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 "recipes" in a cookbook. When these "recipes" 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.
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).

/COMMENT

Comments appear behind a slash (/).

START,
LOOP,

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
TAD I TYPTEX /GET CHAR.
ISZ TYPTEX
SNA /ZERO?
JMP I TYPTEX /YES, JMP TO NEXT LOC.
JMS TYPE /NO, TYPE.
JMP TYPTEX+1

Prog 002
/002 TELETYPewriter TYPE ROUTINE used by prog 001, 003
/INITIALIZES WHEN ENTERED FOR FIRST TIME.
/NOT RESTARTABLE !
/
/ TAD CHARACTER
/ JMS TYPE
/ RETURN ZAC=0

TYPE,
NOP

0
JMP +3 /OVERLAPPED BY "NOP"
TSF
JMP -1
TLS
CLA
TAD TYPE-1
LCA TYPE+1
JMP I TYPE
003 TYPE A CHARACTER CHAIN
TYPE THE CHARACTERS IN THE LIST, POINTED TO
BY THE FIRST ARGUMENT. LIST TERMINATOR = 0

JMS TYPTEX /TYPE "ARC"
LIST
RETURN /AC = 0

/LIST, 301
/ 302
/ 303
/ 0

0 /USED AS POINTER
TYPTEX, 0 /TYPE TEXTSTRING
TAD I TYPTEX /GET ARG
DCO 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
004 BINARY TO DECIMAL CONVERSION AND TYPE; NO SIGN
ROUTINE TO CONVERT A BINARY WORD TO DECIMAL AND TYPE IT.
VALID FOR NUMBERS 0-4095. NO SIGN.
IF USED FOR 3 DIGITS: DELETE 6030; -4=-3 DIGIT COUNT.

/ TAD WORD
/ JMS PRINTD
/ RETURN
/ AC=0

CCC 6030 /-1000 CONVERSION CONSTANTS
7634 /-100
7766 /-10
7777 /-1

-6 r 1204
DIGIT 8 /USED FOR CONV. CONSTANTS
-5 CNTR 0 /DIGIT BCD TO BE TYPED
-4 k260 260 /COUNTER
-3 REMAIN 0 /TO MAKE A CHR.
-2 MA 4 /SAVE AREA
-1 PRINTD 0 /DIGITS TO BE TYPED (-4, -3, -2)

DCA PRINTD-2 /ENTER WITH WORD IN AC
LCA PRINTD-4

LOOP1, TAD PRINTD-6 /CLEAR BCD
LCA PRINTD-4

LOOP2, TAD PRINTD-6 /FETCH CURR. CONV. CONST.
TAD PRINTD-4 /BY ADDING COUNT TO TAL
DCA +1
HLT /SET TO TAD C C C + 0, 1, 2, 3

CLL
TAD PRINTD-2 /VALUE = CONSTANT
SML
JMF +4 /OVERFLOW?
I $ PRINTD-5 /NO, TYPE IT
LCA PRINTD-2 /YES, NEXT TRY
JMP PRINTD+5 /SAVE REMAINDER = REMAIN
CLA
TAD PRINTD-5 /BCD
TAD PRINTD-3 +260
JMS TYPE

ISZ PRINTD-4
JMP PRINTD+4 /NEXT DIGIT

Cal2 23
Print 7
Const 3
Var 3

33

5 instrs shorter than DEC routine
/005 BINARY TO OCTAL CONVERSION AND PRINT
/Routine prints the AC in octal, no sign.
/
TAD WORD
JMS PRINTB
RETURN /AC=0
/
260
7
3
-4
0
PRINTB, 0

RAL CL1
LCA PRINTB-1
TAL PRINTB-2
LCA PRINTB-3 /SET UP COUNT
TAL PRINTB-1

Loop
RAL
RTL
LCA PRINTB-1
TAL PRINTB-1
AND PRINTB-4 /MASK
TAL PRINTB-5 /MAKE ASCII
JMS TYPE
ISZ PRINTB-3 /4 DONE?
JMP PRINTB+5 /NOT YET
JMP I PRINTB

Code 16
Vara Count 5

21
06 HIGH SPEED READER SUBROUTINE
ENTER WTH AC=0: ROUTINE INITIALIZES HSREAD EACH REENTRY
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 HSREAD
OUT OF TAPE RETURN
NORMAL RETURN, CHAR IN AC
/

HSREAL, 3
LCA HSREAD-1
HSRELG, 1
TAD HSRELG
SZA CLA
JMP *+3
HSF
JMP *+5
DCA HSRELG
6016
ISZ HSREAD
JMP I HSREAD
ISZ HSREAD-1
JMP HSRELG
ISZ HSRELG
JMP I HSREAD

0USED AS TIME-OUT COUNT
ENTRY
SET UP COUNT
FLAG SIGNALS TO INIT READER
THESE INSTR. CONTRIBUTE TO LOOP
INIT READER
SKIP?
NO, COUNT TIME-OUT
CLEAR FLAG
REAL
RETURN, CHAR IN AC

14
007 TABULATOR ROUTINE
THE USER HAS TO TAKE CARE OF:
INCREMENTING TABCNT WITH EACH INCOMING CHARACTER, CLEARING
IT WHEN CARRIAGE RETURN. TAB-INTERVAL IS VARIABLE.
A JMS TO TAB WILL MOVE THE TYPING HEAD TO THE NEXT
TABULATOR POSITION.

/CLA
/JMS TAB
/RETURN /AC=0

TABCNT, 0
-10 /TAB INTERVAL
240 /SPACE
TAB, 0 /ENTER WITH AC =0
TAL TABCNT /SUBTRACT N TIMES TO GIVE REMAINDER
TAL TAB-2
SMA
JMP *-2 /USE AS NEGATIVE COUNTER
LCA TABCNT
TAL TAB-1
/JMS TYPE
1SZ TABCNT /REALY?
JMP *-3
JMP I TAB /YES
SUBROUTINE TO MOVE A BLOCK THROUGH CORE.

CALLING SEQUENCE.

JMS MOVE
BEGINADDRESS
ENDADDRESS
DESTINATION OR FIRST WORD
RETURN \( \text{ZAC}=0 \)

IF BEGINADDRESS AND ENDADDRESS ARE
THE SAME ADDRESS, OR BEGINADDRESS
IS GREATER THAN ENDADDRESS,
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

\( 0 \)
\( 1 \)
\( 2 \)
\( 3 \)

MOVE.

TAD I MOVE \(/\text{GET BEGINADDRESS}\\)
LCA MOVE-1
1SZ MOVE
TAD I MOVE \(/\text{GET ENDADDRESS}\\)
LCA MOVE-2
1SZ MOVE
TAD MOVE-2
CMA
TAD MOVE-1 \(/\text{CALCULATE MOVECOUNT}\\)
SNA \(/\text{IS IT POSITIVE OR ZERO?}\\)
JEP MOVSET \(/\text{YES, NO MOVE NEEDED}\\)
LCA MOVE-4 \(/\text{SAVE MOVECOUNT}\\)
TAD MOVE-1
CIA CLL
TAD I MOVE \(/\text{CALCULATE MOVECOUNT}\\)
LCA MOVE-3 \(/\text{AND SAVE}\\)
SZL \(/\text{LINK IS ON IF MOVE TO HIGHER CORE}\\)
JMP \(+3 \)/(\text{skip next instructions})
IAC CML \(/\text{LINK IS OFF}\\)
TAD MOVE-4 \(/\text{FIRST IN ADDRESS IS BEGINADDRESS}\\)
TAD MOVE-2
LCA MOVE-2 \(/\text{SAVE INPUTPOINTER}\\)
SRL ICA RAL \(/\text{TO HIGHER CORE, INC = -1}\\)
IAC
LCA MOVE-1 \(/\text{SAVE INCREMENT}\\)
TAD MOVE-2 \(/\text{SET UP OUTPUTPOINTER}\\)
TAD MOVE-3
LCA MOVE-3 \(/\text{AND SAVE}\\)
MVLOOP, TAD I MOVE-2 \(/\text{GET A WORD}\\)
LCA I MOVE-3 \(/\text{AND STORE IT IN DESTINATION BLOCK}\\)
TAD MOVE-2
TAD MOVE-1 \(/\text{INCREMENT INPUTPOINTER}\\)
LCA MOVE-2
TAD MOVE-3

16
TAD MOVE-1 /INCREMENT OUTPUTPOINTER
LCA MOVE-3
ISZ MOVE-4 /INCREMENT WORDCOUNT
JMP MVLOOP /AGAIN IF NOT ZERO
/
MOURET, ISZ MOVE /SET UP RETURNADDRESS
JMP I MOVE /RETURN
009 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 PUNCHK 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
CLL
TAD 0010
JMS BINPUN
ORIGIN
FIRST LOC.
LAST LOC.
JMS PUNCHK
/PUNCH LEADER, CLEAR CHECKSUM
/FIELD 1
/IN NORMAL USE=FIRST LOC.
/PUNCH CHKSM AND TRAILER

BINEND, 0
6201
100
300
0

BINPUN, 0
DCA BINPUN-1
SZL
JMP BIN3
TAD BINPUN-1
TAD BINPUN-4

BIN3,
DCA PIN2
TAD BINPUN-1
TAD BINPUN-2
JMS TYPE
TAD I BINPUN
ISZ BINPUN
DCA BINPUN-1
TAD BINPUN-1
JMS BINLH
TAL BINPUN-3
JMS BINCHK
TAD BINPUN-1
AND BINLH-1
JMS BINCHK
TAD I BINPUN
DCA BINPUN-1
ISZ BINPUN
TAD I BINPUN
DCA BINEND

BIN2,
CDF 0
TAD I BINPUN-1
JMS BINLH
/SET UP DF IF LINK=0
/MAKEThrown
/MAKEFIELDSETT. AND PUNCH
/NOTE!!! FIELD SETT. NOT IN CHECKSUM!!

 Prog 002

TAD I BINPUN
DCA BINPUN-1
TAD I BINPUN
DCA BINPUN-1
TAD I BINPUN
TAD I BINPUN
DCA BINPUN-1
/SET UP POINTER
ISZ BINPUN
TAD I BINPUN
DCA BINEND

OVERLAI D BY CDF STRANGE FIELD
/GET DATA
/PUNCH LEFT HALF

-- See Page 19
JMS BINCHK  
TAD I BIPNUN-1 /PUNCH RIGHT HALF
AND BINLH-1
JMS BINCHK
TAD BINEND /END REACHED?
CIA
CII
TAD BIPNUN-1
ISZ BIPNUN-1
SNL CLA
JMP RIN2+1 /NO, TAKE NEXT DATA
RIF /YES, RESTORE DF
TAD BIPNUN-4
DCA +1
CLF 0 /OVERLAIN
ISZ BIPNUN
JMP I BIPNUN
SPA CLA

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

/UPDATE THE CHECKSUM AND PUNCH FRAME
/CHECKSUM
FINCHK 0 /ENTER WITH 6 HIT FRAME IN AC
DCA BINLH /USE FINLH TEMPORARILY
TAD BINLH
TAD BINCHK-1 /UPL. CHKS
LCA BINCHK-1
TAD BINLH
JMS TYPE
JMP I BINCHK

/PUNCH 100(8) LEADER OR TRAILER HOLES; CLEAR CHECKSUM
200
LEADER 0
TAD BIPNUN-3 /USE CHKS 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 BIPNUN-1
AND BINLH-1
JMS TYPE
JMS LEADER /CLEAR CHECKSUM
JMP I PUNCHK
/010 PAL MESSAGE PRINTER
/PRINTS A MESSAGE CODED WITH THE PAL PSEUDO-OP
/'TEXT': PAL3 AND PAL8 COMPATIBLE
/
/JMS PRMMSG
/ MESS
/ RETURN /AC=0
/
/MEGG, TEXT 'ABC82' /CODED AS 0102;0370136200

77 /MASK
-40 /TO TEST
240 /TO MAKE ASCII
100 /TO MAKE ASCII
0 /PACKSWITCH 0=LEFT;7777=R
. /POINTER

PRMMSG, 0 /SAVE POINTER(-1)
TAD I PRMSEG
DCA PRMSEG-1
ISZ PRMSEG /FOR RETURN

PRM1, CMA
DCA PRMSEG-2 /PACKSW=RIGHT
ISZ PRMSEG-1 /NEXT WORD
TAD I PRMSEG-1 /FETCH AND ROTATE 6
RTR
RTR
RTR

PRM2, AND PRMSEG-6 /MASK 6 BITS
SNA
JMP I PRMSEG /ZERO ENDS THE LIST
TAD PRMSEG-5 /<40?
SPA
TAD PRMSEG-3 /YES ASCII 301-337
TAD PRMSEG-4 /NO, ASCII 240-277
JMS PRINT
ISZ PRMSEG-2 /LEFT OR RIGHT?
JMP PRM1 /LEFT
TAD I PRMSEG-1 /RIGHT
JMP PRM2
/011 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
/ LF /IF "CHAR"=212 PROGRAM JUMPS TO "LF"
/ 215
/ CH; ETC; ETC; ETC
/ 0 /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
BRANC, 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  --- ASCII coded
/ JKS OCTCHK
/ NOT OCTAL RETURN /AC=0
/ OCTAL RETURN /AC=0

10
-270

OCTCHK, 0
TAD OCTCHK-1
SQA
JMP OCT2
TAL 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
/CLA
/    TAD A
/    AND B

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

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

/XOR
/                 A 1010
/                 B 1100
/                 = 0001
/CLA
/    TAD A
/    CMA
/    LCA TEM
/    TAD B
/    CMA
/    AND TEM

/XNOR
/                A 1010
/                B 1100
/                = 0111
/CLA
/    TAD A
/    AND B
/    CMA

/EXCLUSIVE OR
/              A 1010
/              B 1100
/              = 0110
/CLA
/    TAD A
/    X1 B
/    CMA
/    LCA TEM
/    TAD A
/    AND TEM
/    TAD B
/    AN1 TEM

See Raruein in Vol 2 p13

A+8 1000
CMA 0111 TEM
A+TEM 0000
ADD B 1100
M TEM 0110

See Raruein in Vol 2 p13
/14 PSK-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 OSM 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 331-332,268-271
/
/ TAL (16) /CHECK OPTION 16 (N)
/ JMS OPTION
/ OPTION NOT SET RETURN AC=0
/ OPTION SET RETURN AC=0
/
OPTION 0:

LCA OPTION-2
TAL OPTION-3
LCA OPTION-1
TAL OPTION-2
TAL OPTION-4
ISZ OPTION-1
SNA SZA
JMP -3
TAL OPTION1
LCA OPTION-2
CCL CML
RAL
ISZ OPTION-2
JMP -2
CDR 10
AND I OPTION-1
CDF 0
SZA CLA
ISZ OPTION
JMP I OPTION

/FRM L AND X
/SAVE REMAINDER MODULO 12
/AND ROTATE ONE BIT INTO POSITION
/ROTATE FURTHER

/AND WITH OPT WORD FIELD 1

/IN CASE IT HAD BEEN SET
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
LCA PRNT2-2 /TRY SUBTRACT 10 UNTIL OVFLO
TAD PRNT2-2
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
LCA PRNT2-1 /RESET COUNTER
JMP I PRNT2

Code 17
Var 2
Counts 2
21
016 PRINT THE PS8-US8 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 MMMMDDDDYYYY /M=MONTH, D=DAY, Y=YEAR

JMS DATE
RETURN

AC=0

MASKS

/SLASH

/70 YEARS

/STORAGE

/DATE LOC. IN OS8

PICK THE DATE

SHIFT MONTH OUT

AND (17)

PRINT SLASH

PRINT

SHIFT MONTH OUT AND PRINT

PRINT SLASH

NOW THE YEAR

+70

AC=0

HOE

PRINT2

JMP 1 DATE
/17 PRINT THE AC AS A FOCAL LINENUMBER
/ THE VALUE OF THE AC IS PRINTED AS IN FOCAL: 11.35
/ XX.YY STORED AS FOLLOWS: XXXXXXXYYYYY IN 1 WORD.
/ IF YYYYYYY >= 99 STRANGE DIGITS OCCUR AS IN FOCAL.
/ REQUIRES ROUTINES PRT2 AND PRINT.
/
/ TAD VALUE
/ JMS PRTF
/ RETURN

PRTFM, 37
177
256
0
PRTF, 0
LCA PRTF-1
TAD PRTF-1
CLL RTL
RTL
RTL AND PRTFM
JMS PRT2
TAD PRTF-2
JMS PRINT
TAD PRTF-1
AND PRTFM+1
JMS PRTD
JMP I PRTF

AC=0

MASKS
PERIOD
STORAGE

ISOLATE AND PRINT HIGH ORDER

AND (37)
NOW LOW ORDER

Prog 013
018 PRINT 4 DECIMAL DIGITS USING ROUTINE PRINT2
/ THE CONTENT OF THE AC IS DIVIDED BY 100(10)
/ GIVING TWO LOW ORDER DIGITS AND 2 HIGH ORDER.
/ THESE ARE PRINTED BY PRINT2.
/
/ TAD VALUE
/ JMS PRINT4
/ RETURN
/
7634
0
0
PRINT4, 0
LCA PRINT4-2
CLL
TAD PRINT4-2
TAD PRINT4-3
SNL
JMP *+3
ISZ PRINT4-1
JMP PRINT4+1
CLA
TAD PRINT4-1
JMS PRINT2
TAD PRINT4-2
JMS PRINT2
LCA PRINT4-1
JMP I PRINT4

/*AC=0*/
/*-100(10)*/
/*STORAGE AND LOW ORDER*/
/*HIGH ORDER COUNTER*/

/*TRY TO SUBTRACT 100 UNTIL OVERFLOW*/

/*PRINT HIGH ORDER DIGITS*/

/*PRINT LOW ORDER DIGITS*/

/*RESET COUNTER*/
/019 SUBROUTINE READS A DECIMAL NUMBER FROM KEYBD /
/RUBOUT REMOVES NUMBER COMPLETELY
/
/
/CALL JMS DECINP
/ RETURN WITH NUMBER BINARY IN AC
/
/
DECINP, 0

CLA
LCA 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
TAL M260
SPA
JMP DECOUT /NO, CHARACTER IS DELIMITER
TAD M12 /YES
SNA CLA /CHAR<272?
JMP DECOUT /NO, CHAR IS DELIMITER
TAL DECNUM /YES, CHAR IS FIGURE
CALL RAL
DCA DECTMP /NUMB*2
TAD DECTMP
RTL
TAL DECTMP /NUMB*8+NUMB*2=NUMB*10
TAD CHAR /ADD LAST FIGURE
TAD M260
LCA 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
DECIMAL PRINT ROUTINE,
PRINTS AC DECIMAL IN 4 DIGITS
MAX NUMBER = 4095 DECIMAL
SKIPS LEADING ZERO'S

LPRHTS, 0
LCA DPRREG
TAD DPRINS
LCA DPRTP
TAD M4
DCA DPRFAC
DCA DPRFL
DCA DPRFIG
LPRSUBL, CLL
TAD DPRREG
DPRTP, TAD DPRTE
SNL
JMP +4
DCA DPRREG
ISZ DPRFIG
JMP DPERSUB
CLA CLL
TAD DPRFIG
SNA
JMP LPRZRU
DPRIN, TAD C260
JMS PRINT
ISZ DPRFL
DPRINI, ISZ DPRTP
ISZ DPRFAC
JMP DPERSUB-1
JMP I LPRHT

LPRZRU, TAD DPRFL
SZA CLA
JMP DPRIN
JMP DPRINI

DPRREG, 0
DPRFL, 0
DPRINS, TAD DPRTE
DPRFAC, 0
DPRFIG, 0
DPRTE, 6030 /-1000
7634 /-100
7766 /-10
7777 /-1

GENERAL CONSTANTS
M4, -4
C260, 260

42. Wucer than DEC's.
SUBROUTINE TO PRINT DOUBLE LENGTH DECIMAL
/
/CALL: JMS DDECPR
/
MOST SIGNIFICANT PART
LEAST SIGNIFICANT PART
NUMBER OF DIGITS TO BE PRINTED ( <= 8 )
RETURN
/
DDECPR
TAD I DDECPR
DCA DLX
DCA DLPD
ISZ DDECPR
TAD I DDECPR
DCA DLX+1
ISZ DDECPR
TAD I DDECPR
DCA DLDIG
ISZ DDECPR
TAD DLATPL
DCA DLPDL
TAD DLATPH
DCA DLDEPTH
TAD A10
DCA DLMFAC
DCA DLDIG
DDECSP, CALL
TAD DLX+1
TAD I DLPTPL
DCA DLX+1
RAL
TAD LDV
TAD DLDEPTH
CMPL
JMP *4
DCA DLX
ISZ DLDIG
JMP DLSUB
CLA
TAD I DLPTPL
CLA
TAD DLX+1
DCA DLX+1
TAL DLDIG
SZA
JMP DLPLIN
TAD DLPD
SZA CLA
JMP DLPLIN
TAD DNMFAC
SZA CLA
JMP DLPLIN
TAD DNMFAC
TAD DLNDIG
SPA CLA
JMP DLPTIN
TAL C240
JMP DLPLI.
LDPLIN, 1SZ, LPFD
TAD C260
/CONVERT TO ASCII
LDPPR, J35, PRINT
/PRINT DIGIT
LDPTIN, 1SZ, DLPTPL
/STEP UP POINTER LOW
ISZ, DLPTPH
/STEP UP POINTER HIGH
ISZ, DLNFACT
/READY FACTORIZE?
JMP DLSUB-1
/NO, NEXT DIGIT
TAD DLPL
CIA
TAD DLNDIG
SPA SNA, CLA
JMP *+3
TAD DLNDIG
DCA DLPL
CLL
JMP LLC
/EXIT, END PUNCHOUT ROUTINE

/CONSTANTS PUNCH OUT ROUTINE
DDATPL, DDATPL
DDATPH, DDATPH
DDPTPL, 0
DDPTPH, 0
DLX, 0
DLNFACT, 0
DLDIG, 0
DLPL, 4600
6700
2500
4369
6636
7634
7766
7777
DLTPH, 3166
7413
7747
7775
7777
7777
7777
7777
DLPL, 0
DLNDIG, 2
/
/GENERAL CONSTANTS
M10, -10
C240, 240
C260, 260
022 OCTAL PRINT ROUTINE

/ NONSIGNIFICANT ZERO'S BECOME SPACES
/

CLA
LCA OCTFIG / CLEAR FLAG FIGURE PRINTED
LCA OCTSPC / CLEAR SPACE-COUNTER
TAD NUMBER
/CALL JMS OCTPRM / 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

OCTPRM, 0
RAL   / ROTATE IN LINK
LCA OCTTMP / TEMP. STORAGE
TAD M4 / 4 OCTADES
LCA OCTCNT

OCTPRM, TAD OCTTMP
RAL
HLT
LCA 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
CIA
LCA OCTSPC
TAD C240
JMP PRINT / PRINT THE SPACES
ISZ OCTSPC
JMP -3

OCTNUM, CLA IAC / SET FLAG FIG. PRINTED
LCA OCTFIG
TAD OCTTMP
AND C7
OCTPHI, TAD C260 / MAKE THE FIGURE
JMP PRINT

OCTPRM, ISZ OCTCNT / READY?
JMP OCTPRM / NO
JMP I OCTPRM / YES, EXIT
/
OCTZER, TAD OCTFIG
SZA CLA / FIGURES PRINTED ?
JMP OCTPHI / YES, PRINT THIS ZERO TOO
ISZ OCTSPC / NO COUNT AS SPACE
JMP OCTPRM
/
OCTTMP, 0
OCTCNT, 0
OCTSPC, 0
OCTFIG, 0
M4, -4
C7, 7
C240, 240
C260, 260
/023 DOUBLE WORD OCTAL PRINT ROUTINE
/USES ROUTINE OCTPR
/CALLING: JMS DOCTPR
/HIGH ORDER NUMBER
/LOW ORDER NUMBER
/RETURN AC=0
/
DOCTPR, 0

CLA
DCA OCTFIG
DCA OCTSPC
TAD I DOCTPR
ISZ DOCTPR
JMS OCTPR
TAD I DOCTPR
ISZ DOCTPR
JMS OCTPR
TAD OCTSPC
CIA
SNA
JMP I DOCTPR
IAC
DCA OCTSPC
TAD C246
JMS PRINT
ISZ OCTSPC
JMP -3
TAD C260
JMS PRINT
JMP I DOCTPR

/CLEAR FLAG FIGURE PRINTED
/CLEAR SPACE-COUNTER
/HIGH ORDER PART
/PRINT OCTAL
/LOW ORDER PART
/PRINT OCTAL
/PRINT OCTAL
/PRINT OCTAL
/PRINT OCTAL
/SPACES TO PRINT?
/N0, EXIT
/YES, NUMBER IS ZERO
/PRINT SPACES
/AND A "0"
/EXIT
/024 SUBROUTINE TRANSLATES TEXT TO ASCII

;CALL: JM5 TLXAS WITH TEXT CHARACTER IN AC
; RETURN CHARACTER IS SHIFT
; RETURN WITH ASCII CHARACTER IN AC

/WHO IS TRANSLATED AS $
/? IS TRANSLATED AS *
/RELL 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 /FIGURESIFHT
  TAD M4
  SNA CLA
  JMP TLXSW2 /LETTERSHIFT
  TAD TLXTMP /GET CHARACTER AGAIN
  TAD TLXLA /ALL LISTADDRESS
  DCA TLXTMP /TEMP STORAGE
  TAD TLXSW /WHICH SIDE?
  SNA CLA
  JMP TLXRGST /RIGHT SIDE
  TAD I TLXTMP /GET ASCII 6 BIT
  RTH
  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

/ TLXRGST, TAD I TLXTMP
  JMP TLXMS

/ TLXSW1, IAC
TLXSW2, DCA TLXSW /REMEMBER WHICH SHIFT
  JMP I TLXAS

/ TLXCR, TAD C215
  JMP TLXOUT

TLXNL, TAD C212
  JMP TLXOUT

/ TLXLA, TLXLST
TLXLST, 0
2465 /T 5
37 37
1771 /U 9
4046 /SPACE
1036 /H 1
1654 /N 3
1556 /N 3

-40
1451 /L 0
2264 /R 4
0735 /G 1
1170 /I 8
2060 /P 0
2372 /C :.
2675 /Q ==
0563 /E 3
3853 /Z +
0477 /U WHO=:
0252 /B ?=*2
2347 /S 1
3166 /Y 6
0633 /F 0
3057 /X /
0155 /A -
2762 /W 2
1273 /J HELLE;

100
2567 /U 7
2161 /G 1
1356 /K 0

/ VARIABLES

/ TLXTMP; 0
TLX5; 0

/ GENERAL CONSTANTS
M2; -2
M4; -4
M6; -6
M23; -23
C77; 77
C212; 212
C215; 215
C240; 240
/026 ROUTINE TO TRANSLATE ASCII TO TELEX
/CALL *JMS ASTLX
/ RETURN
/
/BFREJF J ALL CALL INITNIALIZE ASTCFT: =4 AND
/PRINT A LETTERSHIFT
/
/NOT EXISTING CHARACTERS ARE PRINTED AS BLANK
/ALTMOV E IS TRANSLATED AS FIGURESHEET
/EOBUF OUT IS TRANSLATED AS LETTERSHIFT
/
ASTLX, A
DCA ASTTMP /TEMP STORAGE
TAD ASTTMP
ANL C77 /MAKE 6 BIT
SNA
JMP ASTOUT+2 /BLANK=BLANK
TAD ASTLA /LISTADDR
DCA ASTHLP /LISTADDR + 6-BIT CHAR
TAD ASTTMP
TAD C300
SNA CLA
JMP ASTRIG /CHAR>=300; RIGHT HALF OF LIST
TAD I ASTHLP /CHAR<300; LEFT HALF OF LIST
ATR
LTR
LTR
SKP
ASTRIG, TAD I ASTHLP
DCA ASTTMP /TEMP STORAGE
TAD ASTTMP
ANL C77
SNA
JMP ASTOUT+2 /NOT EXISTING IN TELEX: BLANK
AND C49 /GET SHIFT BIT
SZA CLA /WHICH SHIFT
JMP ASTSHF /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
ANL C37 /MASK 5 BITS
JMS PRINT /PRINT
JMP I ASTLY /EXIT
/
ASTSHF, 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 /*A
0023 /*B

39
5773 \text{/=, ALTMOD=HIGHSHIFT}
3000 \\text{/>}
6337 \text{/=, RUBOUT=LETTERSHIFT}

\text{/VARIABLES/}
ASTSFT, 0
ASTYMP, 0
ASTHELP, 0

\text{/GENERAL CONSTANTS/}

C33, 33
C37, 37
C40, 40
C77, 77
M300, -300
/027 INTERRUPT OUTPUT HANDLER
/ WITH HEAD-TAIL COUPLED BUFFER
/
/INITIALIZE ONCE BUFIP=BUFOP=BUFFER
/ BUFIBO=0
/
/
/CHARACTER HANDLER
/
/CALL :JMS BUFINP WITH CHAR IN AC
/ RETURN AC=0
/
/
BUFINP, 0
DCA BUFTMP / TEMP STORAGE
TAD BUFIBO / INPTR BEHIND OUTPTR?
SNA CLA
JMP BUFPUT / NO, STORE CHARACTER
TAD BUFIP
CIA
TAD BUFIP / INPTR = OUTPTR ?
SNA CLA
JMP BUFINP+2 / YES, WAIT FOR PLACE TO STORE
BUFPUT, TAD BUFTMP
DCA I BUFIP
ISZ BUFIP / PRINT BUSY?
TAD BUFBUS
SNA CLA
6046 / NO, INIT WITH AC=0
IAC
DCA BUFBUS
TAD BUFIP
TAD BUFBN
SZA CLA
JMP I BUFIP / NO, EXIT
TAD BUFADN / YES, POINTER TO HEAD
DCA BUFIP
IAC
LCA BUFIBO
JMP I BUFINP / EXIT
028 DEVICE INTERRUPT HANDLER

CALL :JMP BUFOUT /DEVICE INTERRUPT DETECTED!
ROUTINE RETURNS TO INTERRUPT RESTORE "EXIT"

BUFFOUT, CLA
6042 /CLEAR DEVICE FLAG
TAD BUF1PT
CIA
TAD BUF0PT /INPTR = OUTPTR ?
SZA CLA
JMP BUFGET /NO, GET CHAR AND PRINT
TAD BUFIEO /YES, INPTR BEHIND OUTPTR?
SZA CLA
JMP BUFGET /YES, GET AND PRINT
DCA BUFUS /NO, PRINTER READY
JMP EXIT

BUFGET, TAD I BUF0PT /GET CHAR
ISZ BUF0PT
6044 /PRINT CHAR
CLA
TAD BUF0PT
TAD BUFEND /END OF BUFFER?
SZA CLA
JMP EXIT /NO, END OF ROUTINE
TAD BUFALR /YES, POINTER TO HEAD
DCA BUF0PT
DCA BUFIEO /RESET INPTR BEHIND OUTPTR
JMP EXIT /END OF HANDLING

GENERAL INTERRUPT RETURN ROUTINE

EXIT, CLA CLL
TAD LINK
HAL /RESTORE LINK
TAD ACCU /RESTORE ACCU
ION /INTERRUPT ON
JMP 1 0

VARIABLES
BUFTMP, 0
BUFIO, 0
BUF1PT, 0
BUF0PT, 0
BUFUS, 0
BUFEND, -BUFEND
BUFALR, BUFFER
BUFFER, 0

*BUFFER+200
BUFEND, 0

43
SUBROUTINE READS OR WRITES DECTAPE IN BOTH DIRECTIONS

CALL JMS DECTAPE
DEFINING BITS
BLOCKNUMBER
-# WORDS (12 BITS)
BUFFERADDRESS-1
ERROR RETURN OR RETURNADDRESS
NORMAL RETURN OR RETURNADDRESS

DEFINING 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

DECTAPE, 0
CLA
TAD I DECTAPE /DEFINING BITS
DECTCOD /SAVE
LSZ DECTAPE
TAD DECTCOD
AND C7400 /UNIT# & DIRECTION BIT
TAD C10 /SEARCH MOLE
DECTCOD DETRA
I/O
DETXA /CLEAR FIELD REGISTER
TAD DECTWC /WORD COUNT ADDRESS
DECTCA /WORD COUNT=B P L A N K ADDRESS
TAD C200 /GO BIT

DECTCNT, JMS DECTRAN /TURN DECT AND WAIT FOR FLAG
TAD I DECTWC /READ NUMBER
CIA /NEG.
TAD I DECTAPE /NUMBER TO FIND
SNA
JMP DECTMAY /FOUND, CHECK DIRECTION

DECTSET, CLL RAL /SAVE SIGN DIFFERENCE
CLA
DETRA
AND C400 /DIRECTION BIT
SNA CLA
CML /IS FORWARD
SNL /IS REVERSE
TAD C400 /CHANGE DIRECTION
JMP DECTCNT /DIRECTION OK, NEXT NUMBER

DECTMAY, TAD DECTCOD /UNIT# & DIRECTION
AND C400 /MASK DIRECTION
SNA CLA
JMP DECTFAM /MUST BE FORWARD
DETRA /MUST BE REVERSE
AND C400  
SZA CLA  
JMP DCTRDR  
JMP DCTCNT  
---/IS REVERSE, GO READ OR WHITE  
---/IS FORWARD, CONT SEARCHING  
DCTFF,  
DTRA  
AND C400  
SNA CLA  
JMP DCTRDR  
JMP DCTCNT  
---/IS FORWARD, GO READ OR WHITE  
---/IS REVERSE, CONT SEARCHING  
LCTFL,  
ISZ DCTAPE  
TAD I DCTAPE  
---/# WORDS  
LCA I DCT-1C  
---/SET WORD COUNT  
ISZ DCTAPE  
TAD I DCTAPE  
---/CORE ADDRESS-1  
LCA I DCTCA  
---/SET CURRENT ADDRESS  
TAD DCTCOD  
LTLB  
---/LOAD FIELD BITS  
TAD DCTCOL  
RAR  
SZL CLA  
---/READ OR WRITE?  
TAD C80  
---/WRITE  
TAD C136  
---/WRITE  
DTXA  
DTSF DTRB  
JMP -1  
ISZ DCTAPE  
---/ADVANCE TO ERROR RETURN  
SRA CLA  
---/SKIP IF ERROR  
ISZ DCTAPE  
TAL DCTCOL  
---/NORMAL RETURN  
---/DIRECT OR INDIRECT?  
HTR  
SNL CLA  
---/DIRECT  
JMP +3  
---/DIRECT, PREPARE  
TAD I DCTAPE  
---/INDIRECT, PREPARE  
LCA DCTAFE  
LTRA  
AND C200  
---/GO BIT  
TAL C2  
---/PRESERVE ERROR FLAG  
DTXA  
---/STOP TAPE  
JMP I DCTAPE  
---/READY, EXIT  
/  
LCTTER,  
DTXA  
DTSF DTRB  
JMP -1  
SPA  
JMP DCTERR  
CLA  
JMP I DCTTRN  
/  
LCTERR,  
RTL  
RAL  
CLA CML  
SNL  
TAD C400  
JMP DCTCNT-1
/ VARIABLES
/ DCTCOD, 0
DCTWC, 7754
LCTCA, 7755
/
/ GENERAL CONSTANTS
/ C2, 2
C10, 12
C20, 20
C130, 130
C230, 230
C460, 460
C7400, 7400
SUBROUTINE TO PACK CHARACTERS (TSS8)
THREE CHARACTERS IN TWO WORDS (TSS8 FORMAT)
PACKER:111111112222
   222233333333
/
CALL :JMS PACK
   ADDRESS INPUTBUFFER
   ADDRESS OUTPUTBUFFER
   RETURN
/
ROUTEUSE AUTO INDEX 10 AND 11
/
FORMAT INPUTBUFFER= 1 CHAR/WRD
LENGTH OUTPUTBUFFER= 200
LENGTH INPUTBUFFER= 300
/
PACK, 0
   TAD PCKBFL
   STL PAF
   LCA PCKCNT
   CLA CMA
   TAD I PACK
   LCA 10
   ISZ PACK
   CMA
   TAD I PACK
   LCA 11
   ISZ PACK
   PCKLOP, TAD I 10
   CLL RTL
   XTL
   LCA PCKTMP
   TAD I 10
   RTR
   RTR
   LCA PCKTP1
   TAD PCKTP1
   AND C17
   TAD PCKTMP
   LCA I 11
   TAD PCKTP1
   BAR
   ANL C7400
   TAD I 10
   DCA I 11
   ISZ PCKCNT
   JMP PCKLOP
   JMP I PACK
/
/
VARIABLES
/
PCKCNT, 0
PCKTMP, 0
PCKTP1, 0
PCKBFL, -200
/
GENERAL CONSTANTS
C17, 17
C7400, 7400

47
/031 SUBROUTINE PACKS CHARACTERS ONE BY ONE (TSS8)
/THREE CHARACTERS IN TWO WORDS (TSS8 FORMAT)
/PACKED: 111111112222
/ 222233333333
/
/CALL  JMS PCKSGL WITH CHAR IN AC
/ ADDRESS OF OUTPUT BUFFER
/ RETURN BUFFER FULL
/ RETURN NORMAL AC=0
/
/INITIALIZE CE PCKSWT: =0
/
/
PCKSGL, 0
ISZ PCKSWT /INITIALIZE?
JMS PCKINI /YES
DCA I PCKRP /NO PUT CHAR IN TEMP BUF
ISZ PCKRP /INCREMENT POINTER
ISZ PCKRCT /3 CHAR'S IN TEMP BUF?
JMP PCKNRM /NO NORMAL EXIT
JMS PCKRST /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
DCA PCKSWT
TAD PCKSWT /TEMP STORAGE
AND C17 /MOST SIGN. 4 BITS
TAD I PCKPTR
DCA I PCKPTR /FIRST WORD
ISZ PCKPTR
TAD PCKSWT
RAR
AND C7400 /LEAST SIGNIFICANT 4 BITS
TAD I PCKRP /GET THIRD CHAR
DCA I PCKPTR /SECOND WORD
ISZ PCKPTR
JMS PCKRST /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
/
/
JMS PCKRES /SET POINTER TEMP BUF
TAD I PCKSGL /GET BUFFER ADDRESS
DCA PCKPTR
TAD PCKBFL
STL BAR
DCA PCKCNT
TAD PCKSWT
JMP I PCKINI

PCKRES, 0
TAD M3
DCA PCKRCT /TEMP BUF IS 3 WORDS
TAD PCKRBA /TEMP BUF ADDRESS
LCA PCKRP
JMP I PCKRES

/ VARIABLES

PCKSWT, 0
PCKPTR, 0
PCKRP, 0
PCKRCT, 0
PCKCNT, 0
PCKRBA, PCKRB
PCKBFL, -400
PCKRB, 0
0
0

/ GENERAL CONSTANTS
M3, -3
C17, 17
C7400, 7400
/032 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
/DSEP = 400
/
DSOUT, 0
DCA DSTM P /TEMP. STORAGE
PAR
DCA DSLNK /SAVE LINK
TAD DSCNTW /FIRST, SECOND OR THIRD CHAR
CLL RTR
SNL SMA CLA
JMP DSRST /FIRST CHAR OF THREE
SNL
JMP DSSEC /SECOND CHAR OF THREE
TAD DSTM P /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 /BUFFERSLENGTH
STL RAR /DIVIDE BY 2
DCA DSCNT
TAD DSLNK /RESTORE LINK
CLL RAL
JMP I DSOUT /EXIT BUFFER FULL
DSSEC, TAD DSTM P
CLL RTL
BSW /BYTE SWAP
AND C77
TAD I DSPTR
DCA I DSPTR
ISZ DSPTR
TAD DSTM P
AND C17
BSW
CLL RTL
DCA I DSPTR
JMP DSEX2
DSRST, TAD DSTM P
CLL RTL
RTL
DCA I DSPTR
DSEX2, ISZ DSCNTW
DSEX3, TAD DSLNK /RESTORE LINK
CLL RAL
ISZ DSOUT
JMP DSOUT /NORMAL EXIT
/
/VARIABLES
/
DSBFL, -400
DSBFA, DSBUF /OUTPUT BUFFER ADDRESS
DSLNK, 0
LSTMP, 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)
/
PACTED: 111111112222
/
222233333333
/CALL : JMS UNPACK
/
ADDRESS OF INPUTBUFFER
/
ADDRESS OF OUTPUTBUFFER
/
RETURN
/
/Routine uses auto-index 10
/

UNPACK, 0

TAD UNPBFL  
STL BAR  
DCA UNPCNT
TAD I UNPACK
DCA UNPFT
ISZ UNPACK
CLA CMA

TAD I UNPACK
DCA 10
ISZ UNPACK

UNPLOP, TAD I UNPFTR

RTL

XTR

AND C377

DCA I 10

TAD I UNPFTR

CLL RAL

RTL

AND C360

DCA UNPTMP

ISZ UNPFTR

TAD I UNPFTR

CLL RAL

RTL

AND C17

TAD UNPTMP

LCA I 10

TAD I UNPFTR

AND C377

LCA I 10

ISZ UNPFTR

ISZ UNPCNT

JMP UNPLOP

JMP I UNPACK

/VARIABLES
/
UNPFTR, 0
UNPTMP, 0
UNPCNT, 0
UNPFL, -400
/
/GENERAL CONSTANTS
C17, 17
C360, 360
C377, 377
/SC34 SUBROUTINE UNPACKS CHARACTERS ONE BY ONE (TSS6) /PACKED THREE CHARACTERS IN TWO WORDS (TSS6 FORMAT) /PACKED: 111111112222 / 222233333333 / /CALL :JMS UNPSGL / ADDRESS INPUTBUFFER / RETURN BUFFER EMPTY AC=0 / NORMAL RETURN AC=CHAR* / /INITIALIZE ONCE UNPRBF:=UNPREF:=UNPCNT:=0 / / UNPSGL, 0 CLA CLL TAD UNPRBF /* ARE THERE CHAR'S IN SZA CLA /* TEMP. BUFFER ? JMP UNPGET /* YES, GET ONE TAD UNPREF /* NO, INPUTBUFFER EMPTY ? SZA CLA JMP UNPEXP /* YES, RETURN END OF BUFFER TAD UNPCNT /* NO OR YES, MUST I SNA CLA /* START UP POINTERS ? JMS UNPINI /* YES, PLEASE DO TAD UNPRA /* NO, JUST UNPACK NEXT WORDS DCA UNPHP TAD I UNPPTR /* NEXT WORD FROM INPUTBUF RTR RTR AND C377 DCA I UNPHP /* FIRST CHAR IN TEMP. BUF ISZ UNPREF TAD I UNPPTR /* GET WORD AGAIN CLL RTL RTL AND C360 DCA I UNPHP /* TEMP. STORAGE ISZ UNPREF TAD I UNPPTR /* NEXT WORD CLL RAL RTL RTL AND C17 TAD I UNPHP DCA I UNPHP /* SECOND CHAR ISZ UNPR /* THAT WORD AGAIN ISZ UNPREF AND C377 DCA I UNPHP /* THIRD CHAR TAD UNPRBA /* RESET POINTER TEMP. BUF DCA UNPHP CLA CLL CMA RTL /*- 3 DCA UNPGET /* 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
DCA UNPREF
TAD I UNPEP
ISZ UNPRP
ISZ UNPSGL
UNPEMT, ISZ UNPSGL
JMP I UNPSGL
/
UNPEMP, DCA UNPREF
JMP UNPEMT
/
UNPINI, 0
DCA UNPREF
TAD I UNPSGL
DCA UNPEPTR
TAD UNPREFL
STL RAR
DCA UNPCNT
JMP I UNPINI
/
/VARIABLES
UNPREFL, 400
UNPCNT, 0
UNPFCT, 0
UNPPEP, 0
UNPPEF, 0
UNPPEF, 0
UNPREP, 0
UNPRES, 0
UNPRESB, UNPRES
UNPRES, 0
/
/GENERAL CONSTANTS
C17, 17
C360, 360
C377, 377
/35 SUBROUTINE TO READ A NAME FROM KEYBOARD
/
/CALL JMS RDNME
/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
/
BS1=7002
BUFA=400
/
/RNAME, 0
TAD RLNDMBF /ADDRESS ASCII BUFFER
DCUA RDPTR
DCUA RDCNT /CHAR. COUNTER
RDIN, JMS READ /READ CHAR FROM KEYB.
DCUA RDCHAR
TAD RDCHAR /RUN OUT ?
TAD RDERO
SNA
JMP RDROS /YES, TO SERVICE
TAD RKROCHN /NO, CARRIAGE RETURN ?
SNA
JMP RDTWR /YES, TO SERVICE
TAD RMLFD /NO, LINE FEED
SNA
JMP RDTWR /YES, SAME SERVICE AS CR
TAD RMSPEL /NO, CHAR$>240 ?
SPA SNA CLA
JMP RDTSNM /NO, ERROR RETURN
TAD RDCHAR /YES, IN BUFFER
DCA I RDPTR
ISZ RDCNT /+# CHAR'S
ISZ RDPTR
JMP RDIN /NEXT CHAR
JMP RDTSNM /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 RDPTR
DCA RDPTR
TAD I RDPTR
JMS PRINT /PRINT REMOVED CHAR
JMP RDIN /END RO-SERVICE
/
RDTSNR, JMS CRLF
TAD RDCNT
SNA
JMP RDTSNM /NAME WITHOUT CHAR'S IS RUBBISH
TAD M6
SMA SZA
CLA
TAD C6
CIA
DCA RDCNT
TAD RDNMPF
DCA RDPTTR
TAD RDNAME
DCA RDTMP
TAD RDTMP
DCA 10
DCA I 10
DCA I 10
RDNXT:
TAD I RDPTTR
TAD C240
AND C77
BSW
DCA I RDTMP
ISZ RDPTTR
ISZ RDCNT
SKP
JMP RDNMOK
TAD I RDPTTR
TAD C240
AND C77
TAD I RDTMP
DCA I RDTMP
ISZ RDTMP
ISZ RDPTTR
ISZ RDCNT
JMP RDNXT

/READY READING NAME
/NOT READY NEXT CHAR

/RDNMOK, ISZ RDNAME
/RDFTRM, ISZ RDNAME
ISZ RDNAME
ISZ RDNAME
JMP I RDNAME
/EXIT
/
/
/VARIABLES
/RDNMBF, BUFADR /ADDRESS BUFFER
RDPTTR, 0
RDCNT, 0
RDCHAR, 2
RDTMP, C
RDCNP, -377
RDCHR, 377-215
RDCHR, 215-218
RDCHR, 218-240
/
/GENERAL CONSTANTS
C6, -6
C6, 6
C77, 77
C240, 240
SUBROUTINE SEARCHES NAME IN DN-BLOCKS (DISKMON.)
(DISK MONITOR SYSTEM)

CALL JMS DNSRC

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

SUBROUTINE USES AUTO INDEX 11 AND MONITOR DISK HANDLER
BUFFER=400

DNSRC, 0

TAD C177   # FIRST DN-BLOCK
JMS DNSRBNK READ BLOCK
TAD I DNSRC
CLA
DCA DNSMNA /
ISZ DNSRC
TAD I DNSRC
CLA
DCA DNSMME /
ISZ DNSRC

DNSPK, CLA CLL IAC RAL +2
TAD DNSBFNA BUFFER ADDRESS
LCA 11
TAD 431   # ENTITIES IN ONE BLOCK
LCA DNSCNT

LNSNXT, TAD I 11 FIRST HALF OF NAME
TAD DNSNNA COMPARE WITH NAME TO LOOK FOR
SQA CLA EQUAL?
JMP DNSNOT NO TRY NEXT NAME
TAD I 11 YES, TEST 2ND. HALF TOO
TAD DNSMME
SYA CLA EQUAL?
JMP DNSNTO NO NEXT NAME
ISZ 11
ISZ 11
TAD I 11
AND C7 MASK OF INT FILE #
ISZ DNSSEC

DNSERR, JMP I DNSRC

DNSNOT, CLA IAC
DNSNTO, TAD C3
TAD 11
DCA 11
ISZ DNSCNT END OF THIS BLOCK?
JMP DNSNXT NO, COMPARE NEXT NAME
TAD DNSLNK YES NEXT BLOCK?
SNA
JMP DNSERR NO NAME NOT FOUND
JMS DNSRBNK READ THAT BLOCK
JMP DNSRLK

DNSRBK, 0
DCA FSTPLK
TAD C3
DCA FUNCTI
TAD DNSBFA
DCA BUFADR
DCA DNSLNK
JMS I SYSIO

/monitor disk handler

FUNCTI, 0
ESTBLK, 0
BUFADR, 0
DNSLNK, 0

HLT
JMP I DNSRBK

/error return

/ VARIABLES
/
SYSIO, 7642
DNSMNA, 0
DNSOME, 0
INSCNT, 0
DNSBFA, 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 
JMS SAMSRC 
JMP I SAMFIL 
CLA CMA 
TAD 10 
DCA 10 
TAD SAMMSK 
TAD M77 
SNA CLA 
JMP SAMRGT 
TAD SAMSAV 
CALL RTL 
RTL 
RTL 
DCA SAMSAV 
JMP +3 
SAMRGT, TAD I 10 
TAD SAMSAV 
DCA SAMSAV 
CMA 
TAD 10 
DCA 10 
TAD SAMSAV 
DCA I 10 
TAD C5 
DCA FUNCTI 
TAD SAMBFA 
DCA BUFADR 
JMS SAMRDB 
ISZ SAMFIL 
TAD SAMBKN 
JMP I SAMFIL 

VARIABLES 
SAMSAV, 0 
GENERAL CONSTANTS 
M77, -77 
C5, 5
/038 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, 0
DCA SAMIFN /INT FILE # TO SEARCH FOR
TAD SAMIFN /MAKE IT TWO IN ONE WORD
CLL RTL
RTL
TAD SAMIFN
LCA SAMIFN
LCA SAMSCK
TAD C2000 /# FIRST SAMBLOCK
SAMREK, LCA BLKRN
TAD C3 /READ FUNCTION
LCA FUNCTI
TAD SAMPFA /BUFFER ADDRESS
DCA RUFADR
JMS SAMSREB /READ BLOCK
SAMSH, TAD C77
LCA SAMSCK /SEARCH RIGHT HALF
TAD M200
LCA SAMCT
CXA
TAD SAMPFA
DCA 10
SKP
SAMNXT, ISZ SAMSCK /COUNT BLOCKNR
TAD 1 10 /GET WORD
AND SAMSCK /MASK
CIA /NEGATIVE
LCA SAMTMP /TEMP. STORAGE
TAD SAMIFN /INT FILE # TO SEARCH FOR
AND SAMSCK /ASK CORRECT HALF
TAD SAMTMP /SAME #?
SMA CLA
JMP SAMFNL /YES, FOUND IT
ISZ SAMCXT /NO, MORE IN THIS HALF?
JMP SAMNXT /YES, SEARCH
ISZ SAMSRN /NO, UPDATE BLOCKNR
TAD SAMSCK /WHERE WERE WE SEARCHING?
AND C7700
SZA CLA /LEFT OR RIGHT HALF
JMP -.+3 /LEFT HALF, BOTH SILES DONE
TAD C7700 /RIGHT HALF, SO DO LEFT NOW
JMP SAMSST+1
TAD SAMLNK /LAST SAMBLOCK?
SNA
JMP SAMNXV /YES, SO NOT FOUND
JMP SAMREK /NO, READ NEXT BLOCK

60
/*
  SAMFND, I Sz SAMSRC
  TAD SAMEKN
  SAMNOT, JMP I SAMSRC
/*
SAMFDR, 0
  JMS I SYSIO /MONITOR DISK HANDLER
FUNCT, 0 /READ=3, WRITE=5
FLKNR, 0 /BLOCKNR
BUFADR, 0 /BUFFER ADDRESS
SAMLNK, 0 /NR NEXT BLOCK, 0=LAST BLOCK
HLT /ERROR RETURN, SYSTEM ERROR
  JMP I SAMFDR
/

/VARIABLES
/
  SAMTP, 0
  SAMIFN, 0
  SAMEKN, 0
  SAMMSK, 0
  SAMCNT, 0
  SAMBFA, BUFFER
  SYSIO, 7642
/

/GENERAL CONSTANTS
/
  C3, 3
  C77, 77
  C200, 200
  C7700, 7700
  M200, -200
/039 SUBROUTINE READS OR WRITES ON DISK (TSS-8)
/
/BEFORE CALLING CALCULATE DISK ADDRESS AND
/PUT IN HIQR AND LOWQR
/
/CALL :JMS DFILE
/FUNCTION (AFILE OR WFILE)
/INTERNAL FILE NUMBER
/-# WORDS
/CORE ADDRESS
/ERROR RETURN
/NORMAL RETURN
/
/FILE MUST BE OPEN !!!!!!!!
/
/
DFILE, 0
TAD I DFILE /GET FUNCTION
DCA DFINST
ISZ DFILE
TAD I DFILE /GET INT. FILE NR
DCA #6BUF+1
ISZ DFILE
DFTRY, TAD I DFILE /-#WORDS
DCA #6BUF+2
ISZ DFILE
CLA CMA
TAD I DFILE /CORE ADDRESS
DCA #6BUF+3
ISZ DFILE
TAD W6AD /ADDRESS 6 WORD BUFFER
DFINST, 0 /DO FUNCTION
TAD W6BUF+5 /ERROR WORD
SNA
JMP DFOKE /NO ERROR
CLL RTR /ERROR
SZL SNA CLA
SKP CLA /ERROR=2
JMP DFERR /ERROR IS NOT 2
IAC
DCA #2BUF+2 /ERROR IS FILE FULL
TAD W2AD /SO MUST EXTEND FILE
EXT /EXTENDING WITH ONE SEGMENT
SZA CLA
JMP DFERR /ERROR: DISK FULL
TAD DFSEG A /ADDRESS LIST SEGMENT COUNTERS
TAD W2BUF /INT FILE NR
DCA #6BUF+2 /TEMP USE
ISZ I #6BUF+2 /INCREMENT COUNTER
CLL CLA CMA RAL /-2
TAD DFILE
DCA DFILE
JMP DFTRY /GO TRY AGAIN NOW
DFOKE, ISZ DFILE
DFERR, JMP I DFILE
/
/VARIABLES
/
DFSEG A, DFSEG 0
DFSEG 0, 0 /# SEGMENTS FILE 0
DFSEG1, 0  # SEGMENTS FILE 1
DFSEG2, 0  # SEGMENTS FILE 2
DFSEG3, 0  # SEGMENTS FILE 3
W6AD, W6BUF
W2AD, W2BUF
W6BUF, H1OR, 0  # HIGH ORDER DISK ADDRESS
W2BUF, 0  # INT FILE NR
0  # WORDS; # SEG'S TO EXT
0  # CORE ADDRESS-1
LOWOR, 0  # LOW ORDER DISK ADDRESS
0  # ERROR WORD