PDP PROGRAM DOCUMENTATION SERIES
VOLUME 1: PDP8

CONTENTS:

GENERAL DESCRIPTION AND CLASSIFICATION SYSTEM
INQUIRY FORM FOR CONTRIBUTORS
1. SYSTEMS AND SYSTEM PROGRAMS
2. APPLICATION PROGRAMS
3. HARDWARE TEST- AND DEMONSTRATION-PROGRAMS
INDEX
GENERAL_DESCRIPTION_AND_CLASSIFICATION_SYSTEM

THIS MEMORANDUM CONTAINS THE DESCRIPTION OF A SIMPLE DOCUMENTATION SYSTEM OF THE SOFTWARE THAT CAN BE USED, OR IS UNDER DEVELOPMENT FOR USE ON THE PDP-COMPUTER INSTALLATION AT THE Mc.

THE COMPUTER INSTALLATION CONSISTS OF ONE 16 M-PDP8/+ AND ONE 16 M-PDP8/E WITH E/E, AND ONE 48 M-PDP11/45 MUTUALLY CONNECTED THROUGH FAST /&/-LIKE INTERFACES.

TO THE PDP8/+ ARE CONNECTED THE FOLLOWING PERIPHERALS:
2 HIGH SPEED READERS;
2 HIGH SPEED PUNCHES;
1 LINE PRINTER;
1 ASR35 TELETYPE;
2 OLIVETTI TERMINALS AND A MODEM;
1 BRAILLEPRINTER;
1 KV08 VISUAL DISPLAY AND JOYSTICK;
1 PLOTTER;
1 RF08 DISC UNIT;
2 RE8 TAPE UNITS;
3 D/A-CONVERTERS.

TO THE PDP8/E ARE CONNECTED:
1 RM8 DISC UNIT.

TO THE PDP11/45 ARE CONNECTED:
1 WA30 DECWRITER;
1 GP40 VISUAL DISPLAY (INCLUDING A PDP11/05 PROCESSOR WITH 8K MEMORY);
1 MRB1 LASER SCAN DISPLAY/PLOTTER;
1 DQ19A HIGH-SPEED DATACOMMUNICATION UNIT;
2 RM05 DISC UNITS.

THE DOCUMENTATION SYSTEM WILL CONSIST OF A LOOSE-LEAF SERIES OF PROGRAM DESCRIPTIONS. THE PROGRAM DESCRIPTIONS WILL ALL BE IN THE ENGLISH LANGUAGE, AND CONFORM TO A STANDARD FORMAT. THE DOCUMENTATION SYSTEM WILL CONTAIN TWO VOLUMES, ONE VOLUME FOR THE PDP8 SERIES AND ONE VOLUME FOR THE PDP11 SERIES. THE VOLUMES ARE DISTRIBUTED SEPARATELY.

CHANGES AND ADDITIONS WILL BE DISTRIBUTED AT IRREGULAR INTERVALS AMONG THE USERS OF OUR COMPUTER INSTALLATION AND OTHER INTERESTED PEOPLE.

ALL USERS OR OTHER PROGRAMMERS WHO WANT THEIR PROGRAM(S) TO BE ADDED TO THE DOCUMENTATION SYSTEM MUST PRODUCE A DESCRIPTION IN A STANDARD FORM. THE DOCUMENTATION OF EACH PROGRAM WILL BE SUBMITTED TO A CRITICAL REVIEW BY THE EDITORS. IN ORDER TO FACILITATE THE STANDARDIZATION, EACH CONTRIBUTOR WILL RECEIVE AN INQUIRY FORM, WHICH HAS TO BE FILLED OUT AS PART OF THE DOCUMENTATION.
THE SYSTEM ALLOWS FOUR MAIN TYPES OF DOCUMENTATION AS FOLLOWS:

1. **REQUIRED PROGRAM-DOCUMENTATION**: THIS TYPE IS MEANT FOR PROGRAMS THAT ARE EXTENSIVELY DESCRIBED OUTSIDE THE DOCUMENTATION SYSTEM, FOR INSTANCE: MANUFACTURERS SOFTWARE (IN MANUALS), OWN PROGRAMS THAT ARE SUBJECT OF A SO-CALLED MG+MW-REPORT ETC.

2. **NONREPORTED PROGRAM-DOCUMENTATION**: MEANT FOR PROGRAMS OF WHICH THE DESCRIPTION IN THE SYSTEM IS THE ONLY EXISTING DOCUMENTATION, THIS IS THE DEFAULT TYPE; IT IS ASSUMED WHEN THE TYPE IS NOT MENTIONED.

3. **PRELIMINARY DESCRIPTIONS**: THIS TYPE IS USED FOR:
   - PROGRAMS UNDER DEVELOPMENT, INFORMATION THAT FOR SOME REASON OR ANOTHER MUST BE SUPPLIED TO INTERESTED PEOPLE, E.G. EXTERNAL SPECIFICATIONS FOR FUTURE USERS, OR OPERATING INSTRUCTIONS FOR A PRE-RELEASE.
   - IDEAS FOR PROGRAMS THAT SHOULD BE MADE, OR SHOULD BE IMPROVED.

4. **REMARKS AND RECOMMENDATIONS** NOT NECESSARILY IN THE FORM OF A PROGRAM (FOR INSTANCE AN ERROR REPORT).

FROM THESE FOUR TYPES OF DOCUMENTATION THE MAIN PURPOSES OF THE SYSTEM CAN BE DERIVED AS:

- TO SUPPLY A MANAGEABLE OVERVIEW OF WHAT IS (C.Q. WILL BE) POSSIBLE ON THE INSTALLATION;
- TO FACILITATE THE EXCHANGE OF PROGRAMS AND PROGRAMMING CONCEPTS IN THE EARLIEST STAGE;
- TO SUPPLY SOFTWARE INFORMATION TO USERS OUTSIDE OF THE MATHEMATICAL CENTRE.

**CLASSIFICATION OF PROGRAMS**

1. **THE DOCUMENTATION PARAGRAPHS**

THE DESCRIPTION OF A PROGRAM CONSISTS OF THE FOLLOWING PARAGRAPHS:

A) **FUNCTIONAL DESCRIPTION**: SHORT DESCRIPTION (ABSTRACT) OF THE PURPOSE OF THE PROGRAM AND THE MAIN WORKING PRINCIPLES, THIS DESCRIPTION IS THE GUIDE LINE FOR THE SO-CALLED FUNCTION-CLASSIFICATION, WHICH WILL BE FULLY DESCRIBED IN THE NEXT SECTION.

B) **DOCUMENTATION AND/OR REFERENCES**: THIS PARAGRAPH CONTAINS ALL (KEYS TO) THE INFORMATION ABOUT THE WAY THE PROGRAM HAS TO BE USED, I.E. SOFTWARE- AND HARDWARE RESOURCES, REAL-TIME REQUIREMENTS, A USERS MANUAL ETC. THIS INFORMATION ALSO LEADS TO A CLASSIFICATION, THE SO-CALLED ENVIRONMENT CLASSIFICATION, THIS ENVIRONMENT CLASSIFICATION IS SPECIFIED IN SECTION 3.
C) CORRECTIONS AND CHANGES: THESE MAY CONSIST OF:

1. CORRECTIONS-ON-THE-DOCUMENTATION. THIS PARAGRAPH REMAINS ONLY TEMPORARILY WITHIN THE SYSTEM, AT FIXED INTERVALS IT WILL BE REMOVED WHEN THE REAL DOCUMENTATION IS UPDATED. UPDATING AND REPRODUCTION OF THE DOCUMENTATION WILL BE SIMPLIFIED BY KEEPING A COPY ON DEGTAPE THAT ALLOWS ON-LINE EDITING.

2. PROGRAM CHANGES AND PROGRAM CORRECTIONS. THE SO-CALLED UPDATES, THESE CAN BE LOCAL VARIANTS OF IMPORTED PROGRAMS, THEY REMAIN IN THE SYSTEM AS THEY ARE.

IN THE CASE OF HOME-MADE PROGRAMS WE MUST DECIDE BETWEEN REPORTED AND NONREPORTED DOCUMENTATIONS, FOR THE FORMER A COMPLETE NEW VERSION MAY FOLLOW OR THE UPDATE MAY REMAIN. THIS WILL BE DECIDED BY THE EDITORS AFTER CONSULTING THE PROGRAMMER FOR EACH CASE SEPARATELY. FOR THE LATTER CASE THE REMARKS OF 1. APPLY. FOR PRELIMINARY DESCRIPTIONS ONLY CORRECTIONS WILL BE PRODUCED.

THE DOCUMENTATION CAN BE RETRIEVED BY MEANS OF THE PROGRAM-INDEX. THIS IS AN INDEX TO ALL PROGRAMS IN THE DOCUMENTATION, PROGRAMS WITH EQUAL FUNCTION CLASSIFICATION ARE SUB-ORDERED ACCORDING TO THE ENVIRON CLASSIFICATION, AT EACH PERIODIC REVISION A COMPLETELY NEW INDEX IS GENERATED.

2. FUNCTION CLASSIFICATION.

THE FUNCTION CLASSIFICATION CONSISTS OF FOUR GROUPS THAT ARE DIVIDED INTO SUBGROUPS ETC.

THE FOUR MAIN GROUPS ARE:

1. SYSTEM PROGRAMS: ALL PROGRAMS THAT CONTRIBUTE TO THE FUNCTIONING OF AN OPERATING SYSTEM. SOME FUNCTIONS OF AN OPERATING SYSTEM ARE: INPUT-, STORAGE-, COMPILATION-, CONVERSION-, EXECUTION OF PROGRAMS AND ALSO THE MANAGEMENT OF ALL RESOURCES. THESE TYPES OF FUNCTIONS ARE REFLECTED IN THE SUBCLASSIFICATION.

2. APPLICATION PROGRAMS: PROGRAMS THAT PERFORM SOME USEFUL TASK FOR A RESTRICTED GROUP OF USERS. THERE ARE AS MANY SUBGROUPS AS THERE ARE APPLICATION FIELDS. THE DOCUMENTATION SYSTEM SHOULD BE EXTENDIBLE ON THE SUBGROUP LEVEL, FOR INSTANCE FOR THE OPENING OF A NEW APPLICATION AREA.

3. HARDWARE-TEST- AND DEMONSTRATION PROGRAMS: PROGRAMS TO CHECK THE CORRECT FUNCTIONING OF THE HARDWARE. THIS INCLUDES THE MANUFACTURERS TEST SOFTWARE, PROGRAMS THAT ILLUSTRATE THE WORKING PRINCIPLES OF A HARDWARE MODULE ARE ALSO INCLUDED. THEY CAN BE CONSIDERED AS ADDITIONAL HARDWARE DOCUMENTATION. THE DIVISION IN GROUPS FOLLOWS SOME AD HOC HARDWARE CLASSIFICATION.

4. TIPS: TIPS AND OBSERVATIONS ABOUT THE USE OF BOTH HARDWARE AND SOFTWARE, THAT ARE NOT (YET) IN THE FORM OF A COMPLETED PROGRAM. WARNING: THIS CATEGORY IS DEFINITELY NOT INTENDED FOR ERROR REPORTING. ERRORS WILL HAVE TO BE REPORTED UNDER THE CLASSIFICATION OF THE ERRONEOUS PROGRAM.

THE DIVISION IN SUBGROUPS IS NOT FIXED AND AS NEED ARISES NEW SUBGROUPS MAY BE ADDED OR EXISTING SUBGROUPS MAY BE SPLIT OR SUBDIVIDED.
THE FUNCTION CLASSIFICATION SCHEME

1. SYSTEM PROGRAMS
   1.0 MISCELLANEOUS
   1.1 MONITORS, SCHEDULERS, COMPLETE SYSTEMS
   1.2 FILE HANDLERS AND FILE UTILITIES
   1.3 EDITORS
   1.4 ASSEMBLERS AND MACROPROCESSORS
   1.5 COMPILERS
   1.6 INTERPRETERS
   1.7 LOADERS
   1.8 I/O-DRIVERS AND -UTILITIES
   1.9 TRACERS AND DEBUGGERS
   1.11 GENERAL UTILITIES E.G. CONVERSION ROUTINES
      DUMPS
      CROSS REFERENCES

2. APPLICATION PROGRAMS
   2.0 MISCELLANEOUS
   2.1 MATHEMATICAL APPLICATIONS
      2.1.0 MISCELLANEOUS
      2.1.1 NUMERICAL
      2.1.2 STATISTICAL
      2.1.3 COMBINATORIAL
      2.1.4 OPERATIONS RESEARCH
   2.2 TEXT PROCESSING
   2.3 GRAPHICAL APPLICATIONS
   2.4 SIMULATION
      2.4.1 RANDOM GENERATORS
   2.5 DATA-ACQUISITION
   2.6 EDUCATIONAL
   2.7 ADMINISTRATION
   2.8 INFORMATION RETRIEVAL
   2.9 ARTIFICIAL INTELLIGENCE
   2.11 NON-MATHEMATICAL UTILITIES
   2.11 GAMES

3. HARDWARE TEST AND DEMONSTRATION PROGRAMS
   3.0 MISCELLANEOUS
   3.1 GENERAL PROCESSORS
   3.2 BACKGROUND MEMORY
   3.3 BACKGROUND MEMORY
      3.3.1 DISC UNITS
      3.3.2 MAGNETIC TAPE UNITS
   3.4 COMMUNICATION PERIPHERALS
   3.5 LOW SPEED (HARDCOPY) INPUT/OUTPUT
      3.5.1 TELETYPES
      3.5.2 PRINTERS
      3.5.3 READERS
      3.5.4 PUNCHES
   3.6 I/O- AND B/A- CONVERTERS
   3.7 BLOCKS
   3.8 GRAPHICAL DEVICES
   3.9 FLOATING POINT PROCESSORS
3. THE_ENVIRONMENT_CLASSIFICATION_SCHEMA

THIS SCHEME CONTAINS A NUMBER OF CATEGORIES, EACH CATEGORY CONTAINS A NUMBER OF ATTRIBUTES THAT MAY BE SELECTED. SELECTION CONSISTS OF DEFINING THE SUBSET OF ATTRIBUTES THAT APPLY TO THE PROGRAM AT HAND. PRIOR TO SELECTION THE CATEGORY CAN BE DECLARED NOT RELEVANT, IN THAT CASE THE SUBSET REMAINS EMPTY. IF THE SUBSET IS EMPTY BUT THE CATEGORY IS RELEVANT, THE MISSING ATTRIBUTES MUST BE SPECIFIED IF POSSIBLE.

CATEGORY 1: COMPUTER: SPECIFY THE MACHINES ON WHICH THE PROGRAM RUNS.

1.1 PDP8  
    PDP8/0  
    PDP8/1  
    PDP8/E  
    PDP8/A

1.2 PDP11/05  
    PDP11/20  
    PDP11/40  
    PDP11/45  
    PDP11/50  
    PDP11/70


CATEGORY 2: OPERATING_SYSTEM: SPECIFY THE OPERATING SYSTEM ON WHICH THE PROGRAM RUNS.

2.1 MS   
    09/8 (PO/8, 09/12)  
    W.ER  
    RM8  
    RM8  
    RM8+8  
    CBE

2.2 DO9  
    CONTEX  
    R711  
    RX11-D  
    RX11-M (UNMAPPED VERSION)  
    RX11-M (MAPPED VERSION)  
    UNK

CATEGORY 3: INPUT/OUTPUT-RESOURCES: SPECIFY PHYSICAL DEVICES, FILES AND OTHER TYPES OF DATA THAT ARE PRODUCED OR CONSUMED BY THE PROGRAM.

3.1 DEVICES FOR INPUT, OUTPUT, INTERACTION AND BACKUP. (In THAT ORDER)
3.2 FILES FOR INPUT, OUTPUT, INTERACTION AND BACKUP.
3.3 COMMUNICATION WITH OTHER PROGRAMS OR WITH AN OPERATOR (E.G., MESSAGES, PARAMETER LISTS, DATA BLOCKS ETC.
3.4 OTHER RESOURCES, (E.G., CORE MEMORY)

CATEGORY 4: PROGRAMMING_LANGUAGE:

4.1 BINARY CODE
4.2 ASSEMBLY CODE
4.3 MACRO-LANGUAGE
4.4 HIGHLEVEL LANGUAGE
4.5 NATURAL LANGUAGE
CATEGORY 5: PREREQUISITE PROGRAMS OR DATA: SPECIFY ALL PROGRAMS THAT MUST RUN SIMULTANEOUSLY AND ALL DATA THAT HAVE TO BE AVAILABLE IN ADVANCE. (NOT THE OPERATING SYSTEM.)

5.1 ROUTINES FROM PROGRAM LIBRARY.
5.2 DATA PRODUCED BY OTHER PROGRAMS. (SPECIFY BOTH DATA AND PROGRAMS.)
5.3 OTHER PREREQUISITES.

CATEGORY 6: PHYSICAL PROPERTIES:

6.1 PROGRAM LENGTH: WITH AND/OR WITHOUT OVERLAYS.
6.2 DATA SPACE: FOREGROUND AND BACKGROUND.
6.3 EXECUTION TIME: TOTAL AND/OR PER INVOCATION.
6.4 CPU-LOAD AND CORE LOAD (DYNAMIC).
6.5 REAL-TIME BEHAVIOUR. (PRIORITY, INTERRUPTABILITY ETC.)

GENERAL INFORMATION.

THE TWO CLASSIFICATION SCHEMES OF THE PREVIOUS SECTION SERVE TO GIVE AN IMPRESSION ABOUT THE FUNCTION OF THE PROGRAM AND THE ENVIRONMENT IN WHICH IT MUST BE USED. THE READER OF THE DOCUMENTATION SYSTEM CAN USE THESE CLASSIFICATIONS IN ORDER TO FIND OUT QUICKLY WHETHER A CERTAIN TYPE OF PROGRAM IS AVAILABLE OR NOT.

A TOTALLY DIFFERENT TYPE OF DOCUMENTATION IS NECESSARY FOR SOMEONE WHO WANTS TO USE A PROGRAM. TO THIS END THE FOLLOWING INFORMATION MUST BE ADDED:

1. TITLE OF THE PROGRAM.
2. NAME AUTHOR(S).
3. NAME REVISOR(S).
4. ORIGIN OF THE PROGRAM.
5. DATE.
6. DOCUMENTATION TYPE (REPORTED, NOT REPORTED, PRELIMINARY).
7. SHORT DESCRIPTION OF
   A) FUNCTION
   B) WORKING PRINCIPLES
   C) PERFORMANCE.
8. DIRECTIONS FOR USE: N.B., FOR EACH SUBTITLE A COMPLETE SET OF DIRECTIVES MUST BE GIVEN OR REFERRED TO.

8.1 CONDITIONS:
   - OPERATING SYSTEM
   - PROGRAM TYPE:
     A) MAIN PROGRAM (SELF CONTAINED)
     B) SUBRoutines
     C) PROGRAM MODULE
   - LANGUAGE TYPE
   - RESOURCES:
     A) DEVICES
     B) FILES
     C) WORKING SPACE.
8.2 OPERATING INSTRUCTIONS:
   - STARTING SEQUENCE
   - ON-LINE SEQUENCE
   - STOP SEQUENCE AND COLLECTING OF RESULTS.
8.3 EXAMPLES:
   - ILLUSTRATION OF THE WORKING.
   - ILLUSTRATION OF THE USAGE.
8.4 BEELZEBUB:
   1. WINDIRECT (REFERENCE TO PUBLICATION ELSEWHERE).
   2. SOURCE TEXT (ONLY IF VERY SHORT),
--- INQUIRY FORM FOR CONTRIBUTORS. ---

1. PROGRAM TITLE

2. ORIGIN 2.1 AUTHOR(S)
   2.2 REVISOR(S)
   2.3 INSTITUTE

2.4 DATE

3. DOCUMENTATION TYPE

3.1 NEW DOCUMENTATION:
   REPORTED | NON REPORTED | PRELIMINARY

3.2 ADDITIONAL DOCUMENTATION:
   CORRECTION | REVISION | ADDENDUM

3.3 EXPERIENCES:
   ERROR REPORT | UNDOCUMENTED USE |

3.4 IDEAS AND REMARKS:
   ON EXISTING PROGRAMS | FOR A NEW PROGRAM

3.5 REFERENCES TO OTHER DOCUMENTATION:

4. FUNCTION CLASSIFICATION: NUMBER: TITLE:

4.1 PROGRAM FUNCTION:

4.2 WORKING PRINCIPLES:

4.3 PERFORMANCE:

5. ENVIRONMENT CLASSIFICATION

5.1 COMPUTER(S)

5.2 OPERATING SYSTEM(S)

5.3 INPUT/O OUTPUT SPECIFICATIONS:

5.3.1 DEVICES:
   INPUT
   OUTPUT
   INTERACTION

5.3.2 FILES:
   INPUT
   OUTPUT
   INTERACTION

5.3.3 OTHER I/O:
   CORE
   PARAMETERS
   MISCELLANEOUS

5.4 LANGUAGE:

5.5 PROGRAM TYPE:
   MAIN PROGRAM | PROCEDURE OR SUBROUTINE | FRAGMENT | MODULE | OTHER

5.6 PHYSICAL PROPERTIES:
   CORE USAGE:
   RUNTIME: TOTAL: PER INVOCATION
   CPU LOAD: FPP USED:

REAL TIME SPECIFICATIONS:

*SELECT ONE ALTERNATIVE
6. **DIRECTIONS FOR USE:**

6.1 **CONDITIONS:**

6.1.1 MACHINE SETTING

6.1.2 OPERATING SYSTEM SETTING

6.1.3 OTHER PROGRAMS NEEDED:

(LIBRARY)

6.1.4 DATA NEEDED: PRODUCED BY:

FORMAT:

6.1.5 OTHER CONDITIONS:

6.2 **OPERATION:**

6.2.1 START SEQUENCE

6.2.2 ON-LINE SEQUENCE

6.2.3 RESULT SEQUENCE

6.3 **EXAMPLE OF USAGE AND RESULTS**

---

**DESCRIPTION OF DOCUMENTATION:**

**MEDIUM:**

- CARD DECK
- PAPER TAPE(S)
- DECTAPE
- CASSETTE
- PERMANENT FILE ON: SARA (SPECIFY TYPE)
- OTHER:

**NOTE:** HANDWRITTEN OR TYPED DOCUMENTS WILL BE ACCEPTED ONLY IN EXCEPTIONAL CASES.

**FILENAME:** (IF APPLICABLE):

**CODE:**

- MC FLEXOWRITER CODE
- ARBA CODE (SPECIFY THE TYPING BALL REQUIRED)
- SCOPE DISPLAY CODE
- AS99 CODE
- +BM 026 CARD CODE
- +BM 029 CARD CODE
- OTHER:

**IF THE CODE IS NOT AMONG THE LISTED ONES THEN SUPPLY A DESCRIPTION OF THE CODE OR A CONVERSION PROGRAM,**
PDP PROGRAM DOCUMENTATION SERIES

CHAPTER 1: SYSTEMS AND SYSTEM PROGRAMS
NAME: PIP
MODIFICATION BY: A.E., BROUWER, MATH, CENTR., A'DAM
DATE: JANUARY 1972
ENVIRONMENT: OS/8 OPERATING SYSTEM

PIP - CHANGE DEFAULT LISTING DEVICE

MOST VERSIONS OF THE OS/8 PROGRAM PIP HAVE TTY: FOR DEFAULT DIRECTORY LISTING OUTPUT DEVICE, E.G., THE COMMAND LINE
*/E
IS EQUIVALENT TO
*TTY:**/E
TO CHANGE THIS DEFAULT ONE HAS TO REPLACE THE HASHED CODE FOR TTY: BY THE HASHED CODE OF THE DESIRED DEVICE.

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>HASHCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTY:</td>
<td>5524</td>
</tr>
<tr>
<td>LPT:</td>
<td>4020</td>
</tr>
<tr>
<td>TV:</td>
<td>2426</td>
</tr>
<tr>
<td>SC:</td>
<td>2303</td>
</tr>
<tr>
<td>DPL:</td>
<td>6020</td>
</tr>
</tbody>
</table>

IN GENERAL THE HASHCODE IS DETERMINED AS FOLLOWS:
THE DEVICE NAME IS PACKED (IN 6BIT TRIMMED ASCII) INTO ONE OR TWO WORDS (PADDLED WITH A 6BIT ZERO IF ITS NUMBER OF SYMBOLS IS ODD), IF THE DEVICENAME CONSISTS OF ONE OR TWO LETTERS OR DIGITS THEN THE HASHCODE EQUALS THE FIRST (AND ONLY) WORD. IF IT CONTAINS THREE OR FOUR LETTERS OR DIGITS THEN THE CODE EQUALS THE SUM OF BOTH WORDS 0R'I'ED WITH 4000.

PATCH:
.GET SYS PIP
.ODT
12547/5524 2303
+C
.SAVE SYS PIP

IF THE LOCATION SHOWN DOES NOT CONTAIN 5524 YOU MAY SEARCH FOR 5524 IN APPROPRIATE ENVIRONMENT:
.GET SYS PIP
.ODT
F/0000 10
5524w
10566/5524
12546/5524
17155/5524
12545/0012
12546/5524 2303
+C
.SAVE SYS PIP

SEARCH IN FIELD 1
FOR TTY HASH CODE
FOUND AT SOME WILD PLACES
THE PRECEDING LOC SHOULD CONTAIN 0012
CHANGE THE DEVICE CODE
NAME: MCEdit
VERSION: MCEdit AB-V14
AUTHOR: DEC
REVISION AND EXTENSION BY: A.E. BROUWER, MATH. CENTR., A'DAM
DATE: JANUARY 1975
TYPE: REPORTED ELSEWHERE (SEE DEC MANUALS)
ENVIRONMENT:
    COMPUTER: 8/1, 8/E, 12
    OPER. SYSTEM: OS/8, OS/12
    I/O DEVICES: TELETYPET, OPTIONAL KB DISPLAY
    MEMORY REQUIRED: 8K

DESCRIPTION:
MCEdit is the MC version of the DEC PS/8 System Program EDIT.002.
For a description of EDIT see the appropriate DEC Manuals.

THE MAIN DIFFERENCES ARE:

(1) ALL KNOWN BUGS HAVE BEEN ELIMINATED.
    [BUT NOTE: THE CONDITION 'FULL' MAY STILL CAUSE SOME TROUBLE
     IN EXCEPTIONAL CASES.]

(11) KB8/1 SCOPE DISPLAYROUTINES HAVE BEEN ADDED,
     AT ANY TIME (IN COMMAND MODE OR IN TEXT MODE)
     THE COMMANDS $S, $T AND $V CAN BE GIVEN,
     [HERE AND IN THE FOLLOWING $X MEANS CTRL/X FOR ANY LETTER X]

$S: SWITCH TO SCOPE MODE; ALL CHARS ARE ECHOED ON THE
    KB8/1 SCOPE INSTEAD OF ON THE TTY.
    (THERE ARE A FEW EXCEPTIONS: THE MESSAGES '?NAC',
     AND 'FULL' AND 'SURE?' ARE SENT ALWAYS TO THE
     TTY TO ENSURE THEM BEING NOTICED. ALSO WHEN
     MCEdit WANTS TO RING THE BELL (LINE TOO LONG,
     BUFFER FULL) THE CODE 207 IS SENT TO THE TTY.)

$T: SWITCH TO TELETYPETE MODE

$V: ERASE THE SCREEN AND INITIALIZE COORDINATES TO THE
     UPPER LEFT HAND CORNER. (AFTER EACH ERASE THE COORDINATES
     OF THIS STARTING POINT ARE SLIGHTLY CHANGED IN ORDER TO
     PREVENT SCREENBURN.)

IN TELETYPETE MODE MCEdit BEHAVES LIKE EDIT, BUT IN SCOPE MODE
MCEdit DISPLAYS LINE NUMBERS BEFORE EACH LINE. ALSO, IF /C
HAS BEEN GIVEN, CAPITALS ARE DISPLAYED LARGER THAN THE
OTHER SYMBOLS.

(III) A GARBAGE COLLECTOR COMPACTS THE TEXTBUFFER AREA WHEN
     IT IS NEARLY FULL. IT RINGS THE BELL THREE TIMES WHEN THIS GARBAGE
     COLLECTOR WAS UNSUCCESSFUL. AT THIS MOMENT STILL A FEW LINES
     CAN BE ADDED BUT MCEdit SHOWS ITS RELUCTANCE BY RETURNING TO
     COMMAND MODE AFTER EACH LINE ADDED. THE USER SHOULD GIVE A
     DELETE, KILL OR WRITE COMMAND NOW.
(IV) The command \texttt{W} has been added ("Write"), \texttt{M,N W} with integer expressions \texttt{M} and \texttt{N} is (when legal) equivalent to \texttt{M,N P} followed by \texttt{M,N D}, as was to be expected, \texttt{M W} means \texttt{M,M W} and \texttt{W} means \texttt{1/W}. \texttt{I,E P} followed by \texttt{K}.

(This command is much more useful than \texttt{P} since the command \texttt{P} duplicates some information which must be deleted subsequently.)

(V) The search command has been extended to allow searching for the begin or the end of a line. It is also possible to move text from the end of one line in front of the next line (and in particular: to merge lines).

To the search command it looks as if each line begins with a new line and ends with a carriage return, therefore searching for \texttt{AB} finds the beginning of a line (and allows for instance to insert a tag without having to delete and restore the first character of a line) and searching for \texttt{AE} finds the end of a line (and allows easy insertion of comment in a PAL program: just search for \texttt{AE} through the entire program).

To merge two lines: search for \texttt{AE} (now the position is before the carriage return of the first line), then type \texttt{AB} (this deletes the carriage return and leaves the position as it is). At this moment you can insert a space (or whatever you like) or search for another character (e.g., to split the long line just obtained by typing LF) or give Ctrl/Form when ready.

[Note: merging of lines is impossible in edit.]
[Note: the characters \texttt{AB} and \texttt{AE} are not inserted in the textbuffer or in the output file, on the other hand, if the file contains a \texttt{AB} or \texttt{AE} already, these real characters are found also by search (and are treated correctly).]

(VI) ISO code (7-bit ASCII with parity bit) is supported. Specifying the /I option to the command decoder causes all letters to be interpreted as lower case letters. To get an upper case char, type \texttt{A} followed by the letter. (This can be used to edit files containing both upper and lower case chars, but is too cumbersome for creating them) /I implies /C, without /I a \texttt{AA} is ignored.

[Note: in the previous version \texttt{AH} was the upper case shift but the current version recognizes \texttt{AH} as backspace.]
[Note: after /I the code 337 is interpreted (and shown) as underline instead of back arrow.]

(VII) Minor changes have been made to almost every part of the program (Rubout works correctly, \texttt{J} is possible without outputfile, \texttt{K} is ignored when it has arguments etc.)
NAME: EDIT
CONTRIBUTOR: A.E. BROUWER, MATH. CENTR., A'DAM
TYPE: HINT
ENVIRONMENT: OS/8 OPER. SYSTEM

THE CONDITION 'FULL' KEEPS GIVING TROUBLE TO PEOPLE NOT USED TO IT.
THE APPROPRIATE ACTION IS SIMPLE BUT A MISTAKE MAY DESTROY YOUR
SOURCE FILE.

THE GENERAL PATTERN IS AS FOLLOWS:

```
.R EDIT
*A<
#
...
FULL
*DTA1:A2<
#
...
#E
.R PIP
*GARBAGE</D
*A<DTA1:A2
*DTA1:A2</D$.
```

REMARKS:
1. DON'T PANIC! DO NOT TYPE *C TO THE COMMAND DECODER SINCE THIS
   WILL DESTROY THE REMAINDER OF YOUR INPUT FILE.
   (FOR: EDIT CLOSES ITS OUTPUT BEFORE CALLING CD.)
2. WHILE MERGING WITH PIP DO NOT USE /1. /1 LEAVES THE CTRL/Z IN
   THE MIDDLE AND THE SECOND HALF IS EFFECTIVELY LOST.
3. IF YOU SPECIFY A NEW OUTPUTFILE ON THE SAME DEVICE AS THE
   PREVIOUS OUTPUTFILE AND THE INPUTFILE OF THE SAME NAME, THEN
   GIVE #E IMMEDIATELY (IF NOT GIVEN ALREADY) SINCE EXTENSIVE
   EDITING MAY CAUSE YOU TO WRITE IN YOUR INPUT.
4. IF YOU HAD MORE THAN ONE INPUT FILE THEN SPECIFY TO THE
   COMMAND DECODER EXACTLY THE SAME SEQUENCE OF INPUTFILES.
   (THE CALL TO THE COMMAND DECODER ERASES ALL INPUT SPECIFICATIONS
   EXCEPT FOR THE FILE CURRENTLY OPEN.)
   (THERE IS A PROBLEM HERE: AFTER
   *A<P,Q,A,R
   #E
   FULL
   *
   THERE IS NO WAY TO RECOVER THE ORIGINAL INPUT FILE A UNLESS IT
   WAS COPIED ALREADY TO THE OUTPUT. IN GENERAL ONE SHOULD AVOID
   GIVING THE OUTPUT FILE THE SAME NAME AS AN INPUT FILE DIFFERENT
   FROM THE FIRST ONE. (OF COURSE THIS IS A BUG IN EDIT.)
5. IT IS POSSIBLE TO GIVE /D ON THE LINE TO THE COMMAND DECODER,
   THIS CAUSES A PREVIOUS FILE WITH THE
   SAME NAME AS THE SPECIFIED OUTPUT FILE TO BE DELETED FIRST.
   NOTE THAT THIS DOES NOT GUARANTEE YOU THAT THE OUTPUT WILL BE
   WRITTEN TO THE PLACE WHERE THE DELETED FILE WAS LOCATED,

7. IF THE FILE CONTAINS LONG LINES (E.G., BECAUSE OF USE OF BACKSPACE AND UNDERSCORE) AND PIP COMPLAINS 'LINE TOO LONG IN FILE #1' THEN UTIL MAY BE USED FOR THE MERGING:

```
    .R UTIL
    $A<A, DTA1: A2/A$
```

NOTE THAT THE OUTPUT OF UTIL HAS (EVEN) PARITY.
NAME: MCEDIT AB-V16
REVISION BY: A.E. BROUWER
DATE: 75022/
ENVIRONMENT: OS/8

DESCRIPTION:
THIS MONTH MCEDIT HAS BEEN CHANGED TWICE:
FIRST MCEDIT AB-V15 WAS CREATED; THIS VERSION WAS IDENTICAL IN
FUNCTION TO THE PREVIOUS ONE, BUT USED ABOUT 50 LOCATIONS LESS
(WITHOUT DECREASING THE SIZE OF TEXTBUFFERS OR I/O BUFFERS).
THIS ENABLED THE ADDITION OF SOME FEATURES IN MCEDIT AB-V16:

- STRINGSEARCH WITHOUT OUTPUT:
  THE COMMAND #O CAUSES A STRINGSEARCH JUST LIKE #J
  BUT IF THE STRING WASN'T FOUND IN THE CURRENT BUFFER A #Y
  (KILL,READ) INSTEAD OF A #N (PUNCH,KILL,READ) IS EXECUTED
  IN ORDER TO CONTINUE THE SEARCH IN THE NEXT BUFFER.
  THE COMMAND #F (GET THE FOLLOWING ONE) USES THE OUTPUTMODE
  ESTABLISHED BY THE LAST #J OR #O COMMAND; AFTER A #J
  #F SEARCHES WITH OUTPUT WHILE AFTER A #O IT SEARCHES WITHOUT
  OUTPUT.
  [NOTE: SEARCHES BY (ALTMODE) OR " ALWAYS REMAIN WITHIN
  THE CURRENT BUFFER AND HENCE NEVER GIVE ANY OUTPUT.]
  [NOTE: IN THE SEQUENCE #O,...,#(ALTMODE)...,#F, THE #F INDICATES
  A SEARCH WITHOUT OUTPUT (BECAUSE OF #O) FOR THE STRING
  MENTIONED AFTER #(ALTMODE),]

- NO <CR> IN SEARCH STRING:
  PREVIOUSLY A <CR> WAS ACCEPTED IN THE SEARCH STRING, BUT
  SEARCHING FOR A STRING CONTAINING <CR> WAS NEVER SUCCESSFUL.
  TO SUPPRESS ANY MISUNDERSTANDINGS ABOUT THIS MCEDIT NOW
  REPLIES '?' TO A <CR> AND RETURNS TO COMMAND MODE.

- LIST AFTER SEARCH:
  IT APPEARED THAT FOLLOWING A STRINGSEARCH THE LIST COMMAND
  WAS BY FAR THE MOST FREQUENTLY GIVEN COMMAND.
  THE CURRENT VERSION OF MCEDITLISTS THE LINE IN WHICH THE
  STRING WAS FOUND AFTER ONE OF THE COMMANDS #J, #O OR #F.
  [OF COURSE THE SAME RESULT CAN BE OBTAINED FOR " AND (ALTMODE)
  BY STRIKING ONE KEY EXTRA: #"L OR #(ALTMODE)STRING'L ]
1.4 ASSEMBLERS

-NAME: DCP
-DATE: 731003
-VERSION NR: DCP AB=V21
-LAST UPDATE: 741112
-ENVIRONMENT: OS/8 OPER. SYSTEM

DCP (PRELIMINARY DESCRIPTION).

DCP (SOMETIMES CALLED DEASS) IS A PROGRAM TO DEASSEMBLE (OR DISASSEMBLE) A PAL PROGRAM GIVEN IN BINARY OR IN CORE IMAGE FORMAT AS 1ST INPUT FILE. INFORMATION ABOUT THE PROGRAM AND MEANINGFUL TAGS CAN BE GIVEN IN A SECOND INPUT FILE. A WELL READABLE LISTING WITH MEANINGFUL TAGS BUT WITHOUT COMMENT CAN BE OBTAINED IN A FEW PASSES (TYPICALLY FOUR). THE FIRST TIME NO INFORMATION IS SUPPLIED; WHILE READING THE OUTPUT ONE RECOGNIZES CERTAIN PARTS AS MESSAGES ("NO ROOM FOR OUTPUT") OR NUMERIC TABLES (6030, 7634, 7766, 7777) OR SIMPLE SUBROUTINES (TTYOUT, PUSH, PRINT). PUTTING THESE THINGS IN AN INFORMATION FILE AND THEN RUNNING DCP AGAIN GIVES YOU A MUCH NICER OUTPUT THE SECOND TIME.

NOW YOU MAY EMBARK ON THE PROGRAM ITSELF AND OBTAIN AFTER A SMALL NUMBER OF PASSES (DEPENDING ON THE COMPLEXITY OF THE PROGRAM AND YOUR LAZINESS) A SOURCE THAT MIGHT HAVE BEEN THE ORIGINAL ONE EXCEPT FOR ITS LACK OF COMMENT. AT THIS MOMENT YOU COULD PROFITABLY USE THE CTRL/E FEATURE OF MCEDIT TO PROVIDE THE WHOLE SOURCE OF COMMENT. (FOR EXAMPLE, WE OBTAINED A SOURCE OF A FORTRAN COMPILER IN THREE DAYS AFTER FIVE PASSES.)

BELOW WE WILL DESCRIBE THE OS/8 VERSION OF THE PROGRAM

-MEMORY REQUIREMENTS: 16K

-ASSEMBLY INSTRUCTIONS:
.R PAl8
$102, DCP, SBIN, DCPZ/L$
.SAVE SYS DCP

-OPERATING INSTRUCTIONS:
.R DCP
#OUTPUT<: INPUT, INFO(OPTIONS)

.
COMMAND LINE INTERPRETATION:

1. IF NO INPUT AND NO OUTPUT SPECIFIED THEN DELETE DSK:DCPLS.TM
   IF COMMAND CLOSED WITH ALTMODE THEN EXIT TO OS/8 MONITOR
   ELSE CALL COMMAND DECODER AGAIN.

2. IF NO OUTPUT GIVEN BUT AN OUTPUT FILE IS REQUIRED BECAUSE
   CHAINING TO CREF,SV IS REQUESTED THEN DSK:DCPLS.TM IS USED.

3. IF NO INPUT GIVEN THEN USE OUTPUT FILENAME WITH EXTENSIONS
   .SV AND .SM (IF PRESENT)
   E.G.
   *DEASS<
   IS EQUIVALENT TO
   *DEASS<DEASS,SV
   IF DEASS,SM DOES NOT EXIST, AND TO
   *DEASS<DEASS,SV,DEASS,SM
   OTHERWISE.
   IN THIS CASE A PREVIOUS VERSION OF THE OUTPUTFILE IS
   DELETED FIRST (IF NECESSARY).

4. IF THE OUTPUTFILE HAS NO EXPLICIT EXTENSION THEN ADD
   .DC IF A SOURCE IS PRODUCED, AND .LS OTHERWISE
   (ONE WOULD EXPECT .PA INSTEAD OF .DC BUT THAT PROVED DANGEROUS)

OPTIONS AFFECTING INTERPRETATION OF COMMAND LINE:
/B
   EXPECT .BN RATHER THAN .SV FORMAT IN FIRST INPUTFILE
   THIS CHANGES THE DEFAULT EXTENSION INTO .BN IF NO
   INPUT IS SPECIFIED.
/L
   PRODUCE .LS RATHER THAN .DC OUTPUT
/X
   CHAIN TO CREF,SV
   (1ST OUTPUT BECOMES INPUT AND 2ND OUTPUT BECOMES OUTPUT)
   THIS OPTION IMPLIES THE OPTIONS /L AND /T

E.G.
   .R DCP
   *DEASS,TTY:</X/B
   IS EQUIVALENT TO
   .R PIP
   *DEASS,LS</D$
   .R DCP
   *DEASS,LS<DEASS,BN,DEASS,SM/L/T/B
   .R CREF
   *TTY:<DEASS,LS

ALSO
   .R DCP
   *
   *DEASS,SV,SPEC51,SPEC52,SPEC53/X

MEANS
   .R PIP
   *DCPLS.TM</D$
   .R DCP
   *DCPLS.TM<DEASS,SV,SPEC51,SPEC52,SPEC53/L/T
   .R CREF
   *DCPLS,TM
OPTIONS:
/A DO NOT GENERATE A 'START' LABEL
   (BY DEFAULT A LABEL 'START' IS GENERATED WHEN DECODING
    A CORE IMAGE FILE, THIS IS POSSIBLE SINCE THE CORE CONTROL
    BLOCK CONTAINS THE STARTING ADDRESS)
/B EXPECT .BN INSTEAD OF .SV INPUT
/C COPY THE INFO FILE AFTER THE OUTPUT
/D ALLOW 'JMP .-3', 'JMP I .+1' INSTRUCTIONS
   (NORMALLY FOR EACH REFERENCE A TAG IS GENERATED)
/H DO NOT GENERATE LITERALS
/K ALLOW MODIFICATION OF LITERALS
   (NORMALLY AN INSTRUCTION LIKE 1377 WILL BE TRANSLATED BY
    'TAD (1234)' BUT 2377 BY 'ISZ A177' SINCE NO DECENT PROGRAMMER
    EVER WRITES 'ISZ (1234)'. IT WAS FOUND HOWEVER THAT SEVERAL
    DEC PROGRAMS CONTAIN SUCH CONSTRUCTS)
/L PRODUCE OUTPUT IN .LS FORMAT
/N DO NOT GENERATE TABLE OF UNDEFINED SYMBOLS
/S GENERATE TABLE OF ALL SYMBOLS
/T CONVERT TABS INTO SPACES
/W DO NOT INTERPRET 6141 AS THE PDP12 'LINC' INSTRUCTION
/X CHAIN TO CREF, SV

/(F) (WHERE F designates a digit between 0 and 7)
   TRANSLATE FIELD F OF THE PROGRAM (DEFAULT: /0)
   (THE TRANSLATION IS DONE ONE FIELD AT A TIME;
    THIS CAUSES SOME FLAWS IN THE OUTPUT:
    CIF 10
    JMS I (200
    IS TRANSLATED AS
    CIF 10
    JMS I (START
    IF LOC 200 IN THE CURRENT FIELD IS LABELED START,
    NOTE THAT ASSEMBLING THE PRODUCED SOURCE GIVES THE
    CORRECT BINARY)

=NNNNMMMM THE = OPTION CAN BE USED TO SPECIFY A PART
   OF THE PROGRAM TO BE DECODED. NNNN GIVES BEGIN AND
   MMMM END+1 OF THE RANGE.
   (NOTE THAT IF BEGIN>3777 THE COMMAND HAS TO BE CLOSED
   WITH ALT_MODE INSTEAD OF RETURN).
INPUT FORMAT

EACH INPUT SECTION STARTS WITH $X$ (WHERE $X$ IS A LETTER INDICATING THE TYPE OF THE SECTION) AND ENDS WITH $$. $$<CR>$$ INDICATES THE END OF ALL INPUT (WHEN NOT WITHIN A SECTION). BETWEEN THE SECTIONS COMMENT NOT CONTAINING $$ MAY BE INSERTED.

SECTION TYPES:
$A$ TRANSLATE AS 6BIT ASCII (TEXT "STRING")
$D$ DON'T TRANSLATE
$I$ TRANSLATE AS INSTRUCTION (OVERRIDING OTHER SPECS)
$L$ TRANSLATE AS IDENTIFIER RATHER THAN AS INSTRUCTION
$N$ TRANSLATE OCTAL
$S$ SUBROUTINE WITH ARGS
$T$ SYMBOL DEFINITIONS
$Z$ SPECIAL CODING
$$ END OF INPUT

CONTENT OF SECTION:
1. SECTIONS $X$ WHERE $X$ IS A,D,I,L OR N. CONTENTS: LINES OF THE FORM MMMM-NNNN OR NNNN WHERE NNNN AND MMMM ARE OCTAL ADDRESSES, E.G., THE SECTION
$$
1717-1730
1750
$$
SPECIFIES THAT THE LOCATIONS 1717-1730 AND 1750 ARE TO BE TRANSLATED AS OCTAL NUMBERS.

$$
1000:NL
$$
INDICATES THAT EACH CALL TO THE SUBROUTINE AT LOC 1000 HAS TWO ARGUMENTS OF TYPE OCTAL AND LABEL RESPECTIVELY.

3. SECTIONS $T$, CONTENTS: LINES OF THE FORM TAG=NNNN OR TAG MEANING: IF NO OCTAL VALUE OF A TAG IS SPECIFIED THEN ITS VALUE IS TAKEN AS ONE MORE THAN THE VALUE OF THE PREVIOUS TAG.
4. SECTION $Z$.

This is an ad hoc construct to enable the translation of symbol tables like those of PALB and CREF.

E.g.,

$Z = 52; 0 = 240; 1 = 301; 40 = 260$

NNNN-MMMM:(UUUL)

$\$

Indicates that the range NNNN-MMMM is a table of four-word entries.

Three words in a special format and one label.

The special format is as follows:

The value is divided by 52 giving a quotient and a remainder.

Both are converted into a character as follows: 0 gives a space,

1-37 give letters A-_, and 40-51 give digits 0-9.

The coding here is not foolproof yet; a strange command might give strange output instead of an error message.

In later versions this command will be generalized, so we don't describe it in full here.

Errormessages.

These are very poor (because of lack of space): HL TNNNN,

Where NNNN indicates the address of the routine in DCP that detected the error.

Errors are almost always violations of the input format.

A complete list will appear in the final report.
AS NOTED, THE ERRORMESSAGES OF DCP LOOK LIKE 'HLT....'
WHERE .... STANDS FOR THE OCTAL ADDRESS OF THE ROUTINE
THAT DETECTED THE ERROR.
(OF COURSE GIVING INTELLIGIBLE MESSAGES IS HIGHLY DESIRABLE
BUT LACK OF SPACE PREVENTED THIS, SOME FUTURE VERSION OF DCP
WILL CHAIN TO A FILE DCPERR.SV CONTAINING THE MESSAGES.)

BELOW THE ERROR NUMBERS ARE GIVEN FOR DCP AB-V21.
[NOTE: THESE NUMBERS MAY CHANGE SLIGHTLY EACH TIME THAT
DCP IS ASSEMBLED ANEW.)

NUMBER  ERROR
0000  PREMATURE END OF .BN INPUT
0230  CLOSE ERROR
0301  LOOKUP FOR SYS;CREF,SV FAILED
1414  OUTPUT ERROR OR NO ROOM FOR OUTPUT
1451  INPUT ERROR (INFO FILE)
1522  NO CARRIAGE RETURN WHERE EXPECTED IN THE INFO FILE
1755  UPPER BOUND IN BOUND PAIR LESS THAN LOWER BOUND
2031  ASCII STRING CONTAINED A SIXBIT ZERO, BUT NOT AT THE END
     (I.E. A WORD 0XXX), (THIS MIGHT HAVE BEEN AN a,
     BUT IS USUALLY AN ERROR.)
2046  ASCII STRING WITHOUT TRAILING ZERO
2061  DCP COULD NOT FIND A SUITABLE DELIMITER FOR THE ASCII STRING
     IN THE RANGE " " TO ","
2125  IMPOSSIBLE
2214  TEXT BUFFER OVERFLOW (TOO MANY OR TOO LONG IDENTIFIERS),
2234  NO IDENTIFIER WHERE EXPECTED (IN A $T SECTION),
2666  ZERO SUBROUTINE ADDRESS SPECIFIED IN A $S SECTION
2705  S-BUFFER OVERFLOW (TOO MANY SUBROUTINES WITH ARGS),
2761  UNKNOWN TYPE LETTER IN SPECIFICATION OF SUBROUTINE ARGS
3006  $Z NOT FOLLOWED BY =
3011  $Z= NOT FOLLOWED BY A NONZERO NUMBER
3022  NO CARRIAGE RETURN OR SEMICOLON WHERE EXPECTED IN $Z HEADER
3030  NO = WHERE EXPECTED IN $Z HEADER LINE
3041  ZERO LOWER BOUND IN BOUND PAIR IN $Z SECTION
3064  Z-BUFFER OVERFLOW
3117  PREMATURELY EXHAUSTED Z-FORMAT
3135  UNKNOWN Z-FORMAT SYMBOL
3470  T-BUFFER OVERFLOW
3723  NO VALUE ASSIGNED TO FIRST TAG IN $T SECTION
4213  NO INPUT AND NO OUTPUT AND NO DSK;DCPLS,TM TO DELETE
4245  HANDLER FETCH ERROR
4341  LOOKUP FOR INPUTFILE FAILED
4442  OUTPUT OPEN ERROR
4456  NO 16K MEMORY AVAILABLE
4470  CHECKSUM OR FORMAT ERROR IN BINARY INPUT FILE
4613  FORMAT ERROR IN CORE CONTROL BLOCK OF .SV INPUT FILE
4647  ERROR READING CORE CONTROL BLOCK OF .SV INPUT
4723  ERROR READING .SV INPUT FILE
GENERAL I/O ROUTINES.

NAME: 103
AUTHORS: A.E. BROUWER AND R. VAN VLIET
DATE: 750220
ENVIRONMENT: OS/8

103 IS THE SUCCESSOR OF 102 AND COMPATIBLE WITH PROGRAMS USING 102 (EXCEPT PERHAPS BECAUSE OF DUPLICATE LABELS). THIS MODULE RESIDES IN FIELD 1 (NOT IN THE USR AREA) AND TAKES 3 PAGES. IT PROVIDES A NUMBER OF COMMONLY USED ROUTINES, AND IS ASSEMBLED TOGETHER WITH THE MAIN PROGRAM. THE LOCATION OF CORE I/O BUFFERS, OF THE HANDLERS AND OF 103 ITSELF IS SPECIFIED BY PARAMETERS DESCRIBED BELOW.

-NAMES AND FUNCTION OF THE CONSTITUTING SUBROUTINES:

-OPEN: INITIALIZES THE INPUT ROUTINES

-IFOPEN: SELECTS THE INPUT ROUTINE

-WORD: FETCH A WORD (12 BITS)
-CHAR: FETCH A CHARACTER (8 BITS)
-ERROR RETURN: AC>=0 IF END OF FILE
   AC<0 IF READ ERROR

-OOPEN: INITIALIZES THE OUTPUT ROUTINES

-ERROR RETURN: AC>=0 IF NO OUTPUT DEVICE/FILE
   AC<0 IF NO ROOM FOR OUTPUT

-OWORD: OUTPUTS A WORD (12 BITS)
-OCHAR: OUTPUTS A CHAR (8 BITS)
-ERROR RETURN: AC=0 IF NO ROOM FOR OUTPUT
   AC<0 WRITE ERROR

-OCLOSE: CLOSES THE OUTPUT FILE
-ERROR: FILE TOU LARGE TO BE CLOSED OR I/O ERROR

-O TYPE: RETURNS D CB WORD OF OUTDEV IN AC

-PARAMETERS NEEDED:
- INBUF  = ADDRESS OF INPUT BUFFER
- INCTL  = INPUT BUFFER CONTROL WORD
- OUBUF  = ADDRESS OF OUTPUT BUFFER
- OUCTL  = OUTPUT BUFFER CONTROL WORD (MUST BE NEGATIVE)
- INDEVH = ADD FOR INPUT HANDLER
- OUDEH = ADD OF OUTPUT HANDLER
-OPTIONAL PARAMETERS:

IHMT= 1, IF ONLY 1 PAGE AVAILABLE FOR INPUT HANDLER;

OHLTH= 1, IF ONLY 1 PAGE AVAILABLE FOR OUTPUT HANDLER;
DEFAULT: TWO PAGES AVAILABLE

USR= 7700 IF NOT RESIDENT
200 IF RESIDENT IN CORE

CFLD= NO, IF THE ROUTINES IN 103 ARE ALWAYS CALLED FROM FIELD N;
UNDEFINED OTHERWISE.

IOORG= ORIGINSETTING 103
DEFAULT: IOORG=2000

-DEFAULT PARAMETER SETTING:

IFNDEF USR <USR=200>
/ASSUMES I/O MONITOR IS RESIDENT IN CORE
/UNLESS SPECIFIED OTHERWISE
IFNZRO USR-200 <
IFNZRO USR-7700 <
USRX, ? >>

/CAN BE CALLED FROM ANY FIELD,
/WITH BUFFERS IN ANY FIELD
/HOWEVER BY SPECIFYING CFLD ONE PROMISES
/TO CALL 103 FROM THE FIELD CFLD=10 ONLY
/(MINIMIZE THE NR OF CIF, CDF = RDF INSTRUCTIONS
/IN A TIME-SHARING ENVIRONMENT!)
IFNDEF CFLD <IOEXIT=HLT>
IFDEF CFLD <
IFNZRO CFLD=7707 <CFLDX, ?>
IFNZRO CFLD=10 <IOEXIT=CDF CIF CFLD>
IFZERO CFLD=10 <IOEXIT=NOP>
>

/ASSUMES 2 PAGES AVAILABLE FOR INPUT HANDLER
/UNLESS SPECIFIED OTHERWISE
IFNDEF IHLTH <IHLTH=2>
IFNZRO IHLTH=1-7776 <IHLTHX, ?>

/ASSUMES 2 PAGES AVAILABLE FOR OUTPUT HANDLER
/UNLESS SPECIFIED OTHERWISE
IFNDEF OHLTH <OHLTH=2>
IFNZRO OHLTH=1-7776 <OHLTHX, ?>

INFLD=INCTL=70
OUFLD=OUCCTL=70
INRECS=INCTL=200
OUNRECS=OUCCTL=3700-200

-ORIGINSETTING:
IFNDEF IOORG <IOORG=2000>
FIELD 1
*IOORG
NAME: I0G
AUTHORS: A., L. BROUWER & R. VAN VLIET
DATE: 750221
ENVIRONMENT: OS/8

I0G IS A ONE PAGE MODULE USING I03 THAT PROVIDES A SET OF ROUTINES
FOR CHARACTER AND WORD I/O. ALL I/O ERRORS ARE HANDLED BY GIVING AN
APPROPRIATE MESSAGE FOLLOWED BY AN EXIT TO THE OS/8 MONITOR,
ANYONE WISHING TO HANDLE HIS ERRORS HIMSELF SHOULD USE I03
INSTEAD OF I0G+I03).

ROUTINES

INITIO LOCKS USR IN CORE IF SPECIFIED,
CALLS COMMANDDECODER IF ENTERED WITH AC= -1,
OPENS INPUT- AND OUTPUT FILES AS REQUIRED.
N.B.: PROviso IS MADE FOR ONLY ONE OUTPUT FILE,
ALL INPUT FILES ARE COMBINED TO ONE.

INWORD READS A WORD (12 BITS);
ERROR RETURN: END OF FILE REACHED.

INCHAR READS A CHARACTER (8 BITS);
ERROR RETURN: END OF FILE REACHED.

OUWORD WRITES A WORD (12 BITS).

OUCHAR WRITES A CHARACTER (8 BITS).

CLOSE CLOSES THE OUTPUT FILE.

PRTXT PRINTS A TEXTSTRING ON THE TELETYPewriter
ARG 1: POINTER TO TEXTSTRING.

ALL ERROR RETURNS ARE TAKEN WITH AC= 0.

SOME CALLS TO THESE ROUTINES MAY CAUSE ERRORMESSAGES TO BE PRINTED,
FOLLOWED BY AN EXIT TO THE OS/8 MONITOR.
MESSAGES AND MEANING:

"OPEN ERROR"
ONE OF THE FOLLOWING
OUTPUT DEVICE NOT IN SYSTEM
NO ROOM FOR OUTPUTFILE
ATTEMPT TO WRITE ON A READ-ONLY DEVICE.

"INPUT ERROR"
EITHER I/O ERROR DURING INPUT
OR INPUT DEVICE NOT IN SYSTEM.

"OUTPUT ERROR"
EITHER I/O ERROR DURING OUTPUT
OR NO ROOM FOR OUTPUTFILE.

"CLOSE ERROR"
EITHER I/O ERROR DURING FILE CLOSE
OR FILE TO LARGE TO BE CLOSED.
PARAMETERS.

All parameters listed below may be specified by the user, if the user has not specified them default values are taken.

**PARAMETER, DEFAULT VALUE, MEANING.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>INBUF</td>
<td>3000</td>
<td>Address of input buffer</td>
</tr>
<tr>
<td>INCTL</td>
<td>1010</td>
<td>OS/8 input control word</td>
</tr>
<tr>
<td>INDEVH</td>
<td>6600</td>
<td>Address of input device handler</td>
</tr>
<tr>
<td>IMLTH</td>
<td>2</td>
<td>1: One page available for input device handler, 2: Two pages available for input device handler</td>
</tr>
<tr>
<td>INEXT</td>
<td>0</td>
<td>Default input extension</td>
</tr>
<tr>
<td>OUBUF</td>
<td>5000</td>
<td>Address of output buffer</td>
</tr>
<tr>
<td>OUCTL</td>
<td>5010</td>
<td>OS/8 output control word</td>
</tr>
<tr>
<td>OUDEVH</td>
<td>7200</td>
<td>Address of output device handler</td>
</tr>
<tr>
<td>OMLTH</td>
<td>2</td>
<td>1: One page available for output device handler, 2: Two pages available for output device handler</td>
</tr>
<tr>
<td>OUEXT</td>
<td>0</td>
<td>Default output extension</td>
</tr>
<tr>
<td>USR</td>
<td>200</td>
<td>200: USR resident in core, 7700: USR nonresident</td>
</tr>
<tr>
<td>I0ORIG</td>
<td>2000</td>
<td>Address of I/O routines</td>
</tr>
</tbody>
</table>

N.B.: The default values of INBUF, INCTL, OUBUF, OUCTL are chosen such, that by default field 1 is used for I/O from LOC 2000 (I0ORIG) to LOC 6777. One might change the direct assignment to I0ORIG (it may be chosen in the range 2000 <= I0ORIG <= 5600), this will change the default values above such, that all addresses in field 1 with I0ORIG <= ADDRESS < 7600 may be used for I/O.
1.11 GENERAL UTILITIES

---

NAME: FLXISO
AUTHOR: A.E. BROUWER
DATE: 750112
ENVIRONMENT: OS/8 OPERATING SYSTEM

FLXISO IS A SMALL UTILITY PROGRAM THAT CONVERTS
MC-FLEXOWRITER CODE INTO ASCII OR ISO CODE

INPUT: 1-9 FILES IN MC-FLEXOWRITER CODE
       DEFAULT INPUT EXTENSION: .FX
OUTPUT: 1 FILE IN ASCII CODE
       (NO DEFAULT OUTPUT EXTENSION)

OPTIONS:
/A  OUTPUT ASCII
/1  OUTPUT ISO
/F  COMPUTE EVEN PARITY
/O  OUTPUT INCORRECT CODES IN OCTAL: #100#
   (BY DEFAULT AN INCORRECT CODE IS CONVERTED TO #)
/N  DO NOT IGNORE BLANKS (TAPEFEED)
=0  LOWER CASE INITIALLY - THIS IS THE DEFAULT
=1  UPPER CASE INITIALLY

ERROR MESSAGES:

1. FATAL ERRORS
   INPUT ERROR
   OUTPUT ERROR
   OPEN ERROR
   CLOSE ERROR
   TOO MANY PARITY ERRORS
   TOO MANY UNDEFINED CODES
   8-BIT FLEXOCODE?

2. INFORMATIVE MESSAGES
   NNNN PARITY ERRORS
   NNNN UNDEFINED CODES

   NNNN IS THE (OCTAL) NR OF PARITY ERRORS
   NNNN IS THE NR OF UNDEFINED CODES IN THE INPUT
NAME: SBIN
AUTHOR: A.E. BROUWER, MC, ADAM
DATE: 7405/10
ENVIRONMENT: PDP8 (ANY TYPE)

SBIN IS A ONE PAGE MODULE THAT PROCESSES ABSOLUTE BINARY FILES. 
("SHORT BINARY TAPE HANDLER").

CALLING SEQUENCE:
CDF CUR /FIELD OF CALLER
CIF N /FIELD OF SBIN
JMS I (SBIN
INPUT
DATAS
ORIGS
FELDS

ERROR /ERROR RETURN
--- /NORMAL RETURN

HERE INPUT IS A SUBROUTINE CALLED BY SBIN TO GET THE NEXT 8-BIT
FRAME AND DATAS, ORIGS AND FELDS ARE ROUTINES CALLED WITH
DATA (12 BITS), ORIGIN (12 BITS), RESP, FIELD SETTING (00F0) IN AC.
ALL FOUR MUST RESIDE IN THE CALLING FIELD, AND SHOULD RETURN
WITH A CIF N (DF IS NOT IMPORTANT).
SINCE ALL ROUTINES ARE CALLED FROM THE SAME LOCATION AND INPUT IS
CALLED FIRST, ALL ROUTINES MIGHT END WITH
CIF N
JMP I INPUT

THE ERROR RETURN IS TAKEN:
1. WHEN THE CHECKSUM IS INCORRECT OR TRAILER IS FOUND BEFORE EXPECTED.
   IN THIS CASE AC=0.
2. WHEN THE 2ND FRAME OF AN ORIGIN OR DATA WORD IS NOT 0XY.
   IN THIS CASE AC>0.

APPLICATIONS:
SBIN CAN BE USED (AND INDEED IS USED) TO OUTPUT A DUMP OF A BINARY
FILE, OR TO PRODUCE A BITMAP, OR TO LOAD A BINARY FILE ETC.
IT IS INDEPENDENT OF ANY OPERATING SYSTEM AND ACCEPTS EVERYTHING
ACCEPTABLE TO THE BINARY LOADER, IN PARTICULAR IT RECOGNIZES
AND SKIPS THE SEQUENCE <RUBOUT STRING RUBOUT> UNLIKE MANY NEWER
PROGRAMS, IT MAY RESIDE IN ANY FIELD AND IN ANY PAGE.

ACCEPTED DATA FORMAT:
0XY 0ZU DATA WORD XYZU
1XY 0ZU ORIGIN SETTING *XYZU
2XY LEADER/TRAILER FRAME
377 STRING 377 IRRELEVANT INSERTION
3FX FIELD SETTING FIELD F

RECOMMENDED BINARY FORMAT (TO SATISFY OS/8 PROGRAMS LIKE ABLSDLR)
USE ONLY:
200 LEADER/TRAILER
3F0 FIELD SETTING
AND NO OTHER FRAMES STARTING WITH 2 OR 3.
ISOFLX IS A SMALL UTILITY PROGRAM THAT CONVERTS
ISO CODE INTO MC-FLEXOWRITER CODE

INPUT: 1-9 FILES IN ISO CODE
NO DEFAULT INPUT EXTENSION
OUTPUT: 1 FILE IN MC-FLEXOWRITER CODE
DEFAULT OUTPUT EXTENSION: .FX

OPTIONS:
/F OUTPUT INCORRECT CODES IN OCTAL: ?100?
/N DO NOT SKIP NULL CHARACTERS (CODE 200) AND RUBOUT
CHARACTERS (CODE 377)
/C SIMULATES CARRIAGE RETURN WITHOUT LINEFEED
(BACKSPACES) AND LINEFEED WITHOUT CARRIAGE RETURN
(CRLF FOLLOWED BY SPACES)
IF /C NOT SPECIFIED, LF, CR, CRLF AND LF CR ARE
TRANSLATED INTO CRLF
/#NNNN WHERE NNNN IS THE NUMBER OF LINES ON A PAGE (IN
OCTAL). IF A FORMFEED (CTRL/L, CODE 214) IS
ENCOUNTERED, THE CORRECT NUMBER OF CRLF'S IS ADDED,
FOLLOWED BY 50 (OCTAL) BLANK FRAMES,
DEFAULT: CODE 214 IS TRANSLATED INTO 50 BLANK FRAMES.

ERROR MESSAGES:

1. FATAL ERRORS
INPUT ERROR
OUTPUT ERROR
OPEN ERROR
CLOSE ERROR
TOO MANY UNDEFINED FLEXO CODES MORE THAN 4095 OF THESE
TOO MANY NON IMPLEMENTED ISO CODES MORE THAN 4095 OF THESE

2. INFORMATIVE MESSAGES
NNNN UNDEFINED FLEXO CODES
NNNN NON IMPLEMENTED ISO CODES (NNNN IS THE OCTAL NUMBER)
THE PROGRAM BRAIL,

1. BRIEF DESCRIPTION,

BRAIL IS A PROGRAM TO CONVERT TEXTFILES (CONTAINING ASCII CHARACTERS) INTO A FILE CONTAINING THE SAME TEXT IN A SO CALLED "BRAILCODE". THESE FILES MAY SUBSEQUENTLY BE USED TO CONTROL AN AUTOMATIC BRAILWRITER TO EMBOS A BRAILCOPY OF THE ORIGINAL TEXT. PROGRAM BRAIL IS ESPECIALLY INTENDED TO CONVERT PROGRAM SOURCE TEXTS, THEREFORE THE LAYOUT IS TREATED SUCH THAT
1. THE BLIND READER MAY EASILY FIND HIS WAY IN THE BRAIL TEXT,
2. THE LAYOUT OF THE ORIGINAL TEXT CAN ALMOST UNIQUELY BE RECONSTRUCTED FROM THE BRAILCOPY,

THE LINENUMBERS OF THE ORIGINAL TEXT MAY BE PRINTED IN FRONT OF THE CONVERTED LINES IN ORDER TO FACILITATE FUTURE EDITING.

ASSEMBLY INSTRUCTIONS
.R PAL8
*10, BRAIL/L=14000$
.SAVE SYS BRAIL

10 MAY BE ANY IO2-LIKE VERSION OF THE I/O ROUTINES, USING CORE 13200/13777 AND USR NOT LOCKED IN CORE.

OPERATING INSTRUCTIONS
.R BRAIL
*OUTPUT,CHAIN=INPUT (OPTIONS) =NN
:HEADING-TEXT
#XXX
#XXX
...

BRAIL STARTS AT 14000 AND IS RESTARTABLE.
2. PERFORMANCE CHARACTERISTICS.

1. BRAIL DIVIDES THE INPUTTEXT IN "INPUTPAGES", AN INPUTPAGE IS EITHER TERMINATED BY READING A FORMFEEDCHARACTER FROM THE INPUTFILE, OR BY REACHING THE END OF THE INPUTFILE. IF AN INPUTPAGE DOES NOT FIT IN THE "TEXTBUFFER" (3000/5000 CHARACTERS) IT IS ARTIFICIALLY SPLIT IN TWO OR MORE INPUTPAGES. NOTE: THIS SPLITTING AFFECTS THE STRINGSEARCH, BUT DOES NOT AFFECT SKIP OR CONVERSION COMMANDS.

2. INPUTCHARACTERS ARE EITHER GIVEN A SPECIAL TREATMENT (LINEFEED, FORMFEED, TAB, SPACE IN THE LEFT MARGIN) OR skipped (BLANK, CARRIAGERETURN, RUBOUT) OR MARKED AS ILLEGAL (ALL OTHER CONTROLCHARACTERS, CHARACTERS ABOVE 137 (IF ASCI164 INPUT IS ASSUMED), WRONG-PARITYCHARACTERS (IF A PARITYCHECK IS MADE)) OR CONVERTED TO BRAILSYMBOLS (ALL OTHER CHARACTERS), CHARACTERS MARKED AS ILLEGAL SHOW UP AS THE BRAILSYMBOL (1, 4, 6) IN THE OUTPUT.

3. THE LINES OF THE INPUTTEXT ARE NUMBERED STARTING FROM 1 AT THE BEGINNING OF EACH NEW INPUTPAGE, EACH LINEFEED TERMINATES A LINE.

4. THE WIDTH W OF THE LEFT MARGIN OF THE BRAILCOPY IS COMPUTED AS FOLLOWS
   A. LET M BE THE WIDTH OF THE LEFT MARGIN OF THE INPUTTEXT (ASSUMING TABS AT COLUMNS 8 16,..)
   C. W =ENTIER( (M+T-1)/T ),
   THE LEFT MARGIN OF THE BRAILCOPY IS FILLED OUT WITH MARGINCIPHERS (SPECIFIED BY THE USER WITH THE OPTIONS /1/2/3/4/5/6), IF THE CONVERSION OF THE INPUTLINE DOES NOT FIT ON THE BRAILLINE, THE CONVERSION IS CONTINUED ON THE NEXT BRAILLINE PRECEDED BY W SPACES AND A CONTINUE-CHARACTER (HYPHEN), TABS OUTSIDE THE LEFT MARGIN ARE CONVERTED TO ONE SPACE.

5. THE TOPLINE OF EACH BRAILPAGE CONTAINS ITS PAGENUMBER, THE DATE (OS/8 DATEWORD IS USED), AND THE HEADING=TEXT.

6. BRAIL PROVIDES THREE TYPES OF BRAILCODES (DEPENDING ON THE OPTIONS /A /D /H), THE PROPERTIES OF THESE BRAILCODES ARE BRIEFLY SUMMARIZED,
TABLE OF OUTPUT CONVENTIONS,
THE BITS OF THE OUTPUT CHARACTERS ARE CODED AS FOLLOWS, BIT 1: LEAST SIGNIFICANT BIT, BIT 8: MOST SIGNIFICANT BIT, BIT 7 IS THE PARITY BIT. ALL BRAIL SYMBOLS ARE CODED IN BITS 1-6, WHERE BIT 1 CORRESPONDS TO DOT 1. (BRAILDOTS ARE NUMBERED 1 4 2 5 3 6).
CODES WILL BE SPECIFIED AS 3-OCTADE NUMBERS.

<table>
<thead>
<tr>
<th>/A-CODE</th>
<th>/D-CODE</th>
<th>/H-CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYMBOLS</td>
<td>1XX</td>
<td>0XX</td>
</tr>
<tr>
<td>SPACE</td>
<td>100</td>
<td>300</td>
</tr>
<tr>
<td>SKIP</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>CRLF</td>
<td>011</td>
<td>201</td>
</tr>
<tr>
<td>NEW PAGE</td>
<td>003</td>
<td>201</td>
</tr>
<tr>
<td>OTHER CODES</td>
<td>UNDEFINED</td>
<td>SKIP</td>
</tr>
<tr>
<td>PARITY</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>LINE WIDTH</td>
<td>34</td>
<td>36</td>
</tr>
<tr>
<td>PAGE LENGTH</td>
<td>30</td>
<td>27</td>
</tr>
</tbody>
</table>

3. USING PROGRAM BRAIL.

3.1. PROGRAM START AND TERMINATION.
DURING A RUN OF BRAIL A NUMBER OF SEPARATE CONVERSIONS CAN BE MADE, EACH CONVERSION FIRST CALLS THE OS/8 COMMAND DECODER AND - IF NO ERRORS OCCUR - FINALLY CLOSES ITS OUTPUT FILE AND INITIATES THE NEXT CONVERSION.
IF AN ERROR OCCURS, A MESSAGE IS PRINTED ON THE CONSOLE TELETYPET AND AN IMMEDIATE RETURN TO THE OS/8 MONITOR IS TAKEN; NO OUTPUT FILES ARE CLOSED!
IF THE CHAIN OPTION (/X) WAS SPECIFIED, BRAIL CHAINS IN STEAD OF REINITIATING A NEW CONVERSION.
THE LAST CONVERSION CAN BE INDICATED BY TERMINATING THE COMMANDLINE TO THE OS/8 COMMAND DECODER WITH AN ALTMODE, IMMEDIATE RETURN TO THE OS/8 MONITOR CAN BE FORCED BY TYPING \C (CTRL/C) ON THE CONSOLE TELETYPET.

3.2. SPECIFICATIONS TO THE OS/8 COMMAND DECODER.
TO THE OS/8 COMMAND DECODER THE FOLLOWING CAN BE SPECIFIED:
1. OUTPUT FILE. IF NO EXTENSION WAS SPECIFIED, THE EXTENSION .BR WILL BE ADDED, IF THIS FILE IS NOT SPECIFIED, THE FILE BRLBLBR.TM IS TAKEN AS DEFAULT OUTPUT FILE.
2. A CHAIN PROGRAM. THIS PROGRAM MAY BE SPECIFIED AS THE SECOND OUTPUT FILE, IF NO EXTENSION WAS SPECIFIED THE EXTENSION .SV IS ASSUMED. IF NO CHAIN PROGRAM IS SPECIFIED AND ONE IS NEEDED (/X GIVEN) BRCONV.SV IS CHOSEN.
3. TO 9 INPUT FILES, THESE INPUT FILES ARE CONCATENATED TO 1 INPUT FILE, THE DEFAULT INPUT EXTENSION IS .PA.
4. ONE OF THE OPTIONS /A /D /H TO SPECIFY THE TYPE OF OUTPUT CODE. IF NONE OF THESE IS SPECIFIED THE PROGRAM WILL ASSUME /D.
5. THE OPTION /I TO SPECIFY THE INPUT CODE IS ISO. OTHERWISE ASCII-64 IS ASSUMED.
6. ONE OF THE OPTIONS /E OR /O TO FORCE EVEN OR ODD PARITY CHECKING.
7. THE OPTION /N TO SPECIFY THAT LINENUMBERS MUST BE
   GENERATED IN FRONT OF EACH CONVERTED LINE.
   OTHERWISE THESE LINENUMBERS WILL BE OMITTED.
8. THE OPTION /X TO FORCE CHAINING AT THE END OF THIS CONVERSION.
9. SOME COMBINATION OF THE OPTIONS /1 /2 /3 /4 /5 /6 TO SPECIFY
   THE DOTS OF THE DESIRED MARGIN CHARACTER; IF NONE OF THESE
   OPTIONS IS GIVEN, A SPACE WILL BE USED AS MARGIN CHARACTER.
10. =NN, WHERE NN IS AN OCTAL NUMBER SMALLER
    THAN 20. NN IS THE INDENTATION PARAMETER BY WHICH
    THE WIDTH OF THE LEFT MARGIN IS DIVIDED TO FIND THE NUMBER
    OF MARGIN CHARACTERS TO BE PRINTED AT THE START OF EACH
    CONVERTED LINE; IF NN IS NOT SPECIFIED OR SPECIFIED AS 0, A VALUE
    OF 10 (OCTAL) WILL BE ASSUMED.
11. THE COMMAND LINE MAY BE TERMINATED BY AN ALTMODE INSTEAD OF
    A CARRIAGERETURN, TO INDICATE THAT THIS IS THE LAST CONVERSION.

EXAMPLE: THE LINE
   *BRTEST/X
   IS EQUIVALENT TO
   *BRLBR,TM,BRCONV,SV<BRTEST /D/X=10
   THE LINE
   *BRTEST,BRPNCH<BRTEST (26)/E=6
   IS EQUIVALENT TO
   *BRTEST,BR,BRPNCH,SV<BRTEST (26)/D/I/E=6
   AND MEANS THAT THE FILE BRTEST (OR BRTEST,PA) MUST BE CONVERTED,
   EXPECTING EVEN PARITY ISO CODE AS INPUT, THE OUTPUT IS SENT TO THE FILE
   BRTEST, THE WIDTH OF THE LEFT MARGIN IS DIVIDED BY 6 IN THE BRAILCOPY,
   AND QUESTIONMARK (DOTS 2 AND 6) IS USED AS MARGIN CHARACTER.
   NO CHAIN IS MADE AS /X WAS NOT SPECIFIED.

3.3. HEADING=TEXT,

   AFTER THE COMMAND LINE TO THE OS/8 COMMANDDECODER HAS BEEN TYPED
   IN, THE PROGRAM REPLIES BY TYPING A COLON (:) ON THE CONSOLETYPE.
   THE USER MAY THEN ENTER ONE LINE OF TEXT, THE HEADING=TEXT.

3.4. BRAIL COMMANDMODE,

   AFTER THE HEADING=TEXT HAS BEEN ENTERED, BRAIL STORES THE FIRST
   INPUTPAGE IN ITS TEXTBUFFER AND ENTERS ITS COMMANDMODE.
   ENTERING THE BRAIL COMMANDMODE IS INDICATED BY TYPING A NUMBERSIGN
   (#) ON THE CONSOLETYPE. AFTER THE # HAS BEEN TYPED, A BRAIL
   COMMAND MAY BE TYPED IN. EACH TIME BRAIL ENTERS ITS COMMANDMODE, IT
   HAS AN INPUTPAGE IN ITS TEXTBUFFER.
   IF A BRAIL COMMAND TAKES LINENUMBERS AS ARGUMENTS, THESE LINENUMBERS
   APPLY TO LINES OF THE INPUTPAGE CURRENTLY IN THE TEXTBUFFER
   (THE CURRENT INPUTPAGE). THE COMMAND WILL ONLY BE EXECUTED FULLY
   IF ALL DESIGNATED LINES ARE IN THE TEXTBUFFER. AFTER EXECUTING
   THE COMMAND, BRAIL REENTERS ITS COMMANDMODE STILL HOLDING THE
   SAME INPUTPAGE IN ITS TEXTBUFFER.
   COMMANDS CAN BE GIVEN TO SKIP OR CONVERT A NUMBER OF BUFFERS, WHEN
   BRAIL REENTERS ITS COMMANDMODE AFTER THE EXECUTION OF SUCH A COMMAND,
   THE FIRST INPUTPAGE THAT HAS NOT BEEN SKIPPED OR CONVERTED IS THEN
   STORED IN THE TEXTBUFFER.
WHEN A STRINGSEARCH COMMAND IS GIVEN, THE INPUTFILE IS SEARCHED
FORWARD FOR THE SPECIFIED STRING, STARTING AT THE CURRENT INPUTPAGE.
WHEN THE STRING IS FOUND BRAIL REENTERS ITS COMMANDMODE HOLDING THE
INPUTPAGE CONTAINING THE FIRST OCCURRENCE OF THE SPECIFIED STRING IN
ITS TEXTBUFFER.

WHEN A # HAS BEEN TYPED A BRAILCOMMAND MAY BE ENTERED ON THE
CONSOLE/TELETYPewriter. EACH BRAILCOMMAND STARTS WITH A LETTER SPECIFYING
THE COMMAND; THIS LETTER IS FOLLOWED BY 0, 1 OR 2
NUMERIC ARGUMENTS (AN EXCEPTION IS FORMED BY THE STRINGSEARCH
COMMAND); THE COMMAND IS TERMINATED BY A CARRIAGERETURN. NUMERIC
ARGUMENTS CONSIST OF 3 DECIMAL DIGITS; LEADING ZEROS MAY BE OMITTED.
IF NO DIGITS ARE PRESENT A DEFAULT VALUE 1 IS ASSUMED. TWO NUMERIC
ARGUMENTS ARE SEPARATED BY A COMMA.
IF A COMMAND IS UNKNOWN TO BRAIL A QUESTIONMARK WILL BE TYPED IN
REPLY AND THE COMMANDMODE IS REENTERED HOLDING THE SAME INPUTPAGE IN
THE TEXTBUFFER AS BEFORE.

THESE ARE THE BRAIL COMMANDS (M AND N INDICATE NUMERIC ARGUMENTS):
END THE CURRENT CONVERSION;
THE REST OF THE INPUTFILE WILL BE CONVERTED, STARTING AT THE
CURRENT INPUTPAGE.
STRINGSEARCH, AFTER THE LETTER J A DOLLAR MUST BE TYPED,
FOLLOWED BY THE STRING TO BE SEARCHED FOR; SUBSEQUENTLY
THE LINE MUST BE TERMINATED BY AN ALTMODE (THAT Will BE
ECHOED AS A DOLLAR).
A FORWARD SEARCH IS MADE IN THE REST OF THE INPUTFILE FOR
THE FIRST OCCURRENCE OF THE SPECIFIED STRING, STARTING
AT THE CURRENT INPUTPAGE, THE COMMANDMODE
IS REENTERED HOLDING THE INPUTPAGE CONTAINING THE FIRST OCCURRENCE
OF THE SPECIFIED STRING IN THE TEXTBUFFER.
TWO ERRORS MAY OCCUR:
A. THE SPECIFIED STRING DOES NOT OCCUR IN THE REST OF THE
   INPUTFILE;
B. ONE OF THE INPUTPAGES THAT ARE SEARCHED IS ENDED
   ARTIFICIALLY; IT IS NOT ALLOWED
   TO SEARCH OVER SUCH A PAGE END.

CONVERT N INPUTPAGES STARTING AT THE CURRENT ONE.
A SEQUENCE OF ARTIFICIALLY SPLIT INPUTPAGES IS COUNTED FOR ONE.
BRAIL COMMANDMODE IS REENTERED HOLDING THE FIRST INPUTPAGE
THAT HAS NOT BEEN CONVERTED IN THE TEXTBUFFER.
IF THE NUMBER OF INPUTPAGES
IS NOT SUFFICIENT, THE CURRENT CONVERSION WILL BE
TERMINATED, AFTER THE LAST INPUTPAGE HAS BEEN CONVERTED,
QUIT, TERMINATE THE CURRENT CONVERSION.
FIRST THE OUTPUTFILE IS CLOSED, THEN A NEW CONVERSION
WILL START, UNLESS A CHAIN OPTION (/X) WAS SPECIFIED OR
AN ALTMODE HAD TERMINATED THE COMMANDLINE
OF THE OS/8 COMMANDDECODER, THIS CAUSES A RETURN TO THE OS/8
MONITOR.

SKIP N INPUTPAGES STARTING AT THE CURRENT ONE.
A SEQUENCE OF ARTIFICIALLY SPLIT INPUTPAGES IS COUNTED FOR ONE.
BRAIL COMMANDMODE WILL BE REENTERED HOLDING THE FIRST
INPUTPAGE THAT HAS NOT BEEN SKIPPED IN THE TEXTBUFFER.
IF THE NUMBER OF INPUTPAGES IS NOT SUFFICIENT, THE CURRENT
CONVERSION WILL BE TERMINATED.
3.5. Typing a Line to Braille.

The heading-text and all braille commands are entered by typing a line on the console teletype. For entering lines via the console teletype, the following conventions hold:

1. The length of a line is the number of readable characters and spaces typed on it.
2. Control characters (except CTRL/U and CARRIAGERETURN) are skipped and do not increase the length of the line.
3. A rubout deletes the last character or space from the line and decreases its length by one, if the length of the line ≥0 A rubout is simply ignored.
4. \(\backslash u\) (CTRL/U) causes the current line to be entirely ignored.
5. Trying to add a character or space to a line whose length is equal to 64, will cause \(\backslash u\) to be typed on the teletype, and that line to be ignored.
6. A line may be ended by typing a CARRIAGE RETURN or an ALTMODE.

3.6. Errors.

If a fatal error is detected, an error message is printed on the console teletype and a return is taken to the OS/8 monitor. The current output file is not closed. The following error messages may be printed:

"NO INPUT"
No input file(s) have been specified to the OS/8 command decoder.

"CANNOT OPEN OUTPUTFILE"
Either I/O error during the opening of the output file, or trying to open an output file on a read-only device.

"STRING NOT FOUND"
A search was made for a string that did not occur in the rest of the input file.

"PAGE DID NOT FIT IN BUFFER DURING STRING SEARCH"
Trying to search over the end of an artificially ended input page.

"CLOSE ERROR"
I/O error while closing the output file.

"INPUT ERROR"
I/O error during input.

"OUTPUT ERROR"
I/O error during output, or no room for output file.

"CHAIN ERROR"
Lookup for chain program failed.
NAME: RDCOMM
AUTHORS: C.L. PIPEL, A.E. BROUWER (MATH, CENTR)
DATE: 72XXX
LAST UPDATE: 750310
TYPE: PROGRAM MODULE
ENVIRONMENT: PDP8 (ANY TYPE) + TELETYPE
DESCRIPTION: RDCOMM IS A SIMPLE ONE PAGE COMMAND LINE INTERPRETER. EACH COMMAND CONSISTS OF A COMMAND TAG FOLLOWED BY A (POSSIBLY EMPTY) SEQUENCE OF PARAMETERS. TO EACH COMMAND CORRESPONDS A FIXED NUMBER OF PARAMETERS. EACH PARAMETER CONSISTS OF A SIGNED OCTAL NUMBER PRECEDED BY A COMMA. IF THE NUMBER IS ABSENT, IT WILL BE INTERPRETED AS ZERO. RDCOMM PRINTS A CHARACTER -USUALLY A "#"- INDICATING IT IS READY TO ACCEPT A COMMAND. A CR ENDS THE COMMAND LINE. AFTER AN ERRONEOUS COMMAND RDCOMM TYPES A "?", TRY IT AGAIN!
EXAMPLES:
#RU, 20
#l,,
#MO, = 3, 10
#RU, 20, ?RU, 20

ASSEMBLY PARAMETERS:
- COMSYM
- COMGO
THE COMMAND SYMBOL (DEFAULT #)
IF 1 WE DISPATCH TO THE COMMAND ROUTINE
IF 0 WE RETURN NORMALLY (DEFAULT 1)

CALLING SEQUENCE:
CDF CUR
CIF N
JMS RDCOMM
COMTAB /PTR TO COMMAND TABLE
PAR /PTR TO PARAMETER SPACE

REMARKS:
- FORMAT OF THE COMMAND TABLE
A LIST OF TRIPLETS CLOSED BY ZERO
TRIPLET:
ADDRESS OF COMMAND ROUTINE
# OF PARAMETERS FOR COMMAND
-(HASHCODE FOR COMMAND)

HASHCODE:
A COMMAND TAG IS A STRING OF LETTERS
THIS STRING IS PACKED IN TWELVE BITS:
A IS PACKED 0100
AB IS PACKED 0201
ABC IS PACKED 0502, JUST LIKE BE.
THE CODE IS UNIQUE FOR COMMANDS OF AT MOST TWO LETTERS
(FOR THE HASHING ALGORITHM SEE THE SOURCE OF RDCOMM)

-PARAMETER SPACE
RDCOMM PUTS THE PARAMETERS INTO THE PARAMETER SPACE. IF COMGO IS SPECIFIED ZERO THE ROUTINE ADDRESS IS THE 1ST PARAMETER.
NAME: ANTIO
VERSION: ANTIO AB-V03
AUTHOR: A.E. BROUWER, MC, A'DAM
DATE: 750209
ENVIRONMENT:
ANY TYPE PDP8 WITH AT LEAST 8K CORE MEMORY AND A TELETYPewriter (NO OPERATING SYSTEM REQUIRED).

STARTING ADDRESS: 00200
RESTART ADDRESS: 04000

DESCRIPTION:
ANTIO IS A CONVERSATIONAL PROGRAM DESIGNED TO INVESTIGATE THE BEHAVIOR OF ANTICHAINS UNDER THE OPERATOR (STACK, COMP, MAX) DENOTED * AND ITS INVERSE.
IT HANDLES ANTICHAINS OVER SETS OF AT MOST 11 (DECIMAL) ELEMENTS, (LARGER SETS ARE NOT ALLOWED FOR SEVERAL REASONS:
FIRST, THE BITMANIPULATION IS MUCH MORE DIFFICULT TO PROGRAM IF A SET CANNOT BE REPRESENTED IN ONE (12 BIT) WORD;
SECOND, FOR LARGER ANTICHAINS THE PROGRAM CAN HARDLY BE CALLED 'CONVERSATIONAL'. BECAUSE OF THE BIEXponentially GROWing TIME NEEDED FOR THE EXECUTION OF A COMMAND;
THIRD, THE MEMORY REQUIRED (GIVEN THE CURRENT REPRESENTATION OF ANTICHAINS AS A CHARACTERISTIC FUNCTION) IS 16K FOR N=12 WHICH IS NOT (YET) AVAILABLE UNDER TRAC, THE CURRENT PROGRAM REQUIRES 8K ONLY.)

CONVERSATION:
AT ANY TIME:
CR AND LF ARE IGNORED, CTRL/C CAUSES A JUMP TO 07600
CTRL/O STOPS ANY ACTIVITY TO RETURN TO COMMAND MODE,
AND # CLOSES A COMMAND NORMALLY.

INITIAL DIALOG:
ANTIO AB-V03
N=
HERE A VALUE BETWEEN 2 AND 11 (INCLUSIVE) HAS TO BE SPECIFIED.

COMMAND STRUCTURE:
THE COMMAND SIGN IS #, AND A COMMAND CONSISTS OF ONE LETTER.
THE COMMAND IS EXECUTED IMMEDIATELY (NO WAITING FOR CR, WHICH IS IGNORED ANYWAY),
THE FACT THAT THE COMMAND SIGN IS THE SAME SYMBOL AS THE COMMAND CLOSER CAN BE USED ADVANTAGEOUSLY TO READ IN A PAPERTAPE CONTAINING A LISTING OF AN ANTICHAIN CREATED PREVIOUSLY.
COMMAND MEANING
A
APPEND
R
REMOVE
K
KILL
L
LIST
C
SWITCH CYCLIC MODE ON/OFF
S
SWITCH STATISTIC MODE ON/OFF
F
DO ONE FORWARD STEP
I
DO ONE INVERSE STEP
G
GO FORWARD AND LIST EACH ANTICHAIN ENCOUNTERED
T
GO FORWARD AND TYPE PERIOD,
(WHEN NOT IN TRAC THIS COMMAND MAY BE INTERRUPTED WITH
CTRL/O AND CTRL/C)
P
TYPE OUT THE PERIOD COUNTER (USEFUL ONLY AFTER INTERRUPTING
A #T COMMAND).
O
GET THE ORIGINAL ANTICHAIN AGAIN (IT WAS SAVED AT THE
BEGINNING OF #G AND #T),
X
ASK FOR A NEW VALUE OF N.

WITH THE COMMANDS A(PPEND) AND R(EMOVE) A SET IS SPECIFIED
BY TYPING ITS ELEMENTS FOLLOWED BY A /
ELEMENTS ARE DESIGNATED BY LETTERS IN THE RANGE A-K (IF N=11).
FOR EXAMPLE
#K
#A
ABC/ABD/ACD/BCD/
#

CREATES THE ANTICHAIN CONSISTING OF THE FOUR THREE-ELEMENT SUBSETS
OF A 4-SET. IF N=4 THE SAME ANTICHAIN IS PRODUCED BY
#K
#C
ON
#A
ABC/
#

THE 'STEP' MENTIONED WITH THE COMMANDS F AND I CONSISTS IN THE
REPLACING OF THE CURRENT ANTICHAIN BY ITS IMAGE UNDER THE
OPERATOR * (RESP. ITS INVERSE).

REFERENCE:
A.E. BROUWER & A. SCHRIJVER,
ON THE PERIOD OF AN OPERATOR DEFINED ON ANTICHAINS
MATH. CENTR. REPORT ZW 24/74.
2.2 TEXT PROCESSING

NAME: ADDDOC, TE
AUTHOR: A. E. BROUWER, MC, A'DAM
DATE: 750127
TYPE: PRELIMINARY DOCUMENTATION
ENVIRONMENT:
- A TECO VIRTUAL MACHINE (AS IMPLEMENTED BY THE
  DEC PROGRAM TECO, SV UNDER THE OS/8 OPERATING SYSTEM),
- THE FILE INDEX.TX

DESCRIPTION:
ADDDOC IS A SIMPLE TECO PROGRAM TO SUPPLY PAGE NUMBERING AND DATE
TO THE SPECIFIED INPUT FILES, IT ALSO UPDATES THE FILE INDEX.TX.

OPERATING PROCEDURES:
. R TECO
*ER ADDDOC, TE$YHXAAM$S
DATE: JAN 1975
FILENAME: FNAM.PQ
FILENAME: $C

THE ERROR MESSAGES CONSIST OF ONE OF THE MESSAGES BELOW
FOLLOWED BY THE OFFENDING TEXT LINE (IF ANY).
ALL ERRORS ARE FATAL AND CAUSE RETURN TO TECO COMMAND MODE.
IF YOU ARE NOT ACQUAINTED WITH TECO, TYPE $C TO RETURN TO OS/8
OR THE COMMAND LISTED ABOVE TO RESTART ADDDOC.

ERROR MESSAGES:
BAD INDEX.TX
I DON'T LIKE THE FIRST LINE
NO 'NAME:' FOUND

THE FILE INDEX.TX DOES NOT CONFORM TO
THE FIRST LINE OF THE INPUT FILE
THE NAME OF THE PROGRAM COULD NOT
BE DETERMINED.

APART FROM THESE ALSO ERROR MESSAGES FROM TECO ARE POSSIBLE
(E.G., COMPLAINING ABOUT LOOKUP FAILURE OR BUFFER OVERFLOW).
FOR A DESCRIPTION OF THOSE SEE THE APPROPRIATE DEC MANUAL.
2.4.1 RANDOM GENERATORS

NAME: RANDOM
AUTHOR: C. L. PIPPEL
DATE: 1972
TYPE: NONREPORTED
ENVIRONMENT:
    COMPUTER: ............... PDP8 (ANY TYPE)
    PROGRAMMING LANGUAGE: PAL
    MEMORY REQUIRED: .......... 36 LOCATIONS
    EXECUTION TIME: .......... 48 CYCLES

DESCRIPTION:
    EACH CALL OF RANDOM DELIVERS THE NEXT ELEMENT
    OF A SEQUENCE OF PSEUDO RANDOM NUMBERS. THESE
    NUMBERS ARE UNIFORMLY DISTRIBUTED BETWEEN 0 AND 4095.
    THE PERIOD OF THE SEQUENCE IS 2^33-1. THIS TYPE OF
    RANDOM GENERATORS WAS FIRST STUDIED BY TAUSWORTHE
    [RANDOM NUMBERS GENERATED BY LINEAR Recurrence MODULO TWO.
    MATH. COMPUT. 19 (1965), 201-209].

CALLING SEQUENCE:
    JMS RANDOM /DELIVERS A TWELVE BIT RANDOM
    /NUMBER IN AC, (CALL WITH AC=0)

INITIALIZATION:
    THE INITIAL POSITION IN THE RANDOM SEQUENCE IS
    DETERMINED BY C, B AND THE NINE LEAST SIGNIFICANT BITS OF A.
    TO INITIALIZE THE RANDOM GENERATOR WE PUT SOME VALUES INTO THESE
    LOCATIONS. NOTE THAT AT LEAST ONE OF THESE VALUES
    SHOULD BE NONZERO (OTHERWISE THE RANDOM GENERATOR DEGENERATES).
SOURCE:
/RANDOM GENERATOR
/PERIOD: 2^33-1
/MULTIPLY BY X^12 MOD X^33-X^13-1

RANDOM, 0

/  CLA
TAD  B
AND  K7000

CLLRAL
TAD  A
RTL
RAL
DCA  SVC1
TAD  SVC1

CLLRAL
DCA  SVC2
TAD  B
RAR
SPA
CML
RAL
AND  K777
DCA  A
TAD  C
AND  SVC2

CIA

CLLRAL
TAD  C
TAD  SVC2
DCA  B
TAD  SVC1
DCA  C
TAD  C
JMP  I  RANDOM

A,  0527
B,  2527
C,  2527

SVC1,  0
SVC2,  0

K777,  0777
K7000,  7000
2.11 GAMES

NAME: GOMOKU
AUTHORS: C.L. PIPPEL, A.E. BROUWER (MATH, CENTR.)
DATE: 73XXX
TYPE: GAME
ENVIRONMENT: OS/B
CONVERSATIONAL DEVICES: TELETYPEx, KB/1 STORAGE TUBE
CORE LAYOUT:
PRORAM: 0000-2577
DATA: 4000-6377

DESCRIPTION:

THE GAME

GOMOKU IS ALSO KNOWN AS GO-BANG OR QUINIO. USUALLY IT IS
PLAYED ON A GO BOARD, TWO PLAYERS MOVE IN TURN PUTTING A STONE OF
THEIR COLOUR ON THE BOARD, THE PLAYER WHO FIRST MAKES A ROW OF
FIVE ADJACENT PIECES (EITHER HORIZONTAL, VERTICAL OR DIAGONAL) WINS.

THE PROGRAM

TO START GOMOKU TYPE "R GOMOKU" TO THE OS/B MONITOR DOT.
INITIALLY GOMOKU CLEARS THE SCREEN AND TYPES A "#". GOMOKU IS
WAITING FOR A COMMAND, A COMMAND IS A "I" OR A "Y" FOLLOWED BY
A CR, A BOARD APPEARS, WHEN THE COMMAND "I" WAS GIVEN, GOMOKU WAITS
FOR THE MOVE OF HIS ADVERSARY, OTHERWISE GOMOKU MOVES DISPLAYING
A "O", THE USER MOVES AS FOLLOWS, HE PLACES THE CURSOR AT THE RIGHT
PLACE AND HIT A TELETYPEx BUTTON SUBSEQUENTLY, GOMOKU DISPLAYS A
"X" OR A "O" INDICATING HE ACCEPTS THE MOVE, A CR ENDS THE CURRENT
GAME AND STARTS A NEW ONE, GOMOKU EXITS TO THE OS8 MONITOR IF THE
USER TYPES A "C" INSTEAD OF A COMMAND.

MESSAGES

THE MESSAGES YOU WILL SEE MOST OFTEN ARE:

"PDP8 WINS"

AND

"PDP8 WINS AGAIN"

THESE MESSAGES APPEAR WHEN GOMOKU WINS THE GAME, IF THE
USER WINS GOMOKU ANSWERS:

"YOU WIN"

IF NEITHER THE USER NOR GOMOKU CAN WIN THE GAME GOMOKU PRINTS:

"DRAW"
PRPCT is a one page module which enables the user to display a picture, composed of (in)visible incremental vectors, all of equal length. Also routines are available to erase the screen and to reset the current position. We describe them below:

1) CLEARSC
   CLRSC erases the screen.
   Calling sequence:
   /CLEAR SCREEN
   JMS CLRSC

2) SETPNT
   SETPNT defines the X,Y int registers by drawing a short
   invisible absolute vector to the point (C0,C1). When the length
   of the stroke is greater than 40 the routine should be called twice.
   This is faster and more accurate than drawing one invisible long
   vector.
   Calling sequence:
   /SET POINT
   JMS SETPNT
   C0 /X coordinate
   C1 /Y coordinate

3) PRPCT
   PRPCT draws a sequence of incremental short vectors.
   Each vector can be drawn in one of the following directions:
   CODE  DIRECTION
   0    0,DX
   1    DX,DX
   2    DX,0
   3    DX,-DX
   4    0,DX
   5    -DX,-DX
   6    -DX,0
   7    -DX,-DX
   (WHERE 0<= DX <=40)
   The following figure illustrates the relation between the
   direction of each stroke and the code involved:

   FIG (1)  6 . 2
            5 4 3
CALLING SEQUENCE:

/PRINT PICTURE
TAD (DX) /SIZE OF EACH STROKE
JMS PRPICT
ARG1 /-(# OF STROKES)
ARG2
...
ARGN


EXAMPLE:

THE INSTRUCTION SEQUENCE

JMS CLRSC /CLEAR SCREEN
JMS SETPNT /SET CURRENT POSITION
-4
-4
TAD (10) /DIMENSION
JMS PRPICT
-4 /-(# OF STROKES)
7024 /INTENSIFY BITS + DIRECTION CODES
4600

WILL RESULT IN A SMALL SQUARE WITH CENTRE (0,0)

NOTE:

-BEFORE USING PRPICT FOR THE FIRST TIME IT IS NECESSARY TO SET THE SCOPE FLAG (E.G., USE CLRSC)

-BECAUSE OF THE DRIFT OF THE ANALOG REGISTERS OF THE SCOPE IT IS ADVISABLE TO RESTRICT THE NUMBER OF STROKES OF ONE PICTURE. HOWEVER IT IS NECESSARY TO DEFINE THE CURRENT POSITION EXACTLY AT REGULAR TIMES

-AT THE PAGE OF PRPICT ROOM IS AVAILABLE TO PLACE THE A/D CONVERSION ROUTINE
<table>
<thead>
<tr>
<th>ITEM</th>
<th>PAGE(S)</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2  PIP</td>
<td>1</td>
<td>JAN 1975</td>
</tr>
<tr>
<td>1.3  MCEDIT</td>
<td>1-2</td>
<td>JAN 1975</td>
</tr>
<tr>
<td>1.3  EDIT</td>
<td>3-4</td>
<td>JAN 1975</td>
</tr>
<tr>
<td>1.3  MCEDIT AB-V16</td>
<td>5</td>
<td>FEB 1975</td>
</tr>
<tr>
<td>1.4  DCP</td>
<td>1-5</td>
<td>JAN 1975</td>
</tr>
<tr>
<td>1.4  DCP (ERROR TABLE)</td>
<td>6</td>
<td>FEB 1975</td>
</tr>
<tr>
<td>1.8  I03</td>
<td>1-2</td>
<td>FEB 1975</td>
</tr>
<tr>
<td>1.8  I0G</td>
<td>3-4</td>
<td>FEB 1975</td>
</tr>
<tr>
<td>1.11 FLXISO</td>
<td>1</td>
<td>JAN 1975</td>
</tr>
<tr>
<td>1.11 SBIN</td>
<td>2</td>
<td>JAN 1975</td>
</tr>
<tr>
<td>1.11 ISOFIX</td>
<td>3</td>
<td>FEB 1975</td>
</tr>
<tr>
<td>1.11 BRAIL</td>
<td>4-9</td>
<td>FEB 1975</td>
</tr>
<tr>
<td>1.11 RDCOMM</td>
<td>10</td>
<td>MAR 1975</td>
</tr>
<tr>
<td>1.11 SRCCOM (SOURCE COMPARE)</td>
<td>11</td>
<td>MAR 1975</td>
</tr>
<tr>
<td>2.1  ANTIQ</td>
<td>1-2</td>
<td>FEB 1975</td>
</tr>
<tr>
<td>2.2  ADDDOC,TE</td>
<td>1</td>
<td>JAN 1975</td>
</tr>
<tr>
<td>2.4  RANDOM</td>
<td>1-2</td>
<td>JAN 1975</td>
</tr>
<tr>
<td>2.11 GOMOKU</td>
<td>1</td>
<td>MAR 1975</td>
</tr>
<tr>
<td>3.8  PRPICT</td>
<td>1-2</td>
<td>MAR 1975</td>
</tr>
<tr>
<td>INDEX</td>
<td>1</td>
<td>MAR 1975</td>
</tr>
</tbody>
</table>