<table>
<thead>
<tr>
<th>DECUS NO.</th>
<th>8-602B</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td>THE PDP-8 COOKBOOK, VOLUME 2</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>Floor Anthoni</td>
</tr>
<tr>
<td>COMPANY</td>
<td>Medical Biological Laboratory TNO Rijswijk, The Netherlands</td>
</tr>
<tr>
<td>DATE</td>
<td>May 1, 1973</td>
</tr>
<tr>
<td>SOURCE LANGUAGE</td>
<td>PAL III, PAL-D, PAL-8</td>
</tr>
</tbody>
</table>

**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.
THIS VOLUME IS ISSUED IN THREE PARTS:

PART 1 CONTAINS A DIRECTORY OF ALL SUBROUTINE NAMES.
A DIRECTORY, SELECTED BY FUNCTIONAL KEYWORDS.
NAMES AND CONTRIBUTIONS OF AUTHORS.
DEFINITIONS OF KEYWORDS IN USE.

PART 2 CONTAINS REVISIONS TO SUBROUTINES IN VOLUME #1.

PART 3 CONTAINS 44 NEW PROGRAM MODULES.
AUTHORS OF THE PDP8 COOKBOOK AND THEIR CONTRIBUTIONS

============================================

FLOOR ANTHONI
MEDICAL BIOLOGICAL LABORATORY T. N. O.
LANGE KLEIWEG 139; RIJSWIJK (ZH); THE NETHERLANDS
1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72

ANDRIES E. BROUWER
MATHEMATISCH CENTRUM
22 GE BOERHAAVESTRAAT 49; AMSTERDAM
80, 81, 13, 12B

THIERRI DEN DUNNEN
DR. NEHER LABORATORIUM
ST. PAULUSSTRAAT 4; LEIDSCHENDAM; THE NETHERLANDS
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 73, 74, 75, 76, 77, 78, 79, 13, 12A, 10

ADRI HEMELAAR
MEDICAL BIOLOGICAL LABORATORY T. N. O.
LANGE KLEIWEG 139; RIJSWIJK (ZH); THE NETHERLANDS
45, 46, 47, 48, 82, 83

PAUL LOHMAN
MEDICAL BIOLOGICAL LABORATORY T. N. O.
LANGE KLEIWEG 139; RIJSWIJK (ZH); THE NETHERLANDS
16

HANS MEES
MEDICAL BIOLOGICAL LABORATORY T. N. O.
LANGE KLEIWEG 139; RIJSWIJK (ZH); THE NETHERLANDS
8

C. VERWEY
LABORATORIUM VOOR BIOLOGISCHE EN MEDISCHE NATUURKUNDE
MEDISCHE FACULTEIT ERASMUS UNIVERSITEIT ROTTERDAM
P. O. BOX 1738; ROTTERDAM; THE NETHERLANDS
40, 41, 42, 43, 44
FUNCTIONS

ARITHMETIC = ALL TYPES OF OPERATIONS OF AN ARITHMETIC NATURE
(+,-,*, etc. but also BOOLEAN ALGEBRA)

STRINGS = USED TO DENOTE ALL TYPES OF OPERATIONS
RELATED TO THE ORGANISATION AND STRUCTURE OF
DATA. (ALSO: BUFFERS, CONVENTIONAL/STRINGS, ETC.)

FILE = OPERATIONS THAT ORGANIZE A DATA STRUCTURE ON A
LARGE STORAGE, BACKING UP DEVICE.

SORTING = USED TO DENOTE THE TYPE OF OPERATIONS WHERE THE
COMPARISON OF DATA WITH OTHER DATA IS ESSENTIAL.

CONVERSION = CONVERSION FROM ONE DATA STRUCTURE OR FORMAT TO
ANOTHER.

PROGRAM CONTROL = DENOTES OPERATIONS THAT ARE USED TO DIRECT THE FLOW
OF THE PROGRAM IN THE FIRST INSTANCE.

INPUT = OPERATIONS THAT INPUT DATA INTO THE COMPUTER.

OUTPUT = OPERATIONS THAT PUT DATA OUT FROM THE COMPUTER.

interrupt = DENOTES OPERATIONS WHERE THE INTERRUPT IS USED.

GRAPHICS = OPERATIONS ON GRAPHICAL DATA, AND DISPLAY

SIGNAL PROCESSING = OPERATIONS OF A MATHEMATICAL NATURE ON DATA THAT
REPRESENT SOME SIGNAL.

DATA FORMATS

BINARY = THE CONVENTIONAL 'BINARY' FORMAT AS IT IS USED
BY LOADERS.

ASCII = 8 BIT DATA, REPRESENTING ONE CHARACTER FOLLOWING
THE USASCII STANDARD.

CODE = A CHARACTER REPRESENTATION FOLLOWING SOME
OTHER STANDARD.

OCTAL = ASCII, RESTRICTED TO OCTAL VALUES

DECIMAL = ASCII, RESTRICTED TO DECIMAL VALUES

HEXADECIMAL = ASCII, RESTRICTED TO HEXADECIMAL VALUES.

IMAGE = THE 12 BIT WORD AND SEQUENCES OF PDP8 COMPUTER WORDS.

SYSTEM SOFTWARE RESTRICTIONS

DISKMONITOR = CONFINED FOR DISK MONITOR USERS

OS8 = CONFINED FOR OS8 USERS

TSS8 = CONFINED FOR TSS8 TIMESHARING USERS

DEVICE TYPE RESTRICTIONS

DISK = DF32, RF08, RK08 ETC.

DECTAPE = DIGITAL EQUIPMENT'S MAGNETIC TAPE UNITS

MAGTAPE = INDUSTRIAL AND OTHER MAGNETIC TAPE DEVICES

DISPLAY = ALL TYPES OF DISPLAYS, CHARACTER- AND VECTOR TYPES

PAPERTAPE = PAPERTAPE READERS AND PUNCHES.
001 TYPE THE CHARACTERS FOLLOWING THE JMS INSTRUCTION.
K: STRINGS, OUTPUT, ASCII

002 TELETYPE TYPE ROUTINE WITH OVERLAP; NOT RESTARTABLE.
K: OUTPUT, ASCII, BINARY

003 TYPE A CHARACTER CHAIN
K: OUTPUT, ASCII

004 DECIMAL TO DECIMAL CONVERSION AND TYPE
K: OUTPUT, CONVERSION, BINARY, IMAGE

005 BINARY TO OCTAL CONVERSION AND PRINT
K: OUTPUT, CONVERSION, OCTAL

006 HIGH SPEED READER SUBROUTINE
K: OUTPUT, PAPER TAPE, ASCII, BINARY

007 TABULATOR ROUTINE
K: STRINGS, ASCII, OUTPUT

008 SUBROUTINE TO MOVE A BLOCK THROUGH CORE
K: SORTING, STRINGS, IMAGE

009 BINARY PUNCH WITH FIELD SETTING
K: CONVERSION, IMAGE, BINARY, OUTPUT

010 PAL MESSAGE PRINTER
K: OUTPUT, ASCII, CONVERSION,

011 GENERAL BRANCH ROUTINE
K: PROGRAM CONTROL,

012 CHECK IF OCTAL
K: PROGRAM CONTROL, OCTAL, ASCII

013 LOGICAL OPERATORS ON TWO NUMBERS
K: ARITHMETIC

014 Ps8-0s8 OPTION DECODER
K: PROGRAM CONTROL, OSS

015 PRINT TWO DIGITS IN DECIMAL
K: OUTPUT, DECIMAL, CONVERSION

016 PRINT THE Ps8-0s8 DATE
K: OUTPUT, OSS, DECIMAL

017 PRINT THE AC AS A FOCAL LINENUMBER
K: OUTPUT, CONVERSION, DECIMAL

018 PRINT 4 DECIMAL DIGITS USING ROUTINE PRNT2
K: OUTPUT, CONVERSION, DECIMAL

019 SUBROUTINE READS A DECIMAL NUMBER FROM KEYBOARD
K: INPUT, CONVERSION, DECIMAL
020 DECIMAL PRINT ROUTINE
K: OUTPUT, CONVERSION, DECIMAL

021 SUBROUTINE TO PRINT DOUBLE LENGTH DECIMAL
K: OUTPUT, CONVERSION, DECIMAL

022 OCTAL PRINT ROUTINE
K: OUTPUT, CONVERSION, OCTAL

023 DOUBLE WORD OCTAL PRINT ROUTINE, USES OCTPRT
K: OUTPUT, CONVERSION, OCTAL

024 SUBROUTINE TRANSLATES TELEX CODE TO ASCII
K: CONVERSION, TELEX, ASCII

025 SUBROUTINE TO TRANSLATE TELEX CODE TO ASCII
K: CONVERSION, ASCII, CODE

026 ROUTINE TO TRANSLATE ASCII TO TELEX CODE
K: CONVERSION, ASCII, CODE

027 INTERRUPT OUTPUT HANDLER WITH ROTATING BUFFER
K: INTERRUPT, STRINGS, OUTPUT, ASCII

028 DEVICE INTERRUPT HANDLER
K: INTERRUPT, STRINGS, OUTPUT, ASCII

029 SUBROUTINE READS OR WRITES DECTAPE IN 2 DIRECTIONS
K: INPUT, OUTPUT, DECTAPE, IMAGE, FILE

030 SUBROUTINE TO PACK CHARACTERS (TSS8)
K: CONVERSION, STRINGS, ASCII, TSS8

031 SUBROUTINE PACKS CHARACTERS ONE BY ONE (TSS8 FORMAT)
K: CONVERSION, STRINGS, ASCII, TSS8

032 SUBROUTINE TO PACK CHARACTERS ONE BY ONE (TSS8 FORMAT)
K: CONVERSION, STRINGS, ASCII, TSS8

033 SUBROUTINE TO UNPACK CHARACTERS (TSS8 FORMAT)
K: CONVERSION, ASCII, TSS8

034 SUBROUTINE UNPACKS CHARACTERS ONE BY ONE (TSS8 FORMAT)
K: CONVERSION, STRINGS, ASCII, TSS8

035 SUBROUTINE TO READ A NAME FROM KEYBOARD (EXCESS 40 CODE)
K: INPUT, CONVERSION, ASCII, STRINGS

036 SUBROUTINE SEARCHES NAME IN DN-BLOCKS (DISKMONITOR)
K: FILE, STRINGS, DISKMONITOR, DECTAPE, DISK

037 SUBROUTINE SEARCHES UNUSED BLOCK ON DISK (DISKMONITOR)
K: FILE, STRINGS, DISKMONITOR, DECTAPE, DISK

038 SUBROUTINE SEARCHES INTERNAL FILE NUMBER (DISKMONITOR)
K: FILE, STRINGS, DISKMONITOR, DECTAPE, DISK
039 SUBROUTINE READS OR WRITES ON DISK (TSS8)
K: FILE, INPUT, OUTPUT, DISK, TSS8

040 SUBROUTINE TO DISPLAY A BLOCK OF DATA ON VC8E
K: CONVERSION, OUTPUT, DISPLAY, STRINGS

041 SUBROUTINE TO DISPLAY INPUT FROM AN ANALOG CHANNEL
K: OUTPUT, CONVERSION, DISPLAY,

042 SUBROUTINE TO INPUT A BLOCK OF DIGITAL DATA (DR8E)
K: INPUT, STRINGS

043 SUBROUTINE TO SMOOTH A BLOCK OF DATA IN MEMORY
K: STRINGS, ARITHMETIC, SIGNAL, PROCESSING

044 SUBROUTINE TO REDISTRIBUTED A BLOCK OF DATA
K: STRINGS, ARITHMETIC, SIGNAL, PROCESSING

045 PACK A CHARACTER IN A BUFFER IN OSS FORMAT
K: CONVERSION, STRINGS, ASCII, OSS

046 UNPACK A CHARACTER FROM A BUFFER IN OSS FORMAT
K: CONVERSION, STRINGS, ASCII, OSS

047 PARITY GENERATOR
K: CONVERSION, ASCII

048 SKIP ON FLAG WITH TIMED OUT RETURN
K: PROGRAM CONTROL, INPUT, OUTPUT

049 SEARCH A LIST FOR A MATCH
K: STRINGS, PROGRAM CONTROL

050 LIST SEARCH, CROSS-FIELD CALLABLE
K: PROGRAM CONTROL, STRINGS

051 RELATIVE BRANCHER, CROSS-FIELD CALLABLE
K: PROGRAM CONTROL,

052 GENERAL SETUP FOR OSS HANDLERS
K: INPUT, OUTPUT, OSS, ASCII, BINARY

053 UNPACK CHAR-BY-CHAR FOR OSS HANDLERS
K: INPUT, STRINGS, OUTPUT, CONVERSION, ASCII, BINARY, OSS

054 UNPACK CHAR-BY-CHAR FOR OSS HANDLERS
K: INPUT, STRINGS, OUTPUT, CONVERSION, ASCII, BINARY, OSS

055 UNPACK AND PRINT OSS BUFFER (DEVICE HANDLER)
K: OUTPUT, STRINGS, CONVERSION, ASCII, BINARY, OSS

056 PUSH AND POP OPERATORS FOR DIFFERENT STACKS
K: STRINGS, PROGRAM CONTROL

057 FILL A ROTATING BUFFER QUEUE (FIRST IN FIRST OUT)
K: STRINGS

058 RESET (CLEAR) A ROTATING BUFFER QUEUE
K: STRINGS
059 FETCH THE NEXT ITEM FROM THE HEAD OF THE QUEUE
K: STRINGS

060 COMBINED ROTATING BUFFER OPERATORS, CROSS-FIELD CALLABLE
K: STRINGS

061 BINARY LOADER SUBROUTINE
K: INPUT, CONVERSION, BINARY, IMAGE

062 INTERRUPT SERVICE BY LIST-LOOK-UP
K: INTERRUPT, PROGRAM CONTROL, INPUT, OUTPUT

063 FIND THE SMALLEST HOLE IN A LIST, JUST BIG ENOUGH
K: STRINGS, SORTING

064 LINK FOR RELOCATABLE CROSS-PAGE REFERENCING IN ONE FIELD
K: PROGRAM CONTROL

065 INITIALIZE THE FREECORE AREA
K: STRINGS

066 REQUEST A FREE BLOCK
K: STRINGS

067 RELEASE A BLOCK TO FREECORE
K: STRINGS

068 RELEASE A QUEUE OF FORWARD LINKED BLOCKLETS TO FREECORE
K: STRINGS

069 MAKE A BUFFER OR 'QUEUE'
K: STRINGS

070 READ NEXT ELEMENT FROM THE TAIL OF A ROTATING BUFFER (QUEUE)
K: STRINGS

071 WRITE AN ELEMENT ONTO THE TOP OF A BUFFER QUEUE
K: STRINGS

072 KILL THE BUFFER QUEUE
K: STRINGS

073 ROUTINE TO CONVERT OCTAL NUMBERS 0-17 TO EXCESS-40 STRIPPED CODE
K: CONVERSION, OCTAL, CODE

074 INTERRUPT ROUTINE FOR REAL-TIME CLOCK
K: INTERRUPT, CONVERSION

075 PRINT DATE, WEEKDAY AND TIME FROM THE REGISTERS.
K: CONVERSION, DECIMAL

076 SUBROUTINE TO UNPACK TSS8 TIME
K: CONVERSION, DECIMAL, TSS8

077 SUBROUTINE TO UNPACK TSS8 TIME
K: CONVERSION, DECIMAL, TSS8
078 SUBROUTINE TO UNPACK TSS8 DATE.
K: CONVERSION, DECIMAL, TSS8

079 DECIMAL PRINT WITH VARIABLE NUMBER OF DIGITS.
K: OUTPUT, CONVERSION, DECIMAL

080 OCTAL PRINT WITH LEADING SPACES.
K: OUTPUT, CONVERSION, OCTAL

081 DOUBLE WORD OCTAL PRINT ROUTINE
K: OUTPUT, CONVERSION, OCTAL

082 ASCII STRING GENERATOR
K: OUTPUT, ASCII, STRINGS

083 INCREMENT DOUBLE PRECISION COUNTER.
K: ARITHMETIC
ARITHMETIC.

013 LOGICAL OPERATORS ON TWO NUMBERS
043 SUBROUTINE TO SMOOTH A BLOCK OF DATA IN MEMORY
044 SUBROUTINE TO REDISTRIBUTE A BLOCK OF DATA
083 INCREMENT DOUBLE PRECISION COUNTER.

CONVERSION.

004 DECIMAL TO DECIMAL CONVERSION AND TYPE
005 BINARY TO OCTAL CONVERSION AND PRINT
009 BINARY PUNCH WITH FIELD SETTING
010 PAL MESSAGE PRINTER
015 PRINT TWO DIGITS IN DECIMAL
017 PRINT THE AC AS A FOCAL LINENUMBER
018 PRINT 4 DECIMAL DIGITS USING ROUTINE PRNT2
019 SUBROUTINE READS A DECIMAL NUMBER FROM KEYBOARD
020 DECIMAL PRINT ROUTINE
021 SUBROUTINE TO PRINT DOUBLE LENGTH DECIMAL
022 OCTAL PRINT ROUTINE
023 DOUBLE WORD OCTAL PRINT ROUTINE, USES OCTPRT
024 SUBROUTINE TRANSLATES TELEX CODE TO ASCII
025 SUBROUTINE TO TRANSLATE TELEX CODE TO ASCII
026 ROUTINE TO TRANSLATE ASCII TO TELEX CODE
030 SUBROUTINE TO PACK CHARACTERS (TSS8)
031 SUBROUTINE PACKS CHARACTERS ONE BY ONE (TSS8 FORMAT)
032 SUBROUTINE TO PACK CHARACTERS ONE BY ONE (TSS8 FORMAT)
033 SUBROUTINE TO UNPACK CHARACTERS (TSS8 FORMAT)
034 SUBROUTINE UNPACKS CHARACTERS ONE BY ONE (TSS8 FORMAT)
035 SUBROUTINE TO READ A NAME FROM KEYBOARD (EXCESS 40 CODE)
040 SUBROUTINE TO DISPLAY A BLOCK OF DATA ON VC8E
041 SUBROUTINE TO DISPLAY INPUT FROM AN ANALOG CHANNEL
045 PACK A CHARACTER IN A BUFFER IN O88 FORMAT
046 UNPACK A CHARACTER FROM A BUFFER IN O88 FORMAT
047 PARITY GENERATOR
053 UNPACK CHAR-BY-CHAR FOR O88 HANDLERS
054 UNPACK CHAR-BY-CHAR FOR O88 HANDLERS
055 UNPACK AND PRINT O88 BUFFER (DEVICE HANDLER)
061 BINARY LOADER SUBROUTINE
073 ROUTINE TO CONVERT OCTAL NUMBERS 0-17 TO EXCESS-40 STRIPPED CODE
074 INTERRUPT ROUTINE FOR REAL-TIME CLOCK
075 PRINT DATE, WEEKDAY AND TIME FROM THE REGISTERS.
076 SUBROUTINE TO UNPACK TSS8 TIME
077 SUBROUTINE TO UNPACK TSS8 TIME
078 SUBROUTINE TO UNPACK TSS8 DATE.
079 DECIMAL PRINT WITH VARIABLE NUMBER OF DIGITS.
080 OCTAL PRINT WITH LEADING SPACES.
081 DOUBLE WORD OCTAL PRINT ROUTINE
FILE,

029 SUBROUTINE READS OR WRITES DECTAPE IN 2 DIRECTIONS
036 SUBROUTINE SEARCHES NAME IN DN-BLOCKS (DISKMONITOR)
037 SUBROUTINE SEARCHES UNUSED BLOCK ON DISK (DISKMONITOR)
038 SUBROUTINE SEARCHES INTERNAL FILE NUMBER (DISKMONITOR)
039 SUBROUTINE READS OR WRITES ON DISK (TSS8)

INPUT,

019 SUBROUTINE READS A DECIMAL NUMBER FROM KEYBOARD
029 SUBROUTINE READS OR WRITES DECTAPE IN 2 DIRECTIONS
035 SUBROUTINE TO READ A NAME FROM KEYBOARD (EXCESS 40 CODE)
039 SUBROUTINE READS OR WRITES ON DISK (TSS8)
041 SUBROUTINE TO DISPLAY INPUT FROM AN ANALOG CHANNEL
042 SUBROUTINE TO INPUT A BLOCK OF DIGITAL DATA (DR8E)
048 SKIP ON FLAG WITH TIMED OUT RETURN
052 GENERAL SETUP FOR OS8 HANDLERS
053 UNPACK CHAR-BY-CHAR FOR OS8 HANDLERS
054 UNPACK CHAR-BY-CHAR FOR OS8 HANDLERS
061 BINARY LOADER SUBROUTINE
062 INTERRUPT SERVICE BY LIST-LOOK-UP

INTERRUPT,

027 INTERRUPT OUTPUT HANDLER WITH ROTATING BUFFER
028 DEVICE INTERRUPT HANDLER
062 INTERRUPT SERVICE BY LIST-LOOK-UP
074 INTERRUPT ROUTINE FOR REAL-TIME CLOCK

SIGNAL PROCESSING,

043 SUBROUTINE TO SMOOTH A BLOCK OF DATA IN MEMORY
044 SUBROUTINE TO REDISTRIBUTE A BLOCK OF DATA

SORTING,

008 SUBROUTINE TO MOVE A BLOCK THROUGH CORE
063 FIND THE SMALLEST HOLE IN A LIST, JUST BIG ENOUGH
OUTPUT,

082 ASCII STRING GENERATOR
001 TYPE THE CHARACTERS FOLLOWING THE JMS INSTRUCTION.
002 TELETYPewriter TYPE ROUTINE WITH OVERLAP; NOT RESTARTABLE.
003 TYPE A CHARACTER CHAIN
004 DECIMAL TO DECIMAL CONVERSION AND TYPE
005 BINARY TO OCTAL CONVERSION AND PRINT
006 HIGH SPEED READER SUBROUTINE
007 TABULATOR ROUTINE
009 BINARY PUNCH WITH FIELD SETTING
010 PAL MESSAGE PRINTER
015 PRINT TWO DIGITS IN DECIMAL
016 PRINT THE PS8-OS8 DATE
017 PRINT THE AC AS A FOCAL LINENUMBER
018 PRINT 4 DECIMAL DIGITS USING ROUTINE PRNT2
020 DECIMAL PRINT ROUTINE
021 SUBROUTINE TO PRINT DOUBLE LENGTH DECIMAL
022 OCTAL PRINT ROUTINE
023 DOUBLE WORD OCTAL PRINT ROUTINE, USES OCTPRT
027 INTERRUPT OUTPUT HANDLER WITH ROTATING BUFFER
028 DEVICE INTERRUPT HANDLER
029 SUBROUTINE READS OR WRITES DECTAPE IN 2 DIRECTIONS
030 SUBROUTINE READS OR WRITES ON DISK (TSS8)
040 SUBROUTINE TO DISPLAY A BLOCK OF DATA ON VC8E
041 SUBROUTINE TO DISPLAY INPUT FROM AN ANALOG CHANNEL
048 SKIP ON FLAG WITH TIMED OUT RETURN
052 GENERAL SETUP FOR OS8 HANDLERS
053 UNPACK CHAR-BY-CHAR FOR OS8 HANDLERS
054 UNPACK CHAR-BY-CHAR FOR OS8 HANDLERS
055 UNPACK AND PRINT OS8 BUFFER (DEVICE HANDLER)
062 INTERRUPT SERVICE BY LIST-LOOK-UP
079 DECIMAL PRINT WITH VARIABLE NUMBER OF DIGITS.
080 OCTAL PRINT WITH LEADING SPACES.
081 DOUBLE WORD OCTAL PRINT ROUTINE
082 ASCII STRING GENERATOR

PROGRAM CONTROL,

011 GENERAL BRANCH ROUTINE
012 CHECK IF OCTAL
014 PS8-OS8 OPTION DECODER
048 SKIP ON FLAG WITH TIMED OUT RETURN
049 SEARCH A LIST FOR A MATCH
050 LIST SEARCH, CROSS-FIELD CALLABLE
051 RELATIVE BRANCHER, CROSS-FIELD CALLABLE
056 PUSH AND POP OPERATORS FOR DIFFERENT STACKS
062 INTERRUPT SERVICE BY LIST-LOOK-UP
064 LINK FOR RELOCATABLE CROSS-PAGE REFERENCING IN ONE FIELD
082 ASCII STRING GENERATOR
081 TYPE THE CHARACTERS FOLLOWING THE HMS INSTRUCTION.
080 TABULATOR ROUTINE
079 SUBROUTINE TO MOVE A BLOCK THROUGH CORE
078 INTERRUPT OUTPUT HANDLER WITH ROTATING BUFFER
077 DEVICE INTERRUPT HANDLER
076 SUBROUTINE TO PACK CHARACTERS (TSS8)
075 SUBROUTINE PACKS CHARACTERS ONE BY ONE (TSS8 FORMAT)
074 SUBROUTINE TO PACK CHARACTERS ONE BY ONE (TSS8 FORMAT)
073 SUBROUTINE TO UNPACK CHARACTERS (TSS8 FORMAT)
072 SUBROUTINE UNPACKS CHARACTERS ONE BY ONE (TSS8 FORMAT)
071 SUBROUTINE TO READ A Name FROM KEYBOARD (EXCESS 40 CODE)
070 SUBROUTINE SEARCHES NAME IN Dn-BLOCKS (DISKMONITOR)
069 SUBROUTINE SEARCHES UNUSED BLOCK ON DISK (DISKMONITOR)
068 SUBROUTINE SEARCHES INTERNAL FILE NUMBER (DISKMONITOR)
067 SUBROUTINE TO DISPLAY A BLOCK OF DATA ON VCSE
066 SUBROUTINE TO INPUT A BLOCK OF DIGITAL DATA (DRSE)
065 SUBROUTINE TO SMOOTH A BLOCK OF DATA IN MEMORY
064 SUBROUTINE TO REDISTRIBUTE A BLOCK OF DATA
063 PACK A CHARACTER IN A BUFFER IN OS8 FORMAT
062 UNPACK A CHARACTER FROM A BUFFER IN OS8 FORMAT
061 SEARCH A LIST FOR A MATCH
060 LIST SEARCH, CROSS-FIELD CALLABLE
059 UNPACK CHAR-BY-CHAR FOR OS8 HANDLERS
058 UNPACK CHAR-BY-CHAR FOR OS8 HANDLERS
057 UNPACK AND PRINT OS8 BUFFER (DEVICE HANDLER)
056 PUSH AND POP OPERATORS FOR DIFFERENT STACKS
055 FILL A ROTATING BUFFER QUEUE (FIRST IN FIRST OUT)
054 RESET (CLEAR) A ROTATING BUFFER QUEUE
053 FETCH THE NEXT ITEM FROM THE HEAD OF THE QUEUE
052 COMBINED ROTATING BUFFER OPERATORS, CROSS-FIELD CALLABLE
051 FIND THE SMALLEST HOLE IN A LIST, JUST BIG ENOUGH
050 INITIALIZE THE FREECORE AREA
049 REQUEST A FREE BLOCK
048 RELEASE A BLOCK TO FREECORE
047 RELEASE A QUEUE OF FORWARD LINKED BLOCKLETS TO FREECORE
046 MAKE A BUFFER OR 'QUEUE'
045 READ NEXT ELEMENT FROM THE TAIL OF A ROTATING BUFFER (QUEUE)
044 WRITE AN ELEMENT ONTO THE TOP OF A BUFFER QUEUE
043 KILL THE BUFFER QUEUE
042 ASCII STRING GENERATOR
REVISIONS TO THE PDP8 COOKBOOK VOLUME 1

*************************

010 CAN BE IMPROVED BY CHANGING THE FIRST INSTRUCTION
‘CMA’ INTO ‘CLA CMA’.
THIS ROUTINE IS, HOWEVER, SUPERCEDED BY 010A.

*************************

013 EXCLUSIVE OR ROUTINE CAN BE IMPROVED:

TAD A
AND B
CMA IAC
CLL RAL
TAD A
TAD B

OR AS FOLLOWS:

TAD A
AND B
CLL RAL
CIA
TAD A
TAD B

THE NOR ROUTINE CAN BE IMPROVED:

TAD A
CMA
AND B
TAD A
CMA
/010A PAL MESSAGE PRINTER (REPLACES # 010)
/PRINTS A MESSAGE CODED WITH THE
/PALD OR PAL8 PSEUDO-OP "TEXT".
/
PALD AND PAL8 COMPATIBLE EXCEPT CAR RET & LINEFEED.
/
/CALL: JMS PRMESG
/   MESS
/   RETURN (AC=0)
/
/MESS, TEXT 'ABC82' /CODED AS 0102;0370;6200
/
PRMESG,0
CLA CMA
TAD I PRMESG /SAVE POINTER
DCA PRMES3
ISZ PRMESG /FOR RETURN
PRMES1,CLA CMA
DCA PRMES4 /UNPACKSWITCH
ISZ PRMES3 /NEXT WORD
TAD I PRMES3 /FETCH WORD
RTR /*
RTR /MAY BE CODED 'BSW' FOR 8E
RTR /*
PRMES2,AND C77 /MASK 6 BITS
SNA /END OF LIST?
JMP I PRMESG /YES
TAD C240 /RECODE
AND C77 /CHARACTER TO
TAD C240 /BE PRINTED
JMS PRINT /AND PRINT
ISZ PRMES4 /TEST L-R Switch
JMP PRMES1 /LEFT
TAD I PRMES3 /RIGHT
JMP PRMES2
/
PRMES3,0 /POINTER
PRMES4,0 /UNPACKSWITCH 0=R; 1=L
/
/GENERAL CONSTANTS
C77, 77
C240, 240
/012A CHECK IF OCTAL DIGIT (REPLACES # 012)
/Routine checks whether the AC is an octal digit
/
/TAD character
/JMS OCTCHK
/NOT OCTAL RETURN
/OCTAL RETURN
/
OCTCHK, 0
   TAD M260
   AND C7770        /OR "AND M10"
   SNA CLA
   ISZ OCTCHK       /RETURN FOR OCTAL DIGIT
   JMP I OCTCHK
/
/GENERAL constants
M260,  -260
C7770,  7770

/012B CHECK WHETHER OCTAL OR NOT
/Routine can be more practical if 'SZL CLA' becomes
/'SZL' or 'SNL' in which case the 'OCTALRETURN'
/IS with that digit in the AC. The other return is
/with AC=-260

   10
  -270
OCTCHK, 0
   TAD OCTCHK-1
   CLL
   TAD OCTCHK-2
   SZL CLA
   ISZ OCTCHK
   JMP I OCTCHK
/019A REV. 9 FEB 1973

/ SUBROUTINE READS A DECIMAL NUMBER FROM KEYBD
/ 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, CHAR IS DELIMITER
TAD DECNUM /YES, CHAR IS FIGURE
CLL RTL /NUMB*4
TAD DECNUM /NUMB*4+1=NUMB*5
CLL RAL /NUMB*5*2=NUMB*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
/
/
/GENERAL CONSTANTS
M12, -12
M260, -260
M377, -377

16
SUBROUTINE TO DISPLAY A BLOCK OF DATA
DISPLAY CONTROL: VC8-E
CALLING SEQUENCE:

IR1=10

0
7000
0

DISPLA, 0
TAD I DISPLA /GET START ADDRESS
DCA IR1
ISZ DISPLA
TAD I DISPLA /GET NO. OF WORDS
DCA DISPLA-1
TAD DISPLA-2 /GET LEFT POINT ON DISPLAY
DCA DISPLA-3

DISNEX, TAD DISPLA-3 /GET X-POINT

DILX
CLA
TAD I IR1 /GET DATA
DILY
CLA
DISD /WAIT FOR SETTILING
JMP -.1

DIXY
ISZ DISPLA-3 /INCREMENT X-POINT
NOP
ISZ DISPLA-1 /INCREMENT COUNT
JMP DISNEX
ISZ DISPLA /SET RETURN ADDRESS
JMP I DISPLA /RETURN
/SUBROUTINE TO DISPLAY INPUT FROM AN ANALOG CHANNEL
/INPUT: AD8-E +AM8-E, DISPLAY CONTROL: VC8E
/CALLING SEQUENCE:
/ JMS MULDIS
/ MULTIPLEXER CHANNEL
/ RETURN /AC=0

0
0
-1777
7000

MULDIS, 0
TAD I MULDIS /GET CHANNEL
ADLM
TAD MULDIS-1 /GET LEFT POINT ON DISPLAY
DCA MULDIS-4
TAD MULDIS-2 /GET MINUS NO. OF POINTS
DCA MULDIS-3

MULNEX, TAD MULDIS-4 /GET X-POINT
DILX
CLA
JMS MULSAM /GET SAMPLE
DILY
CLA
DISD /WAIT FOR SETTLING
JMP -1
DIXY
ISZ MULDIS-4 /INCREMENT X-POINT
NOP
ISZ MULDIS-3 /INCREMENT COUNT
JMP MULNEX /GO TO NEXT POINT
ISZ MULDIS /SET RETURN ADDRESS
JMP I MULDIS /RETURN

MULSAM, 0
ADST /START CONVERSION
ADSK
JMP -1 /WAIT FOR CONVERSION DONE
ADRB /READ AD BUFFER
JMP I MULSAM
SUBROUTINE TO INPUT A BLOCK OF DIGITAL DATA

INPUT VIA THE DIGITAL I/O DRB-E

CALLING SEQUENCE:

JMS DIGIN

START ADDRESS OF MEMORY BLOCK MINUS 1

MINUS NO OF WORDS

RETURN /AC=0

START DATA BLOCK: BIT 10

WORD COMMAND: BIT 11

END DATA BLOCK: BIT 9

IR1=10

DIGIN, 0

TAD I DIGIN /GET START ADDRESS
DCA IR1
ISZ DIGIN
TAD I DIGIN /GET NO OF WORDS
DCA DIGIN-1
DBCO /CLEAR OUTPUT REGISTER
DBCI /CLEAR INPUT REGISTER
CLA CLL IAC RAL /START DATA BLOCK
DBSO /SET OUTPUT REGISTER
DBCO /CLEAR OUTPUT REGISTER
CLA CLL IAC /WORD COMMAND PULSE
DBSO
DBCO
CLA CLL
DBSK /WAIT FOR FLAG
JMP -1
DBHI /READ INPUT REGISTER
DBCI /CLEAR INPUT REGISTER
DCA I IR1
ISZ DIGIN-1 /INCREMENT COUNTER
JMP -12 /GO TO NEXT WORD
CLA CLL IAC RTL /END DATA BLOCK
DBSO
DBCO
CLA CLL
ISZ DIGIN /SET RETURN ADDRESS
JMP I DIGIN /RETURN
SUBROUTINE TO SMOOTH A BLOCK OF DATA IN MEMORY
DATA WILL BE SMOOTHED WITH THE FILTER:
A(I) = 1/4*A(I-1) + 1/2*A(I) + 1/4*A(I+1)
CALLING SEQUENCE:
JMS FILTER
START ADDRESS OF MEMORY BLOCK
MINUS NO OF POINTS
RETURN ICAC=0

FILTER, 0
TAD I FILTER /GET START ADDRESS
DCA FILTER-1 /INPUT POINTER
ISZ FILTER
TAD I FILTER /GET MINUS NO. OF POINTS
DCA FILTER-2 /POINT COUNT
TAD FILTER-1
DCA FILTER-3 /OUTPUT POINTER
TAD I FILTER-1 /GET FIRST POINT
DCA FILTER-5 /SAVE IT IN OUT-STORE
FILNEX, TAD I FILTER-1 /GET FIRST POINT
JMS FILHAL
JMS FILHAL /DIVIDE BY 4
DCA FILTER-4 /SAVE IT IN IN-STORE
ISZ FILTER-1 /INCREMENT INPUT POINTER
TAD I FILTER-1 /GET SECOND POINT
JMS FILHAL /DIVIDE BY 2
TAD FILTER-4 /ADD AND
DCA FILTER-4 /SAVE IT IN IN-STORE
ISZ FILTER-1 /INCREMENT INPUT POINTER
TAD I FILTER-1 /GET THIRD POINT
JMS FILHAL
JMS FILHAL /DIVIDE BY 4
TAD FILTER-4 /ADD AND
DCA FILTER-4 /SAVE IT IN IN-STORE
TAD FILTER-5 /GET PREVIOUS RESULT
DCA I FILTER-3 /PUT IT BACK IN MEMORY
TAD FILTER-4 /GET CURRENT RESULT
DCA FILTER-5 /SAVE IT FOR NEXT OUTPUT
ISZ FILTER-3 /INCREMENT OUTPUT POINTER
TAD FILTER-1 /GET INPUT POINTER
CIA /AND
CMA /SUBTRACT ONE
DCA FILTER-1
ISZ FILTER-2 /INCREMENT POINT COUNT
JMP FILNEX /GO TO NEXT POINT
ISZ FILTER /SET RETURN ADDRESS
JMP I FILTER /RETURN
FILHAL, 0
SPA /IS IT POSITIVE?
JMP +4
CLL RAR /YES, DIVIDE POS.
CLL /NUMBER BY TWO
JMP FILHAL /AND JMP BACK
STL /NO, DIVIDE NEG.
RAR /NUMBER BY TWO
CLL
JMP FILHAL /AND JMP BACK
SUBROUTINE TO REDISTRIBUTE A BLOCK OF DATA.

A SIGNAL, CONSISTING OF X SAMPLES IS REDISTRIBUTED INTO A SIGNAL CONSISTING OF THE NEAREST LOWER 2^N SAMPLES. (MAX 2048)

END OF SIGNAL CONDITION: 10 SUCCEEDING ZERO'S.

GENERAL: X POINTS BECOME Y=2^N POINTS

\[
Y(I) = (1 - \text{REST OF } I \cdot \text{CONTENTS OF } I \cdot \text{----} + (X-1)
\]

\[
(Y-I) = (Y-1) \cdot (Y-1)
\]

\[
(X-I) = (X-1) \cdot (X-1)
\]

\[
((1 - \text{REST OF } I \cdot \text{CONTENTS OF } (I \cdot \text{----} + 1)
\]

\[
(Y-I) = (Y-1) \cdot (Y-1)
\]

THIS OPERATION IS NECESSARY PRIOR TO APPLYING A FAST FOURIER TRANSFORM TO A PERIODIC SIGNAL, NOT CONSISTING OF 2^N SAMPLES.

NECESSARY HARDWARE: KE8-E EXTENDED ARITHMETIC E L.

CALLING SEQUENCE:

JMS REDIST

START ADDRESS OF MEMORY BLOCK MINUS 1
MINUS NO. OF POINTS OF SIGNAL Buffer
RETURN /AC=0

IRI=10
REDSTA, 0
REDPOI, 0
RED0, 0
REDM11, -11
REDCNT, 0
REDNO, 0
REDCMP, 0
REDNEW, 0
REDMIN, -1
REDRES, 0
REILOC, 0
REDPOS, 4000
REFAC, 0
REDIST, 0

TAD I REDIST /GET START ADDRESS
DCA IRI
TAD I REDIST IAC
DCA REDSTA
ISZ REDIST
TAD I REDIST /GET NO. OF POINTS
DCA REDPOI
TAD I REDIST
DCA REDTLR
ISZ REDIST /SET RETURN ADDRESS

22
REDSNL,  TAD I IR1  /GET WORD
ISZ REDTLP  /INCREMENT WORD COUNT
SKP
JMP I REDIST  /RETURN, NO END OF SIGNAL
SNA
JMP *+3
CLA CLL
JMP *-7  /NO, TAKE NEXT WORD
TAD IR1  /YES, STORE LOCATION
DCA RED0  /OF FIRST ZERO
TAD REDM11
DCA REDCNT
TAD I IR1  /ADD NEXT 9 WORDS
ISZ REDCNT
JMP *-2
SZA CLA  /TEST FOR SUM =0
JMP REDNL  /NO, GO ON
TAD REDSTA  /YES AND
CIA  /CALCULATE
TAD RED0  /NO, OF POINTS
DCA REDNO
CLA CLL CML RAR  /MAKE 40000
DCA REDCMP
TAD REDNO
AND REDCMP  /AND WITH 21N
SNA CLA  /TEST FOR NON ZERO
JMP *+4
TAD REDCMP  /YES, THIS IS THE
DCA REDNEW  /NEW NO. OF POINTS
JMP *+5
TAD REDCMP  /NO, DIVIDE
CIA RAR  /BY TWO
DCA REDCMP
JMP *-12  /AND GO AGAIN
CAM
SWAB
TAD REDNEW  /GET 21N POINTS
CIA
DCA REDTLP  /PUT IN COUNTER
DCA RED0  /CLEAR I
TAD REDSTA  /GET START ADDRESS
TAD REDMIN  /SUBTRACT ONE
DCA IRI
TAD REDNO  /GET ORIGINAL NO. OF POINTS
TAD REDMIN  /SUBTRACT ONE
DCA REDNO  /THIS IS: (X-1)
TAD REDNEW  /GET NEW NO. OF POINTS
TAD REDMIN  /SUBTRACT ONE
DCA REDNEW  /THIS IS: (Y-1)
REDNEW, TAD REDNEW
MQL
MUY
RED0
REDNEW
DCA REDRES
SWP
TAD REDSTA
DCA REDLOC
TAD REDNEW
MQL
TAD REDRES
SAM
MQL
TAD I REDLOC
TAD REDPOS
DCA REDCMP
MUY
REDCMP
DVI
REDNEW
SWP
DCA REDFAC
CAM
ISZ REDLOC
TAD REDRES
MQL
TAD I REDLOC
TAD REDPOS
DCA REDCMP
MUY
REDCMP
DVI
REDNEW
SWP
TAD REDFAC
TAD REDPOS
DCA I IRI
ISZ RED0
ISZ REDTLR
JMP REDNEW
TAD REDNEW
DCA I IRI
TAD REDPOI
TAD REDNEW
IAC
DCA REDTLR
DCA I IRI
ISZ REDTLR
JMP -.2
JMP I REDIST

/GET (X-1)
/MULTIPLY WITH I
/DIVIDE BY (Y-1)
/SAVE REST
/QUOTIENT TO AC
/GET START ADDRESS
/SUM IS LOCATION
/GET (Y-1)
/SUM IS LOCATION
/RESULT: (Y-1)-REST
/PUT IN MQ
/GET CONTENTS OF LOC.
/MAKE POS.
/MULT. (Y-1)-REST
/WITH POS CONTENTS OF LOC.
/DIVIDE RESULT
/SAVE FIRST FACTOR
/INCREMENT LOCATION
/GET REST
/GET CONTENTS OF LOC. +1
/MAKE POS.
/MULT. REST WITH
/POS CONTENTS OF LOC. +1
/DIVIDE BY
/(Y-1)
/ADD FIRST FACTOR
/GET RIGHT SIGN
/AND STORE IN MEMORY
/INCREMENT I
/INCREMENT COUNTER
/NEXT POINT
/CALCULATE END ADDRESS
/OF SIGNAL
/GCONF NUMBER OF POINTS
/IN SIGNAL BUFFER
/CONF REST NO. OF POINTS
/CLEAR REST OF BUFFER
PACK A CHARACTER IN A BUFFER IN OS/8 FORMAT
CAN BE USED FOR BUFFERS UP TO 31 PAGES
( NOT USING LAST PAGE OF FIELD)
PARAMETERS ARE:
CURFLD: FIELD OF SUBROUTINE
BUFFLD: FIELD OF BUFFER
BUFBEg, BUFENd: DEFINE SIZE OF BUFFER
CALL: TAD CHAR
JMS PACKB
BUFFER FULL RETURN (AC=0)
NORMAL RETURN (AC=0)

PACKB, ---
DCA PACKB-1 /SAVE CHARACTER
TAD PACKSW /TEST PACKSWITCH
CDF BUFFLD
SZA
JMP PACKB1 /IF -2
TAD PACKB-1 /GET CHARACTER
DCA I PACPTR /INSERT IN BUFFER
TAD PACPTR
CLL RAR /IS POINTER ODD?
SNL CLA /SKIP IF YES
JMP +4
CLA CLL CMA RAL /SET PACKSWITCH TO -2
DCA PACKSW
SKP
ISZ PACPTR /INCREMENT POINTER IF EVEN
JMP PACKB2 /GO TO EXIT
PACKB1, CLA CLL CMA
TAD PACPTR /DECREMENT POINTER
DCA PACPTR
TAD PACKB-1 /GET CHARACTER
RTL
RTL /SHIFT 4 POSITIONS TO LEFT
DCA PACKB-1 /SAVE TEMPORARY
TAD PACKB-1
AND C7400 /KILL BITS 4-11
TAD I PACPTR
DCA I PACPTR /INSERT IN BUFFER
ISZ PACPTR /INCREMENT ADDRESS POINTER
ISZ PACKSW /INCREMENT PACKSWITCH
JMP PACKB1+3 AGAIN IF PACKSWITCH NONZERO
TAD PACEND
CMA CLL /TEST FOR BUFFER END
TAD PACPTR
SNL CLA /SKIP IF FULL
JMP PACKB2
TAD PACBEG /INITIALIZE POINTER
DCA PACPTR
SKP
PACKB2, ISZ PACKB /NORMAL RETURN
CDF CURFLD
JMP I PACKB

PACPTR, BUFBEG
PACKSW, 0
PACBEG, BUFBEG
PACEND, BUFENd
C7400, 7400
UNPACK A CHARACTER FROM A BUFFER IN OS/8 FORMAT

CAN BE USED FOR BUFFERS UP TO 31 PAGES
(NEAR USING LAST PAGE OF FIELD)

PARAMETERS ARE:
CURFLD: FIELD OF SUBROUTINE
BUFFLD: FIELD OF BUFFER
BUFFREG, BUFEND: DEFINE SIZE OF BUFFER

CALL: CLA
JMS UNPACK
RETURN; IF LINK=1: BUFFER EMPTY
IF LINK=0: NORMAL RETURN

RETURNS WITH CHARACTER IN AC

UNPACK, ...........
\TEMPORARY STORAGE

TAD UNPSW /TEST PACKSWITCH
CDF BUFFLD
SZA CLA
JMP UNPAC1 /IF -2
TAD I UNPPTR /GET FROM BUFFER
AND C377 /KILL BITS 0-3
DCA UNPACK-1/SAVE TEMPORARY
TAD UNPPTR /IS POINTER EVEN?
CLL RAR
SNL CLA /SKIP IF NO
JMP .+4
CLA CMA HAL /SET PACKSWITCH TO -2
DCA UNPSW
SKP /NO INCREMENT
ISZ UNPPTR
JMP UNPAC2 /GO TO EXIT

UNPAC1, DCA UNPACK-1/MAKE TEMP. LOCATION ZERO
TAD I UNPPTR /GET WORD
AND C7400 /KILL BITS 4-11
TAD UNPACK-1
CLL RTR
RTR
DCA UNPACK-1/SAVE TEMPORARY
CLA CMA
TAD UNPPTR /DECREMENT POINTER
DCA UNPPTR
ISZ UNPSW /INCREMENT PACKSWITCH
JMP UNPAC1+1/AGAIN IF NONZERO
CLA CLL CML IAC HAL
TAD UNPPTR
DCA UNPPTR /POINTER +3
TAD UNPEND /TEST FOR BUFFER END
CLA CMA
TAD UNPPTR
SNL CLA /SKIP IF EMPTY
JMP UNPAC2
TAD UNPBE G INITIALIZE POINTER
DCA UNPPTR
SKP /LINK=1
UNPAC2, CLL /NORMAL RETURN
TAD UNPACK-1/GET CHARACTER
CDF CURFLD
JMP I UNPACK

UNPPTR, BUFBEG
UNPSW, 0
UNPBE G, BUFBEG
UNPEN D, BUFE ND

C377, 377
C7400, 7400
PARITY GENERATOR

GENERATES ODD OR EVEN PARITY BIT (8TH BIT)

CALL: TAD CHAR
      JMS PARITY
      RETURN (WITH CHARACTER IN AC)

0 /CHARACTER TEMPORARY
0 /PARITY SUM

PARITY, AND C177 /KILL 8TH BIT
DCA PARITY-2
DCA PARITY-1
TAD PARITY-2/GET CHARACTER

PARITY1, CLL RAR
  SZL /TEST BIT
  ISZ PARITY-1
  SZA /NEXT BIT ?
  JMP PARITY1 /YES
  TAD PARITY-1/GET PARITY SUM
  RAR
  SZL CLA /FOR EVEN PARITY;
  TAD C200 /SET PARITY BIT
  TAD PARITY-2/GET CHARACTER AND
  JMP I PARITY / RETURN

C177, 177
C200, 200
/ SKIP ON FLAG WITH TIMED OUT RETURN
/
/
CALL:  CLA  SKP OUT
/
JMS  SKP OUT
/
TIME OUT (IN MILLISECONDS)
/
SKIP INSTRUCTION
/
TIMED OUT RETURN
/
NORMAL RETURN

0  /TIMER 1
0  /TIMER 2
-140  /PRESET TIMER 2 (ADJUST FOR 1 MILLISECOND)

SKP OUT,     ---
TAD I  SKP OUT  /GET PRESET TIMER 1
CIA
DCA  SKP OUT-3
ISZ  SKP OUT
TAD I  SKP OUT  /GET SKIP INSTRUCTION
DCA  SKP OUT2
ISZ  SKP OUT

SKP OUT1,  TAD  SKP OUT-1
DCA  SKP OUT-2/SET TIMER 2

SKP OUT2,  0  /OVERLAID BY SKIP INSTRUCTION
JMP  +3
ISZ  SKP OUT  /SKIPPED !
JMP  SKP OUT
ISZ  SKP OUT-2/OVERFLOW TIMER 2 ?
JMP  SKP OUT2  /NO
ISZ  SKP OUT-3
JMP  SKP OUT1
JMP  SKP OUT  /TIMED OUT
SEARCH A LIST FOR A MATCH

THE ROUTINE SEARCHES IN A LIST FOR A MATCH WITH AC.

IT HAS TWO RETURNS: FOUND AND NOT FOUND;

BOTH WITH AC=0. THE LIST TERMINATOR IS ALSO 0.

THE FIRST ELEMENT HAS ELEMENT NUMBER 0

/ TAD (ELEMENT
/ JMS SORTC
/ LIST-1
/ NOT IN LIST RETURN /AC=0
/ NORMAL RETURN /AC=0

LIST 301, 302, 303, 304, .......... 0 /ZERO IS TERMINATOR

SORTCN, 0 /COUNTER CAN BE ON PAGE 0
0 /POINTER
0 /AC

SORTC, --- /TEST FOR CHAR. IN LIST
CIA
DCA SORTC-1 /SAVE -AC
DCA SORTCN /CLEAR COUNTER
TAD I SORTC /GET ARG1
ISZ SORTC /FOR CORRECT RETURN
DCA SORTC-2 /SAVE POINTER
ISZ SORTC-2 /POINTER+1
TAD I SORTC-2 /GET LIST ELEMENT
ISZ SORTCN
SNA /ZERO?
JMP I SORTC /YES, ELEMENT NOT FOUND
TAD SORTC-1 /COMPARE
SZA CLA /EQUAL?
JMP -7 /NO, TRY NEXT ELEMENT
ISZ SORTC /YES, SET UP RETURN
JMP I SORTC /AC=0
LIST SEARCH CROSS FIELD CALLABLE
THE ROUTINE MATCHES THE AC AGAINST ALL ELEMENTS
OF A LIST. IN CASE OF A MATCH IT TAKES THE NORMAL
RETURN WITH THE OFFSET IN THE LIST IN THE AC.
IN CASE IT ENCOUNTERS A 0000 IN THE LIST, IT
TAKES THE ERROR RETURN, ALSO WITH OFFSET IN AC.
THE LIST IS ASSUMED TO BE IN FIELD OF CALL.
IN THE EXAMPLE: ELEMENT 301 HAS OFFSET 0

/ TAD (AC
/ CIF 0
/ JMS LIST
/ LIST¬ . /MARK THIS CONSTRUCTION!!!
/ NOT IN LIST RETURN /AC=OFFSET IN LIST
/ NORMAL RETURN (FOUND) /AC=OFFSET IN LIST

LIST, 0 0 0 /ZERO IS TERMINATOR!!!
LIST, 0 CIA
DCA LIST-1
CMA /NOTE! COUNTER OVERFLOWS INSTANTLY
DCA LIST-3
TAD I LIST
TAD LIST
DCA LIST-2
ISZ LIST /FOR RETURN
LIST1, ISZ LIST-3
ISZ LIST-2
TAD I LIST-2
SNA /ZERO?
JMP LISTR
TAD LIST-1
SZA CLA /MATCH FOUND?
JMP LIST1
ISZ LIST
LISTR, RDF
TAD C6203
DCA +2
TAD LIST-3
0
JMP I LIST
RELATIVE BRANCHER; CROSS FIELD CALLABLE

THIS BRANCH ROUTINE CAN BE CALLED FROM ANY FIELD.

IT ASSUMES THE LIST IN THE FIELD OF CALL, AND THE
DESTINATION ADDRESSES TOO. IT WORKS WITH RELATIVE
DISTANCES AND IS THEREFORE USEFUL FOR RUNTIME RELO-
CATABLE PROGRAMS.

TAD (AC) /IF BRANCH RESIDES IN FIELD 0
CIF Ø
/CALL, JMS I (BRANCH
/ LIST--1 /RELATIVE DISTANCE TO LIST
/ NOT IN LIST RETURN /AC=Ø

/LIST, 215; CR--
/ 212; LF--
/ 377; RUB--
/ Ø /ZERO IS TERMINATOR !!!!
Ø /TEMP AC;NEGATIVE
Ø /POINTER
BRANCH, --- /ENTER WITH AC=ELEMENT
CIA
DCA BRANCH-2
TAD I BRANCH
TAD BRANCH /MAKE POINTER ABSOLUTE
DCA BRANCH-1
ISZ BRANCH /FOR RETURN
BRAN1, ISZ BRANCH-1
TAD I BRANCH-1 /FETCH ELEMENT
SNA /IS IT Ø?(END OF LIST)
JMP BRANR /Y
TAD BRANCH-2
ISZ BRANCH-1
SZA CLA /MATCH?
JMP BRAN1 /N;LOOP
TAD I BRANCH-1 /PICK DISTANCE TO DESTINATION
TAD BRANCH-1 /MAKE ABSOLUTE
DCA BRANCH
BRANR, RDF /FIX FIELD AGAIN, AND JUMP
TAD C6203
DCA +1
Ø
JMP I BRANCH
C6203, 6203
GENERAL SETUP FOR OS8 HANDLERS
ENTER WITH LINK=1 FOR READ ONLY DEVICE
/LINK =0 FOR WRITE ONLY DEVICE
/ONLY APPLIES TO NON-BLOCK-ORIENTED DEVICES.
/IT CREATES A WORDCOUNT(WC) SET TO -WORDCOUNT-1
/AND A START ADDRESS OF BUFFER (CA),
/AND A CDF TO THE BUFFER FIELD IN 'SETCDF'.
/'ENTRY' IS THE ENTRYPONT OF THE HANDLER.
/IT LEAVES THE SUBROUTINE WITH 'ENTRY' POINTING
/TO 'STARTING BLK #' FOR BLOCK ORIENTED DEVICES,
/AND DATAFIELD STILL SET TO FIELD OF CALL.

7700
70

TEM=.
SETUP, 0
RDF /SET UP RETURN
TAD C6203
DCA XIT+2
TAD I ENTRY
AND SETUP-1 /70
TAD SETCD /MAKE CDF
DCA SETCDF /FIELD OF BUFFER
RAR /GET LINK
TAD I ENTRY /GET FUNCTION WORD
ISZ ENTRY
SPA /CHECK READ/WHITE
JMP ERR /UNRECOVERABLE ERROR
AND SETUP-2 /7700
CLL CML CMA HAL /MAKE -WORDCOUNT-1
DCA WC
TAD I ENTRY
ISZ ENTRY
DCA CA /CURRENT ADDRESS
JMP I SETUP

ERR, CLA CLL CML RAR /SIGNAL "PERMANENT IO ERROR"
XIT, ISZ ENTRY
ISZ ENTRY
0 /CDF CIF
JMP I ENTRY

SETCD, CDF Ø /SOMEWHERE IN HANDLER
C6203, CDF CIF Ø
CA, Ø
WC, Ø
UNPACK CHAR BY CHAR ROUTINE FOR OS8 HANDLERS.

NEEDS A WORDCOUNT SET TO -WORDCOUNT-1, AND A POINTER
SET TO CURRENT ADDRESS (BEGIN OF BUFFER).
AT THE VERY FIRST ENTRY: PACKSWITCH SHOULD BE 0.
THE WORDCOUNT IS INCREMENTED EVERY 3 CHARACTERS.
IF THE BUFFER IS EXHAUSTED, THE ROUTINE JUMPS TO XIT.
IT EXITS WITH THE DATAFIELD STILL SET TO THE
FIELD OF BUFFER

PACKSW, 0

/ PACKSWITCH-POS. MEANS L.S. 8 BITS
/ NEG. MEANS MOST SIGN 4 AND 4 BITS.

7400
377

UNPACK, ---

SETCDF, 0

/ ENTER WITH AC=0

SET TO CDF X BY SETUP ROUTINE

TAD PACKSW
SPA
JMP UNPI
SZA
CLA CLL CMA RTL / AC=-3

'DCA PACKSW
ISZ PACKSW
TAD I CA
AND UNPACK-1
ISZ CA
ISZ WC
JMP I UNPACK
/ EXIT WITH NEXT CHAR IN AC

7600

C7600, 7600

DCA PACKSW
/ RESET PACKSWITCH FOR NEXT ENTRY

JMP XIT

UNPI,

TAD CA
DCA CA
CLL
DCA TEM
TAD I CA
AND UNPACK-2
TAD TEM
RTL
ISZ CA
ISZ PACKSW
/ LOOP 2 TIMES; ALSO SETS PACKSW=0
JMP UNPI+3
RAL
JMP I UNPACK

CA, 0
WC, 0
TEM, 0

/ TEMP. LOCATION, SOMEWHERE IN HANDLER
/UNPACK CHAR BY CHAR FOR OS8 HANDLERS.
/ROUTINE UNPACKS AN OS8 FORMAT ASCII BUFFER CHARACTER
/BY CHARACTER. IT NEEDS A POINTER (CA) SET TO THE
/BEGINNING OF THE BUFFER, AND A WORDCOUNT (WC) SET
/TO - THE NUMBER OF WORDS IN THE BUFFER -1.
/THE LOCATION 'SETCDF' NEEDS TO BE SET TO THE FIELD
/WHERE THE BUFFER RESIDES.
/THE PACKSWITCH HAS 3 VALUES: 0 FOR THE FIRST OF 3 CHAR.
/1 FOR THE SECOND, AND 2 FOR THE 3RD.
/THE PACKSWITCH SHOULD BE 0 WHEN ENTERED FOR THE FIRST
/TIME. THE ROUTINE LEAVES THE DATAFIELD TO THE FIELD
/OF BUFFER UPON EXIT. IF BUF EMPTY THEN JUMP TO 'XIT'.

PACKSW, 0
7400
377

UNPACK, ---
/ENTER WITH AC=0

SETCDF, CDF
TAD PACKSW
RAR
SZL
JMP UNP2
SZA CLA
JMP UNP3

UNP1,
TAD I CA
AND UNPACK-2
CLL RTR
DCA TEM
TAD I CA
AND UNPACK-1
ISZ CA
ISZ PACKSW
ISZ WC
JMP I UNPACK

C7600, 7600
JMP 'XIT

UNP2,
TAD I CA
AND UNPACK-2
CLL RTR
RTR
TAD TEM
JMP UNP1+2

UNP3,
DCA PACKSW
TAD TEM
JMP I UNPACK

TEM, 0
CA, 0
WC, 0

/ENTER CHAR=0
/OVERLAID
/INCR. TWICE. EVERY 3 BYTES

/SOMEBWHERE IN THE HANDLER
UNPACK AND PRINT OS8 BUFFER; IN HANDLER
THIS PROGRAM PART IS USEFUL FOR HANDLERS THAT HAVE
AN OUTPUT ROUTINE THAT LEAVES BITS 0-3 AS THEY
ARE. THE WORDCOUNT (WC) SHOULD BE SET TO -WORDCOUNT/2
AS IT INCREMENTS EVERY 2 LOCATIONS (3 BYTES).
THE ROUTINE IS ENTERED AND EXITED WITH THE DATAFIELD
SET TO THE FIELD OF BUFFER.

```
UNP, TAD I CA /FIRST BYTE
ISZ CA
JMS OUTPUT
AND C7400
DCA TEM
TAD I CA /2ND BYTE
ISZ CA
JMS OUTPUT
AND C7400 /3RD BYTE
CLL RTR
RTR
TAD TEM
RTR
RTR
JMS OUTPUT
C7600, 7600
ISZ WC
JMP UNP
JMP XIT

TEM, 0 /SOMEWHERE IN HANDLER
CA, 0
WC, 0
C7400, 7400
```
/PUSH AND POP OPERATORS
/THE Routines CAN OPERATE ON DIFFERENT STACKS
/WHICH ARE POINTED TO BY THE ARGUMENT.
/CONVENTION IS THAT THE POINTER ALWAYS POINTS TO
/A N ELEMENT, UNLESS COUNT=0, THEN IT POINTS TO
/ITSELF.
/
/ TAD (AC
/ JMS PUSH
/ STACK
/ FULL RETURN, AC=0
/ NORMAL RETURN, AC=0

/STACK=-21  / MAX SIZE OF STACK=21(8)
/ 0  / COUNTER OF ELEMENTS; 0=EMPTY
/ 1  / FIRST ELEMENT
/ 2;3;4................0 (LAST ELEMENT)

PUSH, 0  / TEMP POINTER
      .-.  
      DCA POPTEM  / SHARED LOCATION
      TAD I PUSH  / SAVE PTR
      ISZ PUSH
      DCA PUSH-1
      TAD I PUSH-1 / STACK FULL?
      ISZ PUSH-1
      TAD I PUSH-1
      SMA CLA
      JMP I PUSH  / YES, FULL RETURN
      ISZ I PUSH-1 / COUNT+1
      ISZ PUSH-1
      ISZ I PUSH-1 / POINTER+1
      TAD I PUSH-1
      DCA PUSH-1 / POINTER FETCHED
      TAD POPTEM  / DEPOSIT IN STACK
      DCA I PUSH-1
      ISZ PUSH
      JMP I PUSH
/THE POP OPERATION FETCHES ONE WORD FROM THE
/STACK.
/
JMS POP
/STACK /POINTS TO STACK
EMPTY RETURN /AC=0
NORMAL RETURN /AC=ELEMENT

POPTERM, 0
7777
/TEMP POINTER
POP, .--.
  TAD I POP
  DCA POP-1 /SET UP PTR TO STACK
  ISZ POP
  ISZ POP-1
  TAD I POP-1 /COUNT=0?BUF EMPTY?
  SNA SPA
  JMP I POP /YES, EMPTY RET.
  TAD POP-2 /DECREMENT COUNT
  DCA I POP-1
  ISZ POP-1 /POINTS TO PTR
  TAD I POP-1
  DCA PUSH-1
  TAD I PUSH-1 /FETCH ELEMENT
  DCA POPTERM
  TAD PUSH-1 /DECREMENT PTR
  TAD POP-2
  DCA I POP-1
  TAD POPTERM /EXIT WITH ELEMENT IN AC
  ISZ POP
  JMP I POP
/*
 * Fill a rotating buffer queue.
 * Fillq is a routine that puts the AC in the next free location in the buffer. The concept is that the routine can be used for various buffers. Each buffer carries its own administration data.
 */

/* Buf*/
/* Full return */
/* Ac = 0 if element still accepted */
/* Ac uneq 0 if not accepted */
/* Normal return */

FILLQ

-JMS FILLQ
- Buf
- Full return
- Ac = 0 if element still accepted
- Ac uneq 0 if not accepted
- Normal return

0

Save size

Enter with the element in the AC

DCA FILLQ-1
- Save AC

TAD I FILLQ
- Argum • points to buf

ISZ FILLQ
- For return

DCA FILLQ-2

TAD I FILLQ-2
- Save size

DCA FILLQ-3

ISZ FILLQ-2

TAD FILLQ-3
- Ac = -size + count + 1

Cla

TAD I FILLQ-2

iac

Sma sza

JMP FILLQ3
- Buf full, AC not accepted

Sza Cla

ISZ FILLQ
- For normal return

ISZ I FILLQ-2
- Count + 1

ISZ FILLQ-2

ISZ FILLQ-2
- For ptr2

ISZ I FILLQ-2

PTR2+1

TAD FILLQ-3
- Size + ptr2 > 0?

Cla

TAD I FILLQ-2

Sma Sza Cla

JMP FILLQ2
- No wrap around

FILLQ2

TAD I FILLQ-2
- Yes, compute real addr

TAD FILLQ-2
- Ptr2 + 'Ptr2'

DCA FILLQ-3

Tad FILLQ-1

DCA I FILLQ-3
- Dep in buf

JMP I FILLQ

FILLQ3

Iac

DCA I FILLQ-2

JMP FILLQ1

Ptr2 = 1

FILLQ3

Cla

TAD FILLQ-1

JMP I FILLQ
FILL A ROTATING BUFFER

BUF, SIZE
COUNT
PTR1 /EMPTY PTR
PTR2 /FILL PTR
1
2...
N

FILLQ
SAVE AC
FETCH & SAVE BUF POINTER
SAVE SIZE

RETURN = RETURN + 1

MAKE NORMAL RETURN

COUNT + 1 = SIZE

COUNT = COUNT + 1

PTR2 > SIZE

PTR2 = PTR2 + 1

FILLQ2

FILLQ1

DEPOSIT AC THERE

RETURN

/FULL RETURN; AC = φ
/NORMAL RETURN; AC = φ
RESET A ROTATING BUFFER QUEUE
CLEAR THE BUFFER BY SETTING COUNT=0, AND MOVING POINTERS TO THEIR INITIAL POSITION (SYMMETRICALLY)

BUFF, 16 /*MAX. SIZE OF BUFFER SPECIFIED HERE*/ 0 /*COUNTS # OF ELEMENTS IN BUF*/ 1 /*EMPTY POINTER(PTR1) SET TO FIRST ITEM*/ 16 /*FILL POINTER(PTR2) SET TO END OF BUF*/ ZLOCK 16 /*THE ACTUAL BUFFER AREA*/

CLRQ, ... /*ENTER WITH AC=0*/
TAD I CLRQ /*GET POINTER TO BUFBOOKKEEPER*/
DCA CLRQ-1
ISZ CLRQ /*FOR RETURN*/
TAD I CLRQ-1 /*SAVE SIZE*/
DCA CLRQ-2
ISZ CLRQ-1 /*FOR COUNT=0*/
DCA I CLRQ-1
ISZ CLRQ-1
IAC /*PTR1=1*/
DCA I CLRQ-1
ISZ CLRQ-1
TAD CLRQ-2
DCA I CLRQ-1 /*PTR2=SIZE*/
JMP I CLRQ
/FETCH THE NEXT ITEM FROM THE HEAD OF THE Q.

BUFF /POINTER TO BUFFER

EMPTY RETURN /AC=0

NORMAL RETURN /AC=ELEMENT

MTQ STANDS FOR EMPTY-THE QUEUE

0 /SAVE ELEMENT

0 /BUF SIZE

0 /POINTER IN BUFBOOKKEEPER

MTQ, --- /ENTER WITH AC=0

TAD I MTQ /PTR TO BUFBOOKKEEPER

DCA MTQ-1

TAD I MTQ-1 /SAVE SIZE

DCA MTQ-2

ISZ MTQ /FOR RETURN

ISZ MTQ-1

CMA

TAD I MTQ-1 /COUNT-1<0?

SPA

JMP MTQ1 /YES, BUF EMPTY

DCA I MTQ-1 /COUNT=COUNT-1

ISZ MTQ-1

IAC /COMPUTE REAL ADDRESS

TAD I MTQ-1 /PTR1+\'PTR1\'+1

TAD MTQ-1

DCA MTQ-3 /SAVE

TAD I MTQ-3 /FETCH THE ELEMENT

DCA MTQ-3 /SAVE

ISZ I MTQ-1 /PTR1=PTR1+1

TAD I MTQ-1 /-PTR1+SIZE<0?

CIA

TAD MTQ-2

SMA CLA

JMP MTQ2 /YES, UPDATE PTR1

IAC

DCA I MTQ-1 /PTR1=1

MTQ2, TAD MTQ-3 /ELEMENT IN AC

ISZ MTQ

JMP I MTQ

MTQ1, CLA

JMP I MTQ /\'EMPTY\' RETURN
EMPTY A ROTATING BUFFER QUEUE

EXAMPLE:
SIZE = 10

PTR2

EMPTY; COUNT = 0

COUNT = 1

FIRST ELEMENT

PTR1

PTR2

NORMAL; COUNT = 5

PTR1

PTR2

FULL; COUNT = 10

RETURN

COUNT = COUNT - 1

COMPUTE THE ABSOLUTE ADDRESS

FETCH ELEMENT FROM BUF 2 SAVE TEMP.

PTR1 = PTR1 + 1

Y

PTR1 = 1

RETURN

N

MTQ

FETCH & SAVE BUF SIZE

RETURN = RETURN + 1

COUNT = COUNT - 1

MTQ

/EMPTY
AC = φ
COMBINED ROTATING BUFFER OPERATORS, CROSS FIELD CALLABLE.
TWO SUBROUTINES HAVE BEEN PUT TOGETHER IN ORDER TO
SAVE SPACE. FILLO PUTS AN ELEMENT INTO THE BUFFER,
AND MTQ TAKES ONE FROM THE BUFFER. EACH BUFFER HAS ITS
OWN ADMINISTRATION, WHICH MAKES THE ROUTINES USEFUL FOR
MULTIPLE BUFFERS WITH EACH ITS OWN LENGTH AND CONTENTS.
WHEN USING THE ROUTINE CROSS FIELD IT ASSUMES THE BUFFER
IN THE FIELD OF CALL. THE ARGUMENT IS RELATIVE, SO THE
ROUTINES CAN BE USED IN A RELOCATABLE ENVIRONMENT.
POINTERS ARE ALSO RELATIVE.

THIS IS WHAT THE BUFFER LOOKS LIKE:
BUF, 16 /MAX. SIZE OF BUF. HERE THE BUF IS 16 LONG
0 /COUNTER OF ELEMENTS IN THE BUFFER
0 /'READ' POINTER, USED BY MTQ
0 /'WRITE' POINTER, USED BY FILLO
ZBLOCK 16 /THE ACTUAL BUFFER AREA

NOTE THAT THE BUFFER SHOULD BE SET UP AS ABOVE.

TAD ELEMENT
CIF MONFLD
JMS I (FILLO
BUF-

BUFFER FULL RETURN/AC=0 MEANS THAT THE ELEMENT
HAS BEEN ACCEPTED (THE VERY LAST ONE)
/AC=ELEMENT MEANS NOT ACCEPTED!
NORMAL RETURN /AC=0

OSIZE, 0 /SIZE OF BUF SAVED HERE
QPTR, 0 /POINTER IN ADMINISTRATION OF BUFFER.
C6203, 6203

FILLO, 0...
JMS SETFIL /INITIAL SETUP, COMMON FOR BOTH.
TAD I QPTR /BUFFER FULL?
CMA
TAD OSIZE /SIZE-COUNT-1
SPA
JMP FILLQR-1 /BUF FULL: AC NOT ACCEPTED
SZA CLA
ISZ FILLO /BUF OK; NORMAL RETURN
ISZ I QPTR /COUNT+1
ISZ QPTR
JMS QWRAP /INCREMENT POINTER AND/OR WRAP
TAD QAC /DEPOSIT
DCA I QPTR
CLA

FILLQR, 0
JMP I FILLO
SETFIL, -- /COMMON SETUP
DCA OAC
TAD I FILLQ /GET ARGUMENT
TAD FILLQ /MAKE ABSOLUTE
DCA QPTR
TAD I QPTR /SAVE SIZE
DCA QSIZE
RDF /FOR RETURN
TAD C6203
DCA FILLQR
ISZ QPTR /-> COUNT
ISZ FILLQ /-> ERROR RETURN
JMP I SETFIL

/INCREMENT POINTER, AND TEST FOR END OF BUFFER.
/ELSE WRAP-AROUND. RETURN WITH QPTR POINTING TO THE ABSOLUTE
/BUFFER LOCATION

QWRAP, -- /ALSO COMMON INSTRUCTION
ISZ QPTR /TRY NEXT LOC. IN BUF
ISZ I QPTR /BEYOND SIZE?
CIA
TAD I QPTR
SMA CLA
DCA I QPTR /WRAPS AROUND
QWR1, TAD I QPTR /COMPUTE ABSOLUTE ADDRESS
TAD QPTR
IAC
DCA QPTR
JMP I QWHAP

/FETCH THE NEXT ITEM FROM THE ROTATING BUFFER
/ CIF MONTLD
/ JMS I QMTQ
/ BUF-
/ EMPTY RETURN /AC=0
/ NORMAL RETURN /AC=ELEMENT

QAC=, -- /TO SAVE SPACE
MTQ, --
TAD MTO
DCA FILLQ
JMS SETFIL /DO COMMON SETUP
CMA
TAD I QPTR /COUNT-1<0?
SPA
JMP FILLQR-1 /Y, BUF EMPTY
ISZ FILLQ /FOR NORMAL RETURN
DCA I QPTR /N, CONFIRM
JMS QWRAP /INC POINTER ETC.
ISZ QPTR /READ POINTER COMES EARLIER.
TAD I QPTR /FETCH
JMP FILLQR
IBINARY LOADER SUBROUTINE

THIS BINARY LOADER IS MUCH DIFFERENT FROM THE STANDARD
ONE. IT WAS DESIGNED ON THE CONCEPT OF A MECHANISM
THAT FREES THE PROGRAMMER FROM THE CUMBERSOME BINARY
FORMAT PECULIARITIES. THE PROGRAM READS BINARY 'TAPE'
FROM AN INPUT DEVICE AND PRODUCES 3 PARAMETERS OF
INTEREST: THE LOAD POINTER 'BINPC', THE LOAD FIELD 'BINFLD'
AND THE CONTENTS OF THE DATA 'BINAC'. THE USER CAN
TEST FORMATS AND BOUNDARIES, AND HAS TO DEPOSIT THE
DATA HIMSELF.
DURING LEADER PHASE THE ROUTINE IgNORS BOTH LEADER
AND BLANK CODE. A JMS TO BINL WITH AC UNEQ 0 ALSO
RESETS ALL PARAMETERS AND RETURNS TO LEADER MODE.

THE BINARY FORMAT TAPE HAS 3 PHASES:

PHASE 1 (LEADER PHASE) CODES 200 OR 000
PHASE 2 (ORIGINS, DATA AND FIELD SETTINGs)
3X0 FIELD SETTING TO FIELD X; ADDS NOT TO CHECKSUM
2XX:0YY ORIGIN SETTING TO PC=XXX:Y; ADDS TO CHECKSUM
.0XX:0YY DATA ELEMENTS XXX:Y ADD TO CHECKSUM
0XX:0YY CHECKSUM XXX:Y RIGHT BEFORE THE STOP SIGN
PHASE 3 (THE STOP SIGN) IS ALSO ONE 200 CODE

HOW TO USE THE ROUTINE AS A STANDARD BINARY LOADER
START, JMS BINL

HLT /EOT RETURN; AC=0
HLT /READY RETURN; AC=DIFERENCE IN CHECKSUMs
TAD BINFLD /AC=0; NORMAL RETURN
DCA TEM
TAD BINAC
TEM, 0 DCA I BINPC
JMP START

-100 /CONSTANTS
-300
6201
-200
BINL, 77 /MASK
BINPH, 0 /PHASE FLAG, 0=LEADER, POS=ORIGIN; NEG=DATA
BINPC, 0 /LOAD POINTER
BINAC, 0 /DATA TO BE DEPOSITED IN LOADPOINT
BINFLD, 0 /FIELD INTO WHICH TO LOAD
CHKSUM, 0 /COMPUTED CHECKSUM
CHKTEM, 0 /CHECKSUM OVER 2 FRAMES
0 /FRAME 2
0 /FRAME 1
BINL, ... /AC UNEQ TO 0 RESET ALL PARAMETERS
SZA CLA
JMP BIN3
BIN2, TAD BINPH /ARE WE IN LEADER PHASE?
SNA CLA
JMP BIN3 /Y
BIN4,
TAD BINL-1 /N, IS FRAME1=FIELD?
TAD BINC-3 /-300
SMA CLA
JMP BIN6 /Y, SET FIELD
TAD BINL-1 /IF 100-177 MAKE PHASE GT 0
TAD BINC-4 /IF 000-077 MAKE PHASE NEG
SMA
IAC /TO AVOID AC=0
DCA BINPH

/ASSEMBLE 6 M.S.BITS IN BINAC AND MAKE TEMP.CHECKSUM
TAD BINL-1
AND BINC
CLL RTL
RTL
DCA BINAC
TAD BINL-1 /MAKE TEMP CHKSUM
DCA CHKTEM

/READ FRAME 2
JMS HSREAD
    JMP BINRET+1 /EOT RETURN
DCA BINL-2

/ASSEMBLE 6 L.S.BITS AND UPDATE TEMP.CHECKSUM
TAD BINL-2
AND BINC
TAD BINAC
DCA BINAC
TAD BINL-2 /TEMP CHKSUM
TAD CHKTEM
DCA CHKTEM
TAD BINPH /ARE WE IN ORIGIN PHASE?
SMA CLA
JMP BIN7 /Y; PROCESS ORIGIN SETTING

/READ FRAME 1 IN ADVANCE; THIS IS FOR DATA ONLY
JMS HSREAD
    JMP BINRET+1 /EOT RETURN
DCA BINL-1
TAD BINL-1
TAD BINC-1 /IS IT 200 CODE?(STOP CODE)
SNA CLA
JMP BIN8 /Y, END OF DATA
ISZ BINPC /NEXT LOCATION
NOP /JUST IN CASE
TAD CHKTEM
TAD CHKSUM
DCA CHKSUM
ISZ BINV
BINRET, ISZ BINL
JMP I BINL
/RESET ALL PARAMETERS FOR START OF NEW DATATAPE
BIN3, TAD BINC-2 /CDF 0
DCA BINFLD
DCA CHKSUM /CHKSUM=0
DCA BINPH /SET TO LEADER PHASE

BIN5, JMS HSREAD /READ 1 FRAME
JMP BINRET /EOT RETURN
DCABINL-1
TAD BINC-1 /IS IT LEADER CODE?
TAD BINC-1
SNA CLA
JMP BIN2 /Y, PROCEED LEADER LOOP
TAD BINC-1 /IS IT 000?
SNA CLA
JMP BIN2 /Y, IGNORE ALSO IF IN LEADER MODE
JMP BIN4 /N, CAN BE ORIGIN ONLY

/MAKE CDF OUT OF FRAME 1
BIN6, TAD BINL-1
AND BINC
TAD BINC-2 /AC=0X0
DCA BINFLD
JMP BIN5

/PROCESS ORIGIN SETTING AND LOOK FOR MORE
BIN7, CMA /PC-1
TAD BINAC /THIS IS NEW PC
DCA BINPC
TAD CHKTEM
TAD CHKSUM /UPDATE CHKSUM
DCA CHKSUM
JMP BIN5

/END OF DATA PROCESSING; COMPARE TWO CHECKSUNS
BIN8, DCA BINPH /SET TO LEADER PHASE
TAD BINAC /TAPE CHKSUM
CIA
TAD CHKSUM
JMP BINRET /END OF DATA RETURN(READY)
**Binary Loader Subroutine**

**Entry**
- **BINL**, AC = ∅?
  - **Y**
    - **BINS**, ARE WE IN LEADER PHASE?
      - **N**
        - **BIN4**, FRAME1 = FIELD SET?
          - **N**
            - **BINS**, MAKE BINFLD = CORRECT CDF
          - **Y**
            - **BIN3**, FRAME1 = ORIGIN?
              - **Y**
                - SET ORIGIN PHASE
              - **N**
                - **ASSEMBLE 6 M.S. BITS IN BINAC; MAKE TEMP. CHECKSUM**
                - **EOT RETURN**
  - **N**
    - **EOT RETURN**

**Read Frame 2**
- **APPLY 6 L.S. BITS IN BINAC, MAKE TEMP. CHECKSUM**
- **ORIGIN PHASE?**
  - **Y**
    - **BINPC = BINAC - 1**
    - **CHKSUM = CHKSUM + CHKTEN**
  - **N**
    - **EOT RETURN**

**Read Frame 2**
- **SET LEADER PHASE; COMPARE CHECKSUMS**
  - **Y**
    - **STOP CODE 200?**
      - **N**
        - **BIN8, BINPC = BINPC + 1**
        - **CHKSUM = CHKSUM + CHKTEN**
      - **Y**
        - **READY RETURN**
  - **N**
    - **LEADER CODE?**
      - **Y**
        - **ZERO CODE?**
          - **Y**
            - **EOT RETURN**
          - **N**
            - **LEADER CODE**
              - **N**
                - **EOT RETURN**
              - **Y**
                - **LEADER CODE**

INTERRUPT SERVICE BY LIST LOOK-UP

THE ROUTINE TRIES TO IDENTIFY THE INTERRUPTING
DEVICE BY LOOKING IN THE SKIP LIST. IF FOUND, IT
CLEARS THE HARDWARE FLAG WITH THE INSTRUCTION
IN THE CLEARLIST. THEN IT JUMPS TO THE LOCATION
POINTED TO IN THE JUMP LIST.

IT TAKES CARE FOR SPECIAL DEVICES THAT SKIP IF THE
FLAG WAS LOW INSTEAD OF HIGH, AS IS NORMALLY THE CASE.
THIS IS NECESSARY FOR THE KV8 INTERRUPTBAR (6051).
FOR THIS PURPOSE LEAVE OUT BIT 4000 (2051) IN SKIPLST.
SKIP LIST ALWAYS TERMINATED BY A ZERO!!!!
OTHER LISTS NOT.

SKIPLST  6031; 6041; 2051; 6141; 6151; 0
CLRLST  6032; 6042; 6052; 6142; 6152
"MP' ST  KBD1; TTY; INTBAR; KBD2; TTY2

*/0  JMP I  +1:INTSERV

INTAC, 0  /AC SAVED HERE
INILNK, 0  /LINK SAVED HERE
INTPTR, 0  /POINTER IN LISTS

4000
SKIPLST-1
CLRLST-SKIPLST /*USED TO FIND LIST
JMPLST-CLRLST /*TO FIND JUMplist

INTSERV, DCA INTAC /*SAVE AC AND LINK
RAL
DCA INILNK
TAD INTSER-3 /*SET UP POINTER
DCA INTOPTR

INT3, CLA /*JUST IN CASE
ISZ INTOPTR
TAD INTOPTR /*GET SKIP INSTR.
SMA /*STRANGE SKIP?
JMP INTO /*YES, BIT 4000 WAS LEFT OUT
DCA *+1
0 /*OVERLAID BY SKIP INSTR.
JMP INTO /*DID NOT SKIP

INT3, TAD INTOPTR
TAD INTO-2 /*WHERE IS CLEAR INSTR?
DCA INTOPTR
TAD INTOPTR /*PICK IT UP
DCA *+1
0 /*OVERLAID BY CLEAR INSTR.
TAD INTOPTR /*WHERE TO GO?
TAD INTO-1
DCA INTOPTR
TAD INTOPTR
DCA INTOPTR
JMP INTO /*JUMP THERE
INT2,  SNA
     HLT
     TAD INTSER-4
     DCA +1
     0
     JMP INT3
     JMP INT1

EXIT,  CLA CLL
       TAD INTLNK
       CLL RAR
       TAD INTAC
       RMF
       ION
       JMP I 0

/END OF LIST?
/HALT IF INTERRUPT NOT FOUND
/SET BIT 4300
/OVERLAID BY SKIPINSTR.

/JUST IN CASE
\textbf{FIND SMALLEST HOLE IN A LIST, JUST BIG ENOUGH.}

\textbf{THE ROUTINE FINDS A HOLE IN A LIST THAT MATCHES A SPECIFIED SIZE (IN THE AC) BEST. IT RETURNS WITH THE POINTERS TO THAT HOLE IN THE AC.}

/ TAD 4 /SEARCH SMALLEST HOLE > OR = 4LOC.
/ JMS SMALL
/ LISTBEGIN
/ -LISTEND
/ NOT FOUND RETURN /AC=0
/ NORMAL RETURN /AC=POINTER TO HOLE BEGIN

\textbf{LISTBEGIN, 1;3;2;4;0;0;0;0;0}
\textbf{LISTEND, 0}
\textbf{ZEROS REPRESENT HOLES!!!!!!}

\textbf{SMALCN, 0} /ENDSWITCH:0=END;7777=NORMAL
\textbf{0} /C OUNTS SIZE OF HOLE
\textbf{0} /POINTS WHERE LAST HOLE WAS
\textbf{0} /HOLEFLAG:0=HOLE;7777=NOT HOLE
\textbf{0} /POINTER IN LIST
\textbf{0} /SIZE OF LAST HOLE
\textbf{0} /SIZE OF HOLE WANTED.

\textbf{SMALL, •••} /ENTRY SIZE OF WANTED HOLE IN AC
\textbf{CIA}
\textbf{DCA SMALL-1} /SAVE -SIZE
\textbf{CMA} /SET LAST HOLE SIZE TO LARGE VALUE
\textbf{DCA SMALL-2}
\textbf{CMA} /RESET END SWITCH
\textbf{DCA SMALCN-1}
\textbf{CMA}
\textbf{TAD I SMALL} /SET UP POINTER (-1)
\textbf{DCA SMALL-3}
\textbf{ISZ SMALL} /POINTS TO -LISTEND)
\textbf{SMALL1, DCA SMALCN} /CLEAR HOLE COUNTER
\textbf{CMA} /SET HOLEFLAG TO NOT HOLE
\textbf{SMAL2, DCA SMALL-4}
\textbf{TAD SMALCN-1} /END SWITCH SET?
\textbf{SNA CLA}
\textbf{JMP SMALND} /YES, REAL END
\textbf{TAD SMALL-3} /OUT OF LIST?
\textbf{CLL}
\textbf{TAD I SMALL}
\textbf{SZL CLA}
\textbf{JMP SMALL5} /YES, SIGNAL END FOR NEXT TURN
\textbf{ISZ SMALL-3} /POINTER+1
\textbf{TAD I SMALL-3} /FETCH

/ THE FOLLOWING INSTRUCTION DETERMINES WHAT IS A HOLE
/ AND WHAT NOT. IN THIS CASE: 0000='HOLE'
\textbf{SZL CLA} /IS IT A ZERO?
\textbf{JMP SMALL3} /NO
\textbf{DCA SMALL-4} /CLEAR HOLEFLAG
\textbf{ISZ SMALCN} /COUNT SIZE OF HOLE
\textbf{JMP SMALL2} /SET HOLEFLAG AND PROCEED
WE HAVE NOW TOUCHED AN OCCUPIED LOCATION. IF THIS
WAS THE FIRST AFTER A SET OF ZEROS, IT TERMINATES
A HOLE. LISTEND ALSO TERMINATES A HOLE!! (ENDFLAG)

SMAL3, TAD SMALL-4 /HOLEFLAG=0?
    SZA CLA
    JMP SMALL1 /NO, WE ARE STILL IN OCC. AREA
    TAD SMALLCN /YES, WE JUST ISOLATED A HOLE
    TAD SMALL-1 /IS HOLE EXACTLY WHAT WE WANT?
    SNA
    DCA SMALCN-1 /SET END FLAG
    SPA CLA
    JMP SMALL1 /NO, TOO SMALL, SEARCH FURTHER
    TAD SMALL-2 /YES, FOUND A POSSIBLE HOLE
    CIA CLL
    TAD SMALCN /IS IT SMALLER THAN LAST HOLE?
    SZL CLA
    JMP SMALL1 /NO, LAST HOLE IS BETTER MATCH
    TAD SMALCN /YES, REPLACE LAST HOLE
    DCA SMALL-2
    TAD SMALL-2 /MAKE IT POINT TO HOLEBEGIN
    CIA
    TAD SMALL-3 /ALSO REMEMBER WHERE LAST HOLE BEGINS
    DCA SMALL-5
    JMP SMALL1

/END OF LIST DETECTED, MAY TERMINATE A SET OF ZEROS (HOLE)
/SO SET FLAG TO SIGNAL THAT THE GAME IS OVER, BUT PROCESS
/THE HOLE FIRST.
SMAL5, DCA SMALCN-1 /SET END FLAG
    JMP SMALL3

/END OF THE ROUTINE. THERE IS EITHER A HOLE OR NO
/HOLE FOUND (LASTHOLE IS STILL 7777)
SMAL5D, TAD SMALL-2 /LASTHOLE=7777?
    CMA
    SNA CLA
    JMP SMALL4+1 /YES, NOTHING FOUND
    TAD SMALL-5 /TAKE POINTER IN AC
SMAL4, ISZ SMALL
    ISZ SMALL
    JMP I SMALL
FIND THE SMALLEST HOLE IN A LIST GREATER THAN OR EQUAL TO THE VALUE IN THE AC.

(SMALL)
  REQUEST = AC
  LASTHOLE = VERY LARGE #
  RESET ENDSWITCH

(SMALL1)
(SMALL)
  HOLOCOUNT = 0
  HOLEFLAG = 7777
  ERROR RETURN
  NOTHING FOUND
  NORMAL RETURN
  POINTER TO BEGINNING OF HOLE IN AC

(SMALL2)
  ENDSWITCH SET?
  Y
  SMALLND
  N
  WITHIN LIMITS? (POINTER)
  Y
  SET ENDSWITCH
  N

(SMALL3)
  IS IT A HOLE?
  Y
  HOLEFLAG = \phi
  Y
  SME = REQUEST
  N
  SET ENDSWITCH
  HOLECOUNT = REQUEST
  Y
  TOO SMALL
  \phi
  TOO SMALL
  Y, POSSIBLE HOLE

55
HOLE < LAST HOLE?

Y

REPLACE LAST HOLE, COMPUTE AND SAVE POINTER

N

A
// LINK FOR RELOCATABLE CROSS-PAGE REFERENCING WITHIN 1 FIELD.
// THIS LINK WORKS IN MUCH THE SAME WAY AS A DEBUGGING TRACER.
// IT EXECUTES THE INSTRUCTION WITH A COMPUTED OPERAND.
// THE ROUTINE HAS BEEN TAILORED FOR AND TAD ISZ DCA.
// JMP AND JMS. THE LINK WILL BE PRESERVED AND OPERATED
// UPON IN THE NORMAL WAY. IN CASE OF A JMS THE ABSOLUTE
// RETURN ADDRESS IS DEPOSITED IN THE SUBROUTINE ENTRY.
// SO THAT THE RETURN FROM THE SUBROUTINE IS THROUGH A
// JMP I SUBR. ENTRY. ARGUMENTS ARE PICKED UP IN THE SAME WAY:
// TAD I SUBR. ENTRY.

TAD AC
JMS LINK // LINK PREFERABLY IN PAGE 0
B- // RELATIVE POINTER
TAD // OR AND ISZ DCA JMP JMS
RETURN
RETURN IF ISZ SKIPPED

1000 // CONSTANT
AND I LINKPC // TO MAKE REAL INSTRUCTION FROM
LINKPC, 0 // COMPUTED ABSOLUTE ADDRESS
LINKAC, 0 // SAVED AC

LINK, ...
DCA LINKAC
TAD I LINK // COMPUTE OPERAND(PC)
SPA // TO PRESERVE THE LINK
CML
TAD LINK // MAKE ABSOLUTE ADDRESS
DCA LINKPC
ISZ LINK
TAD I LINK // AND TAD...ETC
ISZ LINK
SPA
JMP LINKJ // JMP OR JMS
TAD LINKPC-1 // MAKE TAD I LINK INSTR ETC.
DCA +2
TAD LINKAC
0 // OVERLAIEd
JMP I LINK // NORMAL RETURN
ISZ LINK // IF IT SKIPPED, RETURN+1
JMP I LINK

LINKJ, AND LINKPC-2 // AND (1000
SZA CLA // IS IT JMP OR JMS?
JMP LINK3 // JMP
TAD LINK // DEP. 'RETURN' IN SUBR. ENTRY
DCA I LINKPC
ISZ LINKPC
JMP I LINKPC

LINK3, TAD LINKAC
JMP I LINKPC
ONE OF THE METHODS TO ALLOCATE CORE STORAGE DYNAMICALLY IS BY USING THE 'FREE-CORE' TECHNIQUE. FREE-CORE IS AN AREA IN CORE, DIVIDED INTO BLOCKLETS OF A GIVEN SIZE. (ALSO EQUAL IN SIZE) THESE BLOCKLETS CONTAIN ONE POINTER POINTING TO THE FOLLOWING BLOCKLET, IN THIS WAY LINKING THE BLOCKLETS IN ONE DIRECTION. (FIG 1)

IF A PROGRAM (1) REQUIRES CORE STORAGE FOR VARIABLES, DATA, OR PUSH-DOWN LISTS, THE NECESSARY STORAGE CAN BE REQUESTED FROM FREECORE IN UNITS OF A GIVEN SIZE. (ONE BLOCKLET AT A TIME). (FIG 2)


IF PROGRAM (1) NO LONGER NEEDS ITS BLOCKLET, IT CAN RELEASE THAT BLOCKLET TO FREECORE, WHICH RESTORES THE CHAIN AS IN FIGURE 4.

THE ROUTINES DESCRIBED IN THE FOLLOWING PAGES MAKE USE OF A FREECORE AREA CONTAINING BLOCKLETS OF 8 COMPUTER WORDS, THE LAST WORD IN EACH BLOCKLET IS RESERVED FOR THE POINTER.
/INITIALIZE THE FREECORE AREA
/IT ACTS AS IF ALL THE BLOCKS WERE OCCUPIED BY
/SOME STRANGE CHAIN. BY RELEASING THE BLOCKS ONE
/AFTER ANOTHER, THEY BECOME CHAINED TO THE FREE-
/CORE CHAIN OF AVAILABLE BLOCKS. FC ZEROED FIRST.
/FREELO AND FREEHI DELIMIT THIS ACTION.
/
/ CLA
/ JMS INITFC
/ NORMAL RETURN

INITFC, -.-

DCA FREEF /NO FREEBLKS AVAILABLE
DCA FREECT
TAD FREEHI /GET UPPER LIMIT OF FREECORE
DCA INITFC-1
/
AGAIN,
TAD INITFC-1 /RELEASE ALL BLOCKS
JMS RELBLK /AND CHAIN ALL TOGETHER
TAD INITFC-1
TAD C7770 /SET UP FOR NEXT BLOCK
DCA INITFC-1
/
TAD FREELO /GET LOWER LIMIT OF FREECORE
CIA
TAD INITFC-1 /THIS BLOCK BELONGS TO FREECORE?
SMA CLA /SKIP IF NO
JMP AGAIN /YES, RELEASE THIS BLOCK AND SO ON
/
JMP I INITFC /RETURN
IPEQUES1
A FREE BLOCK
ITHE ROUTINE ISOLATES A BLOCK FROM THE TAIL OF THE
/CHAIN OF FREE CORE BLOCKS, AND LINKS IT TO THE BLOCK,
/POINTED TO BY THE AC. IF THE AC=0 NO LINK IS MADE
/THIS IS FOR THE VERY FIRST REQUEST.
/IN CASE OF ERROR RETURN: AC=0; ELSE AC
/POINTS TO FIRST LOC OF REQUESTED NEW BLK.
/
/ TAD ANY LOC. IN PREVIOUS BLOCK, OR AC=0
/ CIF 0
/ JMS REQBLK
/ ERROR RETURN /NO MORE BLKS AVAILABLE
/ NORMAL RETURN

0 /TEMPORARY STORAGE

REQBLK=.
 RELK ..
 DCA RELK-1 /SAVE
 PDF
 TAD C6203
 DCA RBLRET
 CDF 0
 TAD RELK-1
 SNA
 JMP RELK2
 AND C7770 /MAKE POINTER TO LINK LOCATION.
 TAD C7
 DCA RELK-1
 TAD FREEF /POINTS TO TAIL
 SNA
 JMP RBLRET /= NO BLKS AVAILABLE
 DCA I RELK-1 /MAKE LINK TO NEW BLK
 RELK2. TAD FREEF /FIND NEW TAIL.
 SNA
 JMP RBLRET /NO MORE BLKS AVAILABLE
 DCA I RELK-1
 TAD C7
 DCA RELK-1
 TAD I RELK-1 /UPDATE FREEFIRST (THE TAIL)
 DCA FREEF
 DCA I RELK-1 /ZERO THE LINK OF REQ BLK
 CMA
 TAD FREECT /DECREMENT THE NUMBER OF AVAILABLE FREE
 DCA FREECT /BLKS
 TAD RELK-1 /TAKE POINTER TO REQ BLK
 AND C7770 /FST LOC OF BLK
 ISZ RELK
 RELK1: 0
 JMP I RELK /NORMAL RETURN.
RELEASE A BLOCKLET TO FREECHAIN.
THE BLOCK POINTED TO BY THE AC IS LINKED TO THE TAIL OF THE CHAIN OF FREE CHAIN BLOCKS WITH ITS CONTENTS ZERODE.
EXIT IS ALWAYS WITH AC = 0. THERE IS NO ERROR EXIT.

TAD (POINTER
CIF 0
JMS RELBLK
NORMAL RETURN/AC=0

RELBLK=

ELFL, -..-..-..-..
AND C77TO /KILL BIT 9-11
TAD C7 /INSERT BIT 9-11
DCA RBLK /SAVE TEMP
RDF
TAD C6203
DCA RELRET
CDF 0
TAD FREEF /GET ADDR. OF NEXT FREE FREEBLK
DCA I RBLK /CHAIN THIS BLOCK TO THE FREE CHAIN
TAD RBLK
AND C77TO /SET UP ADDR. OF THIS BLOCK
DCA FREEF /STORE AT PAGE 0
TAD M7 /SET UP COUNTER
DCA RBLK-1
TAD FREEF /GET ADDR. OF FIRST WORD IN BLOCK
DCA RBLK
DCA I RBLK
ISZ RBLK
ISZ RBLK-1 /INCR. COUNT
JMP .-3 /AGAIN IF CNT NONZERO
ISZ FREECT /INCR. NR. OF FREE BLOCKS
RELRET, 0
JMP I RELFL /RETURN
/RELEASE A Q OF FORWARD LINKED BLOCKETS TO FC.

/ TAD (POINTER IN FIRST BLKLT
/ CIF MONFLD
/ JMS REL0
/ NORMAL RETURN /AC=0

0   /TEMPORARY
0   /TEMPORARY

REL0,    ---
DCA REL0-1 /SAVE AC
RDF        /MAKE RETURN
TAD C6203
CIF MONFLD
DCA RELOR
TAD REL0-1

REL02,   AND C7773 /MAKE -> FORWARD LINK
TAD C7
DCA REL0-1
TAD I REL0-1 /PICK UP FORWARD LINK
DCA REL0-2
TAD REL0-1 /RELEASE THIS BLKLT
JMS RELBLK
TAD REL0-2 /IF FORWARD LINK=0; END OF CHAIN
SZA
JMP REL02

RELOR,  0
JMP I REL0
MAKE A BUFFER OR QUEUE IN FREECORE.

THE ROUTINE CREATES A 'BOOKKEEPING' BLOCK AND THE FIRST BLOCK OF THE BUFFERQUEUE. IT SETS THE PARAMETERS OF 'BOOK' ACCORDINGLY.

THE CONTENTS OF THE AC IS USED TO SET THE MAXIMUM ALLOWABLE LENGTH OF THE BUFFER QUEUE. RETURN IS WITH THE POINTER TO 'BOOK' IN THE AC. IN CASE OF ERROR RETURN

(NO ROOM AVAILABLE), AC=0 AND THERE WILL ALSO BE NO 'BOOK' BLOCK.

TAD MAXIMUM LENGTH
CIF 0
JMS MAKQ
ERROR RETURN /AC=0
NORMAL RETURN /AC= PTR TO 'BOOK'

0 /TEMP. POINTER
0 /POINTS TO FST LOC IN BUF
0 /POINTS TO 'BOOK'
0 /'MAXIMUM'

MAKQ 0
DCA MAKQ-1 /SAVE
PDF
TAD C6203
DCA MAKRET
CDF 0
TAD MAKQ-1
SNA
JMP MAKRET /ERROR AC=0
DCA MAKQ-1 /SAVE
JMS REQBLK /REQ. 'BOOK' BLK
JMP MAKRET /NO ROOM ERROR
DCA MAKQ-2 /SAVE 'BOOK' PTR
TAD MAKQ-2 /REQ. FIRST BUF BLK
JMS REQBLK
JMP MAKQ2 /NOT AVAILABLE
DCA MAKQ-3 /SAVE PTR TO FST BUFBLK

THE 'BOOK' BLK IS SET UP AS FOLLOWS:
BOOK, 0 /'DATA'
BUF PTR+1 /'BUFTAIL'
0 /'BUFCOUNT'
BUF PTR /'BUFHEAD'
MAXIMUM
0
0
0
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAD MAKQ-2</td>
<td>SET UP TEMP PTR</td>
</tr>
<tr>
<td>DCA MAKQ-4</td>
<td></td>
</tr>
<tr>
<td>ISZ MAKQ-4</td>
<td>POINTS 'BUFTAIL'</td>
</tr>
<tr>
<td>TAD MAKQ-3</td>
<td></td>
</tr>
<tr>
<td>IAC</td>
<td></td>
</tr>
<tr>
<td>DCA I MAKQ-4</td>
<td></td>
</tr>
<tr>
<td>ISZ MAKQ-4</td>
<td>SET UP 'BUFHEAD'</td>
</tr>
<tr>
<td>ISZ MAKQ-4</td>
<td></td>
</tr>
<tr>
<td>TAD MAKQ-3</td>
<td></td>
</tr>
<tr>
<td>DCA I MAKQ-4</td>
<td></td>
</tr>
<tr>
<td>TAD MAKQ-1</td>
<td>SET UP 'MAXIMUM'</td>
</tr>
<tr>
<td>TAD MAKQ-2</td>
<td></td>
</tr>
<tr>
<td>DCA I MAKQ-4</td>
<td>EXIT WITH PTR TO 'BOOK' IN AC</td>
</tr>
<tr>
<td>ISZ MAKQ</td>
<td></td>
</tr>
</tbody>
</table>

**MAKRET, 0**

**JMP I MAKQ**

**MAKQ2,**

**TAD MAKQ-2**

**IF NO ROOM, RELEASE 'BOOK' BLK TOO**

**JMS RELBLK**

**JMP MAKRET**
READ NEXT ELEMENT FROM THE TAIL OF A BUFFER.
THE AC MUST POINT TO A SECTION IN CODE WHERE THE
BOOKKEEPING OF THIS BUFFER IS DONE. FORMAT IS:

/BOOK, DATA /TEMP. SAVE LOCATION, ONE ELEMENT
/BUFTAIL /TAIL OF BUFFER, FIRST LOC.
/BUFCOUNT /# OF ELEMENTS NOW IN BUF
/BUFHEAD /HEAD OF BUFFER, LAST LOC.
/MAXIMUM /MAX # OF ELEMENTS ALLOWED

TAD (BOOK
CIF 0
JMS REBUF
BUF EMPTY RET. AC=0
RETURN WITH ELEMENT IN AC.

0 /TEMPORARY
0 /"

REBUF=.
RB, 0
DCA RB-1 /SAVE POINTER TO BOOK
RDF
TAD C6203
DCA RBRETR
CDF 0
CLL CML RTL
TAD RB-1
DCA RB-2 /POINTS TO BUFCOUNT
TAD I RB-2 /IS BUF EMPTY?
SNA
JMP RBRETR /YES, ERROR RETURN
TAD M1 /NO, COUNT-1
DCA I RB-2
ISZ RB-1 /POINTS TO BUFTAIL
TAD I RB-1 /IS THIS THE LINK?
DCA RB-2
TAD I RB-1
AND C7
TAD M7
SZA CLA
JMP HB2 /NO
TAD I RB-2 /YES, GET LINK TO NXT BLK.
DCA I RB-1 /UPDATE BUFTAIL
TAD RB-2 /NOW RELEASE BLOCK
JMS RELBLK
RB2, TAD I RD-1 /FETCH ELEMENT
DCA RB-2
ISZ I RB-1 /BUFTAIL+1 FOR NXT ENTRY
TAD I RB-2
ISZ HB
RBRET, 0
JMP I RB
Write an element onto the top of a buffer queue.

When buffer full, or no free blocks available.

If full, try full is also taken when the last char

could just be stored. Successive attempts lose data.

TAD ELEMENT /store in 'DATA'
CIF 0
JMS WRUF
BOOK /pointer to book flk, flg
BUF FULL RETURN /AC=0; also if FC• FULL
NORMAL RETURN /AC=0

0 /flag: 1 if last element
0 /data store
0 /temporary

NAME

DCA WP-3
TAD I WP /pick up bookptr
DCA WE-1
ISZ WB
CDF 0
TAD C6203
DCA WRET
CDF 0
ISZ WE-1
ISZ WB-1 /points to 'bufcount'
CLL CML RTL /AC=2
TAD W0-1
DCA WE-2 /points to 'maximum'
TAD I WE-2 /buf full?
CIA
TAD I WE-1
SMA
JMP WRET /yes, error
DCA WR-4 /flag for last element
TAD WE-1
IAC
DCA WR-2 /points to 'bufhead'
ISZ I WP-2 /bufhead+1 gives next free loc.
TAD I WP-2 /is this the link?
AND C7
TAD M7
SZA CLA
JMP WR2 /no
TAD I WE-2 /yes, request a free flk
JMS RERBLK
JMP WRET /no room; error; AC=0
NAME
DCA I WE-2 /update buf 'bufhead'
TAD I WE-2 /put data in buf
ISZ I WB-1 /bufcount+1
DCA WB-2
TAL WE-3
DCA I WE-2
ISZ WE-4 /ERR• RET• if last element
ISZ WB /normal ret
JMP I WE

66
/KILL THE BUFFER QUEUE
/THIS OPERATION, WHICH MAY BE TIME-CONSUMING, RELEASES
/ALL THE BUFFER BLKS TO FREE CORK, ALSO THE 'BOOK' BLK.
/
/
/TAD (POINTER TO 'BOOK'
/
/CIF 0
/
/JMS KILLQ
/
/NORMAL RETURN /AC=0

0 /TEMP. LINK
0 /POINTS LINK OF THIS BLK
0 /POINTS IN 'BOOK'

KILLQ, 0
DCA KILLQ-1 /SAVE
RDF
TAD C6203
DCA KILLRT
CIF 0
ISZ KILLQ-1 /POINTS 'BUFTAIL'
TAD I KILLQ-1
AND C7770

KILLQ2, TAD C7 /POINTS TO LINK
DCA KILLQ-2
TAD I KILLQ-2 /GET LINK
DCA KILLQ-3
TAD KILLQ-2 /NOW RELEASE BLK
JMS RELBK
TAD KILLQ-3 /WAS IT LAST BLK OF QUEUE?
SZA
JMP KILLQ2 /NO, SEARCH NEXT
TAD KILLQ-1 /YES, KILL 'BOOK' TOO
JMS RELBLK

KILLETT, 0
JMP I KILLQ
OUTINES TO CONVERT OCTAL NUMBERS 0-17
/TO EXCESS 40 STRIPPED CODE
/
INPUT OCTAL
OUTPUT OCTAL
/
EXAMPLE: 15 --> 2125
/
07 --> 2027
/
TAD M10
SMA
TAD C70
TAD C2030
/*******************************
*/INPUT OCTAL
/OUTPUT OCTAL
/LEADING ZERO'S BECOME PACKED SPACES
/
EXAMPLE: 15 --> 2125
/
07 --> 0027
/
TAD M10
SMA
TAD C2070
TAD C30
/*******************************
*/INPUT OCTAL
/OUTPUT DECIMAL
/
EXAMPLE: 15 --> 2123
/
07 --> 2027
/
TAD M12
SMA
TAD C66
TAD C2032
/*******************************
*/INPUT OCTAL
/OUTPUT DECIMAL
/LEADING ZERO'S BECOME PACKED SPACES
/
EXAMPLE: 15 --> 2123
/
07 --> 0027
/
TAD M12
SMA
TAD C2066
TAD C32
/*******************************
*/GENERAL CONSTANTS:
/
M10, -10
M12, -12
C30, 30
C32, 32
C66, 66
C70, 70
C2030, 2030
C2032, 2032
C2066, 2066
C2070, 2070
/
68
/INTERUPT ROUTINE FOR REALTIME CLOCK

/TIME AND DATE ARE AVAILABLE IN REGISTERS

MIN
HOUR
DAY
MONTH
YEAR

/OPTIONAL DAY OF WEEK AVAILABLE IN REGISTER

DAYW 1=SUNDAY
2=MONDAY
ETC.

/ENTRY AFTER CLOCK INTERRUPT
/WITH AC AND LINK SAVED

CLOCK, ISZ TICK
JMP EXIT    /RETURN FROM INTERRUPT
TAD M50    /50 Hertz
DCA TICK    /RESET TICK = -50
ISZ SEC
JMP EXIT
TAD M60
DCA SEC    /RESET SEC = -60
ISZ MIN
TAD MIN
TAD M60    /60 MINUTES GONE?
SZA CLA
JMP EXIT    /NO
DCA MIN    /YES, RESET MIN = 0
ISZ HOUR
TAD HOUR
TAD M24
SZA CLA    /24 HOURS GONE?
JMP EXIT    /NO
DCA HOUR    /YES, RESET HOUR = 0
ISZ DAY
TAD DAYW
IAC    /***********
AND C7    /* IF DAY OF WEEK IS
SNA    /* NOT IMPORTANT DELETE
IAC    /* THOSE INSTRUCTIONS
IAC
DCA DAYW    /***********
TAD MONTH
TAD M0NLST    # OF DAY / MONTH
DCA TEMP
TAD DAY
TAD T TEMP
SPA SNA CLA    /LAST DAY OF MONTH?
JMP EXIT

69
IAC  /YES
DCA LAY  /RESET DAY = 1
ISZ MONTH
TAD MONTH
TAD M12
SPA SNA CLA
JMP EXIT
IAC  /LAST MONTH OF YEAR?
DCA MONTH
ISZ YEAR
TAD YEAR
AND C3
SZA CLA
JMP NLY
TAD YEAR
LOOP, CLL
TAD M400
SNL
JMP N400V
SZA
JMP LOOP
JMP NLY-1  /YEAR=MULTIPLE 400
N400V, TAD C100  /LEAP YEAR
SPA
JMP N400V
SZA CLA
CLA CMA
NLY, TAD M28  /SET FEB TO 28 OR 29 DAYS
DCA M0NLST+2
JMP EXIT
/
DECIMAL
M0NLST, M0NLST
-31
-28
-31
-30
-31
-30
-31
-31
-31
-31
-31
/
MIN, 0
HOUR, 0
DAY, 0
DAYW, 0  /DAY OF THE WEEK, OPTIONAL!
MONTH, 0
YEAR, 1973  /
/GENERAL CONSTANTS
/
M50, -50
M62, -60
M24, -24
M28, -28
M12, -12
M400, -400
C100, 100
OCTAL
C7, 7
C3, 3
TICK, 0
SEC, 0
/
GENERAL INTERRUPT RETURN
/
EXIT, CLA CLL
TAD LINK
RAR
TAD ACCU
ION
JMP I 0
/
TEMP, 0
ACCU, 0
PRINT DATE, WEEKDAY AND TIME FROM THE REGISTERS:

MIN /0-59
HOUR /0-23
DAY /1-31
DAYW /1-7
MONTH /1-7
YEAR

USING THE SUBROUTINES:

PRINT MESSAGE
DECIMAL PRINT
STANDARD PRINT ROUTINE

PRDAT, 0
TAD DAYW
TAD DIXTAD
DCA PRTMP
TAD I PRTMP
DCA PRDAT1
JMS PRMESSG

PRDAT1, 0
JMS SPAC
TAD DAY
JMS DPR1T1
2
JMS SPAC
TAD MONTH
TAD MIXTAD
DCA PRTMP
TAD I PRTMP
DCA PRDAT2
JMS PRMESSG

PRDAT2, 0
JMS SPAC
TAD YEAR
JMS DPR1T1
4
JMS SPAC
TAD HOUR
JMS DPR1T1
2
TAD DOT
JMS PRINT
TAD MIN
JMS DPR1T1
2
JMP I PRDAT

PR1MP=.
SPAC, 0
TAD C240
JMS PRINT
JMP I SPAC
/SUBROUTINE TO UNPACK TSS-8 TIME
/
/CALL: JMS UNPTIM /WITH NC=0
/
/RESULT IN REGISTERS:
/    HOUR /0-23
/    MINT /0-59
/    SEC /0-59
/
/THIS ROUTINE ASSUMES THAT THE SYSTEM CLOCK RATE
/IS 20 SYSTEM TICKS PER SECOND,
/IF OTHER, CHANGE CONSTANTS AT UNPTIM UP TO UNPTLM+2
/
/
UNPTIM,0
TAD UNPTLM
TOD
TAD UNP1HA
DCA UNPTPH
TAD UNPTLA
DCA UNPTPL
TAD UNPTOT
DCA UNPTPO
CLA CLL CMA RIL
DCA UNPTCT

UNPTLM, DCA UNPTLM
TAD UNPTLM
CLL
TAD I UNPTPL
DCA UNPTMP
PAL
TAD UNPTH
TAD I UNPTPH
SNL
JMP UNPTND
DCA UNPTH
TAD UNPTLM
DCA UNPTPL
ISZ UNPTLM
/RESTORE HIGH REMAINDER
/RESTORE LOW REMAINDER
/COUNT THIS SUBTRACTION
JMP UNPTLM+1

UNPTND, CLA
TAD UNPTLM
DCA I UNPTPO
ISZ UNPTPH
ISZ UNPTPL
ISZ UNPTPO
ISZ UNPTCT
JMP UNPTIL
JMP I UNPTIM

/GET NUMBER
/PUT IN SPEC. REGISTER
/UPDATE POINTERS
/READY?
/NO, LOOP
/YES, EXIT

/GET TIME OF DAY
/SET POINTERS

/3
/LOW ORDER POINTER
/HI ORDER POINTER
/ADDR. OF HI ORDER CONSTANTS
/ADDR. OF LOW ORDER CONSTANTS
/-# OF TICKS/HOUR=-7200 DEC.
/-# OF TICKS/SEC= - 1200 DEC.
/-# OF TICKS/SEC=- 22 DEC.
/SUBROUTINE TO UNPACK TSS-8 TIME
/
/CALL: JMS UNPTIM /WITH AC=0
/
/RESULT IN REGISTERS:
/HOUR /0-23
/MIN /0-59
/SEC /0-59
/
/THIS ROUTINE WORKS FOR ANY CLOCK RATE
/

UNPTIM,0

ECHR
CIA
DCA UNPTDU
TAD UNPTAD
TOD
TAD UNPTDA
DCA UNPTPD
TAD UNPTOT
DCA UNPTPO
TAD M4
DCA UNPTCT
/PUT IN DIVISOR LIST

UNPTLP, DCA UNPTNM
DCA UNPTNM+1
TAD UNPTL
CLL
TAD I UNPTPD
DCA UNPTMP
SNL
CLA CMA
TAD UNPTH
SNL
JMP UNPTND
DCA UNPTH
TAD UNPTMP
DCA UNPTL
ISZ UNPTNM
SKP
ISZ UNPTNM+1
JMP UNPTLP+2
/RESTORE HIGH REMAINDER

UNPTND, CLA
TAD UNPTNM+1
DCA UNPTH
TAD UNPTNM
DCA I UNPTPO
ISZ UNPTFD
ISZ UNPTPD
ISZ UNPTCT
JMP UNPTLP
/READY?
JMP I UNPTIM
/NO, LOOP
/YES, EXIT
UNPTAL, UNPTH
UNPTNM, 0
0
UNPTCT, 0
UNPTMP, 0
UNPTPD, 0
UNPTOT, UNPTL
HOUR
MIN
SEC
UNPTPD, 0
UNPTDA, UNPTDV
/
DECIMAL
/
UNPTDV, 0
-3600
-60
-1
/
OCTAL
/
UNPTH, 0
UNPTL, 0
HOUR, 0
MIN, 0
SEC, 0
/
M4, -4

/LOW ORDER
/HIGH ORDER
/POINTER IN DLU LIST
/SUBROUTINE TO UNPACK 1558 DATE
/
/RESULT IN REGISTERS
/  DAY  /1-7
/  MONTH /1-12
/  YEAR  /ANY
/
UNPDAT,#
  DATE                   /GET DATE FROM TS8
  DCA DAY
  JMS DATDIV             /DIVIDE BY
  -564                   /564(SUBTRACT)
  TAD TIMES
  TAD C1964
  DCA YEAR
  JMS DATDIV
  -37                    /MONTHS MUST BE +1
  ISZ TIMES
  TAD TIMES
  DCA MONTH
  ISZ DAY
  JMP I UNPDAT
/
/SUBROUTINE DATDIV DEVIDES 'DAY' BY GIVEN NUMBER
/
DATDIV,0
  DCA TIMES             /CLEAR RESULT
  DATDIV, CLL
  TAD DAY
  TAD I DATDIV
  SSL
  JMP DATD2
  DCA DAY
  ISZ TIMES
  JMP DATD1
  DATD2, CLA CLL
  ISZ DATDIV
  JMP I DATDIV
  /EXIT
/
TIMES, 0
C26C, 260
LAY, 0
MONTH, 0
YEAR, 0
/
DECIMAL
C1964, 1964
OCTAL
DECIMAL PRINT WITH VARIABLE NUMBER OF DIGITS

CALL WITH NUMBER TO BE PRINTED IN AC
AND # OF DIGITS TO BE PRINTED
FOLLOWING THE SUBROUTINE CALL.

CALL: TAD NUMBER
JMS DPR1
2 # OF DIGITS TO BE PRINTED (MAX=4)

DPR1, 0
DCA DPRREG /SAVE AC
LCA DPRD /CLEAR # OF PRINTED DIGITS
TAD I DPR1 /FETCH FORMAT
LCA DPRDIG /CORRECT RETURN
ISZ DPR1 /CORRECT RETURN
TAD DPRINS
DCA DPRPTR /SET POINTER
CLA CLL CMA RTL /-3
LCA DPRFAC /FACTORIZE 4 DIGITS
LCA DPRDGT /CLEAR DIGIT

DPRSUB, CLL
TAD DPRREG

DPRPTR, 0
SNL /SUBTRACT
JMP .+4 /NEGATIVE?
DCA DPRREG /STORE RESULT OF SUBTRACTION
ISZ DPRDGT /NO, STEP UP DIGIT
JMP DPRSUB
CLA
TAD DPRDGT /GET DIGIT
TAD DPRD
SZA CLA /PRINT THE DIGIT?
JMP DPRDIN /YES, GO ON
TAD DPRFAC /NO
TAD DPRDIG
SPA SNA CLA /PRINT A SPACE?
JMP DPRTIN /NO
TAD C240 /YES
JMP DPRPR

DPPLIN, ISZ LPRD /FETCH DIGIT
TAD DPRALGT
TAD C260 /CONVERT TO ASCII
DPRPR, JMS PRINT
DPRTIN, ISZ DPRPTR /STEP UP POINTER
ISZ DPRFAC
JMP DPRSUB-1
TAD DPRREG
TAD C260
JMS PRINT
JMP 1 DPR1

79
DPRFAC, 2
DPRD, 0
DPRDIG, 0
DPRFAC, 2
DPRDGTL, 0
DPRINS, TAD DPRP
/
DECIMAL
/
DPRP, -1000
-100
-10
/
OCTAL
/
GENERAL CONSTANTS
/
C263, 260
C240, 240
OCTAL PRINT ROUTINE WITH LEADING SPACES
NONSIGNIFICANT ZERO'S BECOME SPACES
LINK NEEDS NOT TO BE PRESERVED BY 'PRINT' ROUTINE.

/CALL: TAD NUMBER
       JMS OCTPR
       RETURNS WITH AC=0

OCTPR!, 0
   DCA OCTTMP
   TAD M4
   DCA OCTCNT
   CMA
OCTPR!, DCA OCTFIG
   TAD OCTTMP
   RAL
   RTL
   DCA OCTTMP
   TAD OCTTMP
   RAL
   AND C7
   ISZ OCTCNT
   JMP +4
   TAD C260
   JMS PRINT
   JMP OCTPR0

SNA
   ISZ OCTFIG
   JMP +4
   TAD C240
   JMS PRINT
   JMP OCTPR0-1

   TAD C260
   JMS PRINT
   JMP OCTPR0

OCTTMP, 0
OCTCNT, 0
OCTFIG, 3
+4, -4
C7, 7
C240, 240
C260, 260
DOUBLE WORD OCTAL PRINT ROUTINE

CALL: JMS DOCTPR
HI ORDER PART
LOW ORDER PART
RETURN AC=0

DOCTPR, 0
CLA
TAD I DOCTPR
ISZ DOCTPR
DCA OCTTM1
TAD I DOCTPR
ISZ DOCTPR
DCA OCTTM2
TAD M10
DCA OCTCNT
CMA

OCTPH1, DCA OCTFIG
JMS DELROT
JMS DELROT
JMS DELROT
TAD OCTTM2
RAL
AND C7
ISZ OCTCNT
JMP +4
TAD C260
JMS PRINT
JMP I DOCTPR

SNA
ISZ OCTFIG
JMP +4
TAD C240
JMS PRINT
JMP OCTPH1-1
TAD C260
JMS PRINT
JMP OCTPH1

OCTTM1, 0
OCTTM2, 0
OCTCNT, 0
OCTFIG, 0
M13, -10
C7, 7
C240, 240
C260, 260

DELROT, 0
TAD OCTTM2
RAL
LCA OCTTM2
TAD OCTTM1
RAL
LCA OCTTM1
JMP I DELROT
/ASCII STRING GENERATOR
/GENERATES THE 64-CHARACTER ASCII SET IN A STRING
/TERMINATED WITH CARRIAGE RETURN, LINE FEED
/
/CALL: CLA
/ JKS GENSTR
/ RETURN (AC=0)

0 /END OF PARTIAL STRING
0 /POINTER
0 /TEMPORARY STORAGE

GENSTR: 
TAD GENLST
DCA GENSTR-2/INITIALIZE POINTER
TAD GENSTR-2
TAD GENLTH
DCA GENSTR-1
TAD I GENSTR-1/GET NEXT CHARACTER
DCA GENSTR-1
ISZ GENSTR-2/INCREMENT POINTER
TAD I GENSTR-2
SNA /END OF LIST ?
JMP I GENSTR /YES
CIA
DCA GENSTR-3/SET UP TO TEST

GENLP,
TAD GENSTR-1
JMS PRINT
ISZ GENSTR-1/CHARACTER +1
TAD GENSTR-1
TAD GENSTR-3
SPA SNA CLA /EQUAL TO POINTED CHARACTER ?
JMP GENLP
JMP GENSTR+3/YES, GET NEXT CHARACTER

GENLTH, GENNXT-GENLST
GENLST, 

332 /Z
271 /9
257 /SLASH
300 /0
337 /-
215 /RETURN
212 /LINE FEED
0 /TERMINATOR

GENNXT, 301 /A
260 /0
240 /SPACE
272 /:
333 /L
215 /RETURN
212 /LINE FEED
/INCREMENT DOUBLE PRECISION COUNTER
/WITH OVERFLOW RETURN (AFTER 16,777,216 INCREMENTS)
/
/CALL: JMS  LPINC
/
  DPCNTR  /ADDRESS OF COUNTER
  COUNTER OVERFLOWED RETURN (AC=0)
  NORMAL RETURN (AC=0)
/
  DPCNTR> 0  /HIGH ORDER PART
  0  /LOW ORDER PART
/

0  /TEMPORARY ADDRESS

DPINC, .-.  
CLA  CLI  IAC  /SET UP TO GET ADDRESS OF
TAD I  DPINC  /LOW ORDER PART
DCA  DPINC-1
ISZ I  DPINC-1
JMP  DPINC1  /NO OVERFLOW
CML
SZL
JMP  DPINC+2  /INCREMENT HIGH ORDER PART
SKP  /HIGH ORDER PART OVERFLOW

DPINC1, ISZ  DPINC
ISZ  DPINC
JMP I  DPINC