SAMMSK, 0
SAMCNT, 0
SAMEFA, BUFFER
SYSIO, 7642
/ GENERAL CONSTANTS
/ C3, 3
C77, 77
C200, 200
C7700, 7700
M200, -200
/G39 SUBROUTINE READS OR WRITES ON DISK (TSS-8)
/
/BEFORE CALLING CALCULATE DISKADDRESS AND
/PUT IN HIO AND LOWOR
/
/CALL :JXS DFIE
/ FUNCTION (DFILE OR WFILE)
/ INTERNAL FILE NUMBER
/ - # WORDS
/ CORE ADDRESS
/ ERROR RETURN
/ NORMAL RETURN
/
/FILE MUST BE OPEN !!!!!!!!
/
DFILE, 0
TAD I DFIE
DCA DFINST
ISZ DFIE
TAD I DFIE
DCA W6BUF+1
ISZ DFIE
DFTRY, TAD I DFIE
DCA W6BUF+2
ISZ DFIE
CLA CMA
TAD I DFIE
DCA W6BUF+3
ISZ DFIE
TAD W6AD
/ADDRESS 6 WORD BUFFER
DFINST, 0
TAD W6BUF+5
/D0 FUNCTION
SNA
JMP DF0KE
/NO ERROR
CLL RTR
/ERROR
SZL SNA CLA
SKP CLA
/ERROR=2
JMP DFERR
/ERRORS NOT 2
IAC
DCA W2BUF+2
/ERROR IS FILE FULL
TAD W2AD
/SO MUST EXTEND FILE
EXT
/EXTENDING WITH ONE SEGMENT
SZA CLA
JMP DFERR
/ERROR: DISK FULL
TAD DFSEGA
/ADDRESS LIST SEGMENTCOUNTERS
TAD W2BUF
/INT FILE NR
DCA W6BUF+2
/TEMP USE
ISZ I W6BUF+2
/INCRE N COUNTER
CLL CLA CMA RAL
/-2
TAD DFIE
DCA DFIE
JMP DFTRY
/GO TRY AGAIN NOW
DF0KE, ISZ DFIE
DFERR, JMP I DFIE
/
/VARIABLES
/
DFSEGA, DFSEG0
DFSEG0, 0
/# SEGMENTS FILE 0

62
<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSEG1, 0</td>
<td>/ SEGMENTS FILE 1</td>
</tr>
<tr>
<td>LSEG2, 0</td>
<td>/ SEGMENTS FILE 2</td>
</tr>
<tr>
<td>LSEG3, 0</td>
<td>/ SEGMENTS FILE 3</td>
</tr>
<tr>
<td>W6AL, W6BUF</td>
<td></td>
</tr>
<tr>
<td>W2AL, W2BUF</td>
<td></td>
</tr>
<tr>
<td>W6BUF,</td>
<td></td>
</tr>
<tr>
<td>HI0, 0</td>
<td>/HIGH ORDER DISK ADDRESS</td>
</tr>
<tr>
<td>W2BUF, 0</td>
<td>/INT FILE NR</td>
</tr>
<tr>
<td>0</td>
<td>/# WORDS; # SEG'S TO EXT</td>
</tr>
<tr>
<td>0</td>
<td>/CORE ADDRESS-1</td>
</tr>
<tr>
<td>LOW0, 0</td>
<td>/LOW ORDER DISK ADDRESS</td>
</tr>
<tr>
<td>0</td>
<td>/ERROR WORD</td>
</tr>
</tbody>
</table>