A/S REGNECENTRALEN

CUSTOMER SERVICE DIVISION

SCANDINAVIAN INFORMATION PROCESSING SYSTEMS

Dth

OUR REF. PEH/BECA DATE 27th January, 1971.

Dear Sirs,

Please find enclosed the first part of the 'TEST' system which is a set of hardware testprograms for use under the RC 4000 multiprogramming system.

The first part consists of:

descriptions

'TEST'	RCSL: 44-D10
Typewriter and teleterminal	RCSL: 44-D11
Lineprinter	RCSL: 44-D12
Magnetic tape	RCSL: 44-D13
Test of RC 4317 Punched Card Reader	RCSL: 31-D21

program tapes

'TEST'	RCSL: 44-T2.2/2
A format tape for printertest (5 copies)	RCSL: 44-T2.3/F
Further, a RC 4000 troubleshooting scheme is enclosed.	RCSL: 44-RT 183
	RCSL: 44-D40

A/S REGNECENTRALEN

Per Hansen

Per Hansen

Foreløbig vejledning til testsystemet, TEST 2.2/3. 10.8.71.

ALGOL6 compiler og library forudsættes i maskinen.

Den er ændret på følgende punkter:

mt-test:

a kan nu også køre med CDC (men tester ikke read reverse).

b do.

c virker ikke

d virker kun på Ampex

e kan køre med CDC stationer. Operationen read reverse hedder 'b'.

Test a og b spørger om mode, og man kan taste n hhv. p for nrz og pe mode henholdsvis. Mode er irrelevant for Ampex.

Test e arbejder kun i pe mode på CDC.

Desuden er der indlagt to nye tester, nemlig bs (backing storage) og pt (paper tape).

En kort beskrivelse af

bs test

Først prøver testen at reservere et areal på de enheder der specificeres, f.eks. bsC04 på tromlen, (device 4) o.s.v.

test a

Spørger om bufferstørrelsen (i segmenter). Dernæst om mode, der kan være:

c checkall (både status og data).

d0,d1,d2,d3,d4

forskellige testdatamønstre.

d0 er standard worstcase mønstret.

m no check but monitoring of the percentage of statuserrors after every run.

p checkall with printing of dataerror.

r read

w write

t0 or t1 or t2 etc.

number of tries, i.e. rereadings or rewritings in case of statuserror.

(with the present monitor t0 will mean 3 tries).

Hvis både read og write specificeres vil samtlige segmenter først skrives 1 gang og derpå checklæses 1 gang i hvert run.

movetest

Udfører en random hovedbevægelse spørgsmålet 'check, tries 'kan f.eks. besvares således:

yes 5

paper tape test , a og b

Hvis både en læser og en punch er specificeret, udskrives først nogen meter papirstrimmel af punchen som angivet ved length=

parity kan være:

o odd
e even
n no parity

test a checklæser nu den udperforerede strimmel.

test b gør det samme men i modsat paritet og kontrollerer herved paritetskontrollen (n-parity må ikke specificeres her).

test c læser en vilkårlig strimmel uden check (motions test).

test d puncher en opgivet sekvens af tegn.

Venlig hilsen

Per Hansen

Per Hansen

RCSL: 44-D10
Author: Per Hansen
Edited: October, 1970
Type: Algol 5 program

RC 4000

TIMESHARED TESTPROGRAM LIBRARY

'TEST' SYSTEM

Keywords: RC 4000, Diagnostic program, Algol Program, ISO tape

ABSTRACT: The TEST system is an administrator and some procedures which are intended for use during testing and error location on RC 4000 peripheral devices. The system operates under the fileprocessor thus allowing other users to run programs on the part of the computer not under test.

A/S REGNECENTRALEN
Falkoner Ale 1
2000 Copenhagen F

CONTENTS:

'TEST' system

1. Introduction

The system consists chiefly of three types of programs:

1. CPU-test programs
2. Peripheral device test programs
3. System test and routine test programs

All programs publications are identified by a number of a format like this:

<type>.<serial>/<version>

where <type> is one of the three types mentioned above, <serial> is the serial number and <version> the version number of the actual program.

The programs are mainly written in algol utilizing standard zone procedures as an attempt to make the program production easy and at the same time be able to handle any error situation during run time.

However, CPU-tests in ALGOL are too complex to facilitate error location, so it turned out to be a good idea to write these programs in SLANG code. On the other hand, all-over system tests and routine tests can be written in high-level ALGOL if one is satisfied with the standard error reactions.

This gives a system of mixed-type programs which are organized so that all peripheral device tests in ALGOL are linked together to form a set of procedures under administration of the program 'TEST' whereas the SLANG programs and special ALGOL programs are published as independent programs.

2. The TEST administrator

The TEST system consists of a set of start-up and utility procedures plus a number of blocks, each block containing a number of test procedures for one particular devicekind.

At present seven different devicekinds exist. The sense of these kinds differs slightly from that of the monitor:

6	backing storage
8	typewriter, teleterminal
10	tape reader and punch
12	plotter
14	printer
16	card reader
18	magnetic tape

The reason why reader and punch test is combined is that it may be useful to let the tape run from punch to reader to perform an effective test of one of these devices.

The TEST system makes full use of the RC 4000 multiprogramming features. Thus most output-zones are doublebuffered and all tests may be performed on more devices at the same time.

This may give peculiar effects in error situations but the error messages should enable the operator to handle the situation.

The strategy of a set of tests are accommodated to the devicekind in question, e.g. the printertest is made fast to keep the printer 'warm' etc. But if, for some reason, you want to test another kind with this test a warning message is output and the zone description is changed automatically to match the different format of document.

3. Start of TEST

The system is published as a texttape which must be translated in a process having a core area of at least 12000 bytes and the catalog key 3.

The tape is input by the fp command:

```
i tre
```

and if the translation is succesful the program is called once without parameters. The backing storage area occupied is app. 100 segments.

After the translation the program may be run in a core area as small as 6000 bytes (dependent on the number and kind of devices) but the output may be considerably slower.

TEST is called in the following way:

$$\left\{\langle\text{outfile}\rangle=\right\}_{0}^{1}\text{test}\left\{\begin{array}{l}\langle\text{empty}\rangle \\ \langle\text{s}\rangle\langle\text{kind}\rangle\left\{.\langle\text{device}\rangle\right\}_{1}^{\infty}\end{array}\right.$$

where <kind> is one of the following:

bs	backing storage
tw	typewriter
pt	paper tape (reader and punch)
pl	plotter
lp	lineprinter
cr	card reader
mt	magnetic tape

and <device> is the number of a device. If this is the operators console and the devicenumber of this is unknown you may use the letter c.

If the parameter list is empty a listing of the devices in the actual RC 4000 system as implemented in the monitor is performed. After completion of this the run is terminated.

At present the parameter <outfile> has no effect, but in future versions it will be used to define the outputmedium for error-messages.

4. Running of TEST

When the testprogram asks:

testprogram:

you may type a letter defining the program desired followed by <NL>. If only a <NL> is typed the set of testprograms is listed.

Later the program asks:

number of runs =

and you type a number, e.g. 9. This causes nine identical runs to be executed and for every run the runnumber is output on the operators

console. If the number of runs specified is greater than 9 but less than 100 the runnumber is output for every 10 runs and so on.

Any time the programs asks for a character (or a text), i.e. all input but numbers the character may either be typed directly or it may be defined by its numerical value by typing a semicolon followed by the decimal value of the character. The value typed is interpreted modulo 128.

Most device-tests include a sequence test where you may define a certain sequence of characters. When the program asks:

type sequence:

the operator is expected to type a sequence of characters and numbers according to the following syntax:

$$\langle\text{sequence}\rangle\,,\langle\text{repeat}\rangle\left\{\,,\langle\text{sequence}\rangle\,,\text{repeat}\rangle\right\}_{0}^{\infty}\langle\text{NL}\rangle$$

where <sequence> is any sequence of characters except a comma followed by one or more digits (but including characters defined by semicolon + decimal value). <repeat> is a decimal number defining how many times the preceding sequence is to be repeated.

The resulting sequence is stored in an array the size of which depends on the testprogram in question. If the array is too small the following warning is output:

xxxfull

TEST proceeds using the string in the array.

5. Handling of input-output errors

Any transfer to or from the peripheral device (operators console included) are checked and if the result is unexpected the logic status word (see algol 5 manual, P. 46f) and the number of bytes transferred is output on the console. The various bits are named as follows:

0	local
1	parity
2	timer
3	overrun
4	length
5	doc_end
6	loadpoint
7	tapemark
8	we
9	hi
10	10?
11	11?
12	12?
13	13?
14	14?
15	output_lost
16	word_defect
17	pos_error
18	unknown
19	malfunction
20	illegal
21	reservation
22	ok
23	hard_error

The malfunction bit means that the device in question has been either disconnected or has remained busy after an interrupt. The reservation bit means that the device cannot be reserved.

During input from typewriter a parity error is converted to the SUB character. The action of this is

1. The message:

   xinput parity

   is output on the console

2a. In case of number input the entire number is skipped and TEST requests for the operator to type the number again.

2b. In case of character (text) input the SUB character is stored and TEST proceeds.

NOTE: Because of the double-buffering during output the errormessages will not be output immediately after the error has occurred but will be delayed until output of the current and the next buffer has been completed. Double-buffered input from e.g. tape stations has the opposite effect because the zone procedures when checking block n already is busy with transfer of block n+1.

6. Examples

Example 1:

When calling:

```
test
```

the following list is output:

```
bs   4
tw   2 10 18 19 20
pt   0 1
pl
lp   5
cr
mt   6 7 8 9
```

Example 2:

The call:

```
test tw.c.5.20
```

causes the warning:

```
xxxtest device 5 no typewriter
```

and the program proceeds to make testruns of the operators own console (c), the printer and the teletypewriter

Example 3:

When the programs asks:

type sequence:

and you type:

```
abcd ,2,;95,5,,,7,
pølse3,2
```

this is accepted as the sequence:

```
abcd abcd _____,,,,,,,
pølse3
pølse3
```

Example 4:

You have programmed a special sequence of commandsand put it on papertape. Then change current input to RC 2000:

```
(i trf
test lp.5)
```

Example 5:

The call:

```
lp= test mt.6.7
```

will initiate a test of the tapestations device 6 and 7. Error messages will be output at the medium defined by the catalog entry lp. Controlinformation will still be output on the operators console.

Example 6:

The conversation:

```
number of runs = 223
xinput parity
number of runs =
```

indicates that the typewriter has generated parity error during input of the number 223.

7. List of errormessages

x<number>,<status><number><bytes>
During the test of a device there has been status error. The first <number> is the devicenumber, the second one is the number of bytes transferred. The statusbit hard_error means that the device cannot be reserved.

xinput parity
A <SUB> character has been received during input. This is usually due to parity error on the typewriter. TEST proceeds.

xxxfp name
Algol is not present in the machine.

xxxfp syntax
TEST has been called in a wrong way.

xxxfull
During input of a character sequence the storage available is exceeded. TEST proceeds.

xxxline 488 stack
The process area is too small.

xxxpermanent test protected
xxxset test protected
The process does not own the catalog key 3.

xxxtest<devicekind> not implemented
No test for this devicekind. TEST terminates.

xxxtest device <number> no <devicekind>
Is a warning about a wrong devicekind, e.g. the printertest is performed on magtape etc. TEST proceeds.

8. Program text

```
1
1 CLEAR TEST
1 TEST=SET 150
1 IF OK.YES
1 (PERMANENT TEST.3
1 TEST=ALGOL
1 IF OK.YES
1 (END
1 TEST))
1 TEST  2.2/2
1 BEGIN COMMENT
2
```

```
 2 RC 4000 PERIPHERAL DEVICE TEST
 3 RCTS 2.2/2     26.6.70     PEH
 4 VERSION:        1.2.71
 5
 6 NAME IN CATALOG: TESTHEAD;
 7    INTEGER DEVS,KIND,TOPDEV,C,L,I,D,RUNS,RUNSTEP,RUNNO,M,PARITY,OPC;
 8    REAL R;
 9    BOOLEAN NL,BSL;
10    BOOLEAN ARRAY RESERVED(1:24);
11    REAL ARRAY PARAM(0:1);
12    INTEGER ARRAY IA(0:19), DEVICE(1:32), DKIND(1:32);
13    ZONE OUTC(50,1,ERROR), OUTL(256,2,ERROR), INC(16,1,ERROR);
14 COMMENT THE STANDARD ZONE IN IS USED AS DUMMY ZONE AND MUST NOT
15    BE USED FOR INPUT;
16
17 COMMENT GLOBAL UTILITY PROCEDURES;
18    PROCEDURE IND;
19    BEGIN INTEGER ARRAY IA(1:20);
20       GETZONE(INC,IA);
21       IF IA(1)=8 THEN SETPOSITION(INC,0,0);
22    END PROCEDURE IND;
23
24    INTEGER PROCEDURE MODEKIND(PROC);
25    INTEGER PROC;
26    BEGIN SYSTEM (5,PROC,IA);
27       MODEKIND:=CASE IA(0)//2 OF  (0,4,4,8,10+PARITY SHIFT 12,
28       12+PARITY SHIFT 12,14,16,18+PARITY SHIFT 12,
29       0,0,0,0,0,14,0,18+PARITY SHIFT 12,8,14,0,0,0,8);
30    END PROCEDURE MODEKIND;
31
32    BOOLEAN PROCEDURE ANYMESSAGE;
33    BEGIN INTEGER BUF;
34       SYSTEM(6,BUF,PARAM);
35       IF BUF >0 THEN
36       BEGIN MONITOR(20,IN,BUF,IA);
37          IA(9):=1;
38          MONITOR(22,IN,BUF,IA);
39          ANYMESSAGE;
40          ANYMESSAGE:=TRUE;
41       END ELSE ANYMESSAGE:=FALSE;
42    END PROCEDURE ANYMESSAGE;
43
44    PROCEDURE RUNADM(ENDRETURN,RUNEND);
45    LABEL ENDRETURN; PROCEDURE RUNEND;
46    BEGIN
47 AGAIN:
48       IF INCREASE(D)<DEVS THEN GOTO NEXTRUN;
49       RUNEND;
50       D:=1;
51       IF RUNNO=RUNS THEN
52       BEGIN WRITE(OUTC,NL,2,<:TEST END:>,NL,1);
53          GOTO ENDRETURN
54       END;
55 NEXTRUN:
56       IF ANYMESSAGE THEN
57       BEGIN WRITE(OUTC,NL,1,<:OPERATOR TERMINATION:>);
58          GOTO ENDRETURN;
59       END;
60       IF D=1 THEN
61       BEGIN IF INCREASE(RUNNO) MOD RUNSTEP=0 THEN
62          WRITE(OUTC,NL,1,<:RUN NO.:>,<<DDDDDDDD>,RUNNO,NL,1); UD;
63       END;
64       IF ~,RESERVED(D) AND KIND<>18 THEN GOTO AGAIN;
65    END PROCEDURE RUNADM;
66
67 COMMENT
68
```

```
62 RC 4000 PERIPHERAL DEVICE TEST                                    PAGE 2
63
63     INTEGER PROCEDURE TESTPROG;
64     BEGIN INTEGER TP;
65        ANYMESSAGE;
66  RTP:  WRITE(OUTC,<:<10>TESTPROGRAM:  :>); UD; IND;
67        FOR TP:=INCHAR WHILE TP=32 DO;
68        IF TP=10 THEN GOTO A;
69        IF INCHAR<>10 OR TP<97 OR TP>121 THEN GOTO RTP;
70        IF TP=121 THEN GOTO A;
71  READRUNS:
72        WRITE(OUTC,<:NUMBER OF RUNS = :>); UD; IND;
73        RD(READRUNS,RUNS);
74        IF RUNS<=0 THEN GOTO READRUNS;
75        RUNSTEP:=10**(ENTIER(1/LN(10)*LN(RUNS)));
76        D:=RUNNO:=0;
77  A:    TESTPROG:=CASE ROUND SIGN(TP-10)+1 OF (1,TP-95);
78     END PROCEDURE TESTPROG;
79
79     PROCEDURE ERROR(Z,S,B);
80     ZONE Z; INTEGER S,B;
81     BEGIN INTEGER SAVEBYTE,DN,PA;
82        OWN INTEGER DISCONL;
83        SAVEBYTE:=B;
84        IF B< 0 THEN B:=0;
85        PA:=MONITOR(4,Z,I,IA);
86        IF PA=0 THEN PA:=DEVICE(D);
87        DN:=DEVICENUMBER(PA);
88        IF L=OPC THEN
89        BEGIN INTEGER ARRAY IA(1:20),IB(1:20);
90           GETZONE(Z,IA); GETZONE(OUTC,IB);
91           IF IA(19)=IB(19) THEN
92           BEGIN DISCONL:=IF S SHIFT (-2)
93              EXTRACT 10=0 THEN 0 ELSE DISCONL+1;
94              IF DISCONL >100 OR S SHIFT (-8) EXTRACT 1=1
95              THEN GOTO RETURN;
96           END;
97        END;
98        IF DN=L THEN
99        BEGIN PRINT(OUTC,DN,PA,S,SAVEBYTE); UD;
100          IF BSL AND S SHIFT (-18) EXTRACT 1=1 THEN GOTO L_END;
101       END ELSE
102       BEGIN PRINT(OUTL,DN,PA,S,SAVEBYTE); UDL;
103       END;
104 RETURN:
105       S:=2;
106    END PROCEDURE ERROR;
107
107    PROCEDURE PRINT(Z,DN,PA,S,B);
108    ZONE Z; INTEGER DN,PA,S,B;
109    BEGIN INTEGER ARRAY IA(1:1);
110       SYSTEM(5,PA,IA);
111       WRITE(Z,NL,1,FALSE,3,<:*:>,<<DD>,DN,<:  :>);
112       FOR I:=(-23) STEP 1 UNTIL 0 DO
113       IF S SHIFT I EXTRACT 1=1 THEN WRITE(Z,CASE I+24 OF (
114       <:LOCAL :>,<:PARITY :>,<:TIMER :>,<:OVERRUN :>,<:LENGTH :>,
115       <:DOC<95>END :>,<:LOADPOINT :>,<:TAPEMARK :>,<:WE :>,
116       <:HI :>,<:10? :>,<:11? :>,<:12? :>,<:13? :>,<:14? :>,
117       <:OUTPUT<95>LOST :>,<:WORD<95>DEFECT :>,<:POS<95>ERROR :>,
118       <:UNKNOWN :>,<:MALFUNCTION :>,<:ILLEGAL :>,
119       <:RESERVATION :>,<::>,<:HARD<95>ERROR :>)) ELSE IF
120       (IA(1)=18 OR IA(1)=34) AND (I= -15 OR I= -14) THEN
121       WRITE(Z,CASE I+16 OF(<:PROTECT_:>,<:LO<95>DEN_:>));
122       WRITE(Z,<<D>,B,<: BYTES:>,NL,1);
123    END PROCEDURE PRINT;
124
124 COMMENT
125
```

```
125
126
126     PROCEDURE PRINTNAME;
1       WRITE(OUTC, CASE KIND//2-2 OF (<:BACKING STORAGE:>,<:TYPEWRITER:>,
1       <:READER/PUNCH:>,<:PLOTTER:>,<:PRINTER:>,<:CARD READER:>,
129     <:MAGTAPE:>));
130
130     INTEGER PROCEDURE INCHAR;
131     BEGIN INTEGER I,J;
132        READCHAR(INC,I);
133        IF I=26 THEN WRITE(OUTC,NL,1,<:*INPUT PARITY:>,NL,1); UD;
134        IF I=59 THEN
135        BEGIN IF READCHAR(INC,J)=2 THEN
136           BEGIN REPEATCHAR(INC);
137              READ(INC,I);
138           END; REPEATCHAR(INC);
139        END; INCHAR:=I;
140     END PROCEDURE INCHAR;
141
141     PROCEDURE PRINTBITS(MARG,REC,EXPD,CHAR,BITS);
142     VALUE REC,EXPD,CHAR,BITS; REAL REC,EXPD;
143     INTEGER CHAR, BITS,MARG;
1       BEGIN REAL R; INTEGER I;
145        FOR R:=REC,EXPD DO
146        BEGIN WRITE(OUTL,NL,1,FALSE ADD 32,MARG,IF R=REC THEN <:RECEIVED: :>
147           ELSE <:EXPECTED: :>);
148           FOR I:= -47 STEP 1 UNTIL BITS DO WRITE(OUTL,IF R SHIFT I EXTRACT
149           1=1 THEN <:1:> ELSE <:.:>,IF I MOD CHAR=0 THEN <: :> ELSE <::>);
150           WRITE(OUTL,<<-DDDDDDD>,R SHIFT (-24) EXTRACT 24);
151           IF BITS >-24 THEN WRITE(OUTL,<<_-DDDDDDD>,R EXTRACT 24);
152        END;
153        WRITE(OUTL,NL,1);
154     END PROCEDURE PRINTBITS;
155
155
155
155 COMMENT
156
```

```
157
157     INTEGER PROCEDURE PACK(TABLE,FIRST,TOP,MASK,BIT);
1       COMMENT THIS PROCEDURE PACKS A SEQUENCE READ FROM THE TYPEWRITER
159     INTO AN ARRAY NAMED TABLE. THE CHARACTERS ARE INTERPRETED MODULO MASK
160     AND MASKED WITH BIT WHICH RESEMBLES THE NUMBER OF BITS IN THE
161     CHARACTER PACKED. THEN NUMBER OF CHARACTERS PACKED PER WORD IS 48//BIT.
162        THE RETURN VALUE PACK YIELDS THE NUMBER OF THE
163     LAST ELEMENT PACKED AS PACK//6. PACK WILL NEVER EXCEED TOP-2;
164
164     VALUE FIRST, TOP, MASK,BIT; INTEGER FIRST, TOP, MASK,BIT;
165     REAL ARRAY TABLE;
166     BEGIN INTEGER A, B, C, REPEAT, L, L1, L2, CH;
167        PROCEDURE P(I);
168        VALUE I; INTEGER I;
169        BEGIN B:=(CH-A MOD CH-1)*BIT;
170           TABLE(A//CH):=(TABLE(A//CH) SHIFT (-B) ADD I) SHIFT B;
171        END PROCEDURE P;
172  START:CH:=48//BIT;
173        FIRST:=FIRST//6*CH;
174        TOP:=TOP//6*CH;
175  REQUEST:
1          L1:=FIRST;
177        FOR A:=FIRST//CH STEP 1 UNTIL (TOP-1)//CH DO TABLE(A):=0.0 SHIFT 48;
178        WRITE(OUTC,<:<10>TYPE SEQUENCE:<10>:>); UD; IND;
179        FOR A:=FIRST STEP 1 UNTIL TOP-3 DO
180        BEGIN
181  NEXT:    C:=INCHAR MOD MASK;
182           IF C<>44 THEN GOTO T;
183           C:=INCHAR;
184           IF C<58 AND C>=48 THEN
185           BEGIN REPEATCHAR(INC);
186              RD(REQUEST,REPEAT);
187              L2:=A-1;
188              FOR REPEAT:=REPEAT-1 WHILE REPEAT>0 DO
189              FOR L:=L1 STEP 1 UNTIL L2 DO
190              BEGIN IF A>TOP-3 THEN GOTO FULL;
191                 P(TABLE(L//CH) SHIFT ((L MOD CH-CH+1)*BIT) EXTRACT BIT);
192                 A:=A+1;
193              END;
19             L1:=A;
19             REPEATCHAR(INC);
196              GOTO IF INCHAR=10 THEN NL ELSE NEXT;
197           END;
198           REPEATCHAR(INC);
199  T:       P(C EXTRACT BIT);
200        END;
201  FULL: WRITE(OUTC,<:<10>***FULL:>);
202  NL:   IF MASK=128 THEN P(10);
203        PACK:=A//CH*6;
204     END PROCEDURE PACK;
205
205  COMMENT
206
```

```
207
207     PROCEDURE RD(PARITY,NUMBER);
208     LABEL PARITY; INTEGER NUMBER;
2       BEGIN IF INCHAR=26 THEN GOTO PARITY;
210        REPEATCHAR(INC);
211        READ(INC,NUMBER);
212        REPEATCHAR(INC);
213        IF INCHAR=26 THEN GOTO PARITY;
214     END PROCEDURE RD;
215
215     INTEGER PROCEDURE DEVICENUMBER(PROCADDRESS);
216     INTEGER PROCADDRESS;
217     BEGIN INTEGER ARRAY IA(0:5);
218        SYSTEM(5,PROCADDRESS,IA);
219        DEVICENUMBER:=IA(5) SHIFT (IF IA(0)=4 THEN (-13) ELSE (-6));
220     END PROCEDURE DEVICENUMBER;
221
221     PROCEDURE GETPARITY;
222     BEGIN
223 REP:   WRITE(OUTC,NL,1,<:PARITY = :>); UD; IND;
224        C:=INCHAR;
225        PARITY:=IF C=10 OR C=111 THEN 0 ELSE IF C=101 THEN 2 ELSE
2          IF C=110 THEN 4 ELSE (-1);
227        IF PARITY= -1 THEN GOTO REP;
228     END PROCEDURE GETPARITY;
229
229     PROCEDURE UD;
230     SETPOSITION(OUTC,0,0);
231
231     PROCEDURE UDL;
232     IF -,BSL THEN SETPOSITION(OUTL,0,0);
233
233 TESTSTART:
234
234     NL:=FALSE ADD 10;
235     M:= -1 SHIFT 16 + 16381;
236     COMMENT M=ALL ONES BUT WE,HI,OK;
237     PARITY:=0;
238     D:=SYSTEM(7,I,PARAM);
239     OPC:=DEVICENUMBER(D);
2       COMMENT DEVICENUMBER OF OPERATORS CONSOLE;
24      I:=0;
242     OPEN(OUTC,8,STRING PARAM(INCREASE(I)),M);
243     I:=0;
244     OPEN(INC,8,STRING PARAM(INCREASE(I)),M);
245     SYSTEM(5,74,IA);
246     TOPDEV:=(IA(1)-IA(0))//2; COMMENT TOTAL NUMBER OF DEVICES;
247
247
247     BEGIN INTEGER ARRAY DEVICELIST(0:TOPDEV-1),TESTDEVS(0:TOPDEV-6,0:5);
248        INTEGER RESULT;
249        PROCEDURE PRINTDEVICES;
250        COMMENT THIS PROCEDURE LISTS THE DEVICES IMPLEMENTED IN THE
251        ACTUAL SYSTEM;
252        BEGIN FOR KIND:=6 STEP 2 UNTIL 18 DO
253           BEGIN WRITE(OUTL,<:<10>:>,CASE (KIND//2)-2 OF (
254                 <:BS :>,<:TW :>,<:PT :>,<:PL :>,<:LP :>,<:CR :>,<:MT :>));
255              FOR DEVS:=0 STEP 1 UNTIL TOPDEV-1 DO
256              BEGIN SYSTEM(5,DEVICELIST(DEVS),IA);
2                IF IA(0)=36 OR IA(0)=46 THEN
258              IA(0):=8;
259              IF IA(0)=12 AND IA(1)=REAL(<:PUN:>)
260              SHIFT (-24) EXTRACT 24 THEN IA(0):=10;
261
261 COMMENT
262
```

```
263
263                IF IA(0)=34 THEN IA(0):=18;
264                IF IA(0)=KIND THEN WRITE(OUTL,<<DDDD>,DEVS);
265             END;
266          END; UDL;
267          GOTO TESTEND
268       END PROCEDURE PRINTDEVICES;
269
269       PROCEDURE PARAMERROR(VAL);
270       VALUE VAL; REAL VAL;
271       BEGIN WRITE(OUTC,<:<10>***TEST  PARAM :>,<<DDDD>,VAL);
272          GOTO TESTEND
273       END PARAMERROR;
274
274       BSL:=FALSE;
275       SYSTEM(5,IA(0),DEVICELIST);
276       RESULT:=0;
277       IF SYSTEM(4,1,PARAM) SHIFT (-12) =6 THEN
278       BEGIN SYSTEM(4,0,PARAM);
279          D:=3;
280          I:=0;
281          OPEN(OUTL,0,STRING PARAM(INCREASE(I)),0);
282          RESULT:=MONITOR(42,OUTL,I,IA);
283          CLOSE(OUTL,TRUE);
284          IF RESULT<>0 THEN GOTO C;
285          IF IA(0)< 0 THEN FOR I:=0 STEP 1 UNTIL 3 DO
286          PARAM(I//2):=PARAM(I//2) SHIFT 24 ADD IA(I+1);
287          I:=0;
288          IF IA(0)< 0 THEN
289          BEGIN OPEN(OUTL,IA(0) EXTRACT 23,STRING PARAM(INCREASE(I)),
290                M EXTRACT 22);
291             I:=MONITOR(6,OUTL,I,IA);
292          END ELSE
293          BEGIN OPEN(OUTL,4,STRING PARAM(INCREASE(I)),1 SHIFT 18);
294             MONITOR(52,OUTL,I,IA); COMMENT CREATE;
295             I:=MONITOR(8,OUTL,I,IA); COMMENT RESERVE;
296             BSL:=TRUE;
297          END;
298          IF I<>0 THEN
299          BEGIN I:=0;
300             WRITE(OUTC,<:<10>***:>,
301                STRING PARAM(INCREASE(I)),<: RESERVATION:>); UD;
302             CLOSE(OUTL,TRUE);
303             BSL:=FALSE;
304             GOTO C;
305          END ELSE SETPOSITION(OUTL,0,0);
306       END ELSE
307       BEGIN D:=2;
308 C:       SYSTEM(7,I,PARAM);
309          I:=0;
310          OPEN(OUTL,8,STRING PARAM(INCREASE(I)),M EXTRACT 22);
311          IF RESULT<>0 THEN PARAMERROR(0,0);
312       END;
313       L:=DEVICENUMBER(MONITOR(4,OUTL,I,IA));
314       IF SYSTEM(4,D-1,PARAM)< 1 THEN PRINTDEVICES;
315       KIND:=
316          IF PARAM(0)=REAL <:BS:> THEN  6 ELSE
317          IF PARAM(0)=REAL <:TW:> THEN  8 ELSE
318          IF PARAM(0)=REAL <:PT:> THEN 10 ELSE
319          IF PARAM(0)=REAL <:PL:> THEN 12 ELSE
320          IF PARAM(0)=REAL <:LP:> THEN 14 ELSE
321          IF PARAM(0)=REAL <:CR:> THEN 16 ELSE
322          IF PARAM(0)=REAL <:MT:> THEN 18 ELSE
323          4;
324
324 COMMENT
325
```

```
326
326         IF KIND=4 THEN PARAMERROR(PARAM(0));
3           DEVS:=0;
328         FOR I:=SYSTEM(4,DEVS+D,PARAM) WHILE I SHIFT (-12)=8 DO
329         BEGIN IF I<>8 SHIFT 12 +4 THEN
330            PARAM(0):=IF PARAM(0)=REAL <:C:> THEN OPC ELSE 1000;
331            IF PARAM(0)>=TOPDEV THEN PARAMERROR(PARAM(0));
332            FOR I:=0 STEP 1 UNTIL DEVS-1 DO
333            IF TESTDEVS(I,5)=PARAM(0) THEN PARAMERROR(PARAM(0));
334            SYSTEM(5,DEVICELIST(PARAM(0)),IA);
335            DEVICE(DEVS+1):=DEVICELIST(PARAM(0));
336            FOR I:=0 STEP 1 UNTIL 4 DO TESTDEVS(DEVS,I):=IA(I);
337            TESTDEVS(DEVS,5):=PARAM(0);
338
338            IF TESTDEVS(DEVS,0)<>KIND AND -,(KIND=10
339            AND TESTDEVS(DEVS,0)=12
340            AND TESTDEVS(DEVS,1)=REAL(<:PUN:>) SHIFT
341            (-24) EXTRACT 24 OR
342            KIND=8 AND (TESTDEVS(DEVS,0)=36 OR TESTDEVS(DEVS,0)=46)
343            OR KIND=18 AND TESTDEVS(DEVS,0)=34) OR
344            KIND=12 AND TESTDEVS(DEVS,1)=REAL(<:PUN:>) SHIFT
3           (-24) EXTRACT 24 THEN
346            BEGIN WRITE(OUTC,<:<10>***TEST  DEVICE :>,<<DDD>,
347               TESTDEVS(DEVS,5),<:  NO :>); PRINTNAME; UD;
348               IF KIND=18 THEN GOTO TESTEND;
349            END;
350            DEVS:=DEVS+1;
351            RESERVED(DEVS):=FALSE;
352            DKIND(DEVS):=IA(0);
353            R:=DEVS;
354            IF DEVS=TOPDEV THEN PARAMERROR(R);
355         END;
356      END TEST START-UP PROCEDURES;
357
357      CASE KIND//2-2 OF
358      BEGIN COMMENT HERE FOLLOWS 8 BLOCKS EACH CONTAINING A SET OF
359         TESTPROGRAMS FOR ONE PARTICULAR DEVICEKIND.
360         DEVICEKINDS NOT IMPLEMENTED MUST BE REPRESENTED BY A
361         DUMMYBLOCK WHICH WRITES THE MESSAGE <:NOT IMPLEMENTED:>.
3           THE BLOCKS ARE ENTERED WITH THE FOLLOWING PARAMETERS:
3
363                  DEVS      = NUMBER OF DEVICES TO BE TESTED
364         DEVICE(1:DEVS)     = PROCESS DESCRIPTION ADDRESSES
365          DKIND(1:DEVS)     = DEVICEKIND AS DESCRIBED IN THE MONITOR;
366
366
```

```
366 BEGIN COMMENT NAME IN CATALOG: TESTDUMMY;
367       WRITE(OUTC,NL,1,<:***TEST  :>);
3         PRINTNAME;
3         WRITE(OUTC,<:  NOT IMPLEMENTED:>)
370 END;
371
371
371 COMMENT
372
```

```
1949     END CASE TESTPROGRAMS;
1950 TESTEND:
1951     WRITE(OUTC,NL,5,<:<12>:>); UD;
1952     WRITE(OUTL,FALSE ADD 12,1,FALSE ADD 25,1);
1953     SETPOSITION(OUTL,0,0);
1954     CLOSE(OUTL,TRUE);
1955 L_END:
1956     CLOSE(OUTC,TRUE);
1957 END

ALGOL END
```

RCSL: 44-D11
Author: Per Hansen
Edited: November, 1970
Type: Algol 5 program

RC 4000
TIMESHARED TESTPROGRAM LIBRARY
TYPEWRITER and TELETERMINAL

Keywords: RC 4000, Diagnostic program, Algol Program, ISO tape

ABSTRACT: This program contains 15 routines for error location and maintenance purposes on the RC 315 typewriter and the RC 328 teleterminal. It is organized as a block incorporated in the 'TEST' system.

A/S REGNECENTRALEN
Falkoner Alle 1
2000 Copenhagen F

CONTENTS:

TYPEWRITER and TELETERMINAL:

1. Call

The testroutines for the typewriter are called in this way:

<out> = test tw.<device 1>.<device 2>....

The devices thus specified will be used for simultaneous test output whereas run administration and testinput is made from the operators console. The parameter <out> denotes the device for error messages, e.g. the lineprinter. If <out> is missing the error messages will be output on the operators console.

If this console is under test the error message may be delayed until the current share is emptied.

The only error messages will be the statusword and the number of bytes transferred from each share. Only when the transfer of the share which holds 256 bytes = 384 characters has been transferred the statusword is checked.

The transfer may be terminated earlier, however, in the following cases:

1. The share was not full.
2. The output is stopped due to parity error, time-out or disconnected.
3. The operator key has been pressed during output.

2. Echotest

This program tests input from the typewriter.

First the program asks:

number of echos =

and the operator is expected to type a number followed by a <NL>.

After this any character read from the typewriter is repeated a number of times as specified by number of echos.

Each run reads one line from the typewriter.

3. Sequence

When the program asks:

Type sequence:

The operator is expected to type a sequence of characters and numbers as explained in the description of TEST, part 4. The maximum number of characters is 190.

For every run the sequence is output on the devices under test.

4. Write keyboard

This program outputs the characters of the RC 315 keyboard. When all the 'visible' characters have been printed there is performed the control HT (tabulation) once and the BS (backspace) is performed 21 times.

5. Test backspace

Writes a pattern of ones which is overwritten with zeroes by means of the backspace function.

After completion of this 50 backspaces against the left-hand stop are performed to test the function of the cam-pawl.

6. Hyphen and underline

Writes one line consisting of 75 hyphens and one line of 75 underlines to check the linearity of the typing.

7. Test of tabulator

This test outputs 5 lines, each line containing 7 tabulations. Each tabulation is followed by the letter L a number of times:

1.	line:	1	time
2.	line:	2	times
3.	line:	4	times
4.	line:	7	times
5.	line:	15	times

To obtain an appropriate printing six of the tabulator stops should be set at various places.

8. Testhead 1 (RC 315)

This testprogram performs a worstcase movement of the printball on the RC 315 typewriter.

9. Rotate 1, 2, 3, 4 (RC 315)

These programs make a systematic test of the rotate magnets of the RC 315. In the beginning of each test the number of the magnet is announced in the test output.

10. Test tilt (RC 315)

Performs a worstcase test of the tilt magnets by switching between the upper and lower row of characters.

11. Test space

This test produces two lines which form a pattern of ones and spaces to check that no spaces are lost or doubled.

12. Print page

This program outputs an entire page consisting of 75 lines. Each line contains a text of 78 characters.

13. Olivetti head move

This test performs a worstcase side-to-side movement of the head on the RC 328 teleterminal. In each run a line of 74 characters is output.

14. Routine test

This test is intended for use as an overall check of the typewriter. In each run the following tests are performed once:

Test backspace.
Hyphen and underline.
Test tabulator.
Test head 1.
Rotate 1, 2, 3, 4.
Test tilt.
Test space.
Olivetti head move.

15. Examples

☐ This signature denotes input.

```
to s
max
max 8380 30 32 7 6
ready

to s
new peh size 8380 run
ready

to peh
test

bs    4  15
tw    2  10
pt    0   1
pl
lp    5
cr
mt    6   7   8   9  14



end
test tw.c

RC315 typewriter

testprogram:

   a   echo
   b   sequence
   c   write keyboard
   d   test backspace
   e   test hyphen and underline
   f   test tabulator
   g   test of head 1
   h   rotate 1
   i   rotate 2
   j   rotate 3
   k   rotate 4
   l   test tilt
   m   test space
   n   print page
   o   olivetti head move
   x   routinetest
   y   terminate


testprogram: a
number of runs = 3
```

```
number of echos = 7

run no.        1
12345
11111112222222333333344444445555555

run no.        2







run no.        3
!!!???
!!!!!!!!!!!!!!!!!!!!!?????????????????????


test end

testprogram: b
number of runs = 11

type sequence:
dette er en prøve ;125;124;123;32,3,pip,1

run no.        1
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip

run no.       11
dette er en prøve åøæ dette er en prøve åøæ dette er en prøve åøæ pip


test end

testprogram: b
number of runs = 1

type sequence:
a,50,
,1,b,50,
,1,c,50,;9,2000

***full
run no.        1
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
cccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
test end

testprogram: [c]
number of runs =[1]

run no.       1

     1 2 3 4 5 6 7 8 9 0 ¬ ;
     q w e r t y u i o p å
     a s d f g h j k l æ ø
     z x c v b n m , . /

     ! " : * % & ' ( ) _ = +
     Q W E R T Y U I O P̅ Å
     A S D F G H J K L Æ Ø
     Z X C V B N M < > ?


test end

testprogram: [d]
number of runs =[1]

run no.       1

lololololololololololololololololololoððððððððððððððððððððððððððððððððððððð

test end

testprogram: [e]
number of runs =[1]

run no.       1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

________________________________________________________________________________

test end

testprogram: [f]
number of runs =[1]

run no.       1

1 1 1 1 1 1 1
11 11 11 11 11 11 11
1111 1111 1111 1111 1111 1111 1111
1111111 1111111 1111111 1111111 1111111 1111111          1111111
111111111111111 111111111111111 111111111111111          111111111111111          11

test end

testprogram: [g]
number of runs =[1]

run no.       1

HAHBHCHDHEHFHGHIHJHKHLHMHNHOHPHQHRHSHTHUHVHWHXHYHZHÆHØH*H/H=H;HÆHÅH(H)H&H!H

test end
```

```
testprogram: [h]
number of runs =[1]

run no.       1

 W=rotate +5, M=rotate ¬5:
WMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWMWM
 R=rotate ¬3:
WRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWRWR
 V=rotate ¬4:
WVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWV

test end

testprogram: [i]
number of runs =[1]

run no.       1

 rotate 0,+1,+2,+3,+4,+5:
=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW
=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW=>ÆISW
 rotate 0,¬1,¬2,¬3,¬4,¬5:
=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM
=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM=OARVM

test end

testprogram: [j]
number of runs =[1]

run no.       1

 w=rotate +5, m=rotate ¬5:
vmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvmvm
 r=rotate ¬3:
wrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwrwr
 v=rotate ¬4:
wvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwvwv

test end

testprogram: [k]
number of runs =[1]

run no.       1

 rotate 0,+1,+2,+3,+4,+5:
¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw
¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw¬.æisw
 rotate 0,¬1,¬2,¬3,¬4,¬5:
¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm
¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm¬oarvm

test end

testprogram: [l]
```

number of runs = [1]

run no. 1

tilt 0, tilt 3:
zjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzjzj
2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø2ø
5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p5p

test end

testprogram: [m]
number of runs = [1]

run no. 1

11111 11111 11111 11111 11111 11111 11111 11111
11111 11111 11111 11111 11111 11111 11111

test end

testprogram: [n]
number of runs = [1]

run no. 1
Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine
ette er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine D
tte er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine De
te er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Det
e er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dett
er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette
er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette
r en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette e
en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette er
en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette er
n prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette er e
prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette er en
prøvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette er en
røvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette er en p
øvetext som udskrives til kontrol af en rc4000 skrivemaskine Dette er en pr
vetext som udskrives til kontrol af en rc4000 skrivemaskine Dette er en prø
etext som udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøv
text som udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøve
ext som udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvet
xt som udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvete
t som udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetex
som udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext
som udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext
om udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext s
m udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext so
udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som
udskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som
dskrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som u
skrives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som ud
krives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som uds
rives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udsk
ives til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskr
ves til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskri
es til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskriv
s til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrive
til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives
til kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives
il kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives t
l kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives ti
kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til
kontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til
ontrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til k
ntrol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til ko
trol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kon
rol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kont
ol af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontr
l af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontro
af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol
af en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol
f en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol a
en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol af
en rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol af
n rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol af e
rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol af en
rc4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol af en
c4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol af en r

```
4000 skrivemaskine Dette er en prøvetext som udskrives til kontrol af en rc
000 skrivemaskine Dette er en prøvetext som udskrives til kontrol af en rc4
00 skrivemaskine Dette er en prøvetext som udskrives til kontrol af en rc40
0 skrivemaskine Dette er en prøvetext som udskrives til kontrol af en rc400
 skrivemaskine Dette er en prøvetext som udskrives til kontrol af en rc4000
skrivemaskine Dette er en prøvetext som udskrives til kontrol af en rc4000
krivemaskine Dette er en prøvetext som udskrives til kontrol af en rc4000 s
rivemaskine Dette er en prøvetext som udskrives til kontrol af en rc4000 sk
ivemaskine Dette er en prøvetext som udskrives til kontrol af en rc4000 skr
vemaskine Dette er en prøvetext som udskrives til kontrol af en rc4000 skri
emaskine Dette er en prøvetext som udskrives til kontrol af en rc4000 skriv
maskine Dette er en prøvetext som udskrives til kontrol af en rc4000 skrive
askine Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivem
skine Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivema
kine Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivemas
ine Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivemask
ne Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivemaski
e Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskin
 Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine
Dette er en prøvetext som udskrives til kontrol af en rc4000 skrivemaskine


test end

testprogram: [o]
number of runs =[1]



run no.       1

£_/r£_/r£r/_£r/_£_£_/r/r£_£_£r/_/r£r_£_/r£_/r£r/_£r/_£_£_/r/r£_£_£r/_/r£r_

test end

testprogram: [y]



end
```

16. Program text

```
322 RC 4000 TYPEWRITER TEST
323 RCTS   2.0/3  15.11.69 PEH
324 VERSION          1.1.71;
325
326 BEGIN COMMENT NAME IN CATALOG: TESTTW;
327    ZONE ARRAY TW(DEVS,128,2,ERROR);
327
327    PROCEDURE TW_END;
328    BEGIN FOR D:=DEVS-1 STEP -1 UNTIL 1 DO
329       WRITE(TW(D),FALSE ADD 1,368);
330       FOR D:=DEVS STEP -1 UNTIL 1 DO
331       SETPOSITION(TW(D),0,0);
332    END TW_END;
333
333    PROCEDURE TESTBACKSPACE;
334    BEGIN INTEGER BS;
335       WRITE(TW(D),<:<10>L L L L L L L L L L L L L L L L L L L :>,
336       FALSE ADD 108,37);
337       FOR BS:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:<8>0<8>:>);
338       FOR BS:=1 STEP 1 UNTIL 19 DO WRITE(TW(D),<:<8>0<8><8>:>);
339       WRITE(TW(D),FALSE ADD 8,35);
340    END PROCEDURE TESTBACKSPACE;
341
342    PROCEDURE TESTLINE;
343    BEGIN INTEGER TL;
343       WRITE(TW(D),<:<10>:>,FALSE ADD 45,75);
344       WRITE(TW(D),<:<10>:>,FALSE ADD 95,75);
345    END PROCEDURE TESTLINE;
346
346    PROCEDURE TESTTAB;
347    BEGIN INTEGER TT,TTT;
348       FOR TT:=1,2,4,7,15 DO
349       BEGIN WRITE(TW(D),<:<10>:>);
350          FOR TTT:=1 STEP 1 UNTIL 7 DO
351          WRITE(TW(D),FALSE ADD 108,TT,<:<9>:>);
352       END;
353    END  PROCEDURE TESTTAB;
354
354     PROCEDURE TESTHEAD1;
355    BEGIN INTEGER TH;
356       WRITE(TW(D),<:<10>HAHBHCHDHEHFHGHIHJHKHLHMHNHOHPHQHRHSHTHUH:>,
357            <:VHWHXHYHZHÆHØH*H/H=H;HÆHÅH(H)H&H!H:>);
358    END PROCEDURE TESTHEAD1;
359
359    PROCEDURE ROTATE1;
360    BEGIN INTEGER R1;
361       WRITE(TW(D),<:
362  W=ROTATE +5, M=ROTATE -5:
363 :>);     FOR R1:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:WM:>);
364       WRITE(TW(D),<:
365  R=ROTATE -3:
366 :>);     FOR R1:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:WR:>);
367       WRITE(TW(D),<:
368  V=ROTATE -4:
369 :>);     FOR R1:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:WV:>);
370    END PROCEDURE ROTATE1;
371
371
371    PROCEDURE ROTATE2;
372    BEGIN INTEGER R2;
373       WRITE(TW(D),<:
374  ROTATE 0,+1,+2,+3,+4,+5:
375 :>);
376
376 COMMENT
377
```

```
377
378
378         FOR R2:=1 STEP 1 UNTIL 25 DO WRITE(TW(D), IF R2=13
379         THEN <:<10>:> ELSE <:=>AISW:>);
3           WRITE(TW(D),<:
 ROTATE 0,-1,-2,-3,-4,-5:
382 :>);
383         FOR R2:=1 STEP 1 UNTIL 25 DO WRITE(TW(D), IF R2=13
384         THEN <:<10>:> ELSE <:=OARVM:>);
385      END PROCEDURE ROTATE2;
386
386
386      PROCEDURE ROTATE3;
387      BEGIN INTEGER R3;
388         WRITE(TW(D),<:
389  W=ROTATE +5, M=ROTATE -5:
390 :>);      FOR R3:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:VM:>);
391         WRITE(TW(D),<:
392  R=ROTATE -3:
393 :>);      FOR R3:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:WR:>);
394         WRITE(TW(D),<:
395  V=ROTATE -4:
396 :>);      FOR R3:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:WV:>);
3      END PROCEDURE ROTATE3;
398
398
398      PROCEDURE ROTATE4;
399      BEGIN INTEGER R4;
400         WRITE(TW(D),<:
401  ROTATE 0,+1,+2,+3,+4,+5:
402 :>);
403         FOR R4:=1 STEP 1 UNTIL 25 DO WRITE(TW(D), IF R4=13
404         THEN <:<10>:> ELSE <:-,ÆISW:>);
405         WRITE(TW(D),<:
406  ROTATE 0,-1,-2,-3,-4,-5:
407 :>);
408         FOR R4:=1 STEP 1 UNTIL 25 DO WRITE(TW(D), IF R4=13
409         THEN <:<10>:> ELSE <:-OARVM:>);
410      END PROCEDURE ROTATE4;
411
411
411      PROCEDURE TILT;
4        BEGIN INTEGER TI;
413         WRITE(TW(D),<:
414 TILT 0, TILT 3::>);
415         WRITE(TW(D),NL,1);
416         FOR TI:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:ZJ:>);
417         WRITE(TW(D),NL,1);
418         FOR TI:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:2Ø:>);
419         WRITE(TW(D),NL,1);
420         FOR TI:=1 STEP 1 UNTIL 37 DO WRITE(TW(D),<:5P:>);
421      END PROCEDURE TILT;
422
422
422      PROCEDURE TESTSPACE;
423      BEGIN INTEGER TS0, TS1;
424         FOR TS0:= 76, 0 DO
425         BEGIN WRITE(TW(D),<:<10>:>);
426            FOR TS1:=0 STEP 76 UNTIL 1064 DO
427            WRITE(TW(D), FALSE ADD(32+(TS0+TS1) MOD 152),5);
4          END;
4      END PROCEDURE TESTSPACE;
430
430
430 COMMENT
431
```

```
431
432
432     PROCEDURE TESTOL;
433     BEGIN INTEGER TO;
434        WRITE(TW(D),NL,1);
435        FOR TO:=1,2 DO WRITE(TW(D),<:<35><95>/R<35><95>/R<35>R/<95><35>R/:>,
436        <:<95><35><95><35><95>/R/R<35><95><35><95><35>R/<95>/R<35>R<95>:>);
437     END PROCEDURE TESTOL;
438
438 TWTEST_START:
439
439     FOR D:=1 STEP 1 UNTIL DEVS DO
440     BEGIN SYSTEM(5,DEVICE(D)+2,PARAM);
441        I:=0;
442        OPEN(TW(D),MODEKIND(DEVICE(D)),STRING PARAM(INCREASE(I)), M);
443        RESERVED(D):=MONITOR(6,TW(D),I,IA)=0;
444        IF RESERVED(D) THEN SETPOSITION(TW(D),0,0);
445     END;
446
446     WRITE(OUTC,<:<10>RC315 TYPEWRITER<10>:>);
447 RTP:
448     GOTO CASE TESTPROG OF (DIRECTORY,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,
449     RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,X,Y);
450
450 DIRECTORY:
451     BEGIN INTEGER DIR;
452        WRITE(OUTC,<:
453     A    ECHO
454     B    SEQUENCE
455     C    WRITE KEYBOARD
456     D    TEST BACKSPACE
457     E    TEST HYPHEN AND UNDERLINE
458     F    TEST TABULATOR
459     G    TEST OF HEAD 1
460     H    ROTATE 1
461     I    ROTATE 2
462     J    ROTATE 3
463     K    ROTATE 4
464     L    TEST TILT
465     M    TEST SPACE
466     N    PRINT PAGE
467     O    OLIVETTI HEAD MOVE
468     X    ROUTINETEST
469     Y    TERMINATE
470
470 :>);
471     END DIRECTORY;
472     GOTO RTP;
473
473
473 A:     BEGIN COMMENT ECHO;
474        INTEGER LAST, PREVIOUS, ECHOS;
475        LAST:=10;
476        WRITE(OUTC,<:NUMBER OF ECHOS = :>); UD;  IND;
477        RD(A,ECHOS);
478        IF ECHOS <= 0 ! ECHOS>120 THEN GOTO A;  IND;
479 NEXTRUN:
480        IF LAST=10 THEN RUNADM(RTP,TW_END);
481        PREVIOUS:=LAST;
482        LAST:=INCHAR;
483        FOR D:=1 STEP 1 UNTIL DEVS DO
484        WRITE(TW(D),FALSE ADD LAST,
485        IF LAST<>10 OR PREVIOUS=10 THEN ECHOS ELSE 1);
486        GOTO NEXTRUN;
487     END ECHO;
488 COMMENT
489
```

```
489
490
490
490 B:    BEGIN COMMENT SEQUENCE;
491       REAL ARRAY TEXT(0:31);
4         PACK(TEXT,0,32*6,128,8);
493
493 OUTSEQUENCE:
494       RUNADM(RTP,TW_END);
495       I:=0;
496       WRITE(TW(D),STRING TEXT(INCREASE(I)));
497       GOTO OUTSEQUENCE;
498    END SEQUENCE;
499
499
499 C:    BEGIN COMMENT WRITE KEYBOARD;
500       INTEGER CC;
501       RUNADM(RTP,TW_END);
502       WRITE(TW(D),<:
503      1 2 3 4 5 6 7 8 9 0 - ;
504      Q W E R T Y U I O P Å
505      A S D F G H J K L Æ Ø
506      Z X C V B N M , . /
507
5         ! " : * % & ' ( ) <95> = +
508      Q W E R T Y U I O P Å
509      A S D F G H J K L Æ Ø
510      Z X C V B N M < > ?
511      <9>:>,FALSE ADD 8,21,<:<10>:>);
512       GOTO C;
513    END WRITE KEYBOARD;
514
514 D: RUNADM(RTP,TW_END);
515    TESTBACKSPACE;
516    GOTO D;
517
517
517 E: RUNADM(RTP,TW_END);
518    TESTLINE;
519    GOTO E;
520
520
5   F: RUNADM(RTP,TW_END);
5      TESTTAB;
522    GOTO F;
523
523
523 G: RUNADM(RTP,TW_END);
524    TESTHEAD1;
525    GOTO G;
526
526
526 H: RUNADM(RTP,TW_END);
527    ROTATE1;
528    GOTO H;
529
529
529 I: RUNADM(RTP,TW_END);
530    ROTATE2;
531    GOTO I;
532
5
5   J: RUNADM(RTP,TW_END);
533    ROTATE3;
534    GOTO J;
535 COMMENT
536
```

```
537
537
537 K: RUNADM(RTP,TW_END);
5      ROTATE4;
5      GOTO K;
540
540
540 L: RUNADM(RTP,TW_END);
541    TILT;
542    GOTO L;
543
543
543 M: RUNADM(RTP,TW_END);
544    TESTSPACE;
545    GOTO M;
546
546 N:     BEGIN COMMENT PRINTPATTERN;
547        REAL ARRAY P(0:12);
548        INTEGER I, J;
549        FOR I:=0 STEP 1 UNTIL 12 DO
550        P(I):=REAL (CASE I+1 OF (<:DETTE:>,<:ER EN:>,<:PRØVE:>,<:EXT S:>,
551        <:M UDS:>,<:RIVES:>,<:TIL K:>,<:NTROL:>,<:AF EN:>,<:RC400:>,
5          <: SKRI:>,<:EMASK:>,<:NE :>));
553        FOR I:=0 STEP 1 UNTIL 11 DO
554        P(I):=P(I) ADD (CASE I+1 OF (
555        32,32,116,111,107,32,111,32,32,48,118,105));
556 NN:    RUNADM(RTP,TW_END);
557        WRITE(OUTC,NL,1);
558        FOR I:=0 STEP 1 UNTIL 75 DO FOR D:=1 STEP 1 UNTIL DEVS DO
559        BEGIN FOR J:= 0 STEP 1 UNTIL 77 DO
560           WRITE(TW(D),FALSE ADD (P(((I+J) MOD 78)//6)
561           SHIFT (((I+J) MOD 6)*8-40)EXTRACT 8),1);
562           WRITE(TW(D),NL,1);
563        END;
564        GOTO NN;
565    END PRINTPATTERN;
566
566 O: RUNADM(RTP,TW_END);
567    TESTOL;
568    GOTO O;
569
5      X: RUNADM(RTP,TW_END);
570    FOR I:=0 STEP 1 UNTIL DEVS*12-1 DO
571    BEGIN D:=I MOD DEVS +1;
572       CASE I//DEVS+1 OF
573       BEGIN TESTBACKSPACE;
574          TESTLINE;
575          TESTTAB;
576          TESTHEAD1;
577          ROTATE1;
578          ROTATE2;
579          ROTATE3;
580          ROTATE4;
581          TILT;
582          TESTSPACE;
583          TESTOL;
584          WRITE(TW(D),NL,1);
585       END;
586    END;
5      GOTO X;
5
588
588 Y:
589 END  TWTEST;
590
590
```

RCSL: 44-D12

Author: Per Hansen

Edited: December, 1970

Type: ALGOL program

RC 4000

TIMESHARED TESTPROGRAM LIBRARY

LINEPRINTER

N.B!!! Testprogrammerne "a" og "x" må ikke køres, da de sprænger 15A sikringen.

ok ved spacing af til 1800-2000 linier.

Keywords: RC 4000, Diagnostic program, Algol program, ISO tape

---

ABSTRACT: This program contains 6 routines for error location and maintenance purposes on the RC 610 Lineprinter. It is organized as a block incorporated in the 'TEST' system.

A/S REGNECENTRALEN
Falkoner Alle 1
2000 Copenhagen F

NB: Bredt papir skal monteres

LHØ.

CONTENTS:

LINEPRINTER:

1. Call

The test routines for the lineprinter are called in this way:

<out> = test lp. <device 1> . <device 2>.....

The devices thus specified will be used for simultaneous testoutput whereas the run administration is made from the operators console. The parameter <out> denotes the device for errormessages, e.g. a lineprinter. It should not be the same as the one to be tested. If <out> is missing the error messages will appear on the operators console.

The only error messages will be the logical status word and the number of bytes transferred from each share. If the share was full, the number of bytes will be 128 bytes = 192 characters.

First the program asks for

drumsize = 96

and the operator is expected to type a number defining how many characters which are present on the printdrum. The standard drum is defined by the drumsize first characters of the following table. The zeroes represent some of the characters which are not accessible in the shift-in mode:

ISO-value	character	ISO-value	character
46	.	80	P
44	,	79	O
58	:	78	N
59	,	77	M
48	0	76	L
49	1	75	K
50	2	74	J
51	3	73	I

52	4	72	H
53	5	71	G
54	6	70	F
55	7	69	E
56	8	68	D
57	9	67	C
43	+	66	B
45	-	65	A
40	(	00	ü
41	)	00	ö
00	¤	00	ä
00	10	122	z
61	=	121	y
60	<	120	x
62	>	125	å
47	/	124	ø
45	-	123	æ
37	%	119	w
38	&	118	v
42	*	117	u
33	!	116	t
39	´	115	s
00	¨	114	r
63	?	113	q
00	Ü	112	p
00	Ö	111	o
00	Ä	110	n
90	Z	109	m
89	Y	108	l
88	X	107	k
93	Å	106	j
92	Ø	105	i
91	Æ	104	h
87	W	103	g

86	V	102	f
85	U	101	e
84	T	100	d
83	S	99	c
82	R	98	b
81	Q	97	a

2. Highspeed test

This testprogram outputs a sequence of characters chosen in a way which keeps the printing speed at a fixed value. In each line all 132 positions are activated with the same character. The printing speed can be defined when the program asks:

speed lines/min =

Any speed may be selected, but the relevant values are inside the range of app. 750-3000 lines/min. The exact values are depending on the adjustment of the paper-feed delay etc. It should be noted that the DP-printer should not be operated with a higher speed than 2000 lines/min for a longer time.

3. Print characterset

This program tests the shift-in characterset as well as the shift-out characterset. The control characters HT, CAN and ESC are tested as described in 'test of HT, CAN, ESC.'

To obtain the output shown in part 9 a format tape with holes in channel 0 for every 72 lines and in channel one for every 2 lines must be mounted.

The program first outputs the characters SO, FF and SI. In this way FF is tested in the shift-out mode, and the printer is ready for output of the shift-in character set. (If any characters were stored in the line buffer, they are printed before the form feed).

The program now prints:

SHIFT-IN CHAR. SET

and a vertical tabulation is performed so that VT is tested in the shift-in mode.

After this the program outputs the characters:

48 32 48 32 CR 32 49 32 49 NL

so that a line containing

0101

is printed, which is a test of CR and SP in the shift-in mode. Furthermore it is now possible to measure the length of the skip formed by the vertical tabulation mentioned above.

Then 119 lines are printed, that is one line for each character except the 9 control characters so that the line is printed for each of the characters 0-8, 16-23, 25, 26, and 28-127. In each line the character is output 132 times followed by a number of the character (3 digits) followed by NL. So 60 lines each contains 132 equal graphic characters (including the line containing spaces), while the rest of the lines only contain the number of the (blind) character. In this way 1) NL and 2) the printing of only the first 132 characters is tested in the shift-in mode. Furthermore it is tested that 3) blind characters are treated correctly, i.e. the line printer's buffer pointer is not moved by a blind character, and a blind character does not destroy a non-blind character following it.

At last the program writes:

SI END

After having output the characters SI and FF (and in this way tested FF in the shift-in mode) the program prints:

SHIFT-OUT CHAR. SET

and it outputs the characters SO and VT so that the vertical tabulation is tested in the shift-out mode.

The printer is now ready for printing the shift-out character set, and the program outputs the characters:

48 32 48 32 CR 32 49 32 49 NL

so that a line containing

0101

is printed, which is a test of CR and SP in the shift-out mode. Furthermore it is now possible to measure the length of the skip mentioned above.

Then 119 lines are printed as mentioned for the shift-in mode, and at last the program prints:

SO END

Finally it is tested that in the same line it is possible to change between the shift-in and the shift-out character set.
The program outputs the characters:

SI 65 SI SI 65 SI SO 37 SO SO 37 SO SI 66 SI
SI 66 NL

causing the printing of

AAOOBB

after the last run FF is output.

4. Test of HT, CAN, ESC

This program tests the control characters HT, CAN, and ESC in the shift-in mode as well as in the shift-out mode.

HT

The program first outputs the characters:

FF SI H T NL

causing a formfeed and a printing of

HT

then the program outputs:

HT 49...49 (15 times) HT 50...50 (16 times) HT 52...52 (17 times) HT 54 HT

55 HT 56 HT 57 NL

Now the program outputs SO followed by the same sequence of characters as mentioned above, and the line is printed again. In this way HT is tested after 0, 1, 15, 16, and 17 characters, and HT is tested to the right of position 128.

CAN

The program first outputs the characters:

SI NL C A N NL

causing the printing of:

CAN

then the program outputs:

			CAN 67 NL
	48...48	(132 times)	CAN 65 NL
	48...48	(133 times)	CAN 78 NL
SO			CAN 67 NL
SO	48...48	(132 times)	CAN 65 NL
SO	48...48	(133 times)	CAN 78 NL

causing the printing of

C
A
N

twice. In this way CAN is tested after 0, 132, and 133 characters and it is checked that CAN involves a change to the shift-in mode.

ESC

The program first outputs the characters:

SI NL S I - E S C NL

causing the printing of:

SI - ESC

Characters are now output according to the following algorithm:

for m:= 10, 11 do for n:= 48 step 1
until 63 do write (false add 24, 1, n, false add 32, 1, m, false add 27, 1, false add n, 1, false add m, 1);

The program then prints:

SO - ESC

and after SO the above mentioned algorithm is executed again so that ESC is tested in shift-out mode too.

The last run is terminated by a formfeed.

5. Cyclical printing

This testprogram performs a printing of 132 lines.

The first line contains the characters of the drum printed cyclically in the order that they appear in front of the hammers, as defined by the table in part 1:

, : ; 0 1 2 3 4 ... etc.

In the next line all the characters are shifted one position:

: ; 0 1 2 3 4 5... etc.

All blind characters and characters on the drums not accessible in the shift-in mode are skipped.

As the sequence of characters causes printing for every 1/64 revolution of the drum the hammers are activated with a frequency of app. 1 K c/s.

After completion of each run a SI is output and after the last run a formfeed is output too.

6. Sequence

When the program asks:

Type sequence:

The operator is expected to type a sequence of characters and numbers as explained in the description of TEST, part 4. The maximum number of characters is 400.

For every run the sequence is output on the devices under test. Note that all control characters may be inserted in the sequence by means of the numerical delimiter

;

After completion of each run a shift-in character is output and after the last run a formfeed is output too.

7. Test VT

The purpose of this program is to make a worst-case test of the vertical tabulation. First the program asks:

Mount a format tape with holes for every second line in channel one

and waits for a NL to be typed.

Next, for each run, the following sequence is output 4 times:

xxxxx ... xxxxx VT
44 characters

The actual value of the 44 characters in the lines depends on the size of the drum. They are chosen so that the printing speed is app. 1333 lines/min.

Each run is terminated by a SI and the last run is terminated by a SI followed by a form-feed

8. Routine test

To perform an overall testing of the printer a routine test (test x) is composed in the following way:

1. The highspeed test is performed three times, at 1000, 2000 and 3000 lines/min.
2. Next the cyclical test is performed.
3. Finally the VT-test is performed 5 times thus outputting 20 lines.

9. Termination

When in the situation 'testprogram:' an y is typed the TEST terminates and returns to the fileprocessor. At the same time the reservation of the printer(s) is cancelled.

10. Examples

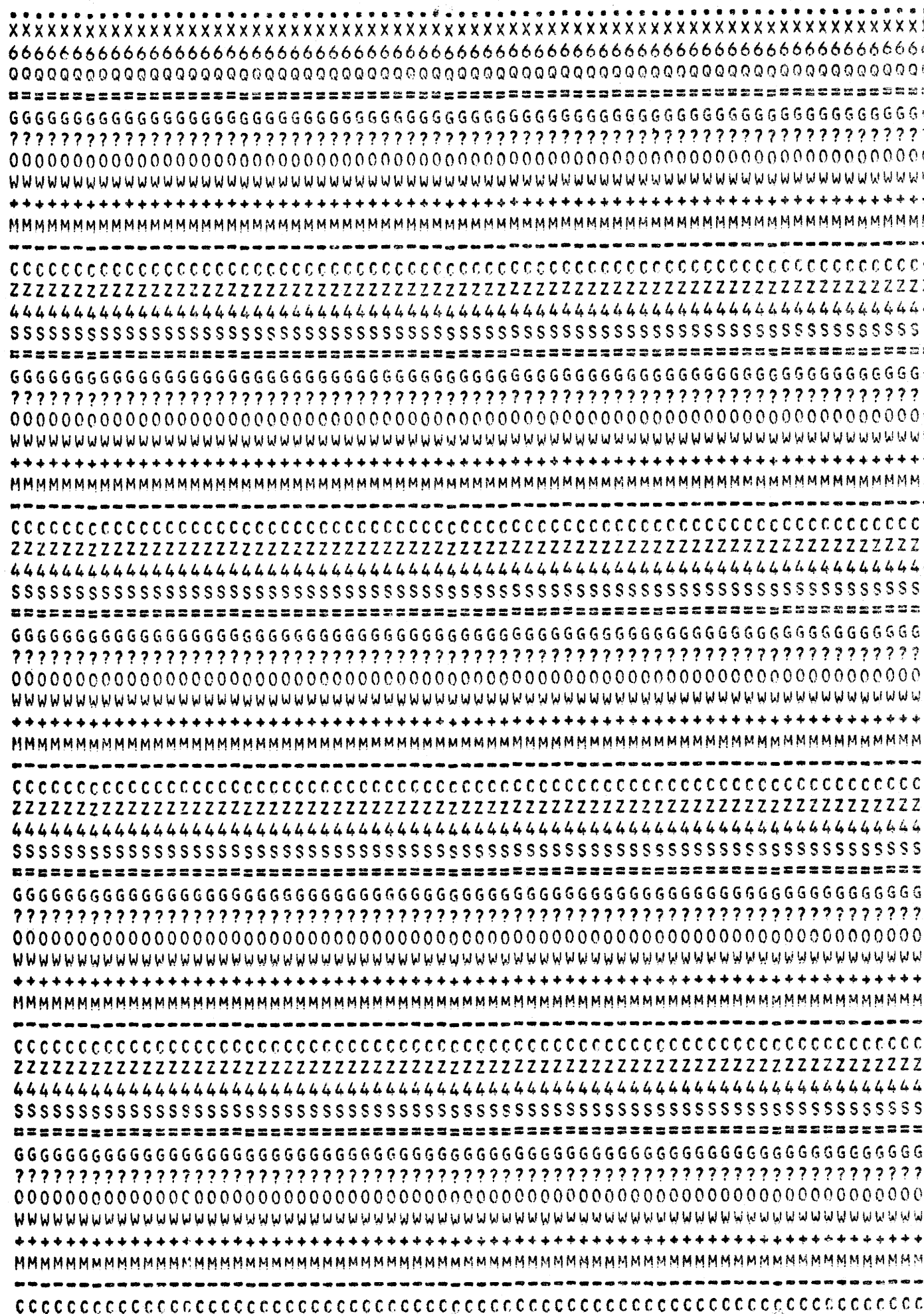
In the following is shown examples of the output from the various tests. The printing was performed on a printer equipped with the 64-character standard drum and the options shift-out, horizontal tabulation, cancel, and escape implemented.

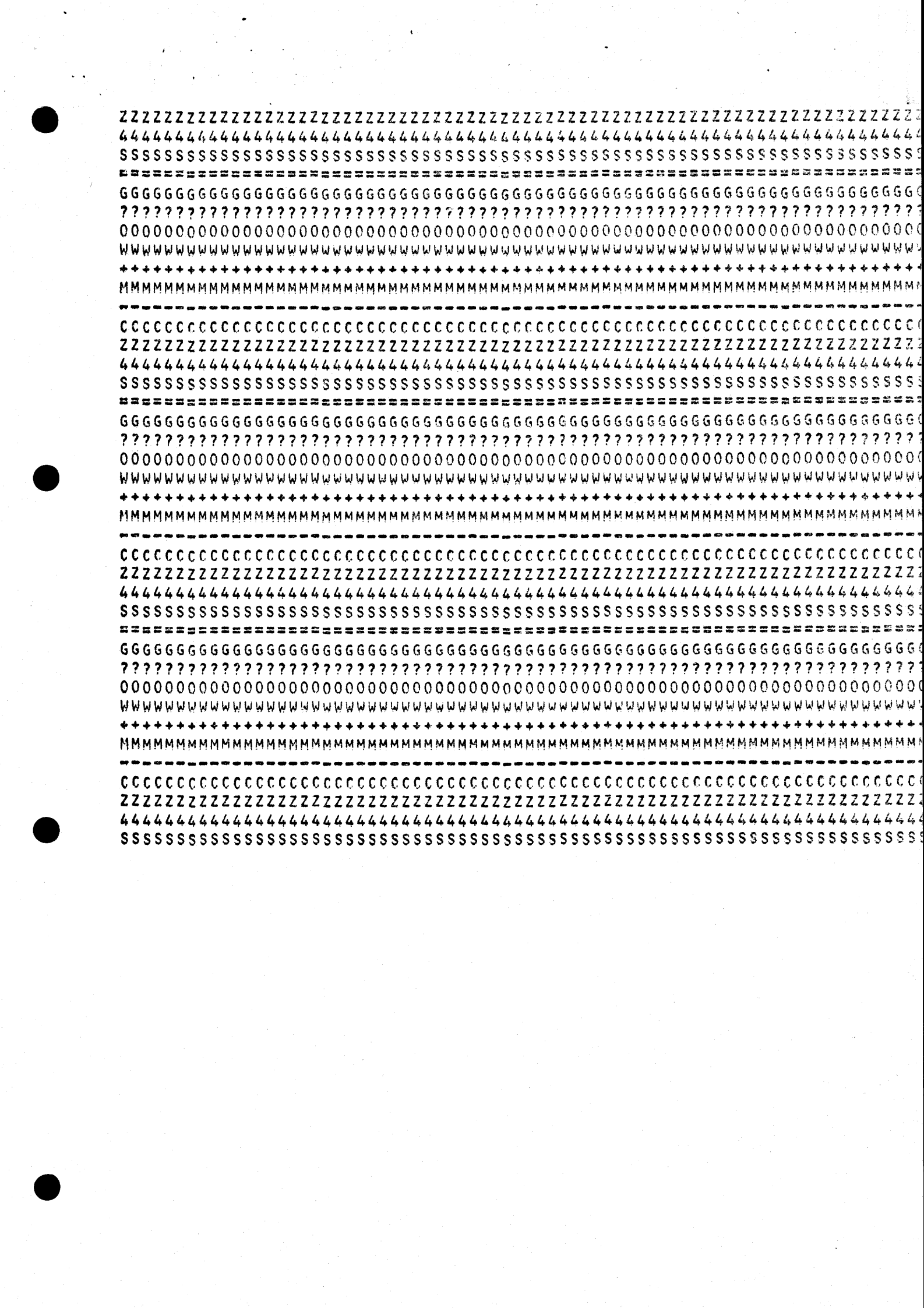
Furthermore a format tape with the following code was mounted:

channel 0 a hole for every 72 lines
channel 1 a hole for every 2 lines
channel 2 a hole for every 3 lines
channel 3 a hole for every 4 lines
channel 4 a hole for every 6 lines
channel 5 a hole for every 8 lines
channel 6 a hole for every 12 lines
channel 7 a hole for every 18 lines

Example 1 :

Highspeed test performed at 1700 lines/min.

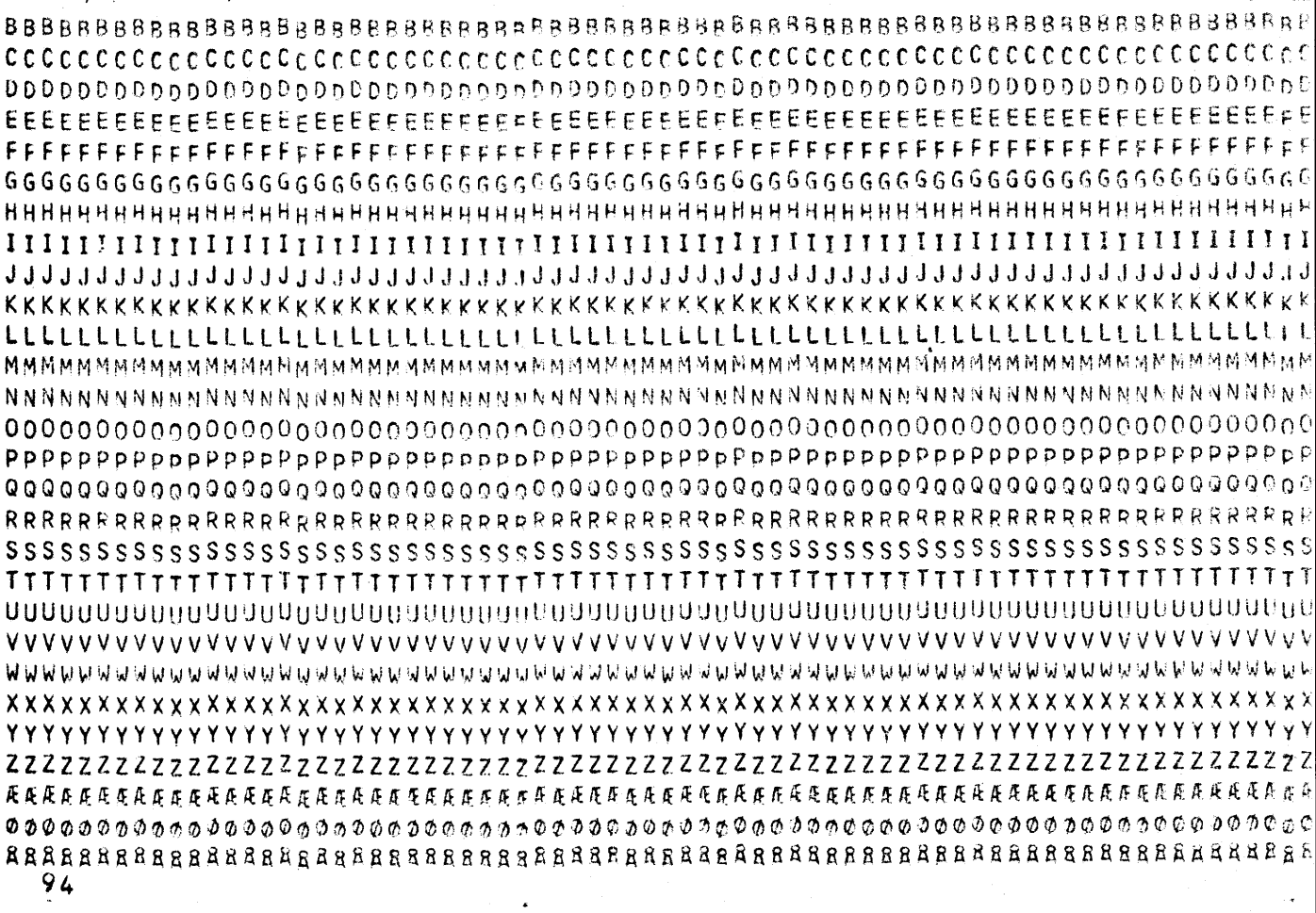




Example 2 :

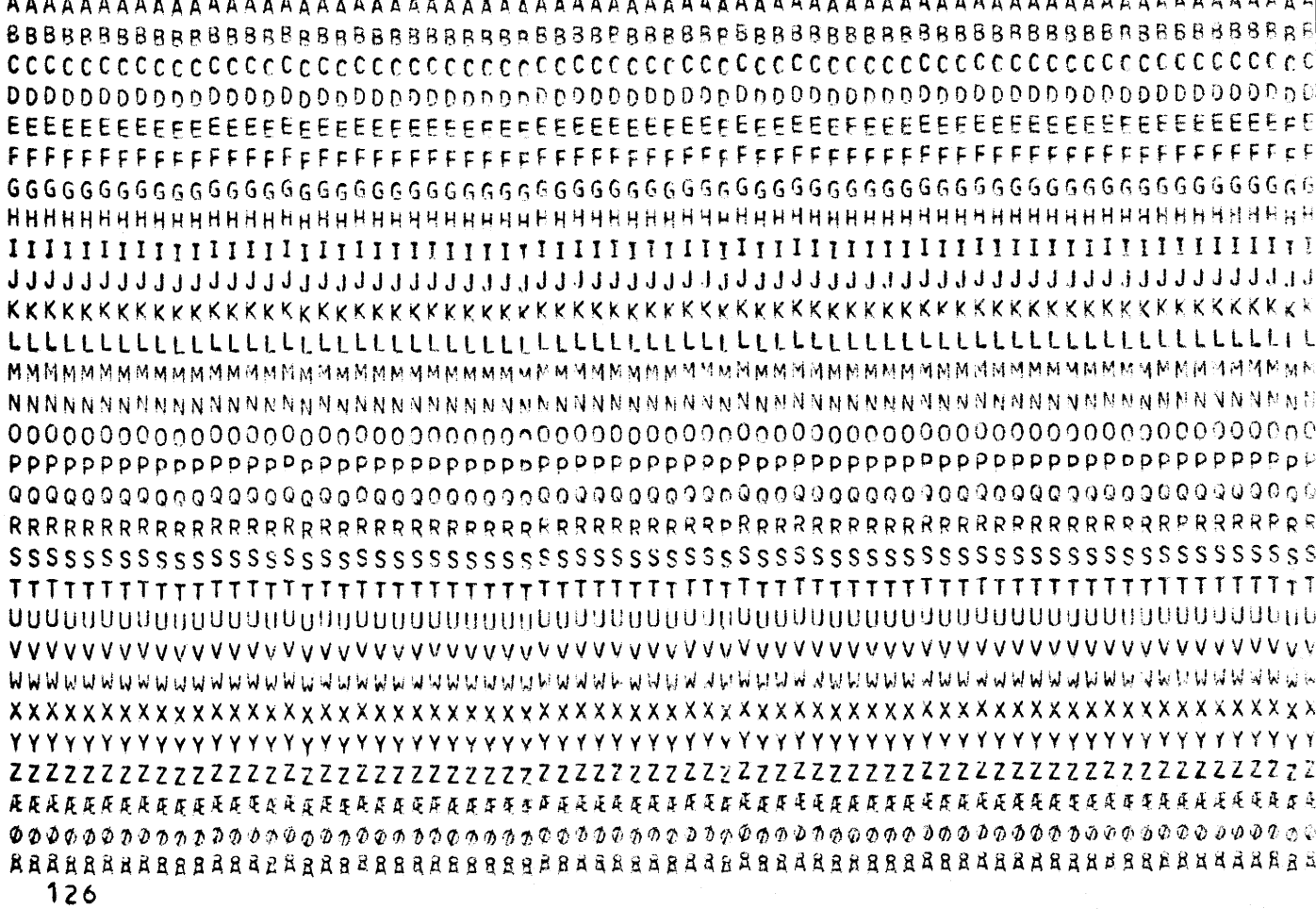
Print characterset.

```
SHIFT IN CHAR. SET

0101
   0
   1
   2
   3
   4
   5
   6
   7
   8
   16
   17
   18
   19
   20
   21
   22
   23
   25
   26
   28
   29
   30
   31

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
   35
   36
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
************************************************************************
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
------------------------------------------------------------------------
........................................................................
////////////////////////////////////////////////////////////////////////
000000000000000000000000000000000000000000000000000000000000000000000000
111111111111111111111111111111111111111111111111111111111111111111111111
222222222222222222222222222222222222222222222222222222222222222222222222
333333333333333333333333333333333333333333333333333333333333333333333333
444444444444444444444444444444444444444444444444444444444444444444444444
555555555555555555555555555555555555555555555555555555555555555555555555
666666666666666666666666666666666666666666666666666666666666666666666666
777777777777777777777777777777777777777777777777777777777777777777777777
888888888888888888888888888888888888888888888888888888888888888888888888
999999999999999999999999999999999999999999999999999999999999999999999999
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
========================================================================
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
????????????????????????????????????????????????????????????????????????
   64
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```



94

96



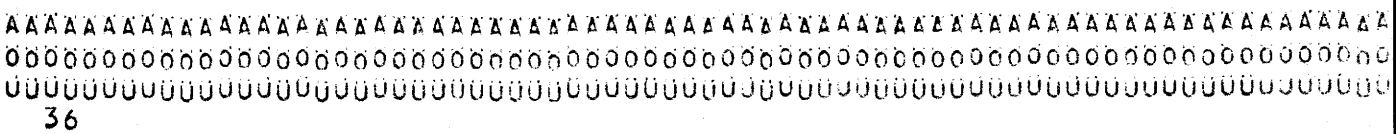
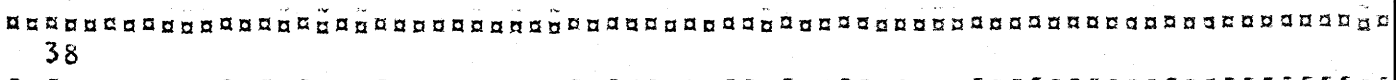
126

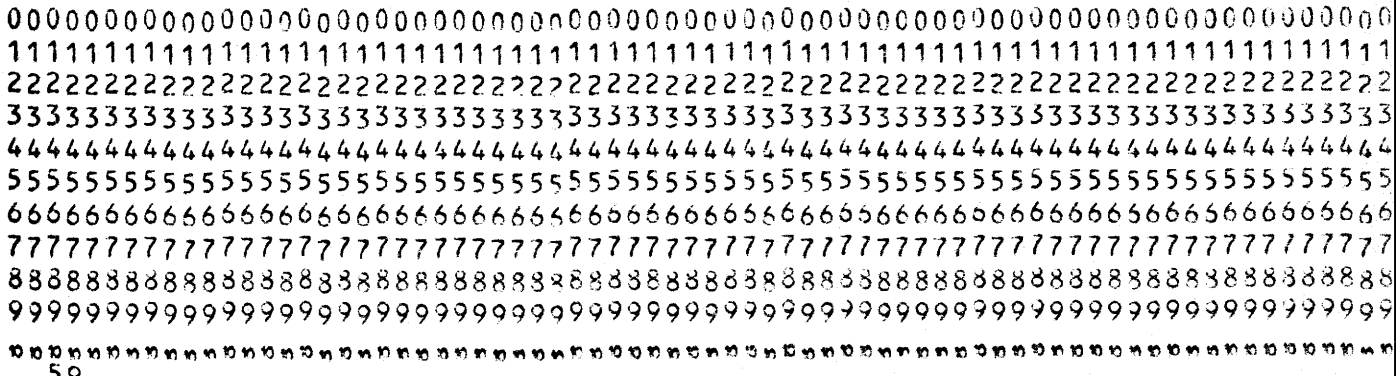
127

SI END

SHIFT OUT CHAR. SET

0101
0
1
2
3
4
5
6
7
8
16
17
18
19
20
21
22
23
25
26
28
29
30
31


36

38

42
43
44
45
46
47


59
60
61
62
63
64
65

66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127

AA¤¤BB

Example 3 :

Test of HT, CAN, ESC.

HT

1111111111111111 2222222222222222 44444444

111111111111111 2222222222222222 444444444

CAN
C
A
N
C
A
N

SI-ESC
048 010
050 010
051 010
052 010
053 010
054 010
055 010
056 010
057 010
058 010

059 010

060 010

061 010

062 010

063 010

048 011

049 011

050 011
051 011
052 011

053 011

054 011

055 011

056 011

057 011

058 011
059 011
060 011

061 011

062 011

063 011

048 010
050 010

051 010

052 010

053 010

054 010

055 010

056 010

057 010

058 010

059 010

060 010

061 010

062 010

063 010

048 011

049 011

050 011
051 011
052 011

053 011

054 011

055 011

056 011

057 011

058 011
059 011
060 011

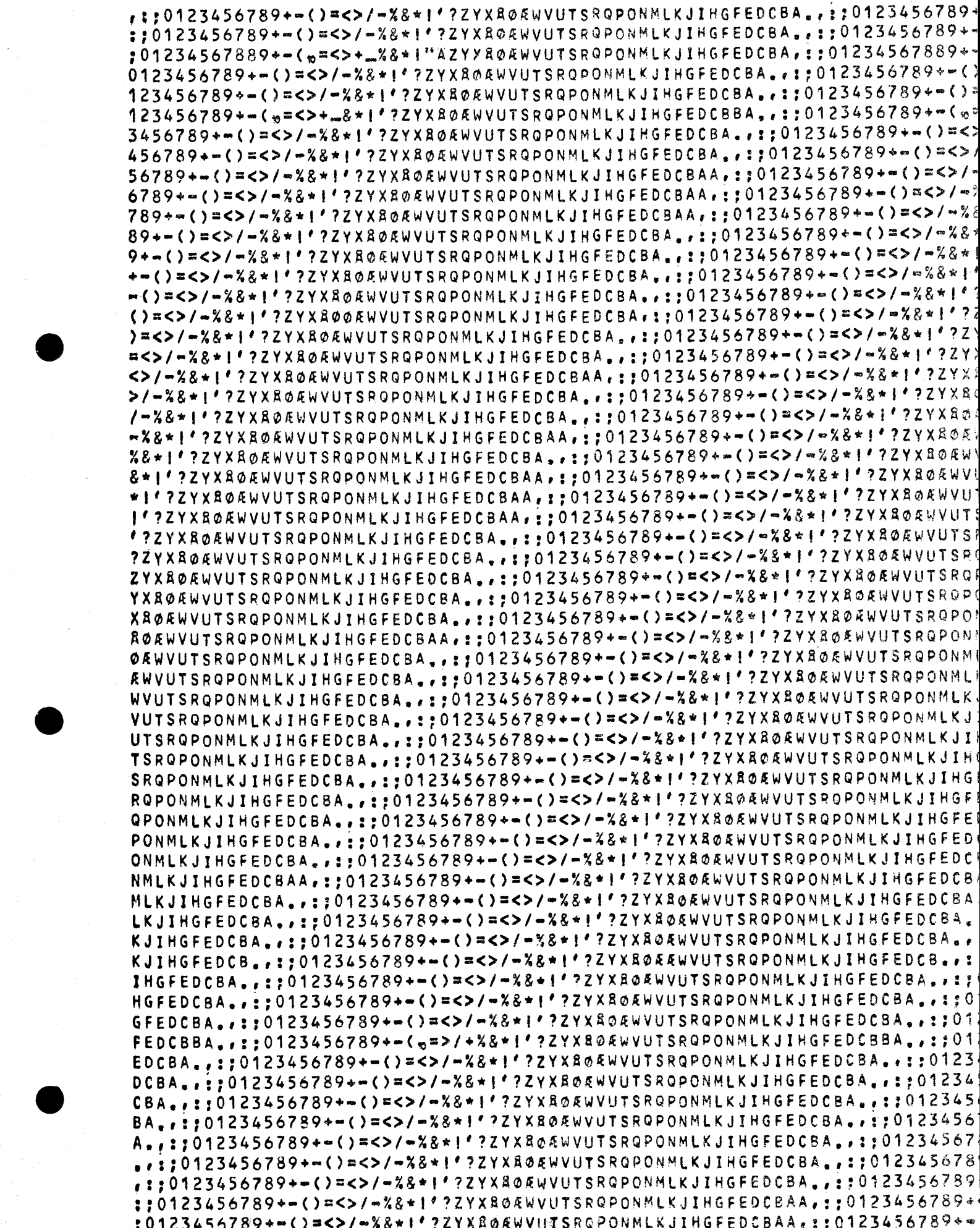
061 011

062 011

063 011

Example 4 :

Cyclical printing.



Example 5 :

Call of program and performance of sequence test.

[ ] This signature denotes input.

```
to [s]
[new peh size 20000 run]
ready

to peh
[test lp.5]

RC 610 lineprinter

drumsize= [64]

testprogram: [e]
number of runs = [10]

type sequence:
[a,20, ,20,c,20,;10,1,d,20, ,20,f,20,;10,2,g,20, ,20,i,20,;10,3]

run no.       1


test end

testprogram: [y]




end
```

The sequence of characters generated is shown on the next page.
Note that any control character may inserted in the sequence.

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

AAAAAAAAAAAAAAAAAAAA CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD FFFFFFFFFFFFFFFFFFFF

GGGGGGGGGGGGGGGGGGGG IIIIIIIIIIIIIIIIIIII

Example 6 :

Test of vertical tabulation.

****

7777777777777777777777777777777777777777

EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

11. Program text

```
RC 4000 PRINTERTEST
RCTS 2.3/1   28.8.70    PEH
VERSION       1.1.71
NAME IN CATALOG: TESTLP;

BEGIN ZONE ARRAY LP(DEVS,64,2,ERROR);
   INTEGER DRUMSIZE,SP;

   INTEGER PROCEDURE CHAR(C);
   VALUE C; INTEGER C;
   COMMENT THIS TABLE DEFINES THE CHARACTERSET OF THE PRINTER;
   CHAR:=CASE C+1 OF (
   46,44,58,59,48,49,50,51,52,53,54,55,56,57,43,45,
   40,41,00,00,61,60,62,47,45,37,38,42,33,39,00,63,
   00,00,00,90,89,88,93,92,91,87,86,85,84,83,82,81,
   80,79,78,77,76,75,74,73,72,71,70,69,68,67,66,65,
   000,000,000,122,121,120,125,124,123,119,118,117,116,115,114,113,
   112,111,110,109,108,107,106,105,104,103,102,101,100,099,098,097);

   PROCEDURE HIGHSPEED(SPEED);
   VALUE SPEED; INTEGER SPEED;
   BEGIN INTEGER  LINESTEP, POS, C;
      LINESTEP:=64000//SPEED;
      FOR POS:=0 STEP LINESTEP UNTIL LINESTEP*100 DO
      BEGIN FOR C:=CHAR(POS MOD DRUMSIZE) WHILE C=0 DO POS:=POS+1;
         FOR D:=1 STEP 1 UNTIL DEVS DO
         IF RESERVED(D) THEN WRITE(LP(D),FALSE ADD C,132,NL,1);
      END;
   END PROCEDURE HIGHSPEED;

   PROCEDURE CHARSET;
   BEGIN INTEGER SISO;
      FOR SISO:=1,2 DO
      BEGIN WRITE(LP(D), CASE SISO OF (
         <:<14><12><15>SHIFT IN CHAR. SET<10><11>:>,
         <:<15><12>SHIFT OUT CHAR. SET<10><14><11>:>),
         <:<48><32><48><32><13><32><49><32><49><10>:>);
         FOR I:=0 STEP 1 UNTIL 8,16 STEP 1 UNTIL 23,25,26,
         28 STEP 1 UNTIL 127 DO WRITE(LP(D),
         FALSE ADD I,132,I,NL,1);
         WRITE(LP(D), CASE SISO OF
         (<:SI END<10>:>,<:SO END<10>:>));
      END;
      WRITE(LP(D),<:<15><65><15><15><65><15><14><37><14><14>:>,
         <:<37><14><15><66><15><15><66><10>:>);
   END PROCEDURE CHARSET;

   PROCEDURE HTCANESC;
   BEGIN INTEGER SISO,M,N;
      FOR SISO:=2,1 DO WRITE(LP(D),
      FALSE ADD 12,1,FALSE ADD (SISO+13),1,<:HT<10><9>:>,
      FALSE ADD 49,15,FALSE ADD 9,1,FALSE ADD 50,16,
      FALSE ADD 9,1,FALSE ADD 52,17,FALSE ADD 9,1,
      <:<54><9><55><9><56><9><57><10>:>);

      WRITE(LP(D),<:<15><10>CAN<10><24><67><10>:>,
      FALSE ADD 48,132,<:<24><65><10>:>,
      FALSE ADD 48,133,<:<24><78><10><14><24><67><10><14>:>,
      FALSE ADD 48,132,<:<24><65><10><14>:>,
      FALSE ADD 48,133,<:<24><78><10>:>);

      FOR SISO:=1,2 DO
      BEGIN

COMMENT

```

```
658
659
659          WRITE(LP(D),CASE SISO OF (
660          <:<15><10>SI-ESC<10>:>,<:<14><10>SO-ESC<10>:>));
661          FOR M:=10,11 DO FOR N:=48 STEP 1 UNTIL 63 DO
662          WRITE(LP(D),<<ZDD>,FALSE ADD 24,1,N,FALSE ADD 32,1,M,
663              FALSE ADD 27,1,FALSE ADD N,1,FALSE ADD M,1);
664       END;
665    END PROCEDURE HTCANESC;
666
666    PROCEDURE CYCLICAL;
667    BEGIN INTEGER A,B,C,I1,I2;
668       I1:=0;
669       FOR A:=0 STEP 1 UNTIL 131 DO
670       BEGIN FOR A:=A WHILE CHAR((A+I1) MOD DRUMSIZE)=0 DO I1:=I1+1;
671          FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
672          BEGIN I2:=0;
673             FOR B:=1 STEP 1 UNTIL 132 DO
674             BEGIN FOR C:=CHAR((A+B+I1+I2) MOD DRUMSIZE)
675                WHILE C=0 DO I2:=I2+1;
676                WRITE(LP(D),FALSE ADD C,1);
677             END; WRITE(LP(D),NL,1);
678       END;  END;
679    END PROCEDURE CYCLICAL;
680
680    PROCEDURE LP_END;
681    BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO
682       BEGIN RESERVED(D):=MONITOR(6,LP(D),0,IA)=0;
683          IF RESERVED(D) THEN
684          BEGIN WRITE(LP(D),<:<15>:>);
685             IF RUNNO=0 OR RUNNO=RUNS THEN
686             BEGIN WRITE(LP(D),FALSE ADD 12,1);
687                SETPOSITION(LP(D),0,0);
688             END;
689          END ELSE ERROR(LP(D),4,0);
690       END;
691    END PROCEDURE LP_END;
692
692    PROCEDURE TEST_VT;
693    BEGIN INTEGER VT;
694       FOR VT:=27 STEP 48 UNTIL 3*48+27 DO
695    WRITE(LP(D),FALSE ADD CHAR(VT MOD DRUMSIZE),44,FALSE ADD 11,1);
696    END;
697
697
697 LPTEST_START:
698
698    FOR D:=1 STEP 1 UNTIL DEVS DO
699    BEGIN SYSTEM(5,DEVICE(D)+2,PARAM);
700       I:=0;
701       OPEN(LP(D),MODEKIND(DEVICE(D)),STRING PARAM(INCREASE(I)),M);
702       SETPOSITION(LP(D),0,0);
703    END; LP_END;
704
704    WRITE(OUTC,<:<10>RC 610 LINEPRINTER<10>:>);
705 RDR:
706    WRITE(OUTC,<:<10>DRUMSIZE=  :>);  UD; IND;
707    READ(IN,DRUMSIZE);
708    IF DRUMSIZE<0 OR DRUMSIZE>96 THEN GOTO RDR;
709
709 RTP:
710    GOTO CASE TESTPROG OF (DIRECTORY,A,B,C,D,E,F,RTP,RTP,RTP,
711    RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,X,Y);
712
712
712 COMMENT
713
```

```
713 RC 4000 PRINTERTEST
714
714 DIRECTORY:
715    WRITE(OUTC,<:
716
716    A    HIGHSPEED TEST
717    B    PRINT CHARACTERSET
718    C    TEST HT, CAN, ESC
719    D    CYCLICAL PRINTING
720    E    SEQUENCE
721    F    TEST VT
722    X    ROUTINE TEST
723    Y    TERMINATE
724
724 :>);    GOTO RTP;
725
725
725 E: BEGIN COMMENT SEQUENCE;
726       REAL ARRAY TEXT(0:66);
727       PACK(TEXT,0,67*6,128,8);
728
728 OUTSEQUENCE:
729       RUNADM(RTP,LP_END);
730       SP:=0;
731       WRITE(LP(D),STRING TEXT(INCREASE(SP)));
732       GOTO OUTSEQUENCE;
733    END SEQUENCE;
734
734
734 A: WRITE(OUTC,<:SPEED LINES/MIN = :>);  UD; IND;
735    RD(A,SP);
736    IF SP<1 THEN GOTO A;
737 AA:
738    RUNADM(RTP,LP_END);
739    HIGHSPEED(SP);
740    GOTO AA;
741
741
741 B: RUNADM(RTP,LP_END);
742    CHARSET;
743    GOTO B;
744
744
744 C: RUNADM(RTP,LP_END);
745    HTCANESC;
746    GOTO C;
747
747
747 D: RUNADM(RTP,LP_END);
748    CYCLICAL;
749    GOTO D;
750
750 F:    WRITE(OUTC,<:
751 MOUNT A FORMAT TAPE WITH HOLES FOR EVERY
752 SECOND LINE IN CHANNEL ONE:>); UD; IND;
753       INCHAR;
754 FF:   RUNADM(RTP,LP_END);
755       TEST_VT;
756       GOTO FF;
757
757
757 COMMENT
758
```

```
758
759
759
759 X: RUNADM(RTP,LP_END);
760    HIGHSPEED(1000);
761    HIGHSPEED(2000);
762    HIGHSPEED(3000);
763    CYCLICAL;
764    FOR I:= 1,2,3,4,5 DO FOR D:=1 STEP 1 UNTIL DEVS DO TEST_VT;
765    GOTO X;
766
766
766 Y:
767    FOR D:=1 STEP 1 UNTIL DEVS DO CLOSE(LP(D),TRUE);
768
768
768
768 END TESTLP;
769
769
```

RCSL: 44-D13
Author: Per Hansen
Edited: December, 1970
Type: ALGOL 5 program

RC 4000

TIMESHARED TESTPROGRAM LIBRARY

MAGNETIC TAPE

KEYWORDS: RC 4000, Diagnostic program, Algol program, ISO tape

ABSTRACT: This program contains 5 routines intended for error location and maintenance purposes on the RC 747 and RC 749 magnetic tape stations. It is organized as a block incorporated in the 'TEST' system.

A/S REGNECENTRALEN
Falkoner Alle 1
2000 Copenenhagen F

CONTENTS:

MAGNETIC TAPE

1. Main characteristics

1.1. Initiation

The process in which the testprogram is run must be created with a function mask which allows creation of peripheral processes, i.e. the function bits 0-6 must be specified.

Example:
new peh size 20000 function 0 1 2 3 4 5 6 run

this is necessary as the program automatically creates the peripheral processes needed when the tapestations under test are set to the remote state.
In the beginning of every new run and in some other cases it is checked whether the tapestations are in the remote state and a peripheral process is assigned to it. If this is not fulfilled one of the following situations may come up:

a. The message

mount <document name>

is output on the console. This is usually a consequence of sudden 'local' during a write operation. Set the station in the 'remote' state, press the attention key on the console and type to s:

call <device no> <document name>

where <device no> is the device number of the station in question and <document name> is the corresponding process name (see below).

b. If the operation was an input the statusbit 'unknown' is set and the device is suspended from further testing until it is

returned to 'remote'.

c. If the document name already has been assigned to device which is reserved by another process the following message is output:

xxxtest create <document name> reserved

and 'TEST' terminates.

d. If the document name already has been assigned to another device which is not reserved by another process the following message is output:

xxxtest create <document name> mounted on station <station no>

and 'TEST' terminates.

e. If the function mask of the process differs from the one mentioned above the following message is output:

xxxtest create <document name> not allowed

and 'TEST' terminates.

f. If the process is not a user of the device the following message is output:

xxxtest create <document name> not user

and 'TEST' terminates

An undesired 'local' during rewind can be remedied in this way:

rewind the tape to loadpoint and set the station to the 'remote' state. Now the process is automatically recreated.
The <document name> mentioned above is composed (by 'TEST') of the letters mt followed by a three-digit number which is the device-number.

Example:
The tapestation with the devicenumber 6 becomes the following document name:

mt 006

1.2 Call of the program

When an internal process has been created the program is called in this way:

<out file> = test mt.< device 1 > .< device 2 > .....

where <out file> describes the device, e.g. a lineprinter used for output of statuswords, bitpatterns and error messages concerning the tapestations actually under test whereas the operators console is used for control information as usual.
<outfile> may also describe a backingstorage area which is not described in a fp-note (r, s, t, u, v).

<device 1> , <device 2> etc. denotes the devicenumbers of the stations to be tested. Any number of 7- and 9 track stations may be tested simultaneously provided that the message buffers needed are available.
Note. If the devicenumber typed does not correspond to a tape-station a warning message is output and 'TEST' terminates.
Every time a new run is initiated it is checked whether a message has arrived (from e.g. the operators console). If this is the case the message:

operator termination

is output and a new testprogram may be selected. The contents of the message received is irrelevant.

1.3 Size and speed

The program may be run in a process area as small as 10000-15000 bytes depending on which program is started and on the number of devices. However, a process size of 20000-30000 bytes is recommended.

The number of message buffers needed is 2 per tapestation + 3 for communication with console and printer. If the buffer claim of the process is too little the message

xxxbuffer claim ‹buffers›

is typed out and a new process with ‹buffers› buffers available must be created.

Normally the test keeps the stations at full speed. However, when running test on more stations simultaneously the speed may be lower, especially when the test 'write and read' is executed with complete checking against contents. To obtain higher speed the remedies are: larger process area, shorter blocks, fewer tapestations at the same time, no other users on the RC 4000.

2. Write and read

2.1 Purpose

This program can operate the tapestations with reading and writing in the odd/even parity, hi/lo density, write enable/protect modes. Furthermore checking of the statuswords and the data transferred may be performed.

Both reading and writing is performed in the double-buffered mode, i.e. two core areas each of the same size as one block are used

in turn for input or output to the magnetic tape. This enables a faster program execution as the former block may be checked while the latter one is on its way in or out. The test is performed simultaneously on the devices specified in the call.
If a station goes to 'local' it is suspended from testing, but every time the tapes are rewound it is checked whether it is remote or not. If remote the station is accepted in the test sequence again.
When EOT is sensed the rest of the current reading or writing is skipped and the tapes are rewound.

2.2 Initiation

When the program has been selected it asks:

blocksize =

and the operator is expected to type a number defining the number of words in each block. If only a NL is typed or if the area needed is not available the following message is output:

max blocksize = <size> words

this means that the blocksize has been adjusted according to the storage available. The message

xxx core area too small

means that not even single-word buffers can be created if the program shall be executed at a reasonable speed.
TEST terminates.
Next the program asks:

blocks, files =

and the operator must type two numbers, one defining the number

of blocks in each file, the other defining the number of files. If only NL is typed blocks and files are initialized to 20. Then the parity is asked for:

parity =

and you may type o for odd parity and e for even parity. If only a NL is typed odd parity is selected.
Further the program asks for the other modes:

mode =

The modes possible, some of which must be typed on the same line separated by spaces, are the following:

write
erase
read
find
statuscheck
checkall
dataprint
lo-den
protect
type
octal
binary
monitor

Only the first letter of the word in question needs to be typed. If only a NL is typed the mode is initialized to:

Statuscheck, dataprint, write, erase, read, find, monitor, write enable and high density.

The effect of the various modes is:

write — In each run the blocks and files specified are written in sequence starting from loadpoint.

read — In each run the blocks and files specified are read in sequence starting from loadpoint
If both write and read are specified all the write operations will be executed at first. Then the tape is rewound and all the read operations are executed.

erase — If a parity error is detected during write the tape is backspaced behind the block before the bad spot, then it is upspaced 1 block and a number of inches of the tape is erased. This is repeated up to 5 times if necessary. If the error persists action is taken as if erase mode was not selected. (See statuscheck). After this writing of the next block is initiated.

find — If a parity error is detected during read the operation is repeated up to 5 times. If the error persists action is taken as if the find mode was not selected, (see statuscheck and checkall). After this reading of the next block is initiated.

statuscheck — In this mode the statusword, the number of bytes transferred, and the block identifier (if any) is checked. Furthermore, if any error in this is detected after a read operation the contents of the entire block is checked.
If statuscheck is not selected erase and find are still effective, but in case of persistant error nothing happens.

checkall — This mode has the same effect as statuscheck except

that the entire contents of the block is always checked. Any error detected in the statuscheck or the checkall mode gives rise to a message as described in section 2.5.

dataprint — If an error in the contents of the block read is detected the error-stricken words will be printed out as described in section 2.5.

lo-den — Low density is selected. In this mode the density bit is expected to be 0 otherwise 1.
If the density is wrong this is treated as a status error.

protect — File protect is expected. In this mode the write enable bit is expected to be 0 otherwise it is expected to be 1. If the bit in the status word is wrong this is treated as a status error. Further, if the write-enable is false during write the message:

mount ring < document name >

is output on the operators console and the program waits for the ring to be mounted

type , octal , binary , see section 2.3.

when the mode has been defined it is printed on the <outfile> .

monitor — After completion of read or write of the number of files specified the relative number of parity errors per device is printed on <out files> . The number of errors is printed in 3 categories: write, erase, and read, as a percentage of the total number of operations in that category since the previous printing or since the first run was started.

The errors are counted by the monitor, and if a parity error e. g. causes more re-readings (in the find mode) every re-reading will increase the error counter.
The effect of monitor is independent of the modes statuscheck and checkall.

2.3 Testdata

During write a certain bitpattern is generated to form the contents of the block. During read the data transferred is checked against this bitpattern.
The pattern is composed as follows:
The first two words, called the identifier contains the current block and file number (the blocks and files are counted from 0 and up). This block identifier is omitted if the block consists only of one or two words or if the even parity mode is selected (because even zeroes must be avoided).
The rest of the block is filled cyclically with the contents of a table, testdata, which consists of 16 elements (1 element = 48 bits). The contents of testdata are initialized to a worst-case bit pattern. It may, however, be changed by means of the type mode which may be modified with the octal or the binary mode. If the type mode is selected the program asks:

type sequence:

Now the operator must type a sequence of characters as explained in the description of 'TEST', section 4.
The ISO-value of the characters are for 7-track interpreted modulo 64 i.e. 8 characters are packed in each element and for 9-track they are interpreted modulo 256, i.e. 6 characters are packed in each element. If the value wanted does not correspond to any of the characters on the typewriter's keyboard the value may be defined by the number delimiter (see the examples). Only the elements

filled are used and only 15 elements may be filled.
Whether the characters are packed according to 7- or 9-track depends on the kind of the first device in the call of 'TEST'.
If the octal mode is selected the bitpattern can be defined by typing a sequence of octal digits (the digits 0-7). In the binary mode the bitpattern is defined by using the character 0 or . to represent a binary zero and the character 1 or ! to represent a binary one.

Example

In octal mode the sequence

0707 , 10

will be interpreted as a sequence of 3x4x10 = 120 bits. As only full elements are used 2 elements (96 bits) will be used cyclically to generate the information in the block.

2.4 Clearings of buffers

In the checkall mode and other cases when data transferred from tape have been checked the buffer contents are set to zero after the checking so that it is well-defined before the next transfer.
During write, when the blocks and files specified have been output a tapemark is written. After this further two blocks containing all ones (including the identifier) and another tape mark is written.
This is done for two reasons: first, during read one excessive block is read due to the double-buffer mode, second, if the program is going to read after the writing the buffer areas are filled with a welldefined bitpattern which usually will be different from that of the two first blocks.
In the case 'End of tape' sensed the writing of the excessive blocks is omitted. However, still one block will be output after the EOT and in the error messages the two last blocks will be mentioned with end of document status. During read only these two blocks will be input but only the first one will be checked.
NOTE: If the blocklength is bigger than 7500 words the tape may fall off the reel during write.

2.5 Errormessages

If a statuserror, identifiererror, or dataerror is detected or a wrong number of characters have been transferred an errormessage will be output on the <outfile>. The message consists of two lines, the first one containing the devicenumber followed by the logical status word as described in 'TEST', section 5. The second line contains the current runnumber, the state, which may be either writing, reading, or dataerror (dataerror may occur during read only). After the state the current block- and filenumber are output.

If a dataerror is detected in the dataprint mode the error-stricken words will be output as a message consisting of the word-number, the received and expected word. The words are numbered 0, 1, 2, 3, 4, ... Two words are printed in each line. The words are printed as binaries and as integers.

The binaries are printed in groups of six or eight bits depending on the actual device being a 7-track or a 9-track station.

When checking the data of the blocks read only the bytes actually transferred (as computed by the monitor) are checked. This means, that if the blocklength expected was 512 bytes and the block on the tape had a length of only 20 bytes the checking and errormessages will comprehend no more than the 20 bytes.

NOTE: the 'TEST' system normally does not operate directly with the statusword yelding the number of characters transferred, but with the number of bytes transferred. The number of bytes will always be even. This convention enables a homogenous handling of 7- and 9-track stations.

If the number of characters transferred does not match the 24-bit word-format of RC 4000, i.e. the last word of the block is only partially filled the 'bytes transferred' is rounded to the nearest higher even number. To indicate this 'defect' word the statusbit 'word defect' is set. The characters of the defect word are placed in the leftmost positions and the rest of the word is filled with zeroes.

NOTE: If only one word is transferred and this is defect 'word defect' will as an exception, not be set.

Example:

From a 7-track station is transferred 5 characters. This will give rise to a message which tells that 4 bytes have been transferred and that worddefect exists.

If error is detected in 10 consecutive blocks during read or 12 consecutive blocks during write on a certain device the message:

give up

is printed on the outfiles and the current run is terminated. The tapes are rewound and the next run is initiated.

2.6 Table of modes in write and read test.

action / mode	checking against status	checking against identifier	checking against dataerror	printing of bad words
no check	no	no	no	no
status check	yes	yes	no	no
checkall	yes	yes	yes	no
statuscheck + dataprint	yes	yes	in case of identifier error or statuserror	yes
checkall + dataprint	yes	yes	yes	yes

3. Print contents

3.1 Purpose

This program is intended for printing out the contents of an arbitrary block on each of the tapestations under test.

3.2 Initiation

First the program creates a buffer for each device. The buffer

size is the maximum available and is announced in this way:

max. blocksize = <size> words

If a longer block is input blocklength error will occur and only the part of the block transferred will be printed.
When the program asks:

first block, file =

the operator must type the numbers corresponding to the first block to be printed. The blocknumber must point to a block inside the file in question.
Next the program writes:

number of blocks =

and the operator types the number of blocks wanted.
Now the tapestations in question are positioned simultaneously to the file and block mentioned and the blocks from the first device are printed on the <outfile>. When this is completed the corresponding blocks from the next device will be printed and so on.
If more runs were specified this will be repeated in each run.
If EOT is sensed the reading on the current device is terminated and the tape is rewound.

3.3 Printing

A block is printed in the following way: First the devicenumber followed by the logical statusword is output. Furthermore the runnumber, the state (reading) and the current block- and file-number are output. In the blockcounting a filemark is treated as the last block in the file.
Next the contents are printed with two words in each line.
First the number of the word, then the two words printed as

binaries and at last printed as text, i.e. every 8 bit byte interpreted as an ISO character are output. The special characters 0-31, and 127-255 are printed as spaces.
In case of statuserror the block reading is repeated up to 5 times before the block is printed.

4. CRC-test

4.1 Purpose

The purpose of this test is to check the CRC correction circuit of the RC 749. This is done by reading a block which has been correctly written, with one read amplifier missing. The bit pattern is chosen so that dropout will not occur.

4.2 Initiation

When the program is started, it first writes 50 similar blocks 24 characters long. Next the tapes are rewound and the message:

remove read amplifier track 1

is output. When the read amplifier has been removed, a NL must be typed and the program now proceeds with reading the blocks.
In case of parity error the tape is backspaced and the read-operation is repeated. Now the automatic correction mechanism should generate the information missing. If not successful this is repeated up to 4 times and a message is output as described for write and read, section 2.5.
When all of the 50 blocks have been read the tapes are rewound and the reading is repeated after requesting the operator to remove read amplifier 2, 3, 4, ... Only one read amplifier must remain removed at the same time.
The tapestation is expected to be in the high-density mode, otherwise errormessages will be output.

5. Test of functions

5.1 Purpose

This testprogram performs testing of the various modes and checking systems in the MTC.
The test is executed in a number of states, each state identified by a number. This number is part of the errormessages. If the test is performed on more devices at the same time these will not be operated simultaneously but one by one.
The tapestations must be equipped with tapes with write-enable ring mounted.

5.2 Testing

First the program asks:

density =

and the operator types lo or hi dependant on the density wanted. If only NL is typed high density is selected. If the density selector on the tapestation is set in a wrong position status error messages will be output.
Notice that 9-track stations only can be operated correctly in high-density (800 bpi), and that the testprogram will go wrong if another density is selected.
The test is executed in the following states:

1. First the program writes 3 blocks in
2. odd, even, and odd parity respectively
3. (state 1, 2, and 3). The blocklengths are 1 word (containing the characters 1, 2, 3, 4 for 7-track or 1, 2, 3 for 9-track), 2 words (containing the characters 5, 6, 7, 8, and 9, 10, 11, 12, for 7-track or 4, 5, 6, and 7, 8, 9, for 9-track) and 3 words (containing the characters 13, 14, 15, 16, and 17, 18, 19, 20, and 21, 22, 23, 24, for 7-track or 10, 11, 12,

and 13, 14, 15, and 16, 17, 18 for 9-track) respectively.

4. Next the 3 blocks are read (state 4,
5. 5, and 6) in even, odd, and even
6. parity respectively into a buffer 2 words long. It is tested that

   a. parity error is detected in each block.
   b. The correct number of characters is counted (i.e. 4, 8, and 8 characters respectively).
   c. Blocklength error occurs in state 6 only.
   d. The characters are correctly transferred.

7. The tape is rewound and 1 block is written with odd parity (state 7). This block consists of 400 words containing the integers:

   0, 1, 2, 3, ...398, 399, respectively

8. This block is read 3 times (state 8,
9. state 9, and state 11) into a buffer
10. 400, 200, and 600 words long,
11. respectively. Having read the block the second time an erase instruction is performed (state 10), which must not destroy the rest of the block. It is tested that the number of input characters is 1600, 800, and 1600, for 7-track and 1200, 600, and 1200, for 9-track respectively.
12. The tape is rewound and the program now writes in even parity a block consisting of one word containing the characters 3, 2, 1, 0 (for 9-track the characters 2, 1, 0). It is tested that parity error occurs, because it is attempted to output the character 0 in even parity.
13. Next the block is read (state 13) and it is tested that the block is read correctly and without parity error.
14. The tape is rewound and the program
15. writes, for 7-track, 2 blocks (state 14 and 15) in odd and even parity, respectively. Each block consists of one word, the first containing the characters 15, 15, 15, 15, the next

containing the characters 15, 15, 0, 0 or in other words two tapemarks.

14. For 9-track only a normal tapemark is written (state 14) and state 15 is omitted.
    It is tested that tapemark is sensed and that the correct number of characters is written.
16. For 7-track the two blocks are read
17. (state 16 and 17) and it is tested that tapemark is sensed and that the number of characters is 4 and 2, respectively.
17. For a 9-track state 16 is omitted. In state 17 the block is read and it is tested that tapemark is sensed.
18. Now the program rewinds the tape and writes 5 blocks:

block No.	No. of words	character contents	parity
1	1	1	odd
2	400	2	even
3	2	3	odd
4	1	15*	even
5	500	5	even

*On 9-track tape this block is replaced by 1 character of the value 19 (a tapemark)

19. The tape is rewound and it is tested that the loadpint is sensed.
20. A 'backspace block' operation is performed and it is tested that loadpoint is sensed and that the contents of the two first words of the buffer are unchanged.
21. Upspace 1 block.
22. Read 1 block and check that this was block No. 2.
23. Upspace 1 file, check that tapemark is sensed and that the two first words of the buffer are unchanged.
24. Read 1 block and check that this was block No. 5.
25. Backspace 1 file and check that tapemark is sensed.
26. Backspace 1 block.
27. Read 1 block and check that this was block No. 3.

28. Upspace 1 block and check that tapemark was sensed.
29. Backspace 1 block and check that tapemark was sensed.
30. Backspace 1 file and check that loadpoint is sensed.
31. Now the tape is rewound and 80 erase operation are executed thus erasing a length of tape corresponding to 12 seconds movement. After this a tapemark is written, the tape is rewound and a read operation is initiated. It is tested that this is terminated with timer status sensed and 0 characters transferred. If the timer does not work tapemark status will be sensed instead.
32. The tape is rewound and 2 blocks containing all ones are written in odd parity. The first block is 40 words long, the second one is 600 words long. Now the tape is positioned to the beginning of the second block and a block of one word is written. Next the rest of the 600-word block (old block 2) is read. It is checked that this is longer than 12 characters, otherwise a noise block may exist in the blockgap. In this case the rest of the test is skipped. Otherwise the number of characters read is used to compute the stopdistance, i.e. the distance the tape moved after writing the new block 2. The distance must be within the range of 25-60 dmm (1dmm = 10-4m). The accuracy depends on the distance from read to erase head which is assumed to be 250 $\pm$ 10dmm for 7-track and 167 $\pm$ 10 dmm for 9-track. If a negative stopdistance is computed this usually means that the erase head is ineffective. This will cause wrong results in state 33, 34, too.
33. Now the tape is rewound and upspaced 2 blocks, i.e. to the blockgap after the new block 2 and again a one word block is written. By counting the characters now remaining in the old block 2 the startdistance may be computed. The startdistance is the distance which the tape moves over the write head before the writing is initiated. The actual length depends on the start delay adjustment as computed from this equation:

startdistance + stopdistance + hc2 = blockgap

where hc2 is the distance between write and read head and stopdistance is the one measured in state 32. It is checked that the blockgap lies inside the range of 192 $\overset{+5}{+25}$ dmm for 7-track and 154 $\overset{+5}{+25}$dmm for 9-track. The accuracy of this measurement depends on the distance between write and read head which is assumed to be 76 $\pm$ 1dmm for 7-track and 38 $\pm$ 1 dmm for 9-track.

34. Next the tape is rewound and positioned to the blockgap right after block 1 and a block of 600 words is written.
Then a backspace block and writing of a 1-word block is performed. Then the remaining part of the 600-word block is read and the number of characters is compared with that of state 29, and the reverse stopdistance is computed.
The reverse stopdistance is the distance which tape moves over the read head in reverse direction after having passed the beginning of the block. (This movement includes the moving during elapse of the reverse delay time). This may be expressed in this way:

revstopdistance = hc2 + startdistance

Actually the reversedistance must be slightly shorter. It is checked that it lies inside the range of 5-25 dmm below the sum of hc2 + startdistance.
The accuracy of this checking will depend only on the accuracy of the tapetransport movements.

5.3 Errormessages

If an error is detected in a state a message is output on the ⟨outfile⟩. The message consists of the devicenumber, the statenumber and a text which tells the nature of the error. Next the received and expected values are printed as integers and, if relevant, as bitpattern. If more than 20 consequtive words of a block are erroneous the rest of the states will be skipped in the current run.

The words of the block are numbered 0, 1, 2... etc.

6. Operations

6.1 Purpose

An arbitrary sequence of output on, input from, sensing and movement of a magnetic tape may be performed by means of this program. It is checked that any operation is finished after elapse of 10 seconds.

6.2 Initiation

A tape is mounted, equipped with a ring if necessary. When the program writes:

blocksize =

The operator types the number of words of a block. If only NL is typed or the core area needed is not available in the process the maximum size is selected and the message:

max blocksize = <size> words

is output on the console.
next the program asks:

parity =

and the operator types odd or even, followed by a NL. If only NL is typed odd parity is chosen. When the program asks:

operations =

the operator specifies the tapeoperations desired by typing 1-10 characters followed by a NL. The characters must be selected

among those shown below:

r	read	(1 block)
w	write	(1 block)
t	write tapemark	
e	erase	
s	sense	
0	upspace file	
1	upspace block	
2	backspace file	
3	backspace block	
4	rewind	
5	rewind and lock out	

During write the value of the characters are:

7-track

1, 2, 3,....62, 63, 0, 1, 2, 3, ...etc.

9-track

1, 2, 3, ... 253,254,255, 1, 2, 3, ...etc.

Notice that if the 7-track stations are operated in even parity and the blocklength is 16 words or longer the block will be terminated when the zero-character is output.
In each run the specified tape operations are performed in an uncritical way, that is without caring about the status. However, interrupt is required to terminate the operation.

6.3 Time supervision

If more tapestations are tested simultaneously and interrupt from one of them is missing the program waits for app. 10 seconds, then the message

<device no> busy for 10 seconds is output on the <out

file> and the device is temporarily suspended from the testing while the other tapestations proceed.
If, however, all the tapestations under test goes busy for a time longer than 10 seconds the program stops until one of them is ready again.
The message mentioned above will be repeated every 10 seconds indicating the total number of seconds elapsed, until an interrupt is generated. When this happens the device in question is returned to the test sequence.
The interrupt necessary may be generated manually by rewinding the tape in local mode and, when loadpoint is reached, pushing the 'remote' button.
If the tapestation is set in local during a read/write operation it will not remain busy but the intervention status will force rejection of any operation.
If end of tape is sensed the message:

<device no.> end of tape

is output and the tape is rewound.
NOTE: If the current operation is 'upspace block' or 'upspace file' the tape can not be stopped at EOT unless a blockgap respectively a filemark is found.

7. Testprogram x and y

If testprogram x is specified a routinetest will be executed. It is composed of

1. The functions test (d)
2. The write and read test (a).

The latter is executed in the standard mode with the number of blocks and files set to 100 and the blocksize set to the maximum available.
If testprogram y is selected the devices under test are released

(the reservation is cancelled) and the tapes are rewound.

8. Examples:

```
att s
new peh size 20000 catalog 0 3 function 0 1 2 3 4 5 6 run
ready

to peh
i tre

from s
message peh load reader

from peh
TEST 2.2/2
    1 begin
algol end


bs    4  13
tw    2  11  18  19  20  21  22
pt    0   1
pl
lp    5
cr
mt    6   7   8   9  10




end
test mt.6.7.8

***buffer claim 10




end

att s

to peh


to s
remove buf=10 run

to peh


from s

ready
to peh

; comment Now the message buffers needed are available
```

```
test mt.6.7.8

RC 747/749 magtape


testprogram: [a]
number of runs =[2]

blocksize =

max blocksize = 364 words
from s
message peh mount mt006
message peh mount mt007
message peh mount mt008

from peh

blocks, files =

parity =

mode =

mode = odd statuscheck dataprint write erase read find wre hi monitor

run no.       1

* 6 monitor   wr        %   er        %   re        %
* 7 monitor   wr  0.468 %   er        %   re        %
* 8 monitor   wr  0.235 %   er        %   re        %

* 6 monitor   wr        %   er        %   re        %
* 7 monitor   wr        %   er        %   re        %
* 8 monitor   wr        %   er        %   re        %
run no.       2

* 6 monitor   wr  0.235 %   er        %   re        %
* 7 monitor   wr  0.235 %   er        %   re        %
* 8 monitor   wr  0.235 %   er        %   re        %
* 6 monitor   wr        %   er        %   re  0.236 %
* 7 monitor   wr        %   er        %   re        %
* 8 monitor   wr        %   er        %   re        %

test end

testprogram: [a]
number of runs =[2]

blocksize = 1

blocks, files = [1000 1]

parity =[e]

mode =[w r c lo ty bin]

type sequence:
[111...111.....111...111,2]
```

```
mode = even checkall write read wre lo type binary

run no.       1

* 7  parity we lo_den word defect 4 bytes
   1'run  dataerr  block 174 file 0

* 7  parity we lo_den word defect 4 bytes
   1'run  dataerr  block 175 file 0

* 7  parity we lo_den word defect 4 bytes
   1'run  dataerr  block 176 file 0

* 7  parity we lo_den word defect 4 bytes
   1'run  dataerr  block 177 file 0

run no.       2

* 6  protect lo_den unknown 0 bytes
   2'run  reading  block 321 file 0
; comment unknown means that the device has gone to 'local'


test end

testprogram:  c
number of runs = 1

from s
message peh mount mt006

from peh

***device  6 no 9-track
***device  7 no 9-track
***device  8 no 9-track




end
test mt.10

RC 747/749 magtape

testprogram: c
number of runs = 1


from s
message peh mount mt010

from peh

run no.       1
```

```
remove readamp. track 1

*10 monitor   wr        %   er        %   re        %
remove readamp. track 2

*10 monitor   wr        %   er        %   re        %
remove readamp. track 3

*10 monitor   wr        %   er        %   re        %
remove readamp. track 4

*10 monitor   wr        %   er        %   re        %
remove readamp. track 5

*10 monitor   wr        %   er        %   re        %
remove readamp. track 6

*10 monitor   wr        %   er        %   re        %
remove readamp. track 7

*10 monitor   wr        %   er        %   re        %
remove readamp. track 8

*10 monitor   wr        %   er        %   re        %
remove readamp. track 9

*10 monitor   wr        %   er        %   re        %
*10 monitor   wr        %   er        %   re        %

test end

testprogram: [d]
number of runs =[1]

density =[hi]

run no.       1

*10    1'run,state 32 stopdistance (dmm)
     received:       64
     expected:       49


test end

testprogram: [b]
number of runs =[1]

max blocksize = 2212 words
parity =[o]

first block, file =[0,0]

number of blocks =[4]

run no.       1
```

```
*10  we hi 80 bytes
   1'run  reading  block 0 file 0

    0,    1  11111111 11111111 11111111 11111111 11111111 11111111
    2,    3  11111111 11111111 11111111 11111111 11111111 11111111
    4,    5  11111111 11111111 11111111 11111111 11111111 11111111
    6,    7  11111111 11111111 11111111 11111111 11111111 11111111
    8,    9  11111111 11111111 11111111 11111111 11111111 11111111
   10,   11  11111111 11111111 11111111 11111111 11111111 11111111
   12,   13  11111111 11111111 11111111 11111111 11111111 11111111
   14,   15  11111111 11111111 11111111 11111111 11111111 11111111
   16,   17  11111111 11111111 11111111 11111111 11111111 11111111
   18,   19  11111111 11111111 11111111 11111111 11111111 11111111
   20,   21  11111111 11111111 11111111 11111111 11111111 11111111
   22,   23  11111111 11111111 11111111 11111111 11111111 11111111
   24,   25  11111111 11111111 11111111 11111111 11111111 11111111
   26,   27  11111111 11111111 11111111 11111111 11111111 11111111
   28,   29  11111111 11111111 11111111 11111111 11111111 11111111
   30,   31  11111111 11111111 11111111 11111111 11111111 11111111
   32,   33  11111111 11111111 11111111 11111111 11111111 11111111
   34,   35  11111111 11111111 11111111 11111111 11111111 11111111
   36,   37  11111111 11111111 11111111 11111111 11111111 11111111
   38,   39  11111111 11111111 11111111 11111111 11111111 11111111

*10  we hi 2 bytes
   1'run  reading  block 1 file 0

    0        11111111 11111111 11111111

*10  parity we hi hard_error 682 bytes
   1'run  reading  block 2 file 0

    0,    1  11..1... 11111111 11111111 11111111 11111111 11111111
    2,    3  11111111 11111111 11111111 11111111 11111111 11111111
    4,    5  11111111 11111111 11111111 11111111 11111111 11111111
    6,    7  11111111 11111111 11111111 11111111 11111111 11111111
    8,    9  11111111 11111111 11111111 11111111 11111111 11111111
   10,   11  11111111 11111111 11111111 11111111 11111111 11111111
   12,   13  11111111 11111111 11111111 11111111 11111111 11111111
att[s]

from peh
111 11111111 11111111
att[s]

from peh

*11  tapemark output_lost 344 bytes

to s
[remove run]

from peh
11
from s
ready

to peh
```

```
test mt.6.10

RC 747/749 magtape

testprogram: e
number of runs = 1000

blocksize = 1000

from s
message peh mount mt006
message peh mount mt010

from peh

parity =

operations =w

run no.       1

*10   busy for  10 seconds
*10   busy for  20 seconds
* 6   end of tape              ; now device 6 is rewound
*10   busy for  30 seconds
*10   busy for  40 seconds
*10   busy for  50 seconds
*10   busy for  60 seconds
*10   busy for  70 seconds
*10   busy for  80 seconds
*10   busy for  90 seconds
*10   busy for 100 seconds
*10   busy for 110 seconds
*10   busy for 120 seconds
*10   busy for 130 seconds
*10   busy for 140 seconds
*10   busy for 150 seconds
*10   busy for 160 seconds
*10   busy for 170 seconds
*10   busy for 180 seconds
*10   busy for 190 seconds


testend


; comment Now the 1000 write operations have been executed
; on device 6
```

```
testprogram: y

from s
message peh suspend mt006
message peh suspend mt010

from peh




end

; If no printer ia available it may be useful to use a backingstorage
; area as outfile and later read the information by means of the
; editor:
Example:
Get the contents of block 333, word 777 on tapes device 6 and 10:

rr=set 100
rr=test mt.6.10

RC 747/749 magtape

testprogram: b
number of runs =1

max blocksize = 1098 words
parity =o

first block, file =333,0

number of blocks =1

run no.       1


test end

testprogram: y

from s
message peh suspend mt006
message peh suspend mt010

from peh




end

edit rr
edit begin.
```

```
1./,777/,p,f
  776,  777  ..1...1. ..1...11 ..1..1.. ..1..1.1 ..1..11. ..1..111 "%&'
edit end.
```

; writing of a block generating the CRC character 0 may be performed
; in this way (note that although only 24 bits are used at least 48
; bits must be defined) :

```
test mt.10

RC 747/749 magtape

testprogram: a
number of runs = 1

blocksize = 1

blocks, files = 100,2

parity = o

mode = w ty bin

type sequence:
11101110111011100111001111,2
mode = odd write wre hi type binary

run no.       1


test end

testprogram: y

from s
message peh suspend mt010

from peh




end

; comment Now all tapestations are released
```

```
testprogram: e
number of runs = 99999

blocksize =

max blocksize = 2230 words
parity =

operations = ww

run no.       1

to peh
stop

operator termination
testprogram:

; comment As a message was sent to the process the test terminates
; at the end of the current run

Note, that this will only work in system 1




testprogram: d
number of runs = 2

density = hi

run no.       1

*10    1'run,state 32 stopdistance (dmm)
     received:       65
     expected:       49

run no.       2

*10    2'run,state 32 stopdistance (dmm)
     received:       66
     expected:       49


test end

testprogram: a
number of runs = 1

blocksize = 60

blocks, files = 1,1

parity = o

mode = w s
```

```
mode = odd statuscheck write wre hi

run no.       1


test end

testprogram: [b]
† comment Now the worst-case bitpattern generated in test a is printed.
; Note, that the 2 first words contains the block- and file number, i.e. 0

number of runs =[1]

max blocksize = 2212 words
parity =

first block, file =[0 0]

number of blocks =[1]

run no.       1

*10  we hi 120 bytes
   1'run  reading  block 0 file 0

    0,    1  ........ ........ ........ ........ ........ ........
    2,    3  ........ ........ ..1..... .1...... 1......1 ......1.
    4,    5  .....11. ....11.. ...11... ..11.... .11..... 11.....1    0
    6,    7  .1....1. 1....1.1 ....1.1. ...1.1.. ..1.1... .1..1... B   (H
    8,    9  1..1...1 ..1...1. .1...1.. 1...1..1 ...111.. ..111...  "D  8
   10,   11  .111.... 111....1 11....11 1....1.1 1...1.11 ...1.11. p
   12,   13  ..1.11.. .1.11... 1.11...1 1.1...11 .1...11. 1...11.1 ,X  F
   14,   15  ...11.1. ..11.1.. .1..11.. 1..11..1 ..11..1. .11..1..  4L 2d
   16,   17  11..1..1 1..1.1.1 ..1.1.1. .1.1.1.. 1.1.1..1 .1.1..1.   *T R
   18,   19  1.1..11. 1.11.1.1 .11.1.1. 11.1.1.1 1.1.1.11 .1.1.11.   j  V
   20,   21  11..11.1 1..11.11 ..11.11. .11.11.. 11.11..1 1..1.111   61
   22,   23  ..1.111. .1.111.. 1.111..1 .111..1. 111.1..1 11.1..11 .Ø r
   24,   25  1.1..111 .1..111. 1..111.1 ..111... 1111...1 111...11  N 8
   26,   27  11...111 1...1111 ...1111. 11.111.1 1.111.11 .111.11.      v
   28,   29  111.11.1 11.11.11 1.1.1111 .1.1111. 1.1111.1 .1111.1.     z
   30,   31  1111.1.1 111..111 11..1111 1..11111 ..11111. .11111..     >ø
   32,   33  11111.11 1111.111 111.1111 11.11111 1.111111 .111111.
   34,   35  ........ ........ ..1..... .1...... 1......1 ......1.
   36,   37  .....11. ....11.. ...11... ..11.... .11..... 11.....1    0
   38,   39  .1....1. 1....1.1 ....1.1. ...1.1.. ..1.1... .1..1... B   (H
   40,   41  1..1...1 ..1...1. .1...1.. 1...1..1 ...111.. ..111...  "D  8
   42,   43  .111.... 111....1 11....11 1....1.1 1...1.11 ...1.11. p
   44,   45  ..1.11.. .1.11... 1.11...1 1.1...11 .1...11. 1...11.1 ,X  F
   46,   47  ...11.1. ..11.1.. .1..11.. 1..11..1 ..11..1. .11..1..  4L 2d
   48,   49  11..1..1 1..1.1.1 ..1.1.1. .1.1.1.. 1.1.1..1 .1.1..1.   *T R
   50,   51  1.1..11. 1.11.1.1 .11.1.1. 11.1.1.1 1.1.1.11 .1.1.11.   j  V
   52,   53  11..11.1 1..11.11 ..11.11. .11.11.. 11.11..1 1..1.111   61
   54,   55  ..1.111. .1.111.. 1.111..1 .111..1. 111.1..1 11.1..11 .Ø r
   56,   57  1.1..111 .1..111. 1..111.1 ..111... 1111...1 111...11  N 8
   58,   59  11...111 1...1111 ...1111. 11.111.1 1.111.11 .111.11.      v


test end

testprogram:
```

9. Alphabetic list of errormessages

x <number>,<status><number> bytes

During test of a device there has been statuserror. The first <number> is the devicenumber, the second one is the number of bytes transferred. The statusbits are printed as texts.

xxx buffer claim <claim>

The process does not own the messagebuffers needed for that number of tapestations. A new process with a bigger buffer claim must be created.

xxx core area too small

The process cannot hold bufferareas for communication with the tapestations.

xxx device <deviceno> no 9-track

The CRC test cannot be executed on 7-track stations.

xxx fp name

Algol is not in the machine during load of TEST.

xxx fp syntax

xxx test call

TEST has been called in a wrong way

xxx full

Occurs after typing a character sequence in case of overflow in the sequence-table.

x input parity

Parity error on the typewriter.

xxx line 18·1 undeclared

Algol library is not in the machine during load of TEST.

xxx line 488 stack

The process area is smaller than 11000 during translation of TEST.

xxx permanent test protected

xxx set test protected

The process does not own the catalog key 3

xxx clear test protected

The name test is used by another process.

xxx test create <doc name> function not allowed

The process has not the function mask necessary

xxx test create <doc name> not user

The process is not user of <doc name>.

xxx test create <doc name> reserved.

xxx test create <doc name> name not unique.

The device in question is reserved by another process.

xxx Test create <doc name> mounted on station no device no

The document (tape) mt 007 is e.g. mounted on station 8.

xxx test <devicekind> not implemented.

No test for this device-kind, e.g. bs.

xxx test device <number> no <devicekind>

Is a warning about a wrong devicekind, e.g. performing magnetic tape test on the printer. TEST terminates.

10. Program text

```
825 RC 4000 MAGNETIC TAPE TEST
826 RCTS 2.1/1   27.10.70 PEH
827 VERSION        1.2.71
828
828 NAME IN CATALOG: TESTMT;
8
829 BEGIN INTEGER BLOCKSIZE, FILE,FILES,FI,BLOCK,BLOCKS,BL,LASTELEM,
830               TP,M1,LOW_DEN,PROTECT,BY,TYPE,ST,SHARES;
831    BOOLEAN CHECKALL,DATAPRINT,FIND,ERASE,MTE,LÆS,READING,SKRIV,WRITING,
832            MONIT,STATUSCHECK,EOT,Q;
833    INTEGER ARRAY ERRORS(1:DEVS),MON(1:DEVS,1:6);
834    REAL ARRAY TESTDATA(1:16), NAME(1:DEVS);
835    BOOLEAN ARRAY TESTBITS(1:9);
836
836    PROCEDURE GETMODE;
837    BEGIN GETPARITY;
838 L:    WRITE(OUTC,<:<10>MODE = :>); UD; IND;
839       C:=INCHAR;
840       IF C=10 THEN
841       BEGIN INITMODE;
842          GOTO RETURN;
843       END ELSE
844       BEGIN LOW_DEN:=PROTECT:=TYPE:=0;
845          CHECKALL:=DATAPRINT:=ERASE:=FIND:=LÆS:=STATUSCHECK:=
8          MONIT:=SKRIV:=FALSE;
847       END;
848       GOTO N;
849 M:    C:=INCHAR;
850 N:    IF C= 10 THEN GOTO D ELSE
851       IF C= 32 THEN GOTO M ELSE
852       IF C= 98 THEN
853       BEGIN IF TYPE>5 THEN TYPE:=1 ELSE GOTO L;
854       END ELSE
855       IF C= 99 THEN CHECKALL:=STATUSCHECK:=TRUE ELSE
856       IF C=100 THEN DATAPRINT:=TRUE ELSE
857       IF C=101 THEN ERASE:=TRUE ELSE
858       IF C=102 THEN FIND:=TRUE ELSE
859       IF C=108 THEN LOW_DEN:=1 SHIFT 14 ELSE
860       IF C=109 THEN MONIT:=TRUE ELSE
861       IF C=111 THEN
862       BEGIN IF TYPE>5 THEN TYPE:=3 ELSE GOTO L;
863       END ELSE
8         IF C=112 THEN PROTECT:=1 SHIFT 15 ELSE
8         IF C=114 THEN LÆS:=TRUE ELSE
866       IF C=115 THEN STATUSCHECK:=TRUE ELSE
867       IF C=116 THEN TYPE:=IF DKIND(1)=18 THEN 6 ELSE 8 ELSE
868       IF C=119 THEN SKRIV:=TRUE ELSE GOTO L;
869       FOR C:=INCHAR WHILE C<>32 AND C<>10 DO;
870       GOTO N;
871 D:    IF TYPE>0 THEN LASTELEM:=(PACK(TESTDATA,6,6*17,256,TYPE))//6-1
872       ELSE INITDATA;
873       IF LASTELEM<1 THEN GOTO D;
874       PRINTMODE;
875 RETURN:
876    END PROCEDURE GETMODE;
877
877    PROCEDURE INITMODE;
878    BEGIN DATAPRINT:=ERASE:=FIND:=LÆS:=STATUSCHECK:=SKRIV:=MONIT:=TRUE;
879       CHECKALL:=FALSE;
880       PROTECT:=LOW_DEN:=TYPE:=0;
881       INITDATA;
8         PRINTMODE;
8      END;
884
884
884
884 COMMENT
885
```

```
886
886     PROCEDURE PRINTMODE;
887     BEGIN WRITE(OUTL,<:<10><12>MODE =:>,
888        IF PARITY=0 THEN <:_ODD:> ELSE <:_EVEN:>,
8          IF CHECKALL AND LÆS THEN <:_CHECKALL:> ELSE IF STATUSCHECK THEN
89            <:_STATUSCHECK:> ELSE <::>,
891        IF DATAPRINT AND STATUSCHECK AND LÆS THEN <:_DATAPRINT:> ELSE <::>,
892        IF SKRIV AND ERASE THEN <:_WRITE_ERASE:> ELSE IF SKRIV THEN
893           <:_WRITE:> ELSE <::>,
894        IF LÆS AND FIND THEN <:_READ_FIND:> ELSE IF LÆS THEN
895           <:_READ:> ELSE <::>,
896        IF PROTECT >0 THEN <:_PRO:> ELSE <:_WRE:>,
897        IF LOW_DEN >0 THEN <:_LO:> ELSE <:_HI:>,
898        IF MONIT THEN <:_MONITOR:> ELSE <::>,
899        IF TYPE>0 THEN <:_TYPE:> ELSE <::>,IF TYPE=3 THEN <:_OCTAL:>
900        ELSE IF TYPE=1 THEN <:_BINARY:> ELSE <::>,NL,1);
901        UDL;
902     END PROCEDURE PRINTMODE;
903
903     PROCEDURE INITDATA;
904     BEGIN INTEGER J,K,L,M,N;
905        LASTELEM:=16;
906        N:=10;
9          TESTDATA(1):=0,0 SHIFT 24;
908        FOR J:=1, -1 DO FOR K:= -4 STEP 1 UNTIL 4 DO
909        FOR L:=0 STEP (-1) UNTIL (-6) DO
910        BEGIN M:=INCREASE(N)//8;
911           TESTDATA(M):=TESTDATA(M) SHIFT 6 ADD (((TESTBITS(J*K+5)
912           SHIFT L EXTRACT 6)*J-(IF J<0 THEN 1 ELSE 0)) EXTRACT 6);
913           IF PARITY<>0 THEN TESTDATA(1):=TESTDATA(2) SHIFT 1;
914        END;
915     END PROCEDURE INITDATA;
916
916     PROCEDURE EXECUTE(Z,OPERATION,PARAM1,WORDS);
917     ZONE Z; INTEGER OPERATION,PARAM1,WORDS;
918     BEGIN INTEGER BASE;
919        INTEGER ARRAY IA(1:20);
920        GETZONE(Z,IA);
921        BASE:=IA(19);
922        GETSHARE(Z,IA,1);
923        IA(1):=0;
924        IA(4):=OPERATION+(IF OPERATION=8 SHIFT 12 THEN PARITY ELSE 0);
9          IA(5):=IF OPERATION=8 SHIFT 12 OR OPERATION=0
926        THEN PARAM1 ELSE BASE+1;
927        IA(6):=BASE+WORDS*2;
928        IA(12):=IA(6)+1;
929        SETSHARE(Z,IA,1);
930        MONITOR(16,Z,1,IA); COMMENT SEND MESSAGE;
931     END PROCEDURE EXECUTE;
932
932     PROCEDURE CREATE(Z,D);
933     COMMENT  THIS PROCEDURE TRIES TO CREATE A PERIPHERAL PROCESS AND
934        CONNECT IT TO A TAPESTATION. IF SUCCESFUL IT IS TESTED THAT THE
935        STATION IS IN THE REMOTE STATE OTHERWISE THE PERIPHERAL PROCESS
936        IS REMOVED (BY MEANS OF THE SENSE OPERATION). IF REMOTE AND
937        RESERVED THE BOOLEAN RESERVED(D) IS SET TO TRUE OTHERWISE IT IS
938        SET TO FALSE. AFTER RETURN EITHER SETPOSITION, OR CKECK, OR CLOSE,
939        MUST BE CALLED;
940     ZONE Z; INTEGER D;
941     BEGIN INTEGER RESULT, K, I; INTEGER ARRAY IA(1:20);
9          PROCEDURE TESTREMOTE;
9          BEGIN MONITOR(6,Z,I,IA); COMMENT RESERVE DEVICE;
944           EXECUTE(Z,0,0,0); COMMENT SENSE DEVICE;
945           K:=MONITOR(18,Z,1,IA); COMMENT WAIT ANSWER;
946           RESERVED(D):=(K=1 OR K=4);
947        END PROCEDURE TESTREMOTE;
948  COMMENT
```

```
949 RC 4000 MAGNETIC TAPE TEST
950
950 CREATE_START:
951       CLOSE(Z,FALSE);
          IF RESERVED(D) THEN
          OPEN(Z,MODEKIND(DEVICE(D)),STRING NAME(D),M1);
954       TESTREMOTE;
955       IF RESERVED(D) THEN
956       BEGIN K:=MONITOR(4,Z,I,IA);
957          IF K<>DEVICE(D) THEN
958          BEGIN WRITE(OUTL,NL,1,<:***TEST CREATE_:>,STRING NAME(D),
959             <:_MOUNTED ON STATION_:>,<<D>,DEVICENUMBER(K));
960             GOTO TESTEND;
961          END;
962       END ELSE
963       BEGIN I:=DEVICENUMBER(DEVICE(D));
964          RESULT:=MONITOR(54,Z,I,IA);
965          IF RESULT<>0 THEN
966          BEGIN WRITE(OUTC,NL,1,<:***TEST CREATE_:>,
967             STRING NAME(D),CASE RESULT OF (<:_FUNCTION NOT ALLOWED:>,
968             <:_NOT USER:>,<:_NAME NOT UNIQUE:>,<:_PIP4:>,
969             <:_RESERVED:>,<:_PIP6:>));
970             GOTO TESTEND;
          END;
91[illegible]    TESTREMOTE;
973       END;
974       IF RESERVED(D) THEN
975       BEGIN CLOSE(Z,FALSE);
976          OPEN(Z,MODEKIND(DEVICE(D)),STRING NAME(D),M1);
977          EXECUTE(Z,8 SHIFT 12,4,0); COMMENT REWIND DEAD SURE;
978       END;
979       GETZONE(Z,IA);
980       IA(14):=IA(19);
981       IA(16):=IA(20);
982       SETZONE(Z,IA);
983    END PROCEDURE CREATE;
984
984    PROCEDURE MAXBLOCK(SHARES);
985    INTEGER SHARES;
986    BEGIN BLOCKSIZE:=(SYSTEM(2,I,PARAM)-9000)//(SHARES*DEVS*4)*2;
987       IF BLOCKSIZE >0 THEN WRITE(OUTC,NL,1,<:MAX BLOCKSIZE = :>,
988       <<D>,BLOCKSIZE,<:_WORDS:>) ELSE
          BEGIN WRITE(OUTC,NL,1,<:***CORE AREA TOO SMALL:>);
990          GOTO TESTEND;
991       END; UD;
992    END PROCEDURE MAXBLOCK;
993
993    PROCEDURE PRINTBLOCKNO(TEXT,I);
994    STRING TEXT; INTEGER I;
995    BEGIN WRITE(OUTL,<<_DDD>,RUNNO,<:'RUN_:>,TEXT,<<B>,I,<<_D>,
996             <:_BLOCK:>,BLOCK,<:_FILE:>,FILE,NL,1); UDL;
997    END PROCEDURE PRINTBLOCKNO;
998
998    PROCEDURE SUPPRESS(Z,WORDS);
999    COMMENT THIS PROCEDURE CHANGES THE ZONE DESCRIPTOR FOR THE ZONE Z SO
1000       THAT <WORDS> WORDS IN THE LAST PART OF THE BLOCK WILL NOT BE OUTPUT;
1001   VALUE WORDS; ZONE Z; INTEGER WORDS;
1002   BEGIN INTEGER ARRAY IA(1:20);
1003      GETZONE(Z,IA);
1004      IA(14):=IA(14)+WORDS EXTRACT 1*2;
1005      IA(16):=IA(16)-(WORDS+1)//2;
1006      SETZONE(Z,IA);
1007   END PROCEDURE SUPPRESS;
1008
1008 COMMENT
1009
```

```
1010
1010      PROCEDURE ACY(PROGRAM);
1011      COMMENT THIS PROCEDURE CAN PERFORM THE TESTPROGRAMS A, C, AND Y;
1012      INTEGER PROGRAM;
1013      BEGIN ZONE ARRAY MT(DEVS,(BLOCKSIZE+1)//2*2,2,MT_ERROR);
1014         BOOLEAN FM,ERR,SYNC;
1015
1015         PROCEDURE INIT_MT;
1016         BEGIN INTEGER ARRAY IA(1:6);
1017            INTEGER OP;
1018            REAL R;
1019            M1:=IF (WRITING AND (LÆS AND -,FIND OR -,LÆS AND =,ERASE)
1020            OR -,WRITING AND (SKRIV AND -,ERASE OR -,SKRIV AND -,FIND))
1021            THEN (-1) ELSE 13 SHIFT 18+1 SHIFT 16+63;
1022            BLOCK:=FILE:=0;
1023            MTE:=EOT:=ERR:=WRITING:=SYNC:=READING:=FALSE;
1024            FOR D:=1 STEP 1 UNTIL DEVS DO CREATE(MT(D),D);
1025            FOR D:=1 STEP 1 UNTIL DEVS DO
1026            BEGIN SYSTEM(5,DEVICE(D)+28,IA);   læsning af monitor label
1027               IF RUNNO=0 OR -,MONIT THEN
1028               BEGIN FOR I:=1 STEP 1 UNTIL 6 DO MON(D,I):=IA(I);   lokal gemning af monitor label
1029               END ELSE
1030               BEGIN WRITE(OUTL,NL,1,<:*:>,<<DD>,
1031                  DEVICENUMBER(DEVICE(D)),<:_MONITOR:>);
1032                  FOR I:=2,3,1 DO
1033                  BEGIN OP:=IA(I)-MON(D,I);   OP = forskel mellem 2 ord i hhvl af monitorlabel.
1034                     R:=IF OP=0 THEN 0.0 ELSE (IA(I+3)-MON(D,I+3))*100/OP;
1035                     WRITE(OUTL,<:____:>,CASE I OF (<:RE:>,<:WR:>,<:ER:>),
1036                        <<_BD.D00>,R,<:_%:>);
1037                     MON(D,I):=IA(I);
1038                     MON(D,I+3):=IA(I+3);
1039                  END;
1040                  WRITE(OUTL,FALSE ADD 13,1); UDL;
1041               END;
1042            END MONIT;
1043            FOR D:=1 STEP 1 UNTIL DEVS DO CREATE(MT(D),D);
1044            FOR D:=1 STEP 1 UNTIL DEVS DO
1045            BEGIN CLOSE(MT(D),FALSE);
1046               IF RESERVED(D) THEN
1047               BEGIN OPEN(MT(D),MODEKIND(DEVICE(D)),STRING NAME(D),M1);
1048                  SETPOSITION(MT(D),0,0);
1049               END;
1050               ERRORS(D):=0;
1051            END;
1052            MTE:=EOT:=ERR:=FALSE;
1053         END PROCEDURE INIT_MT;
1054
1054         PROCEDURE CHECKDATA(CHECKIDENT);
1055         BOOLEAN CHECKIDENT;
1056         BEGIN BOOLEAN DTE;
1057            INTEGER I,CHAR,BYTES;
1058            REAL R;
1059            PROCEDURE CHECKWORD(REC,EXPD,ELEM);
1060            INTEGER ELEM; REAL REC,EXPD;
1061            IF REC SHIFT (-24)<>EXPD SHIFT (-24) THEN
1062            DATAERROR(REC,EXPD,ELEM,CHAR,-24);
1063
1063            BOOLEAN PROCEDURE SKIP;
1064            SKIP:=-,CHECKALL AND -,(MTE AND DATAPRINT);
1065
1065 COMMENT
1066
```

```
1067
1067
1067          PROCEDURE DATAERROR(RECEIVED,EXPECTED,ELEM,CHAR,BIT);
10            VALUE RECEIVED,EXPECTED,ELEM,CHAR,BIT; REAL RECEIVED,EXPECTED;
1069          INTEGER ELEM,CHAR,BIT;
1070          BEGIN INTEGER I;
1071             IF ~,MTE THEN
1072             BEGIN MTE:=TRUE;
1073                ERRORS(D):=ERRORS(D)+1;
1074                I:=BY;
1075                ERROR(MT(D),ST,I);
1076             END;
1077             IF ~,DTE THEN
1078             BEGIN DTE:=TRUE;
1079                PRINTBLOCKNO(<:_DATAERR:>,0);
1080             END;
1081             IF ~,DATAPRINT THEN GOTO RETURN;
1082             IF BIT<>0 THEN WRITE(OUTL,<:<10>WORD:   :>,ELEM*2-2,<:::>)
1083             ELSE WRITE(OUTL,<:<10>WORDS::>,<<DDDDDD>,ELEM*2-2,<:::>,
1084                        FALSE ADD 32,24-CHAR//2,ELEM*2-1,<:::>);
1085             PRINTBITS(0,RECEIVED,EXPECTED,CHAR,BIT);
1086          END PROCEDURE DATAERROR;
10
1087 CHECKDATA_START:
1088          DTE:=FALSE;
1089          CHAR:=IF DKIND(D)=18 THEN 6 ELSE 8;
1090          R:=IF FM THEN (CASE CHAR//4 OF (0.0 ADD 15 SHIFT 6 ADD 15 SHIFT
1091          6 ADD 15 SHIFT 6 ADD 15 SHIFT 24, 0.0 ADD 19 SHIFT 40)) ELSE
1092          IF CHECKIDENT THEN 0.0 SHIFT 24 ADD BLOCK SHIFT 24
1093          ADD FILE ELSE TESTDATA(1);
1094          IF R<>MT(D,1) AND (~,SKIP OR CHECKIDENT OR FM) THEN
1095          BEGIN IF BY=2 THEN CHECKWORD(MT(D,1),R,1)
1096             ELSE IF BY >2 THEN DATAERROR(MT(D,1),R,1,CHAR,0);
1097             MT(D,1):=0.0 SHIFT 48;
1098          END;
1099          IF SKIP THEN GOTO NDTE;
1100          BYTES:=BY//4-2;
1101          FOR I:=0 STEP 1 UNTIL BYTES DO
1102          BEGIN IF MT(D,I+2)<>TESTDATA(I MOD LASTELEM+1) THEN DATAERROR
1103             (MT(D,I+2),TESTDATA(I MOD LASTELEM+1),I+2,CHAR,0);
11             MT(D,I+2):=0.0 SHIFT 48;
11          END;
1106          IF I*4+4<BY THEN
1107          BEGIN CHECKWORD(MT(D,I+2),TESTDATA(I MOD LASTELEM+1),I+2);
1108             MT(D,I+2):=0.0 SHIFT 48;
1109          END;
1110 NDTE:    IF ~,DTE THEN
1111          BEGIN IF MTE THEN PRINTBLOCKNO(<:_READING:>,0)
1112             ELSE ERRORS(D):=0;
1113          END;
1114 RETURN:  MTE:=FALSE;
1115          UDL;
1116       END PROCEDURE CHECKDATA;
1117
1117 COMMENT
1118
```

```
1119
1119        PROCEDURE MT_ERROR(Z,S,B);
1120        ZONE Z; INTEGER S,B;
1121        BEGIN INTEGER PA,STATUS;
1122           BY:=B;
1123           ST:=S;
1124           IF BLOCK<0 THEN BLOCK:=0;
1125           IF S SHIFT (-18) EXTRACT 1=1 THEN EOT:=TRUE;
1126           IF S SHIFT (-5) EXTRACT 1=1 THEN RESERVED(D) :=FALSE;
1127           IF STATUSCHECK THEN
1128           BEGIN STATUS:=S+M+PROTECT+LOW_DEN+1;
1129              FM:=S SHIFT (-16) EXTRACT 1=1 AND (B=2 OR B=0);
1130              IF (B<>BLOCKSIZE*2 OR STATUS<>0) AND
1131              -,(FM AND S SHIFT (-18) EXTRACT 1=0 OR B<0
1132              OR S SHIFT (-17) EXTRACT 1=1 AND BLOCK=0 AND FILE=0) OR
1133              (FM AND BLOCK<>BLOCKS OR -,FM AND BLOCK=BLOCKS)
1134              AND READING THEN
1135              BEGIN ERROR(Z,S,B);
1136                 ERRORS(D):=ERRORS(D)+1;
1137                 MTE:=TRUE;
1138                 IF WRITING THEN
1139                 BEGIN PRINTBLOCKNO(<:_WRITING:>,0);
1140                    UDL;
1141                 END;
1142              END ELSE IF WRITING THEN ERRORS(D):=0;
1143              IF ERRORS(D)>10 THEN ERR:=TRUE;
1144           END;
1145           IF SYNC THEN BLOCK:=BLOCK+1;
1146           COMMENT SYNCHRONIZE BLOCKNO. AT FILE END;
1147           B:=(BLOCKSIZE+1)//2*4;
1148        END PROCEDURE MT_ERROR;
1149
1149        PROCEDURE BITPATTERN(CHECKIDENT,SKIPREADFM);
1150        COMMENT WRITE AND READ TEST;
1151        VALUE CHECKIDENT,SKIPREADFM; BOOLEAN CHECKIDENT;
1152        INTEGER SKIPREADFM;
1153        BEGIN INTEGER J,OLDBLOCK;
1154           BOOLEAN SC;
1155           IF SKRIV THEN
1156           BEGIN WRITING:=TRUE;
1157              FOR FILE:=0 STEP 1 UNTIL FILES-1 DO
1158              BEGIN FOR BLOCK:= -2 STEP 1 UNTIL BLOCKS-3 DO
1159                 BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D)
1160                    THEN OUTREC(MT(D),0);
1161                    IF EOT OR ERR THEN
1162                    BEGIN BLOCK:=BLOCK+1;
1163                       GOTO TERM;
1164                    END;
1165                    FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
1166                    BEGIN OUTREC(MT(D),(BLOCKSIZE+1)//2);
1167                       IF BLOCK < 0 AND FILE=0 THEN
1168                       BEGIN MT(D,1):=TESTDATA(1);
1169                          FOR I:=0 STEP 1 UNTIL (BLOCKSIZE-1)//2-1 DO
1170                          MT(D,I+2):=TESTDATA(I MOD LASTELEM+1);
1171                       END;
1172                       IF CHECKIDENT THEN MT(D,1):=
1173                            0,0 SHIFT 24 ADD (BLOCK+2) SHIFT 24 ADD FILE;
1174                       SUPPRESS(MT(D),BLOCKSIZE EXTRACT 1);
1175                    END DEVS;
1176                 END BLOCKS;
1177
1177  COMMENT
1178
```

```
1178
1179
1179 TERM:           SYNC:=TRUE;
1180                 OLDBLOCK:=BLOCK;
1181                 FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
1182                 BEGIN BLOCK:=OLDBLOCK;
1183                    SETPOSITION(MT(D),FILE+1,0);
1184                 END;
1185                 SYNC:=FALSE;
1186                 IF EOT THEN GOTO ENDWRITE;
1187                 IF ERR THEN GOTO ERRORRETURN;
1188              END FILES;
1189              SC:=STATUSCHECK;
1190              STATUSCHECK:=FALSE;
1191              FOR J:=1,2 DO FOR D:=1 STEP 1 UNTIL DEVS DO
1192              IF RESERVED(D) THEN
1193              BEGIN OUTREC(MT(D),(BLOCKSIZE+1)//2);
1194                 COMMENT SET BUFFER CONTENTS TO ALL ONES;
1195                 FOR I:=1 STEP 1 UNTIL (BLOCKSIZE+1)//2 DO
1196                 MT(D,I):=0.0 SHIFT 24 ADD (-1) SHIFT 24 ADD (-1);
1197                 SUPPRESS(MT(D),BLOCKSIZE EXTRACT 1);
1198                 IF J=2 THEN SETPOSITION(MT(D),FILE+1,0);
1199              END;
1200              STATUSCHECK:=SC;
1201 ENDWRITE:    IF LÆS THEN INIT_MT;
1202           END WRITE;
1203
1203           IF LÆS THEN
1204           BEGIN READING:=TRUE;
1205              FOR FILE:=0 STEP 1 UNTIL FILES-1 DO
1206              FOR BLOCK:=0 STEP 1 UNTIL BLOCKS-SKIPREADFM DO
1207              BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
1208                 BEGIN INREC(MT(D),(BLOCKSIZE+1)//2);
1209                    IF STATUSCHECK THEN CHECKDATA(CHECKIDENT);
1210                 END DEVS;
1211                 IF ERR THEN GOTO ERRORRETURN;
1212                 IF EOT THEN GOTO RETURN;
1213              END BLOCKS, FILES;
1214           END READ;
1215           GOTO RETURN;
1216 ERRORRETURN:
1217           WRITING:=FALSE;
1218           WRITE(OUTL,<:<10>GIVE UP<10><12>:>); UDL;
1219 RETURN:
1220        END PROCEDURE BITPATTERN;
1221
1221 ACY_START:
1222        FOR D:=1 STEP 1 UNTIL DEVS DO
1223        BEGIN OPEN(MT(D),MODEKIND(DEVICE(D)),STRING NAME(D),M1);
1224           CREATE(MT(D),D);
1225        END;
1226        GOTO CASE PROGRAM OF (A,C,Y,X,END_ACY);
1227
1227 A:     WRITE(OUTC,<:<10>BLOCKS, FILES = :>); UD; IND;
1228        IF INCHAR=10 THEN BLOCKS:=FILES:=20 ELSE
1229        BEGIN REPEATCHAR(INC);
1230           RD(A,BLOCKS); RD(A,FILES);
1231        END;
1232        GETMODE;
1233 A1:    RUNADM(RTP,INIT_MT);
1234        BITPATTERN(BLOCKSIZE>2 AND PARITY=0,0);
1235        GOTO A1;
1236
1236 COMMENT
1237
```

```
1237  RC 4000 MAGNETIC TAPE TEST                                    PAGE 8
1238
1238 C:     BEGIN COMMENT CRC TEST;
1239           INTEGER TRACK;
1240           BOOLEAN MT747;
1241           MT747:=FALSE;
1242           FOR D:=1 STEP 1 UNTIL DEVS DO
1243           BEGIN IF DKIND(D)=18 THEN
1244              BEGIN WRITE(OUTC,NL,1,<:***DEVICE:>,<<_DD>,
1245                 DEVICENUMBER(DEVICE(D)),<:_NO 9-TRACK:>);
1246                 MT747:=TRUE;
1247              END;
1248           END;
1249           IF MT747 THEN GOTO TESTEND;
1250           PARITY:=2;
1251           INITDATA;
1252           TYPE:=PARITY:=PROTECT:=LOW_DEN:=0;
1253 CC:       ERASE:=DATAPRINT:=SKRIV:=STATUSCHECK:=CHECKALL:=MONIT:=TRUE;
1254           LÆS:=FALSE;
1255           RUNADM(RTP,INIT_MT);
1256           BLOCKS:=50;
1257           FILES:=1;
1258           BITPATTERN(FALSE,0);
1259           LÆS:=FIND:=TRUE;
1260           SKRIV:=FALSE;
1261           FOR TRACK:=1 STEP 1 UNTIL 9 DO
1262           BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
1263              SETPOSITION(MT(D),0,0);
1264              WRITE(OUTC,NL,1,
1265              <:REMOVE READAMP. TRACK:>,TRACK); UD; IND;
1266              INCHAR;
1267              INIT_MT;
1268              BITPATTERN(FALSE,1);
1269           END;
1270           GOTO CC;
1271        END CRC TEST;
1272
1272 X:     RUNADM(RTP,INIT_MT);
1273        MONIT:=FALSE;
1274        FOR D:=1 STEP 1 UNTIL DEVS DO TEST_FUNCTIONS;
1275        BLOCKS:=100;
1276        FILES:=100;
1277        PARITY:=0;
1278        INIT_MT;
1279        MONIT:=TRUE;
1280        BITPATTERN(BLOCKSIZE>2,0);
1281        GOTO X;
1282
1282 Y:     FOR D:=1 STEP 1 UNTIL DEVS DO CLOSE(MT(D),TRUE);
1283        COMMENT RELEASE TAPESTATIONS;
1284        GOTO TESTEND;
1285 END_ACY:
1286     END PROCEDURE ACY;
1287
1287     PROCEDURE PRINT_CONTENTS;
1288     BEGIN ZONE ARRAY MT(DEVS,BLOCKSIZE//2,1,M_ERROR);
1289        INTEGER BL,CH,CHAR,HALF,I,J,BLO,FIL;
1290        REAL R;
1291
1291
1292  COMMENT
1293
```

```
1292 RC 4000 MAGNETIC TAPE TEST
1293
1293        PROCEDURE M_ERROR(Z,S,B);
1294        ZONE Z; INTEGER S,B;
1295        BEGIN BY:=B;
1296           ST:=S;
1297           IF B<=0 AND S SHIFT (-16)=0 OR RUNNO=0 THEN GOTO RETURN;
1298           ERROR(Z,S,B);
1299           PRINTBLOCKNO(<:_READING:>,0);
1300           IF ST SHIFT (-16) EXTRACT 1=1 THEN
1301           BEGIN FILE:=FILE+1;
1302              BLOCK:=0;
1303           END ELSE IF B>0 THEN BLOCK:=BLOCK+1;
1304 RETURN:   B:=IF BY<=0 THEN 0 ELSE BLOCKSIZE//2*4;
1305           EOT:= ST SHIFT (-18) EXTRACT 1=1;
1306           IF ST SHIFT (-5) EXTRACT 1=1 THEN GOTO NEXT;
1307        END PROCEDURE M_ERROR;
1308
1308 PRINTCONTENTS_START:
1309        M1:=13 SHIFT 18+1 SHIFT 16+2;
1310        GETPARITY;
1311        FOR D:=1 STEP 1 UNTIL DEVS DO
1312        BEGIN OPEN(MT(D),MODEKIND(DEVICE(D)),STRING NAME(D),M1);
1313           CREATE(MT(D),D);
1314        END;
1315 L:     WRITE(OUTC,NL,1,<:FIRST BLOCK, FILE = :>); UD; IND;
1316        RD(L,BLO); RD(L,FIL);
1317 M:     WRITE(OUTC,NL,1,<:NUMBER OF BLOCKS = :>); UD; IND;
1318        RD(M,BLOCKS);
1319        FOR D:=1 STEP 1 UNTIL DEVS DO CREATE(MT(D),D);
1320        FOR I:=0 STEP 1 UNTIL FIL-1 DO FOR D:=1 STEP 1 UNTIL DEVS DO
1321        IF RESERVED(D) THEN SETPOSITION(MT(D),I,0);
1322        FOR I:=0 STEP 1 UNTIL BLO DO FOR D:=1 STEP 1 UNTIL DEVS DO
1323        IF RESERVED(D) THEN SETPOSITION(MT(D),FIL,I);
1324 NEXT:  WRITE(OUTL,FALSE ADD 12,1);
1325 R:     RUNADM(RTP,UDL);
1326        IF -,RESERVED(D) THEN GOTO R;
1327        FILE:=FIL; BLOCK:=BLO;
1328        CHAR:=IF DKIND(D)=18 THEN 6 ELSE 8;
1329        FOR BL:=1 STEP 1 UNTIL BLOCKS DO
1330        BEGIN INREC(MT(D),BLOCKSIZE//2);
1331           FOR I:=4 STEP 4 UNTIL BY DO
1332           BEGIN R:=MT(D,I//4);
1333              WRITE(OUTL,NL,1,<<DDDDD>,I//2-2,<:,:>,I//2-1,<:  :>);
1334              FOR J:= -47 STEP 1 UNTIL 0 DO WRITE(OUTL,IF R
1335              SHIFT J EXTRACT 1=1 THEN <:1:> ELSE <:.:>,
1336              IF J MOD CHAR=0 THEN <: :> ELSE <::>);
1337              FOR J:= -40 STEP 8 UNTIL 0 DO
1338              BEGIN CH:=R SHIFT J EXTRACT 8;
1339                 IF CH< 32 OR CH >126 THEN CH:=32;
1340                 WRITE(OUTL,FALSE ADD CH,1);
1341              END;
1342           END;
1343           IF I-BY=2 THEN
1344           BEGIN WRITE(OUTL,NL,1,<<DDDDD>,I//2-2,FALSE ADD 32,8);
1345              FOR J:= -47 STEP 1 UNTIL -24 DO WRITE(OUTL,IF
1346              MT(D,I//4) SHIFT J EXTRACT 1=1 THEN <:1:> ELSE <:.:>,
1347              IF J MOD CHAR=0 THEN <: :> ELSE <::>);
1348              FOR J:= -40, -32, -24 DO
1349              BEGIN CH:=MT(D,I//4) SHIFT J EXTRACT 8;
1350                 IF CH<32 OR CH >126 THEN CH:=32;
1351                 WRITE(OUTL,FALSE ADD CH,1);
1352              END;
1353           END;
1354
1354
1354 COMMENT
1355
```

```
1356
1356            WRITE(OUTL,NL,1);
1357            IF EOT THEN GOTO RET;
1358         END BLOCK;
1359      RET:  SETPOSITION(MT(D),FIL,BLO);
1360         GOTO NEXT;
1361      END PROCEDURE PRINT CONTENTS;
1362
1362      PROCEDURE TEST_FUNCTIONS;
1363      BEGIN INTEGER STATE, STATUS, BIT, ERROR, DEV, BY, ERRORS, BLOCKGAP;
1364         INTEGER ARRAY IA(1:20);
1365         REAL STARTDISTANCE, STOPDISTANCE, REVSTOPDISTANCE,
1366         HEADCONST1, HEADCONST2, DENSITY;
1367         REAL ARRAY R(0:5);
1368         PROCEDURE WRITESTATE(TEXT, LAYOUT, NO, REC, EXPD, BITS);
1369         STRING TEXT, LAYOUT; INTEGER REC, NO, EXPD, BITS;
1370         BEGIN IF STATE<>ERROR OR DEV<>D THEN
1371            BEGIN ERROR:=STATE;
1372               DEV:=D;
1373               WRITE(OUTL,NL,1,<:*:>,<<DD>,DEVICENUMBER(DEVICE(D)),
1374               <<__DDD>,RUNNO,<:'RUN,STATE:>,<<_DD>,STATE);
1375            END ELSE WRITE(OUTL,NL,1,FALSE ADD 32,14);
1376            WRITE(OUTL,TEXT,LAYOUT,NO);
1377            PRINTBITS(5,0,0 SHIFT 24 ADD REC SHIFT 24, 0,0 SHIFT 24 ADD
1378            EXPD SHIFT 24, IF DKIND(D)=18 THEN 6 ELSE 8,BITS-48);
1379            UDL;
1380         END PROCEDURE WRITESTATE;
1381
1381         PROCEDURE BLOCKPROC(Z,S,C);
1382         ZONE Z; INTEGER S,C;
1383         BEGIN PROCEDURE CHECKS(STATUS);
1384            VALUE STATUS; INTEGER STATUS;
1385            IF S<>STATUS-LOW_DEN THEN WRITESTATE(<:_STATUSERROR_:>,<<B>,0,S,
1386               STATUS-LOW_DEN,24);
1387
1387            PROCEDURE CHECKC(BYTES);
1388            VALUE BYTES; REAL BYTES;
1389            BEGIN INTEGER CHARS;
1390               CHARS:=BYTES*(IF DKIND(D)=18 THEN 2 ELSE 1.5);
1391               IF C<>CHARS AND CHARS >=0 THEN
1392               WRITESTATE(<:_CHARS. TRANSFERRED_:>,<<B>,0,C,CHARS,0);
1393            END PROCEDURE CHECKC;
1394
1394            IF S SHIFT (-5) EXTRACT 1=1 THEN GOTO END_TF;
1395            IF STATE<4 THEN
1396            BEGIN CHECKS(-M-1);
1397               CHECKC(STATE*2);
1398               GOTO RETURN;
1399            END;
1400            IF STATE<7 THEN
1401            BEGIN CHECKS(IF STATE=6 THEN 9 SHIFT 19-M-1 ELSE 1 SHIFT 22-M-1);
1402               CHECKC(CASE STATE-3 OF (2,4,4));
1403               GOTO RETURN;
1404            END;
1405            IF STATE<12 THEN
1406            BEGIN CHECKS(IF STATE=9 THEN 1 SHIFT 19-M-1 ELSE -M-1);
1407               CHECKC(IF STATE=9 THEN 400 ELSE 800);
1408               GOTO RETURN;
1409            END;
1410            IF STATE<18 THEN
1411            BEGIN CHECKS(CASE STATE-11 OF (1 SHIFT 22,
1412               IF DKIND(D)=18 THEN 0 ELSE 1 SHIFT 22,
1413               1 SHIFT 16,1 SHIFT 16,1 SHIFT 16,1 SHIFT 16)-M-1);
1414               CHECKC(IF DKIND(D)=18 THEN (CASE STATE-11 OF (
1415               1.5,1.5,2,1,2,1))
1416               ELSE (CASE STATE-11 OF (1.333,1.333,0,0,0,0.667)));
1417 COMMENT
```

```
1418
1419
1419              GOTO RETURN;
1420           END;
1421           CHECKS(CASE STATE-17 OF (0,1 SHIFT 17,1 SHIFT 17,0,0,
1422              1 SHIFT 16,0,1 SHIFT 16,0,0,1 SHIFT 16,1 SHIFT 16,
1423              1 SHIFT 17,(IF DKIND(D)=34 THEN 3 ELSE 1) SHIFT 21)-M-1);
1424           CHECKC(CASE STATE-17 OF(-1,0,0,0,800,0,1000,0,0,4,0,0,0,0));
1425  RETURN: UDL;
1426        END PROCEDURE BLOCKPROC;
1427
1427        PROCEDURE CHECKCONT(REC,EXPD,ELEM,WORDS);
1428        VALUE REC, EXPD; REAL REC, EXPD; INTEGER ELEM, WORDS;
1429        BEGIN INTEGER I,R,E;
1430           FOR I:= -24 STEP 24 UNTIL WORDS-2 DO
1431           BEGIN R:=REC SHIFT I EXTRACT 24;
1432              E:=EXPD SHIFT I EXTRACT 24;
1433              IF R<>E THEN
1434              BEGIN ERRORS:=ERRORS+1;
1435                 WRITESTATE(<:_DATAERROR, WORD_:>,
1436                 <<D>,ELEM*2-1+I//24,R,E,24);
1437              END ELSE ERRORS:=0;
1438           END;
1439           IF ERRORS >20 THEN
1440           BEGIN WRITE(OUTL,NL,1,<:GIVE UP:>);
1441              GOTO END_TF;
1442           END;
1443        END;
1444
1444        PROCEDURE CHECKSPEC(Z);
1445        ZONE Z;
1446        COMMENT THIS PROCEDURE WORKS LIKE THE CHECKROUTINE EXCEPT THAT
1447        IT CANNOT GENERATE THE STATUSBITS POSITION ERROR, WORD DEFECT,
1448        AND STOPPED. FURTHER, THE BLOCKPROCEDURE IS CALLED WITH THE NUMBER
1449        OF CHARACTERS TRANSFERRED, INSTEAD AF THE NUMBER OF BYTES;
1450        BEGIN INTEGER RESULT;
1451           RESULT:=MONITOR(18,Z,1,IA);
1452           BLOCKPROC(Z,IA(1)+1 SHIFT RESULT,IA(3));
1453        END PROCEDURE CHECKSPEC;
1454
1454        PROCEDURE WAIT(Z);
1455        ZONE Z;
1456        IF RESERVED(D) THEN MONITOR(18,Z,1,IA);
1457
1457        PROCEDURE REWIND(Z);
1458        ZONE Z;
1459        BEGIN EXECUTE(Z,8 SHIFT 12,4,0);
1460           MONITOR(18,Z,1,IA);
1461        END PROCEDURE REWIND;
1462
1462
1462  FUNCTIONS_START:
1463        BIT:=IF DKIND(D)=18 THEN 6 ELSE 8;
1464        DENSITY:=(IF DKIND(D)<>18
1465           THEN 800 ELSE IF LOW_DEN=0 THEN 556 ELSE 200)/256;
1466        BLOCKGAP:=IF DKIND(D)=18 THEN 192 ELSE 154;
1467        FILE:=ERROR:=DEV:=0;
1468        HEADCONST1:=IF DKIND(D)=18 THEN 250 ELSE 168;
1469        HEADCONST2:=IF DKIND(D)=18 THEN 76 ELSE 38;
1470        IF Q THEN GOTO ST32;
1471        FOR I:=BIT STEP BIT UNTIL 96 DO
1472        R(I//49):=R(I//49) SHIFT BIT ADD (I//BIT);
1473
1473
1473
1473  COMMENT
1474
```

```
1474 RC 4000 MAGNETIC TAPE TEST
1475
1475        BEGIN ZONE MT(2,1,BLOCKPROC);
1476           MT(1):=R(0);
1477           MT(2):=R(1);
1478           RESERVED(D):=TRUE;
1479           CREATE(MT,D);
1480           WAIT(MT); COMMENT TERMINATE REWIND;
1481 FOR STATE:= 1, 2, 3 DO
1482           BEGIN PARITY:=(STATE+1) EXTRACT 1*2;
1483              EXECUTE(MT,5 SHIFT 12+PARITY,0,STATE); COMMENT WRITE;
1484              CHECKSPEC(MT); COMMENT WAIT ANSWER AND CHECK TRANSFER;
1485           END;
1486           REWIND(MT);
1487        END;
1488
1488        BEGIN ZONE MT(1,1,BLOCKPROC);
1489           CREATE(MT,D);
1490           WAIT(MT);
1491 FOR STATE:=4,5,6 DO
1492           BEGIN PARITY:=(STATE+1) EXTRACT 1*2;
1493              EXECUTE(MT,3 SHIFT 12+PARITY,0,2); COMMENT READ;
1494              CHECKSPEC(MT);
1495              ERRORS:=0;
1496              CHECKCONT(MT(1),R(0),1,(STATE-1)//2);
1497           END;
1498        END;
1499
1499        BEGIN ZONE MT(200,1,BLOCKPROC);
1500           CREATE(MT,D);
1501           WAIT(MT);
1502 STATE:=7;
1503           FOR I:=1 STEP 1 UNTIL 200 DO
1504           MT(I):=0,0 SHIFT 24 ADD (I*2-2) SHIFT 24 ADD (I*2-1);
1505           EXECUTE(MT,5 SHIFT 12,0,400);
1506           CHECKSPEC(MT);
1507           REWIND(MT);
1508 STATE:=8;
1509           EXECUTE(MT,3 SHIFT 12,0,400);
1510           CHECKSPEC(MT);
1511           REWIND(MT);
1512        END;
1513        BEGIN ZONE MT(100,1,BLOCKPROC);
1514 STATE:=9;
1515           CREATE(MT,D);
1516           WAIT(MT);
1517           EXECUTE(MT,3 SHIFT 12,0,200);
1518           CHECKSPEC(MT);
1519 STATE:=10;
1520           EXECUTE(MT,6 SHIFT 12,0,0); COMMENT ERASE;
1521           WAIT(MT);
1522           REWIND(MT);
1523        END;
1524        BEGIN ZONE MT(300,1,BLOCKPROC);
1525 STATE:=11;
1526           CREATE(MT,D);
1527           WAIT(MT);
1528           EXECUTE(MT,3 SHIFT 12,0,600);
1529           CHECKSPEC(MT);
1530           REWIND(MT);
1531        END;
1532
1533        BEGIN ZONE MT(1,1,BLOCKPROC);
1533           CREATE(MT,D);
1534           WAIT(MT);
1535
1535
1535 COMMENT
```

```
1536 RC 4000 MAGNETIC TAPE TEST
1537
1537 STATE:=12;
1538           R(1):=IF DKIND(D)=18 THEN (0,0 ADD 3 SHIFT 6 ADD 2 SHIFT 6 ADD
1539            1 SHIFT 30) ELSE (0,0 ADD 2 SHIFT 8 ADD 1 SHIFT 32);
1540           MT(1):=R(1);
1541           EXECUTE(MT,5 SHIFT 12+2,0,2);
1542           CHECKSPEC(MT);
1543           REWIND(MT);
1544 STATE:=13;
1545           EXECUTE(MT,3 SHIFT 12+2,0,2);
1546           CHECKSPEC(MT);
1547           CHECKCONT(MT(1),R(1),1,1);
1548           REWIND(MT);
1549
1549 STATE:=14;
1550           IF DKIND(D)=18 THEN
1551           BEGIN R(1):=0,0 ADD 15 SHIFT 6 ADD 15 SHIFT 6 ADD 15
1552              SHIFT 6 ADD 15 SHIFT 24;
1553              MT(1):=R(1);
1554              EXECUTE(MT,5 SHIFT 12,0,1); COMMENT WRITE IN ODD;
1555              CHECKSPEC(MT);
1556 STATE:=15;   MT(1):=R(1) SHIFT (-36) SHIFT 36;
1557              EXECUTE(MT,5 SHIFT 12+2,0,1); COMMENT WRITE IN EVEN;
1558              CHECKSPEC(MT);
1559              REWIND(MT);
1560 STATE:=16;   EXECUTE(MT,3 SHIFT 12,0,2); COMMENT READ 2 WORDS;
1561              CHECKSPEC(MT);
1562              CHECKCONT(MT(1),R(1),1,1);
1563 STATE:=17;   EXECUTE(MT,3 SHIFT 12,0,2); COMMENT READ 2 WORDS;
1564              CHECKSPEC(MT);
1565              CHECKCONT(MT(1),R(1) SHIFT (-36) SHIFT 36,1,1);
1566           END ELSE
1567           BEGIN EXECUTE(MT,10 SHIFT 12,0,0);
1568              CHECKSPEC(MT);
1569              REWIND(MT);
1570 STATE:=17;   EXECUTE(MT,3 SHIFT 12,0,2);
1571              CHECKSPEC(MT);
1572           END;
1573           REWIND(MT);
1574        END;
1575        BEGIN ZONE MT(250,1,BLOCKPROC);
1576           CREATE(MT,D);
1577           WAIT(MT);
1578 STATE:=18;
1579           MT(1):=IF DKIND(D)=18 THEN (0,0 ADD 1 SHIFT 6 ADD 1 SHIFT 6
1580           ADD 1 SHIFT 6 ADD 1 SHIFT 24) ELSE (0,0 ADD 1 SHIFT 8 ADD 1
1581           SHIFT 8 ADD 1 SHIFT 24);
1582           EXECUTE(MT,5 SHIFT 12,0,1); COMMENT WRITE IN ODD;
1583           CHECKSPEC(MT);
1584           R(2):=IF DKIND(D)=18 THEN (0,0 ADD 2 SHIFT 6 ADD 2 SHIFT 6
1585           ADD 2 SHIFT 6 ADD 2 SHIFT 6 ADD 2 SHIFT 6 ADD 2 SHIFT 6
1586           ADD 2 SHIFT 6 ADD 2) ELSE (0,0 ADD 2 SHIFT 8 ADD 2 SHIFT 8
1587           ADD 2 SHIFT 8 ADD 2 SHIFT 8 ADD 2 SHIFT 8 ADD 2);
1588           FOR I:=1 STEP 1 UNTIL 250 DO MT(I):=R(2);
1589           EXECUTE(MT,5 SHIFT 12+2,0,400); COMMENT WRITE IN EVEN;
1590           CHECKSPEC(MT);
1591           R(3):=IF DKIND(D)=18 THEN (0,0 ADD 3 SHIFT 6 ADD 3
1592           SHIFT 6 ADD 3 SHIFT 6 ADD 3 SHIFT 6 ADD 3 SHIFT 6 ADD 3
1593           SHIFT 6 ADD 3 SHIFT 6 ADD 3) ELSE (0,0 ADD 3 SHIFT 8 ADD 3
1594           SHIFT 8 ADD 3 SHIFT 8 ADD 3 SHIFT 8 ADD 3 SHIFT 8 ADD 3 );
1595           MT(1):=R(3);
1596           EXECUTE(MT,5 SHIFT 12,0,2); COMMENT WRITE IN ODD;
1597           CHECKSPEC(MT);
1598
1598
1598 COMMENT
1599
```

```
1600
1600          R(4):=0.0 ADD 15 SHIFT 6 ADD 15 SHIFT 6
1601          ADD 15 SHIFT 6 ADD 15 SHIFT 24;
1602          MT(1):=R(4);
1603          IF DKIND(D)<>18 THEN EXECUTE(MT,10 SHIFT 12,0,0) ELSE
1604          EXECUTE(MT,5 SHIFT 12+2,0,1); COMMENT WRITE TAPEMARK;
1605          WAIT(MT);
1606          R(5):=IF DKIND(D)=18 THEN (0.0 ADD 5 SHIFT 6 ADD 5 SHIFT 6
1607          ADD 5 SHIFT 6 ADD 5 SHIFT 6 ADD 5 SHIFT 6 ADD 5 SHIFT 6
1608          ADD 5 SHIFT 6 ADD 5) ELSE (0.0 ADD 5 SHIFT 8 ADD 5 SHIFT 8
1609          ADD 5 SHIFT 8 ADD 5 SHIFT 8 ADD 5 SHIFT 8 ADD 5);
1610          FOR I:=1 STEP 1 UNTIL 250 DO MT(I):=R(5);
1611          EXECUTE(MT,5 SHIFT 12+2,0,500); COMMENT WRITE IN EVEN;
1612          CHECKSPEC(MT);
1613 STATE:=19;
1614          EXECUTE(MT,8 SHIFT 12,4,0); COMMENT REWIND;
1615          CHECKSPEC(MT);
1616
1616 STATE:=20;
1617          EXECUTE(MT,8 SHIFT 12,3,0); COMMENT BACKSPACE_BLOCK;
1618          CHECKSPEC(MT);
1619          CHECKCONT(MT(1),R(5),1,2);
1620 STATE:=21;
1621          EXECUTE(MT,8 SHIFT 12,1,0); COMMENT UPSPACE_BLOCK;
1622          CHECKSPEC(MT);
1623          CHECKCONT(MT(1),R(5),1,2);
1624 STATE:=22;
1625          EXECUTE(MT,3 SHIFT 12+2,0,400);
1626          CHECKSPEC(MT);
1627          ERRORS:=0;
1628          FOR I:=1 STEP 1 UNTIL 200 DO CHECKCONT(MT(I),R(2),I,2);
1629 STATE:=23;
1630          EXECUTE(MT,8 SHIFT 12,0,0); COMMENT UPSPACE_FILE;
1631          CHECKSPEC(MT);
1632          CHECKCONT(MT(1),R(2),1,2);
1633 STATE:=24;
1634          EXECUTE(MT,3 SHIFT 12+2,0,500); COMMENT READ 500 WORDS;
1635          CHECKSPEC(MT);
1636          ERRORS:=0;
1637          FOR I:=1 STEP 1 UNTIL 250 DO CHECKCONT(MT(I),R(5),I,2);
1638 STATE:=25;
1639          EXECUTE(MT,8 SHIFT 12,2,0); COMMENT BACKSPACE_FILE;
1640          CHECKSPEC(MT);
1641          CHECKCONT(MT(1),R(5),1,2);
1642 STATE:=26;
1643          EXECUTE(MT,8 SHIFT 12,3,0); COMMENT BACKSPACE_BLOCK;
1644          CHECKSPEC(MT);
1645          CHECKCONT(MT(1),R(5),1,2);
1646 STATE:=27;
1647          EXECUTE(MT,3 SHIFT 12,0,2); COMMENT READ 2 WORDS;
1648          CHECKSPEC(MT);
1649          ERRORS:=0;
1650          CHECKCONT(MT(1),R(3),1,2);
1651          FOR I:=1,2,3 DO
1652          BEGIN
1653 STATE:=27+I;
1654             EXECUTE(MT,8 SHIFT 12,CASE I OF (1,3,2),0);
1655             COMMENT UPSPACE OR BACKSPACE BLOCK, BACKSPACE FILE;
1656             CHECKSPEC(MT);
1657             CHECKCONT(MT(1),R(3),1,2);
1658          END;
1659
1659
1659
1659 COMMENT
1660
```

```
1661
1661 STATE:=31;
1662          REWIND(MT);
1663          FOR I:=1 STEP 1 UNTIL 80 DO
1664          BEGIN EXECUTE(MT,6 SHIFT 12,0,0); COMMENT ERASE;
1665             WAIT(MT); COMMENT WAIT ANSWER;
1666          END;
1667          EXECUTE(MT,10 SHIFT 12,0,0); COMMENT WRITE TAPEMARK;
1668          WAIT(MT);
1669          REWIND(MT);
1670          EXECUTE(MT,3 SHIFT 12,0,0);
1671          CHECKSPEC(MT);
1672          REWIND(MT);
1673       END;
1674 ST32:
1675       BEGIN ZONE MT(300,1,BLOCKPROC);
1676 STATE:=32;
1677          FOR I:=1 STEP 1 UNTIL 300 DO
1678          MT(I):= 0.0 SHIFT 24 ADD (-1) SHIFT 24 ADD (-1);
1679          CREATE(MT,D);
1680          WAIT(MT);
1681          EXECUTE(MT,5 SHIFT 12,0,40); COMMENT WRITE 40 WORDS;
1682          WAIT(MT); COMMENT WAIT ANSWER;
1683          EXECUTE(MT,5 SHIFT 12,0,600); COMMENT WRITE 600 WORDS;
1684          WAIT(MT);
1685          REWIND(MT);
1686          EXECUTE(MT,8 SHIFT 12,1,0); COMMENT UPSPACE BLOCK;
1687          WAIT(MT);
1688          EXECUTE(MT,5 SHIFT 12,0,1); COMMENT WRITE ONE WORD;
1689          WAIT(MT);
1690          EXECUTE(MT,3 SHIFT 12,0,600); COMMENT READ DEFECT BLOCK;
1691          WAIT(MT); COMMENT NOW IS IA(3)=NUMBER OF CHARACTERS READ;
1692          EXECUTE(MT,8 SHIFT 12,4,0); COMMENT REWIND;
1693          IF IA(3)< 13 THEN
1694          BEGIN
1695 NOISE:      WRITESTATE(<:_NOISEBLOCK, NO. OF CHARS.::>,
1696             <<B>,0,IA(3),IF DKIND(D)=18 THEN 1760 ELSE 880,0);
1697             GOTO END_TF;
1698          END;
1699          STOPDISTANCE:=(14400/BIT-IA(3))/DENSITY-HEADCONST1-9;
1700          COMMENT 14400=NUMBER OF BITS (PARITY BITS NOT INCLUDED) IN THE
1701          ORIGINAL 600-WORD BLOCK, BIT=6 FOR 7-TRACK AND 8 FOR 9-TRACK;
1702          IF STOPDISTANCE< 25 OR STOPDISTANCE >60 OR Q THEN WRITESTATE
1703          (<:_STOPDISTANCE (DMM):>,<<B>,0,ROUND STOPDISTANCE,49,0);
1704 STATE:=33;
1705          WAIT(MT);
1706          EXECUTE(MT,8 SHIFT 12,1,0); COMMENT UPSPACE BLOCK;
1707          WAIT(MT);
1708          EXECUTE(MT,8 SHIFT 12,1,0);
1709          WAIT(MT);
1710          EXECUTE(MT,5 SHIFT 12,0,1); COMMENT WRITE ONE WORD;
1711          WAIT(MT);
1712          EXECUTE(MT,3 SHIFT 12,0,600); COMMENT READ DEFECT BLOCK;
1713          WAIT(MT);
1714          EXECUTE(MT,8 SHIFT 12,4,0);
1715          IF IA(3)< 13 THEN GOTO NOISE;
1716          STARTDISTANCE:=(14400/BIT-IA(3))/DENSITY-STOPDISTANCE*2
1717          -HEADCONST1-HEADCONST2-18;
1718          IF STARTDISTANCE< BLOCKGAP-STOPDISTANCE-HEADCONST2+5 OR
1719          STARTDISTANCE >BLOCKGAP-STOPDISTANCE-HEADCONST2+25 OR Q THEN
1720          WRITESTATE(<:_STARTDISTANCE (DMM):>,<<B>,0,ROUND
1721          STARTDISTANCE,ROUND(BLOCKGAP-STOPDISTANCE-HEADCONST2+5),0);
1722
1722
1722 COMMENT
1723
```

```
1723 RC 4000 MAGNETIC TAPE TEST
1724
1724 STATE:=34;
1725           WAIT(MT);
1726           EXECUTE(MT,8 SHIFT 12,1,0); COMMENT UPSPACE BLOCK;
1727           WAIT(MT);
1728           EXECUTE(MT,5 SHIFT 12,0,600); COMMENT WRITE 600 WORDS;
1729           WAIT(MT);
1730           EXECUTE(MT,8 SHIFT 12,3,0); COMMENT BACKSPACE BLOCK;
1731           WAIT(MT);
1732           EXECUTE(MT,5 SHIFT 12,0,1); COMMENT WRITE ONE WORD;
1733           WAIT(MT);
1734           EXECUTE(MT,3 SHIFT 12,0,600); COMMENT READ DEFECT BLOCK;
1735           WAIT(MT);
1736           EXECUTE(MT,8 SHIFT 12,4,0);
1737           IF IA(3)< 13 THEN GOTO NOISE;
1738
1738           REVSTOPDISTANCE:=STARTDISTANCE+STOPDISTANCE-(
1739           (14400/BIT-IA(3))/DENSITY-HEADCONST1-9)+HEADCONST2;
1740           IF REVSTOPDISTANCE< STARTDISTANCE+HEADCONST2-25 OR
1741           REVSTOPDISTANCE >STARTDISTANCE+HEADCONST2-5 OR Q THEN
1742           WRITESTATE(<:_REV.STOPDISTANCE (DMM):>,<<B>,0,ROUND
1743           REVSTOPDISTANCE,ROUND(STARTDISTANCE+HEADCONST2-5),0);
1744       END;
1745 END_TF:
1746    END PROCEDURE TEST_FUNCTIONS;
1747
1747
1747
1747 MTTEST_START:
1748    SYSTEM(5,SYSTEM(6,I,PARAM),IA);
1749    I:=IA(13) SHIFT (-12) EXTRACT 12;
1750    IF I< (DEVS+1)*2 THEN
1751    BEGIN WRITE(OUTC,NL,1,<:***BUFFER CLAIM_:>,<<D>,(DEVS+2)*2);
1752       GOTO TESTEND;
1753    END;
1754    FOR I:=1 STEP 1 UNTIL 9 DO TESTBITS(I):=FALSE ADD (CASE I OF (
1755    1 SHIFT 6, 3 SHIFT 5, 1 SHIFT 11+5 SHIFT 4, 1 SHIFT 10+9 SHIFT 3,
1756    1 SHIFT 11+7 SHIFT 4, 3 SHIFT 10+11 SHIFT 3, 1 SHIFT 10+13 SHIFT 3,
1757    3 SHIFT 9+1 SHIFT 6+3 SHIFT 2, 5 SHIFT 9+21 SHIFT 2));
1758
1758    WRITE(OUTC,<:<10>RC 747/749 MAGTAPE<10>:>);
1759    FOR D:=1 STEP 1 UNTIL DEVS DO
1760    BEGIN I:=DEVICENUMBER(DEVICE(D));
1761       NAME(D):=REAL(<:MT000:>) ADD ((I//100) SHIFT 8 ADD
1762       ((I MOD 100)//10) SHIFT 8 ADD
1763       ((I MOD 100) MOD 10) SHIFT 8);
1764    END;
1765
1765
1765 RTP:
1766    PARITY:=0;
1767    BLOCKSIZE:=8;
1768    Q:=FALSE;
1769    TP:=TESTPROG;
1770    GOTO CASE TP OF (DIRECTORY,A,B,C,D,E,RTP,RTP,RTP,RTP,RTP,RTP,RTP,
1771         RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,X,Y);
1772
1772
1772 COMMENT
1773
```

```
1773
1774
1774
1774 A:E:
1775     SHARES:=IF TP=2 THEN 2 ELSE 1;
1776     WRITE(OUTC,NL,1,<:BLOCKSIZE = :>); UD; IND;
1777     IF INCHAR=10 THEN MAXBLOCK(SHARES) ELSE
1778     BEGIN REPEATCHAR(INC);
1779        RD(A,BLOCKSIZE);
1780        IF BLOCKSIZE< 1 THEN GOTO A;
1781        IF BLOCKSIZE >(SYSTEM(2,I,PARAM)-9000)//(SHARES*DEVS*2) THEN
1782        MAXBLOCK(SHARES);
1783     END;
1784     IF TP=6 THEN GOTO EE;
1785     ACY(1);
1786
1786 B: MAXBLOCK(1);
1787     PRINT_CONTENTS;
1788
1788 C:Y:
1789     ACY(ROUND SIGN(TP-4)+2);
1790
1790 D: WRITE(OUTC,NL,1,<:DENSITY = :>); UD; IND;
1791     I:=INCHAR;
1792     LOW_DEN:=0;
1793     IF I=108 THEN LOW_DEN:= 1 SHIFT 14 ELSE IF I=113 THEN
1794     Q:=TRUE ELSE IF I<>104 AND I<>10 THEN GOTO D;
1795     ACY(5); COMMENT REWIND TAPES;
1796     WRITE(OUTL,FALSE ADD 12,1);
1797 DD:RUNADM(RTP,UDL);
1798     TEST_FUNCTIONS;
1799     GOTO DD;
1800
1800 X: MAXBLOCK(2);
1801     INITMODE;
1802     ACY(4);
1803
1803 DIRECTORY:
1804
1804     WRITE(OUTC,<:
1805     A    READ AND WRITE
1806     B    PRINT CONTENTS
1807     C    CRC TEST
1808     D    TEST OF FUNCTIONS
1809     E    OPERATIONS
1810     X    ROUTINE TEST
1811     Y    RELEASE AND TERMINATE
1812
1812 :>);  GOTO RTP;
1813
1813 EE:
1814     BEGIN COMMENT OPERATIONS;
1815        INTEGER RESULT, BUSYDEVS, STOPCOUNT, BUF;
1816        ZONE TIME(1,1,STDERROR);
1817        ZONE ARRAY MT(DEVS,(BLOCKSIZE+1)//2,1,BLOCKPROC);
1818        INTEGER ARRAY OP(1:11), IA(1:20), BUSY(1:DEVS);
1819        BOOLEAN INPUT;
1820        BOOLEAN ARRAY RWDG(1:DEVS), REMOTE(1:DEVS);
1821
1821
1822
1822 COMMENT
1822
```

```
1822
1823        PROCEDURE BLOCKPROC(Z,S,B);
1824        ZONE Z; INTEGER S,B;
1825        IF B<0 THEN B:=0;
1826
1827        PROCEDURE SETDATA(D);
1828        VALUE D; INTEGER D;
1829        BEGIN INTEGER I,J,K,L,M,O,BITS,CHARS;
1830           BITS:=IF DKIND(D)=18 THEN 6 ELSE 8;
1831           CHARS:=24//BITS;
1832           FOR I:=1 STEP 1 UNTIL CHARS DO
1833           BEGIN K:=K SHIFT BITS ADD CHARS;
1834              L:=L SHIFT BITS ADD I;
1835           END;
1836           O:=L;
1837           M:=IF DKIND(D)=18 THEN 16 ELSE 85;
1838           FOR I:=1 STEP 1 UNTIL BLOCKSIZE DO
1839           BEGIN J:= (I+1) SHIFT (-1);
1840              MT(D,J):=MT(D,J) SHIFT 24 ADD O;
1841              O:=O+K;
1842              IF I MOD M=0 THEN
1843              BEGIN IF DKIND(D)=18 THEN
1844                 BEGIN O:=MT(D,J) EXTRACT 24-64;
1845                    MT(D,J):=MT(D,J) SHIFT (-24) SHIFT 24 ADD O;
1846                 END;
1847                 O:=L;
1848              END;
1849           END;
1850        END PROCEDURE SETDATA;
1850
1850
1850        M1:= -1;
1851        OPEN(TIME,0,<:CLOCK:>,0);
1852        STOPCOUNT:=BUSYDEVS:=DEVS;
1853        INPUT:=FALSE;
1854        FOR D:=1 STEP 1 UNTIL DEVS DO
1855        BEGIN OPEN(MT(D),MODEKIND(DEVICE(D)),STRING NAME(D),M1);
1856           CREATE(MT(D),D);
1857           IF -,RESERVED(D) THEN EXECUTE(MT(D),8 SHIFT 12,4,0);
1858           COMMENT REWIND;
1859           RWDG(D):=RESERVED(D):=REMOTE(D):=TRUE;
1860           BUSY(D):=-1000;
1861        END;
1862        GETPARITY;
1863  SYNTAX:
1864        WRITE(OUTC,NL,1,<:OPERATIONS = :>); UD; IND;
1865        FOR I:=1,2,3,4,5,6,7,8,9,10,11 DO
1866        BEGIN OP(I):=INCHAR-48;
1867           IF OP(I) >6 THEN
1868           BEGIN IF OP(I)=57 OR OP(I)=66 THEN
1869              BEGIN OP(I):=6;
1870                 INPUT:=TRUE;
1871              END ELSE
1872              IF OP(I)=63 OR OP(I)=71 THEN OP(I):=7 ELSE
1873              IF OP(I)=68 THEN OP(I):=8 ELSE
1874              IF OP(I)=53 THEN OP(I):=9 ELSE
1875              IF OP(I)=67 THEN OP(I):=10 ELSE GOTO SYNTAX;
1876           END ELSE IF OP(I)< 0 THEN GOTO ENDTYPEOP;
1877        END;
1878  ENDTYPEOP:
1879        IF OP(I)<>(-38) OR I=1 THEN GOTO SYNTAX;
1880        FOR D:=1 STEP 1 UNTIL DEVS DO SETDATA(D);
1881        I:=I-2;
1882        EXECUTE(TIME,0,4,0); COMMENT START CLOCK;
1883        GOTO E2;
1884
1884  COMMENT
1885
```

```
1886
1886 E1:    RUNADM(RTP,UDL);
1887        I:=0;
1888 E2:    FOR I:=I+1 WHILE OP(I)>=0 DO
1889        BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO
1890           IF RESERVED(D) AND ~,RWDG(D) THEN
1891           BEGIN IF OP(I)=7 AND INPUT THEN SETDATA(D);
1892              COMMENT IF OPERATION=WRITE THEN INITIALIZE DATA;
1893              EXECUTE(MT(D),IF OP(I)< 6 THEN 8 SHIFT 12 ELSE
1894              (CASE (OP(I)-5) OF (3,5,10,6,0)) SHIFT 12 ADD PARITY,
1895              OP(I),BLOCKSIZE);
1896              STOPCOUNT:=STOPCOUNT+1;
1897              BUSY(D):=0;
1898           END;
1899 WAIT:     BUF:=0;
1900           FOR RESULT:=MONITOR(24,IN,BUF,IA) WHILE RESULT=0 DO;
1901           FOR D:=1 STEP 1 UNTIL DEVS DO
1902           BEGIN GETSHARE(MT(D),IA,1);
1903              IF IA(1)=BUF THEN
1904              BEGIN STOPCOUNT:=STOPCOUNT-1;
1905                 IF RWDG(D) THEN
1906                 BEGIN RWDG(D):=FALSE;
1907                    BUSYDEVS:=BUSYDEVS-1;
1908                 END;
1909                 IF ~,RESERVED(D) THEN
1910                 BEGIN RESERVED(D):=TRUE;
1911                    BUSYDEVS:=BUSYDEVS-1;
1912                 END;
1913                 BUSY(D):= -1000;
1914                 IF MONITOR(18,MT(D),1,IA)=5 AND ~,REMOTE(D) THEN
1915                 BEGIN BUF:=DEVICENUMBER(DEVICE(D));
1916                    MONITOR(54,MT(D),BUF,IA); COMMENT CREATE PROCESS;
1917                    MONITOR(6,MT(D),BUF,IA);
1918                    REMOTE(D):=TRUE;
1919                 END ELSE IF IA(1) SHIFT (-18) EXTRACT 1=1 THEN
1920                 BEGIN EXECUTE(MT(D),8 SHIFT 12,4,0); COMMENT REWIND;
1921                    RWDG(D):=TRUE;
1922                    BUSYDEVS:=BUSYDEVS+1;
1923                    STOPCOUNT:=STOPCOUNT+1;
1924                    WRITE(OUTL,NL,1,<:*:>,<<DD>,DEVICENUMBER(DEVICE(D)),
1925                    <:___END OF TAPE:>);
1926                 END;
1927                 GOTO NEXTANSWER;
1928              END;
1929           END;
1930           MONITOR(18,TIME,1,IA); COMMENT WAIT CLOCK;
1931           EXECUTE(TIME,0,2,0); COMMENT START CLOCK;
1932           FOR D:=1 STEP 1 UNTIL DEVS DO
1933           BEGIN BUSY(D):=BUSY(D)+2;
1934              IF BUSY(D) MOD 10=0 AND BUSY(D) >0 AND ~,RWDG(D) THEN
1935              BEGIN WRITE(OUTL,NL,1,<:*:>,<<DD>,DEVICENUMBER(DEVICE(D)),
1936                 <:___BUSY FOR_:>,<<DDD>,BUSY(D),<:_SECONDS:>);
1937                 RESERVED(D):=FALSE;
1938                 IF BUSY(D)< 15 THEN BUSYDEVS:=BUSYDEVS+1;
1939              END;
1940              REMOTE(D):=REMOTE(D) SHIFT (-11//DEVS-1);
1941           END;
1942 NEXTANSWER: UDL;
1943           IF STOPCOUNT >BUSYDEVS OR BUSYDEVS=DEVS THEN GOTO WAIT;
1944        END;
1945        D:=DEVS;
1946        GOTO E1;
1947     END OPERATIONS;
1948 END TESTMT;
1949
1949
```

RCSL: 44-D14
Author: Per Hansen
Edition: 1.8.71
Type: FP-utility program

RC4000

TIMESHARED TESTPROGRAM LIBRARY

SLADREHANK

ITR

RESET

CLEARTMX

CLEANTMX

Keywords: RC4000, diagnostic program, fileprocessor utility program, binary tape

---

ABSTRACT: The program is used to check the communication with the RC4000 peripheral devices. It operates essentially on the interrupts and delivers the statusword or the busy/disconnected state. The program needs a minimum core area of 5200 bytes.

A/S REGNECENTRALEN
Falkoner Alle 1
2000 Copenhagen F

CONTENTS:

1. Sladrehank

1.1 Method

The program consists mainly of two parts, one operating in disabled mode, is activated on any interrupt and if the interrupt channel is under supervision the statusword from a device is sensed and buffered in a cyclic buffer of 256 elements. More channels may be supervised, in this case each channel has a buffer.

The other part of the program is executed in enabled mode, i.e. in normal timeslices. Every second the contents of the cyclical buffer(s) are investegated, and if some statuswords have arrived they are printed on current output as explained in 1.5.

1.2 Call of program

The program occupies 5 segments which are stored on the backing storage described by the catalog name sla, which is protected with catalog key 3.

The program must be run in a process area of minimum 5200 bytes (depending of the number of devices supervised) and protection key=0.

The program is called in this way:

$$\text{sla} \left\{ \begin{array}{l} <\text{SP}><\text{deviceparam}> \\ <\text{SP}><\text{modeparam}> \end{array} \right\}_{0}^{\infty}$$

$$<\text{deviceparam}> ::= <\text{deviceno}>.\ <\text{channelno}> \left\{ .\ <\text{skipchar}> \right\}_{0}^{5}$$

$$<\text{modeparam}> ::= \left\{ \begin{array}{l} \text{su} \left\{ .\ <\text{bit}> \right\}_{1}^{\infty} \\ \text{se}.\ <\text{sensings}> \end{array} \right.$$

<deviceno> is the number of the device from which the statusword is sensed.

<channelno> is the number of the interrupt which initiates the sensing.

<skipchar> specifies up to 5 values of the rightmost 12 bits of the statusword (regarded as a positive integer) which should be suppressed in the output. If the status is from the timer (device 3) or the character-counter from a tapestation the value applies for all 24 bits.

< bit > specifies bitnumbers, (0-23), which should be suppressed in the status printing.
< sensing > is the number of senseoperations executed for every interrupt. Up to 2047 sensings may be specified.

If these syntical rules are not followed, the message

xxxcall

is output, and the program terminates.

If no parameters are specified, <deviceno> is initialized to 2 (the main console), <channelno> is initialized to the number belonging to this, and <skipchar> is set to zero, i.e. if the statusword is zero nothing will be printed.

1.3 Interrupt response characteristics.

The part of the code executed in disabled mode is entered any time an interrupt occurs. If the interrupt is uninteresting, it is passed on to the monitor with a delay of only 15 microseconds.

If the interruptnumber is one of those specified in the call of the program, the corresponding device is sensed after a time of 50 microseconds, and the status is stored in an internal buffer. If more sensings were specified, the second sense will take place 126 microseconds after the first one, and the following ones will be executed every 56 microseconds, i.e. with a frequency of 18 k c/s. For every sensing the statusword is compared with the first one, and if they are equal the next sense is executed. Otherwise the statusword is stored together with a flag bit indicating 'bus noise'.

If bus-noise occurs, there will be 98 microseconds between the sensings. If exceptions, i.e. busy or disconnected occur, the inter-sense time will be increased with 20 microseconds.

If 10 sensings are specified, the interrupt will thus be delayed with app. 700 microseconds which normally is quite unimportant. 100 sensings will delay the interrupt (and keep the RC4000 in disabled mode) for app. 5.7 milliseconds and in case of bus-noise for app. 10 milliseconds. This is usually acceptable but if more than 100 sensings are specified special care should be taken.

If the device in question is an interrupt expander only one sensing per interrupt will be executed.

If the internal buffer runs full, because the output device is too slow, (this may especially happen in case of bad noise on the IO-bus), the following action is taken: If the interrupt channel in question is 0, 1 or 2, or the channel is used for timer interrupt, the senseoperation is still performed but the statusword is lost and the interrupt is passed on to the monitor as normal.

In all other cases the interrupt is delayed until there is room in the buffer. This means that the device in question will be stopped until interrupt is released.

No channel may be specified more than one time in the parameterlist, i.e. one interrupt can only initiate sensing of one device. The only exception is teleterminals connected to the same interrupt expander.

If the device specified is a tapestation, the sense operation is executed twice, once for the normal statusword, and once for the charactercounter. This doublesense feature increases the time for a senseoperation with app. 60%.

1.4 Teleterminals

The interrupts from teleterminals connected to a multiplexer are collected in a interruptexpander, which is a device itself.

Sensing of this expander yields the contents as statusword but at the same time the expander is reset. This gives the following restrictions in the deviceparameters:
The devicenumber of an expander must only be specified together with the interruptnumber of the expander, otherwise the message

xxxexpander

will be output.

If more expanders are specified as interrupt sources, or if a device not belonging to the multiplexer is specified with an expander as interrupt source, the message

xxxmulti itr trouble

is output.

If, however, an arbritrary terminal is specified together with the channelno of one of its interruptexpanders, the program will automatically compute the proper subchannel, and the terminal will only be sensed when an interrupt arrives here independent of possible traffic on the other terminals.
If the sensing of the expander is rejected, the message

xxxexpander <dev> <cause>

is output. <dev> is the devicenumber of the expander and <cause> is either busy or disconnected.

1.5 Printing of statuswords

The programpart operating in enabled mode is started every second by the timer. It checks the cyclical buffers one by one, and if statuswords have arrived in one of them, it is printed on current output. First is the devicenumber, preceded by an asterisk output. If the devicenumber is identical with that of the previous message only four spaces are output. Next bits 0-11 are output (if not suppressed by the <su> parameter in the call) as names:

bit	name
0	local
1	parity
2	timer
3	overrun
4	length
5	end-doc
6	loadpoint
7	tapemark
8	we
9	hi
10	reading-error
11	card-rejected

Then the twelve rightmost bits are printed as a possitive integer surrounded by sharp brackets. This printing may be suppressed by the parameter <skipchar> in the call. Note, that the <su>parameter affects all the devices specified whereas the <skipchar> parameter is special for each device.

If the statusword in question belongs to the timer (device no 3) or the character counter of a tapestation, it is printed as a 24-bit signed integer. In this case the <su>parameter is effectless.

If the integer (12 or 24 bit) is inside the range of the ISO-alphabet it is printed as a character instead. This is usually desireable if the device supervised is a typewriter.

If the statusword is member of a series in a busnoise sensing the heading

bus-noise

is printed before the statuswords, and each statusword is preceded by 2 dashes. Furthermore possible suppression or skipchar specifications are suspended.

If the sensing was rejected either

busy

or

disconnected

is printed. After a rejected sense it is checked that the w0 register (which should contain the statusword) still contains zeroes. If it has changed, the message

w-reg

is printed.

1.6 System protection and termination

As the sladrehank during runtime has changed a few cells (cell 12 plus 2 cells in the itr-responsecode of the monitor), it must be terminated in a special way. To prevent removal of the process (e.g. from the main console) the process name is changed to the text

<SP> sla

which cannot be typed in to the operating system S.

Termination then takes place in this way: On the console from which the program is started the text

stopsla

is typed by the operator. The text is immidiately detected by the sladrehank, as this console is always supervised. When the message is received, the monitor cells are reestablished, the process name is changed to the original one, and when the current output buffer is empty, the program returns to the fileprocessor.

2. ITR (Simulate interrupt)

This program is able to simulate an interrupt by loading cell 8 with the interrupt number (times 2) upon which the monitor is entered by jumping to the address stored in cell 12.

The program is called by

itr <SP> { <channelno> / <expanderchannel> . <subchannel> }$_{1}^{\infty}$

where <channelno> must be the number of a single interrupt channel 0-23. If the channel is connected to an interrupt expander, <expander-channel> is the channel of the expander, and <subchannel> is a number within the range 0-23 defining the bitposition in the expander-register.

Itr is an entry in the sladrehank and must be executed in monitormode (protection key=0).

Note, that if interrupt is simulated on a channel belonging to a device which is busy the following will happen: The interrupt response routine of the monitor will detect the "busy" state and return an answer with result=4 (malfunction). This will normally cause termination of the program using the device. On the other hand, this may be useful in order to release the monitor driver when it is waiting forever for an interrupt from a bad device.

If interrupt is simulated on an unused channel, it will be ignored by the monitor.

3. Reset

Reset works exactly as itr exept that the devicenumber instead of the interruptnumber is specified:

reset $\left\{\text{<deviceno>}\right\}_{1}^{\infty}$

Reset retrieves the channelnumber corresponding to <deviceno> in the monitortable and generates an interrupt.
The device must not be a teleterminal or another device connected to interrupt expander but may be an interrupt expander.
If no interrupt channel is found, the message

xxxno itr

is output.

4. Cleartmx,cleantmx

These programs are intended for reset of the telemultiplexer(s) by provoking the monitor to sense the interruptexpanders. This may be necessary if PCBA's have been removed or inserted with power on, or if the power has been switched off-on.

The call

cleartmx

has the following effect: The telemultiplexer(s) is retrieved in the devicetable of the monitor, and an interrupt is simulated on the channel connected to the "intervention" expander (the second channel) and next on the channel connected to the "finish" interruptexpander. Note: Cleartmx may be called at any time without interferering with the normal traffic on the TMX. The call:

cleantmx

has a similar effect: An interrupt is simulated with the "finish" expander loaded with all ones, i.e. interrupt on all 24 subchannels. Note: Cleantmx may disturb the traffic on the TMX seriously and should not be used unless in rare cases.

If no telemultiplexer is found in the devicetable, the message

xxxno tmx

is displayed.

Both programs must be executed with protection key=0.

5. Examples

␣ denotes typewriter input

Example 1:

Supervision of the main console can be obtained just by typing

```
sla
```

```
from sla

*02 timer
    timer lp=test mt.6.7.8.9.10

RC 747/749 magtape

testprogram: a
number of runs = 9

blocksize = 1212

blocks, files =
    timer
    timer
    timer 50 50

parity =

mode =
    timer

run no.        1
```

The sladrehank is terminated in this way (the operator key is pressed):

```
att stopsla
unknown

from p
```

Example 2 :
If more devices are to be supervised it is convenient to have the programcall stored as a text in a backing storage area:

```
tt=set 1
tt=edit
edit begin.
i/
(i c
sla 6.10.0.4848 7.11.0.4848 8.12.0.4848 9.13.0.4848,
     10.9.0.3636 se.10 su.8.9)
/,f
edit end.
```

Note, thet the normal number of characters (4848 or 3636) is different on device 10 which is a 9-track station.

The sladrehank is now called by

```
i tt
```

```
from  sla

*09 tapemark
    tapemark
*06 tapemark
    4
*07 tapemark
    4
*08 tapemark
    4
*09 tapemark
    4
*10 tapemark
    1
*06 parity
    tapemark
    4
*07 tapemark
    4
*08 tapemark
    4
*09 tapemark
    4
*10 tapemark
    1
*06 tapemark
    4
*07 tapemark
    4
*08 tapemark
    4
*09 tapemark
    4
*10 tapemark
    1
*07 bus-noise
 -- we <0>
 -- 4848
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
    local we hi
*08 tapemark
    4
    local
    local
*09 tapemark
    4
*09 bus-noise
 -- we <0>
 -- 4848
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
```

```
 -- local we <0>
 -- local we <0>
    local we hi
*10 tapemark
    1
    local
    local
*06 tapemark
    4
*06 bus-noise
 -- we <0>
 -- 4848
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
 -- local we <0>
    local we hi
*10 loadpoint
*07 loadpoint
*08 loadpoint
*09 loadpoint
*06 loadpoint
    loadpoint
*07 loadpoint
*08 loadpoint
*09 loadpoint
*10 loadpoint
    parity
    3635
    parity
*06 tapemark
    4
*07 tapemark
    4
*08 tapemark
    4
*09 tapemark
    4
*10 tapemark
att stopsla
unknown

from p
```

Note, that setting the tapestation local/remote generates a series of interrupts after which the statusword is changing during sense. This gives rise to the message: "bus-noise"

If the interval timer is specified with more sensings a similar effect is observed:

```
sla 3.18 se.22

from  sla

*03 bus-noise
 -- 2176
 -- 2177
 -- 2178
 -- 2179
 -- 2180
 -- 2181
 -- 2182
 -- 2183
 -- 2184
 -- 2185
 -- 2186
 -- 2187
 -- 2188
 -- 2189
 -- 2190
 -- 2191
 -- 2192
 -- 2193
 -- 2194
 -- 2195
 -- 2196
 -- 2197
*03 bus-noise
 -- 2432
 -- 2433
 -- 2434
 -- 2435
 -- 2436
 -- 2437
 -- 2438
 -- 2439
 -- 2440
 -- 2441
 -- 2442
 -- 2443
 -- 2444
 -- 2445
 -- 2446
 -- 2447
 --
att stopsla
unknown

from p
2448
 -- 2449
 -- 2450
 -- 2451
 -- 2452
 -- 2453
```

Example 3:

```
itr 4
```

```
att
wait
```

It should be noticed, thet channel 4 is used by the interrupt key of the operators console.

Example 4:

A similar interrupt on a teletrminal may be generated in theis way (interruptno. of itr-expander is 6, and the terminal is connected to the first channel in the multiplexer):

```
itr 6.0
```

Example 5:

A discfile is suspected to give spurious status error. The call

```
sla 13.8.0
```

will write all statuswords differing from zero. In this way an single error which not causes any harm (because of the rereading performed by the monitor) will be displayed.

Example 6:

The printer (device 5) has lost an interrupt:

```
reset 5
```

will release it.

6. Alphabetic list of errormessages

xxxcall

sla or itr has been called with wrong parameters.

xxxcore area too small

The process area should be increased. Every device specified needs app. 1100 bytes extra.

xxxexpander

An interrupt expander is specified together with a wrong channel.

xxxexpander <dev> <cause>

When sensed, the interrupt expander with devicenumber <dev> is busy or disconnected.

xxxmultiple itr trouble

Teleterminal is specified with a wrong interrupt channel or more telemultiplexers are specified.

xxxno itr

reset cannot find a channel which corresponds to the device.

xxxno tmx

itr, cleartmx or cleantmx cannot find any multiplexer.

xxxprimary input trouble

The device and/or interrupt number of the console from which the program is started cannot be found. If it is a teleterminal no terminals or expanders must be specified in the call.

xxxprotection key <key>

The process has the protection key <key> instead of key 0.

xxxslasla

The sladrehank is already running in another process.

xxxsum error <sum>

The checksum of the program is <sum> instead of 0.

7. Program text

```
0
0  ; RCTS   3.0/2        17.10.69   PEH
0  ; VERSION             18.07.71
0
0
0  S.   A250, B100, C50, D50
0  W.
0  B0:            0            ;   ENTRYNO;
2  B1:            0            ;   FP BASE;
4  A0:  AM        -2           ;ENTRY SLA:
6       AM        -2           ;ENTRY ITR:
8       AM        -1           ;ENTRY CLEARTMX:
10      AM        -1           ;ENTRY CLEANTMX:
12      AL   W0   8            ;ENTRY RESET:
14      JL.       A207.        ;   GOTO INIT SLA;
16
16 ; DEVICEBUFFER DESCRIPTORS HAVE THE FOLLOWING FORMAT:
16
16 ; <LINK>
16 ; JL.  W3  (-2)            ;   GOTO SLA ITR RESPONSE;
16      D10=0                 ;   DEVICENUMBER SHIFT 6;
16      D11=2                 ;   CHANNEL NO. SHIFT 1;
16      D12=4                 ;   WRITE POINTER;
16      D13=6                 ;   READ POINTER;
16      D14=8                 ;   SKIPCHAR(1:6);
16      D15=20                ;   DEVICEKIND;
16      D16=22                ;   BOOLEAN CHANNEL=VITAL;
16      D17=24                ;   OVERFLOW FLAG;
16      D18=26                ;   OVERFLOW LINK;
16      D49=30                ;   FIRST BUFFER ELEMENT;
16
16 ; CONSTANTS:
16      D0=D49+2              ;   SIZE OF DESCRIPTOR IN BYTES;
16      D1=256<2              ;   SIZE OF BUFFER IN BYTES;
16
16
16 T.
16
```



```
16
16  B2:            1            ;   SENSINGS;
18  B3:            0            ;   DEVICES;
20  B4:            0,R.24       ;   ENTRYTABLE FOR DEVICEBUFFERS;
68  B5:            4<12+4       ;   SPACE,INTEGER;
70  B6:            4<12+10      ;   SPACE,TEXT;
72  B8:            8<12+4       ;   POINT,INTEGER;
74  B9:            8<12+10      ;   POINT,TEXT;
76  B7:            0,R.24       ;   SUB ITR TABLE;
124 B10:           0            ;   MONITOR ITR RESPONSE;
126                0            ;   FIRST TWO WORDS
128 B12:           0            ;   OF PROCNAME;
130 B13:           -1           ;   -,SUPPRESSED STATUSBITS;
132 B11:           0            ;   ENTRY MULTIPLE ITR ROUTINE;
134 B14:           0            ;   ENTRY SINGLE ITR ROUTINE;
136 B15: JL.  W3  (-2)          ;
138
138 H.
138 B16: 115,116,111,112,115,108,97,0  ;   <:STOPSLA:>;
146 W.
146
146 B17:           0            ;   CHARCOUNT;
148 B18:           0            ;   DEVICENO.*64 OF PRIMARY INPUT;
150 B19:           0            ;   W0;
152 B20:           0            ;   W1;
154 B21:           0            ;   W2;
156 B22:           0            ;   W3;
158 B23:           0            ;   EX;
160 B24:           0            ;   LINKPOINT IN MONITOR;
162       JL.      (2)          ;   MULTIPLE ITR RESPONSE;
164 B25:           0            ;   SWAPCODE;
166                0            ;   SAVE FOR MULTIPLE
168 B26:           0            ;   ITR CODE;
170 B27:           0            ;   SAVE FOR ITR EXPANDER;
172 B29:           0            ;   DEVICENO.*64 OF EXPANDER;
174                0            ;
176 B30:           0            ;   STATUS REFERENCE;
178                0            ;
180                0            ;
182                0            ;   BOOLEAN BUSNOISE;
184 B31:           0            ;   SENSE;
186 B32:           0            ;   DEVICE;
188 B33:           0            ;   STOP PROGRAM;
190 B34:           -1           ;   NUMBER OF NOISE WORDS;
192 B35:           -1           ;   LAST DEVICE;
194 B36:           0            ;   BOOLEAN OUTPUT;
196 B37:           0            ;   BOOLEAN DEVICENO;
198 B38:           0            ;   FIRST BUFFERWORD IN CURRENT SENSE;
200 B39:           0            ;   TEST;
202 B40:           48<12+2      ;   <<ZD>;
204 B41:           0            ;   EXPANDERERROR;
206 B42:           2,11<21      ;
208
```

```
208
208
208
208  C1:  <:<10>***CALL<0>:>
214  C2:  <:<10>***SUM ERROR <0>:>
224  C3:  <:<10>***PROTECTION KEY <0>:>
238  C4:  <:<10>***CORE AREA TOO SMALL<0>:>
254  C5:  <:<10>***NO TMX<0>:>
262  C6:  <:<32>SLA<0>:>
266  C7:  <:SE:>
268  C8:  <:SU:>
270  C9:  <:<10>***PRIMARY INPUT TROUBLE<0>:>
288  C10: <:<10>***EXPANDER <0>:>
298  C11: <:<10>***MULTIPLE ITR TROUBLE<0>:>
316  C12: <:CLOCK:>,0,0,0
326  C13: <:BUS-NOISE<0>:>
334  C14: <: -- <0>:>
338  C15: <:*<0>:>
340  C16: <:    <0>:>
344       <: <127><127>:>
346  C17: <:BUSY <0>:>
350       <: <127><127>:>
352  C18: <:DISCONNECTED <0>:>
362  C19: <:W-REG <0>:>
368  C20: <:LOCAL <0>:>
374  C21: <:PARITY <0>:>
380  C22: <:TIMER <0>:>
386  C23: <:OVERRUN <0>:>
392  C24: <:LENGTH <0>:>
398  C25: <:END_DOC <0>:>
404  C26: <:LOADPOINT <0>:>
412  C27: <:TAPEMARK <0>:>
420  C28: <:WE <0>:>
424  C29: <:HI <0>:>
428  C30: <:READING_ERROR <0>:>
438  C31: <:CARD_REJECTED <0>:>
448  C32: <:<60><0>:>
450  C33: <:<62> <0>:>
452  C34: <:<10>***SLASLA<0>:>
460  C35: <:<10>***NO ITR<0>:>
468
468
468
468
468
468  ;   COMMENT THE CODE ON PAGES 3-7 IS EXECUTED IN ITR-DISABLED
468  ;   MODE AND IS INITIATED BY ANY INTERRUPT;
468
468  A97: RL.  W0   B29.      ;ERROR11:
470       LS   W0   -6        ;
472       XS        0         ;   EXPANDERERROR:=
474       RS.  W0   B41.      ;   EX(21:23) SHIFT 12+DEVICENO;
476       AL   W0   -1        ;
478       JL.       A98.      ;   GOTO SWAP;
480
```



```
480
480  A100:DS.  W3   B22.      ;ITR RESPONSE:
482       AM        (8)       ;   SAVE W3:
484       JL.       (B4.)     ;   GOTO ITR ACTION(WORD(8));
486  ;   COMMENT THE ACTION IS ONE OF THE FOLLOWING:
486  ;   1. SINGLE ITR (VIA DESCRIPTOR TO A106)
486  ;   2. MULTIPLE ITR (A101)
486  ;   3. SINGLE ITR, PRIMARY INPUT (A124)
486  ;   4. PASS ON TO MONITOR (I.E. NO ACTION, (B10));
486
486  A101:DS.  W1   B20.      ;MULTIPLE ITR:
488       XS.       B23.      ;
490       IO.  W0   (B29.)    ;   SENSE EXPANDER;
492       SX.       2.11      ;   IF EX(22:23)<>0
494       JL.       A97.      ;   THEN GOTO ERROR11;
496  A98: DL.  W2   B25.      ;   CHANGE MONITOR
498       DS.  W2   (B24.)    ;   MULTIPLE RESPONSE CODE;
500       AL   W1   0         ;   BIT:=0;
502       RS.  W0   B27.      ;
504  A102:SN   W0   0         ;   IF EXPANDER EMPTY
506       JL.       A108.     ;   THEN GOTO MONITOR ITR RESPONSE;
508       SH   W0   -1        ;   IF EXPANDER(BIT) THEN
510       JL.  W2   (X1+87.)  ;   GOTO ITR ACTION;
512  ;   COMMENT THE ACTION IS ONE OF THE FOLLOWING:
512  ;   1. SENSEACTION (VIA DESCRIPTOR TO A103)
512  ;   2. SENSE PRIMARY INPUT (A125)
512  ;   3. NEXTBIT (I.E. NO ACTION, A104);
512
512  A104:LS   W0   1         ;NEXTBIT:
514       AL   W1   X1+2      ;   BIT:=BIT+1;
516       JL.       A102.     ;
518
518  A103:DS.  W1   A107.     ;SENSEACTION:
520       JL.  W2   A121.     ;   GOTO SENSE(BUFFER);
522       DL.  W1   A107.     ;
524       JL.       A104.     ;
526                 0         ;
528  A107:          0         ;
530
530  A105:DL.  W3   B26.      ;   REESTABLISH MONITORS
532       DS.  W3   (B24.)    ;   MULTIPLE INTERRUPT ROUTINE;
534       RL.  W2   B27.      ;   SIMULATE SENSING OF
536       JL.       (B24.)    ;   EXPANDER;
538
538  A106:DS.  W1   B20.      ;SINGLE ITR:
540       XS.       B23.      ;   COMMENT W3=BUFFER ADDR;
542       JL.  W2   A121.     ;   GOTO SENSE(BUFFER);
544  A108:DL.  W1   B20.      ;MONITOR ITR RESPONSE:
546       DL.  W3   B22.      ;   REESTABLISH REGISTERS;
548       XL.       B23.      ;
550       JL.       (B10.)    ;
552
552  A123:          0         ;   LINK;
554  A121:AL   W0   0         ;SENSE:
556       SE   W0   (X3+D17)  ;   IF OVERFLOW(BUFFER)
558       JL.       A130.     ;   THEN GOTO TREAT OVERFLOW;
560       IO   W0   (X3+D10)  ;
562       RS.  W2   A123.     ;
564       RL   W2   X3+D12    ;   SAVE CURRENT WRITEPOINTER;
566       RS.  W2   B38.      ;
568       LD   W2   -100      ;   SENSE:=0;
570       DS.  W2   B31.      ;   BUSNOISE:=FALSE;
572       JL.       A122.     ;
574
```

```
574
574  A128:AL   W2   1          ;NEXTSENSE:
576       WA.  W2   B31.       ;   IF SENSE >=SENSINGS
578       SL.  W2  (B2.)       ;   THEN GOTO EXIT;
580       JL.       A131.      ;
582       RS.  W2   B31.       ;   SENSE:=SENSE+1;
584       LD   W1   -100       ;   CLEAR FLAGS;
586       IO   W0  (X3+D10)    ;
588  A122:SX        2.11       ;   IF EX(22:23)<>0 THEN
590       JL.  W2   A127.      ;   GOTO TREAT EXCEPTIONS;
592       SH   W2   0          ;   IF FIRST SENSE THEN
594       JL.       A126.      ;   GOTO SET REFERENCE;
596       SN.  W0  (B30.-2)    ;
598       SE.  W1  (B30.)      ;   IF STATUS<>REFERENCE THEN
600       JL.  W2   A134.      ;   BEGIN BUSNOISE:=TRUE;
602                            ;     GOTO WRITESTATUS;
602       AL   W1   D50        ;   END;
604       JL.       A133.      ;   GOTO TEST KIND;
606  A126:DS.  W1   B30.       ;   REFERENCE:=STATUS;
608       JL.  W2   A132.      ;   WRITE STATUS IN BUFFER;
610       AL   W1   D50        ;
612  A133:RL   W2   X3+D15     ;   IF KIND=EXPANDER THEN
614       SH   W2   126        ;   GOTO MULTITR;
616       JL.       X1+A131.   ;   IF -,MAGTAPE THEN GOTO
618       SE   W2   127        ;   IF OVERFLOW THEN ENDSENSE
620       JL.       A98.       ;   ELSE NEXTSENSE;
622       LD   W1   -100       ;
624       AM       (X3+D10)    ;   SENSE CHARCOUNTER;
626       IO   W0   +4         ;
628       SX        2.11       ;   IF EX(22:23)<>0 THEN
630       JL.  W2   A127.      ;   TREAT EXCEPTIONS;
632       AL   W1   X1+16      ;   FLAG16:=TRUE;
634       AL   W2   0          ;
636       SL.  W2  (B31.)      ;   IF FIRST SENSE THEN
638       JL.       A137.      ;   GOTO SET REFERENCE1;
640       SN.  W0  (B30.+2)    ;
642       SE.  W1  (B30.+4)    ;   IF STATUS1<>REFERENCE1 THEN
644       JL.  W2   A134.      ;   BEGIN BUSNOISE:=TRUE;
646                            ;     GOTO WRITESTATUS;
646       JL.       A128.      ;   END;
648                 0          ;   GOTO NEXTSENSE;
650  A137:DS.  W1   B30.+4     ;   REFERENCE1:=STATUS1;
652       JL.  W2   A132.      ;   WRITE STATUS IN BUFFER;
654       JL.       A128.      ;   IF -,OVERFLOW THEN GOTO NEXTSENSE;
656
656  A131:RL.  W0   B31.-2     ;ENDSENSE:
658       SN   W0   0          ;   IF -,BUSNOISE THEN
660       JL.      (A123.)     ;   RETURN;
662       RL   W1   X3+D12     ;
664       WS.  W1   B38.       ;
666       SH   W1   0          ;   FLAG(0:11) OF FIRST
668       AL   W1   X1+D1      ;   BUFFERWORD OF CURRENT
670       LS   W1   -2         ;   SENSE:=
672       RL.  W2   B38.       ;     NUMBER OF NOISEWORDS;
674       SL   W2   X3+D1+D49-4;
676       AM        -D1        ;
678       HS   W1   X2+4       ;
680       JL.      (A123.)     ;
682  D50=A128-A131
682
```



```
682
682  A127:RS.  W2   A136.      ;TREAT EXCEPTIONS:
684       XS        3          ;
686       SE   W0   0          ;   FLAG1:=BUSY; FLAG2:=DISCONNECTED;
688       AL   W1   X1+8       ;   IF W0 CHANGED THEN FLAG8:=TRUE;
690       RL.  W2   B31.       ;
692       JL.      (A136.)     ;
694  A136:              0      ;   LINK;
696
696  A130:SE   W0  (X3+D16)    ;TREAT OVERFLOW:
698       JL         X2         ;   IF CHANNEL=VITAL THEN RETURN;
700       RS   W2   X3+D18      ;
702       AL   W1   2           ;   OVERFLOW:=2;
704       RS   W1   X3+D17      ;
706       XL.       B23.        ;   DELAY INTERRUPT AND
708       DL.  W3   B22.        ;   RETURN TO CURRENT TIMESLICE;
710       IC.      (B42.)       ;   IR(1:2):=0;
712       SP       (10)         ;   IF WORD(10) PROTECTED
714       JL.       A129.       ;   THEN GOTO TEST KEY;
716  A135:DL.  W1   B20.        ;
718       JL       (10)         ;
720  A129:AM       (66)         ;TEST KEY:
722       BZ   W1   +33          ;   IF KEY(CURRENT PROC)
724       SN   W1   0            ;   <>0
726       JL.       A135.       ;   THEN
728       RS   W0   8           ;   GENERATE ITR(0);
730       DL.  W1   B20.        ;
732       JL.      (B10.)       ;
734
734  A134:RS.  W2   B31.-2      ;   BUSNOISE:=TRUE;
736       AL   W2   X2+8         ;
738
738  A132:RS.  W2   A138.       ;   PROCEDURE WRITE(STATUS,BUFFER);
740       AL   W2   4           ;
742       WA   W2   X3+D12      ;
744       SL   W2   X3+D1+D49   ;
746       AL   W2   X2-D1       ;   ADVANCE READPOINTER;
748       RS   W2   X3+D12      ;
750       DS   W1   X2          ;   WRITE(STATUS,FLAGS);
752       WS   W2   X3+D13      ;   IF ABS(WRITEPOINTER-READPOINTER)
754       SH   W2   -1          ;   < BUFFERSIZE-10
756       AL   W2   X2+D1       ;
758       SH   W2   D1-10       ;   THEN
760       JL.      (A138.)      ;   RETURN ELSE
762       AL   W1   1           ;   OVERFLOW(BUFFER):=TRUE;
764       RS   W1   X3+D17      ;
766       AM.      (A138.)      ;   ERRORRETURN;
768       JL        +2          ;
770  A138:              0      ;   LINK;
772
772  A124:DS.  W1   B20.        ;PRIMARY INPUT SINGLE ITR:
774       XS.       B23.        ;
776
```

```
776
776  A125:IO.   W0   (B18.)      ;SENSE PRIMARY INPUT:
778       SH    W0    2047       ;   IF STATUSERROR
780       SX          2,11       ;   OR EXCEPTIONS
782       JL.         A140.      ;   THEN GOTO RESET CHARCOUNT;
784       RL.   W1    B17.       ;
786       BL.   W3    X1+B16.    ;
788       SE    W0    X3         ;   IF TYPE=TEXT(CHARCOUNT)
790       JL.         A140.      ;   THEN BEGIN
792       AL    W1    X1+1       ;     CHARCOUNT:=CHARCOUNT+1;
794       BL.   W3    X1+B16.    ;
796       SN    W3    0          ;     IF TEXT(CHARCOUNT)=0
798       JL.         A141.      ;     THEN GOTO STOP;
800       RS.   W1    B17.       ;
802       JL.         A108.      ;     GOTO RETURN;
804                              ;   END;
804  A140:AL    W1    0          ;   CHARCOUNT:=0;
806       RS.   W1    B17.       ;
808       JL.         A108.      ;   GOTO RETURN;
810
810  A141:RL.   W0    B10.       ;STOP:
812       RS    W0    12         ;   REESTABLISH MONITOR ITR RESPONSE
814       AM.         (B1.)      ;   ADDRESS;
816       RL    W2    H16        ;
818       DL.   W1    B12.       ;   REESTABLISH NAME
820       DS    W1    X2+4       ;   OF OWN PROCESS;
822       AL    W0    1          ;   STOP PROGRAM:=TRUE;
824       RS.   W0    B33.       ;
826       JL.         A108.      ;   GOTO MONITOR ITR;
828
828  A165:                       ;TERMINATE:
828       AL    W2    0          ;   DEVICE:=0;
830       AL.   W1    A200.      ;
832  A166:RL    W0    X1+D17     ;
834       SH    W0    1          ;   IF -,OVERFLOW(BUFFER) THEN
836       JL.         A168.      ;   GOTO NEXTDEVICE;
838       AL.   W0    A168.      ;
840       RL    W3    X1+D11     ;   CORE(8):=CHANNEL(BUFFER);
842       DS    W0    10         ;   CORE(10):=RETURN;
844       JL.         (B10.)     ;   SIMULATE ITR;
846  A168:AL    W2    X2+1       ;   DEVICE:=DEVICE+1;
848       SL.   W2    (B3.)      ;   IF DEVICE >=DEVICES THEN
850       JE.         A33.       ;   GOTO END;
852       AL    W1    X1+D0+D1   ;
854       JD.         A166.      ;
856
856  A169:RX    W0    X1+D17     ;DISABLED RELEASE DEVICE:
858       SH    W0    1          ;   OVERFLOW:=0;
860       JE.         A153.      ;   IF OVERFLOW WAS=1 THEN RETURN;
862       AL    W3    X1         ;
864       RL    W2    X1+D18     ;
866       RL    W0    X1+D11     ;
868       AL.   W1    A153.      ;   RETURN:=NEXSTATUS;
870       DS    W1    10         ;
872       JL.         A121.      ;   GOTO SENSE;
874
```



```
874
874  ;   COMMENT THE CODE ON THE FOLLOWING PAGES IS EXECUTED IN
874  ;   ITR-ENABLED MODE, I.E. IN NORMAL TIMESLICES;
874
874  A150:AL    W1    0          ;WAIT:
876  A152:RS.   W1    B32.       ;   DEVICE:=0;
878       AL    W0    D0+D1      ;
880       WM    W1    0          ;   W1=ADDRESS(DEVICE);
882       AL.   W1    X1+A200.   ;
884       RS.   W1    A154.      ;
886  A153:RL.   W1    A154.      ;NEXTSTATUS:
888       RL    W2    X1+D13     ;   IF WRITEPOINTER<>READPOINTER
890       SE    W2    (X1+D12)   ;   THEN
892       JL.         A180.      ;   GOTO PRINT STATUS;
894       RL.   W1    B32.       ;
896       AL    W1    X1+1       ;   DEVICE:=DEVICE+1;
898       SL.   W1    (B3.)      ;   IF DEVICE >=DEVICES
900       JL.         A155.      ;   THEN GOTO START TIMER;
902       JL.         A152.      ;
904  A154:              0          ;   BUFFERBASE;
906
906  A155:RL.   W1    B39.       ;START TIMER:
908       AL    W1    X1+1       ;   TEST:=TEST+1;
910       RS.   W1    B39.       ;
912       SH    W1    1          ;   IF TEST< 2
914       JL.         A150.      ;   THEN GOTO WAIT;
916       RL.   W2    B33.       ;
918       SE    W2    0          ;
920       JD.         A165.      ;   IF PROGRAM STOP THEN GOTO TERMINATI
922       SN.   W2    (B36.)     ;   IF OUTPUT THEN
924       JL.         A157.      ;   BEGIN
926       RS.   W2    B36.       ;     OUTPUT:=FALSE;
928       AM.         (B1.)      ;     OUTEND(0);
930       JL    W3    H33-2      ;     GOTO WAIT;
932       JL.         A150.      ;   END;
934  A157:AL.   W1    A160.      ;
936       AL.   W3    C12.       ;
938       JD          1<11+16    ;   SEND MESSAGE(<:CLOCK:>,1);
940       AL.   W1    A161.      ;
942       JD          1<11+18    ;   WAIT ANSWER(<:CLOCK:>);
944       JL.         A150.      ;   GOTO WAIT;
946
946  A160:              0          ;   MESSAGE; OPERATION=0;
948                     1          ;   SECONDS=1;
950  A161:              0,R,8      ;   ANSWERBUFFER;
966
966  A162:AL.   W0    C10.       ;WRITEEXPERR:
968       JL.   W3    A202.      ;
970       BL.   W0    B41.+1     ;   WRITE(OUT,<:<10>***EXPANDER :>,
972       JL.   W3    A199.      ;   EXPANDER,
974       XL.         B41.       ;   IF EX(23) THEN
976       SX          2,1        ;   <:BUSY:> ELSE
978       AM          C17-C18    ;   <:DISCONNECTED:>);
980       AL.   W0    C18.-2     ;
982       JL.   W3    A202.      ;
984       AL    W0    0          ;   EXPANDERERROR:=FALSE;
986       RS.   W0    B41.       ;
988       RL.   W1    A154.      ;
990       JL.         A181.      ;
992
```

```
 992
 992 A170:AM.      (B37.)      ;   PROCEDURE WRITEDEVICENO;
 994       SE   W3  X3          ;   IF DEVICENO
 996       JL       X3          ;   THEN RETURN;
 998       DS.  W1  A172.       ;
1000       DS.  W3  A174.       ;
1002       AL   W0  1           ;
1004       RS.  W0  B37.        ;   DEVICENO:=TRUE;
1006       AL   W2  10          ;   WRITE(OUT,<:<10>:>);
1008       AM.      (B1.)       ;
1010       JL   W3  H26-2       ;
1012       RL.  W1  B34.        ;   IF NOISE-WORDS
1014       AL   W1  X1-1        ;
1016       SL   W1  -1          ;
1018       RS.  W1  B34.        ;
1020       SH   W1  -1          ;   THEN
1022       JL.      A171.       ;   BEGIN
1024       AL.  W0  C14.        ;     WRITE(OUT,<: -- :>);
1026       JL.      A177.       ;   END;
1028 A171:RL.  W0  (A154.)      ;
1030       SN.  W0  (B35.)      ;   IF DEVICE=LASTDEVICE
1032       JL.      A176.       ;   THEN GOTO PRINT SPACES;
1034       RS.  W0  B35.        ;   LASTDEVICE:=DEVICE;
1036       AL.  W0  C15.        ;
1038       JL.  W3  A202.       ;
1040       RL.  W0  B35.        ;
1042       LS   W0  -6          ;   WRITE(OUT,<<*ZD_>,DEVICE);
1044       JL.  W3  A199.       ;
1046       AM       2           ;
1048 A176:AL.  W0  C16.        ;PRINTSPACE:
1050 A177:JL.  W3  A202.       ;   WRITE(OUT,<:    :>);
1052 A178:DL.  W1  A172.       ;
1054       DL.  W3  A174.       ;
1056       JL       X3          ;   RETURN;
1058                0           ;   WORKING REGISTERS;
1060 A172:            0         ;
1062                0           ;
1064 A174:            0         ;
1066
1066 A180:AL   W0   0           ;PRINT STATUS:
1068       RS.  W0  B37.        ;   DEVICENO:=FALSE;
1070       RS.  W0  B39.        ;   TEST:=0;
1072       SE.  W0  (B41.)      ;   IF EXPANDER-ERROR THEN
1074       JL.      A162.       ;   GOTO WRITEEXPERR;
1076 A181:SE.  W0  (B33.)      ;   IF PROGRAM STOP
1078       JD.      A165.       ;   THEN GOTO TERMINATE;
1080       SL   W2  X1+D1+D49-4 ;
1082       AM       -D1         ;
1084       AL   W2  X2+4        ;   ADVANCE READPOINTER;
1086       RS   W2  X1+D13      ;
1088       RL   W0  X2          ;
1090
```



```
1090
1090       SH   W0  2047        ;   IF BUS-NOISE THEN
1092       JL.      A182.       ;   BEGIN
1094       RS.  W0  B35.        ;     LASTDEVICE:=NONSENS;
1096       JL.  W3  A170.       ;     WRITE DEVICENO;
1098       AL.  W0  C13.        ;     WRITE(OUT,<:_BUS-NOISE:>);
1100       JL.  W3  A202.       ;
1102       AL   W0  0           ;
1104       RS.  W0  B37.        ;     DEVICENO:=FALSE;
1106       DL.  W1  A172.       ;
1108       BZ   W3  0           ;     NOISEWORDS:=FLAG(0:11);
1110       RS.  W3  B34.        ;
1112       DL.  W3  A174.       ;   END;
1114 A182:SZ   W0  2.11         ;   IF FLAG1 OR FLAG2 THEN
1116       JL.      A190.       ;   GOTO EXCEPTIONAL;
1118       RL   W3  X1+D15      ;
1120       SE   W3  2           ;   IF DEVICEKIND=2
1122       SZ   W0  16          ;   OR FLAG16 THEN
1124       JL.      A187.       ;   GOTO PRINT COUNTER;
1126
1126       RL   W0  X2-2        ;PRINT STATUSBITS:
1128       RL.  W3  B34.        ;   GET STATUS;
1130       SL   W3  0           ;   IF NOISEWORDS
1132       JL.      A191.       ;   THEN WRITE DEVICENO
1134       LA.  W0  B13.        ;   ELSE SUPPRESS;
1136 A183:SN   W0  0           ;   IF STATUS=0 THEN
1138       JL.      A186.       ;   GOTO TEST SUPPRESSION;
1140       AL   W1  0           ;   FOR BIT:= 0 STEP 1 UNTIL 11
1142 A184:SL   W0  0           ;   DO BEGIN
1144       JL.      A185.       ;     IF STATUS(0)=1 THEN
1146       JL.  W3  A170.       ;     BEGIN WRITE DEVICENO;
1148       DS.  W1  A189.       ;       WRITE(OUT,CASE BIT+1 OF (
1150       BL.  W1  X1+A188.    ;       LOCAL,PARITY,TIMER,OVERRUN,LENGT
1152       AL.  W0  X1+C20.     ;       END_DOC,LOADPOINT,TAPEMARK,
1154       JL.  W3  A202.       ;       WE,HI,READING_ERROR,CARD_REJECTE
1156       DL.  W1  A189.       ;     END;
1158 A185:LS   W0  1           ;     STATUS:=STATUS SHIFT 1;
1160       AL   W1  X1+1        ;
1162       SH   W1  11          ;
1164       JL.      A184.       ;   END;
1166       LS   W0  -12         ;   STATUS:=STATUS SHIFT (-12);
1168       SN   W0  64          ;   IF STATUS<>64 AND
1170       JL.      A186.       ;   (STATUS=10
1172       SE   W0  10          ;   OR STATUS < 127
1174       SL   W0  32          ;   AND STATUS >31)
1176       SL   W0  127         ;   THEN
1178       JL.      A186.       ;   BEGIN
1180       RL.  W2  (A154.)     ;   IF DEVICE<>LASTDEVICE
1182       SE.  W2  (B35.)      ;   THEN WRITE DEVICENO;
1184       JL.  W3  A170.       ;
1186       RL   W2  0           ;     WRITE(OUT,FALSE ADD STATUS,1);
1188       LS   W2  16          ;
1190       AL   W0  4           ;
1192       JL.  W3  A202.       ;     GOTO TEST OVERFLOW;
1194       JL.      A194.       ;   END;
1196
```

```
1196
1196  A186:JL.  W3   A196.       ;   TEST SUPPRESSION;
1198       JL.  W3   A170.       ;   WRITE DEVICENO;
1200       RS.  W0   A179.       ;
1202       AL.  W0   C32.        ;   WRITE(OUT,<:<_:>);
1204       JL.  W3   A202.       ;
1206       RL.  W0   A179.       ;
1208       JL.  W3   A201.       ;   PRINTNUMBER(STATUS(12:23));
1210       AL.  W0   C33.        ;   WRITE(OUT,<:>:>);
1212       JL.  W3   A202.       ;
1214       JL.       A194.       ;   GOTO TEST OVERFLOW;
1216  A187:RL   W0   X2-2        ;PRINT COUNTER:
1218       RL.  W3   B34.        ;
1220       SH   W3   -1          ;   IF -,NOISEWORDS THEN
1222       JL.  W3   A196.       ;   TEST SUPPRESSION;
1224       JL.  W3   A170.       ;   WRITE DEVICENO;
1226       JL.  W3   A201.       ;   PRINTNUMBER;
1228       JL.       A194.       ;   GOTO TEST OVERFLOW;
1230
1230  H.
1230  A188:C20-C20,C21-C20,C22-C20,C23-C20,C24-C20,C25-C20,
1236       C26-C20,C27-C20,C28-C20,C29-C20,C30-C20,C31-C20
1242  W.
1242  A179:          0            ;   W0;
1244  A189:          0            ;   W1;
1246
1246  A191:JL.  W3   A170.       ;
1248       JL.       A183.       ;
1250
1250  A194:RL.  W1   A154.       ;TEST OVERFLOW:
1252       AL   W0   0            ;
1254       SN   W0   (X1+D17)     ;   IF -,OVERFLOW THEN
1256       JL.       A153.+2     ;   GOTO NEXTSTATUS;
1258       RL   W2   X1+D12       ;
1260       WS   W2   X1+D13       ;
1262       SH   W2   -1           ;   IF ROOM IN BUFFER >=2
1264       AL   W2   X2+D1        ;   THEN GOTO DISABLED
1266       SH   W2   D1-10        ;   RELEASE DEVICE ELSE
1268       JD.       A169.       ;   GOTO NEXTSTATUS;
1270       JL.       A153.+2     ;
1272
1272  A190:JL.  W3   A170.       ;EXCEPTIONAL:
1274       SO   W0   8            ;   WRITE DEVICENO.;
1276       JL.       A192.       ;   IF FLAG8 THEN
1278       AL.  W0   C19.        ;   WRITE(OUT,<:W-REG_:>);
1280       JL.  W3   A202.       ;
1282       DL.  W1   A172.       ;
1284  A192:SO   W0   1            ;   IF FLAG1 THEN
1286       JL.       A193.       ;   WRITE(OUT,<:BUSY_:>);
1288       AL.  W0   C17.        ;
1290       JL.  W3   A202.       ;
1292       DL.  W1   A172.       ;
1294  A193:SO   W0   2            ;   IF FLAG2 THEN
1296       JL.       A194.       ;   WRITE(OUT,<:DISCONNECTED_:>);
1298       AL.  W0   C18.        ;
1300       JL.  W3   A202.       ;
1302       JL.       A194.       ;   GOTO TEST OVERFLOW;
1304
```



```
1304
1304  A196:RL.  W1   A154.       ;   PROCEDURE TEST SUPPRESSION(VALUE);
1306       AL   W2    -1         ;   N:=1;
1308       SL.  W2   (B34.)      ;   IF NOISEWORDS OR
1310  A197:SN   W2   (X1+D14)    ;   IF SKIPCHAR(N)= -1
1312       JL        X3          ;   THEN RETURN;
1314       SN   W0   (X1+D14)    ;   IF VALUE = SKIPCHAR(N)
1316       JL.       A194.       ;   THEN GOTO TEST OVERFLOW;
1318       AL   W1   X1+2        ;   N:=N+1;
1320       JL.       A197.       ;
1322
1322  A199:RL.  W1   B40.        ;   PROCEDURE WRITEINTEGER;
1324       JL.       +4          ;   LAYOUT:=<<ZD>;
1326  A201:AL   W1   0            ;   LAYOUT:=<<D>;
1328       RS.  W1   A203.       ;
1330       RS.  W3   A205.       ;
1332       AL   W3   1            ;
1334       RS.  W3   B36.        ;   OUTPUT:=TRUE;
1336       AM.       (B1.)       ;
1338       JL   W3   H32-2       ;
1340  A203:          0            ;   LAYOUT;
1342       JL.       (A205.)     ;
1344
1344  A202:RS.  W3   A205.       ;   PROCEDURE WRITETEXT(TEXT,LINK);
1346       AL   W3   1            ;
1348       RS.  W3   B36.        ;   OUTPUT:=TRUE;
1350       AM.       (B1.)       ;
1352       JL   W3   H31-2       ;
1354       JL.       (A205.)     ;
1356  A205:          0            ;   LINK;
1358
1358  A33: AL   W2   10           ;SUCCESSFUL TERMINATION:
1360       AM.       (B1.)       ;   OUTEND(10);
1362       JL   W3   H33-2       ;
1364       AL   W2   0            ;
1366       AM.       (B1.)       ;
1368       JL   W3   H7          ;
1370
1370
1370
1370
1370  ; START OF DEVICEBUFFERS;
1370                 0            ;
1372                 0            ;
1374  A200:          2<6          ;   MAIN CONSOLE;
1376                 0            ;   COMMENT THIS BUFFERDESCRIPTOR
1378                 0            ;   IS OVERWRITTEN IF ANOTHER DEVICE
1380                 0            ;   IS SPECIFIED;
1382                 0, -1,R.5    ;
1394                 8            ;
1396                 0            ;
1398                 0            ;
1400                 0            ;
1402
```

```
1402
1402
1402  ;  THE FOLLOWING CODE IS USED FOR INITIALIZATION ONLY AND
1402  ;   IS OVERWRITTEN BY THE CONTENTS OF THE FIRST BUFFER;
1402
1402                                ;INITSLA:
1402  A207:RS.   W0    B0.          ;    SAVE ENTRY;
1404       RS.   W1    B1.          ;    SAVE FP BASE;
1406       AL.   W1    A0.          ;SUMCHECK:
1408       AL    W2    0            ;    SUM:=0;
1410  A1:  WA    W2    X1           ;    FOR ADDR:=FIRST STEP 2
1412       AL    W1    X1+2         ;    UNTIL LAST DO
1414       SH.   W1    A210.        ;    SUM:=SUM+WORD(ADDR);
1416       JL.         A1.          ;
1418       SE    W2    0            ;    IF SUM<>0 THEN
1420       JL.         A22.         ;    GOTO ERROR2;
1422       KL.   W2    +0           ;
1424       SE    W2    0            ;    IF PROTECTION KEY<>0 THEN
1426       JL.         A23.         ;    GOTO ERROR3;
1428       RL    W0    12           ;    SAVE CORE(12);
1430       RS.   W0    B10.         ;
1432       AL    W1    0            ;
1434  A2:  RS.   W0    X1+B4.       ;    INITIALIZE
1436       AL    W1    X1+2         ;    ENTRYTABLE TO
1438       SH    W1    46           ;    MONITOR ITR RESPONSE;
1440       JL.         A2.          ;
1442       AL.   W0    A104.        ;
1444       AL    W1    0            ;
1446  A15: RS.   W0    X1+B7.       ;    INITIALIZE SUB
1448       AL    W1    X1+2         ;    ITR ENTRYTABLE
1450       SH    W1    46           ;    TO NO ACTION;
1452       JL.         A15.         ;
1454       AL.   W0    A103.        ;    SET MULTIPLE RESPONSE;
1456       AL.   W1    A106.        ;    SET SINGLE ITR RESPONSE;
1458       DS.   W1    B14.         ;
1460       DL.   W1    B15.         ;    SET RESPONSE ADDRESS
1462       DS.   W1    A200.-2      ;    FOR MAIN CONSOLE;
1464       AL.   W0    A105.        ;
1466       RS.   W0    B25.         ;
1468       AM.       (B1.)          ;
1470       SE    W3    (H8)         ;    IF OUTFILE THEN
1472       JL.         A21.         ;    GOTO ERROR1;
1474       RL.   W1    B0.          ;    GET ENTRY;
1476       BA    W3    X3+1         ;    GET NEXT PARAM;
1478       BL    W2    X3           ;
1480       SH    W2    2            ;    IF LIST END THEN
1482       JL.         A3.          ;    GOTO TEST2;
1484       JL.         X1           ;TEST1:   GOTO CASE ENTRY OF (
1486       JL.         A4.          ;    FIRSTPARAM,
1488       JL.         A4.          ;    FIRSTPARAM,
1490       JL.         A21.         ;    ERROR1,
1492       JL.         A4.          ;    FIRSTPARAM);
1494  A3:  JL.         X1           ;TEST2:   GOTO CASE ENTRY OF (
1496       JL.         A111.        ;    START1,
1498       JL.         A21.         ;    ERROR1,
1500       JL.         A51.         ;    TMXSTART,
1502       JL.         A21.         ;    ERROR1);
1504
```



```
1504
1504  A4:  RL         X3           ;FIRSTPARAM:
1506       RL.   W0    B0.          ;    GET ENTRY;
1508       SH    W0    2            ;    IF ENTRY >1 THEN
1510       JL.         A5.          ;    BEGIN
1512       SE.   W1    (B5.)        ;      IF PARAM<>INTEGER THEN
1514       JL.         A21.         ;      GOTO ERROR1;
1516       RL    W1    X3+2         ;      GET INTEGER;
1518       BA    W3    X3+1         ;      NEXTCOMMAND;
1520       SL    W0    8            ;      IF ENTRY=8 THEN
1522       JL.         A211.        ;      GOTO RESET;
1524       SL    W1    0            ;      IF ITR < 0 OR
1526       SL    W1    24           ;      ITR >23 THEN
1528       JL.         A21.         ;      GOTO ERROR1;
1530       RL    W0    X3           ;
1532       SE.   W0    (B8.)        ;      IF NO INTEGER
1534       JD.         A71.         ;      THEN GOTO DISABLED START ITR;
1536       RL    W0    X3+2         ;
1538       JD.         A70.         ;      ELSE GOTO DISABLED MULTIITR;
1540                                ;    END;
1540
1540  A5:  SE.   W1    (B6.)        ;NEXT DEVICE:
1542       JL.         A7.          ;    IF TEXTPARAM THEN
1544       RL    W2    X3+2         ;    BEGIN GET PARAM;
1546       AL    W0    0            ;      M:=0;
1548       SN.   W2    (C7.)        ;      IF PARAM=<:SE:> THEN
1550       AL    W0    1            ;      M:=1;
1552       SN.   W2    (C8.)        ;      IF PARAM=<:SU:> THEN
1554       AL    W0    2            ;      M:=2;
1556       BA    W3    X3+1         ;NEXTPARAM:
1558       DL    W2    X3+2         ;      GET PARAMTYPE;
1560       BA    W1    4            ;
1562       SE    W0    0            ;      IF M=0 OR PARAM >2047 OR
1564       SE.   W1    (B8.)        ;      PARAMTYPE<>INTEGER
1566       JL.         A21.         ;      GOTO ERROR1;
1568       SL    W0    2            ;      IF M<2 THEN
1570       JL.         A6.          ;      BEGIN
1572       RS.   W2    B2.          ;        SENSINGS:=INTEGER;
1574       BA    W3    X3+1         ;
1576       RL    W1    X3           ;        GET NEXT PARAMETER;
1578       JL.         A14.         ;
1580  A6:  AC    W2    X2           ;      END;
1582       AL    W1    1            ;NEXTBIT:
1584       LS    W1    X2+23        ;
1586       AC    W1    X1+1         ;
1588       LA.   W1    B13.         ;
1590       RS.   W1    B13.         ;
1592       BA    W3    X3+1         ;      GET NEXT PARAM;
1594       DL    W2    X3+2         ;
1596       SN.   W1    (B8.)        ;      IF PARAMTYPE=INTEGER
1598       JL.         A6.          ;      THEN GOTO NEXTBIT;
1600  A14: BL    W0    2            ;
1602       SL    W0    4            ;      IF -,ENDOFCOMMANDS
1604       JL.         A5.          ;      THEN GOTO NEXTDEVICE;
1606       AL    W0    0            ;
1608       SE.   W0    (B3.)        ;      IF DEVICES >0 THEN
1610       JL.         A112.        ;      GOTO START2 ELSE
1612       JL.         A111.        ;      GOTO START1;
1614                                ;    END TEXTPARAM;
1614
1614  A7:                          ;GETDEVICENO:
1614       SE.   W1    (B5.)        ;    IF PARAMTYPE<>INTEGER
1616       JL.         A21.         ;    THEN GOTO ERROR1;
1618       RL.   W2    B3.          ;
1620       AL    W0    X2+1         ;    DEVICES:=DEVICE+1;
1622       RS.   W0    B3.          ;
```

```
1624
1624        AL   W0   D0+D1          ;
1626        WM   W2   0              ;
1628        AL.  W2   X2+A200.       ;   BUFFER;
1630        SL   W2   X3-D0-D1       ;   IF CORE AREA TOO SMALL
1632        JL.       A24.           ;   THEN GOTO ERROR4;
1634        RL   W0   X3+2           ;
1636        LS   W0   6              ;   DEVICE:=INTEGER SHIFT 6;
1638        RS   W0   X2+D10         ;
1640        LS   W0   -5             ;
1642        RL   W1   74             ;
1644        WA   W1   0              ;
1646        SL   W1   (76)           ;   IF DEVICE >=TOPDEVICE THEN
1648        JL.       A21.           ;   GOTO ERROR1;
1650        RL   W1   (X1)           ;
1652        SE   W1   18             ;   IF KIND=18
1654        SN   W1   34             ;   OR KIND=34 THEN
1656        AL   W1   127            ;   KIND:=127;
1658        SN   W1   60             ;   IF DEVICEKIND=60 THEN
1660        AL   W1   127            ;   DEVICEKIND:=127;
1662        RS   W1   X2+D15         ;   DEVICEKIND(BUFFER):=
1664        AL   W1   X2+D49         ;   DEVICEDESCRIPTOR(0);
1666        RS   W1   X2+D12         ;   WRITEPOINTER(BUFFER):=
1668        RS   W1   X2+D13         ;   READPOINTER(BUFFER):=FIRST BUFFER;
1670        BA   W3   X3+1           ;
1672        RL   W1   X3             ;
1674        SE.  W1   (88.)          ;   IF NEXTPARAM<>INTEGER
1676        JL.       A21.           ;   THEN GOTO ERROR1;
1678        RL   W0   X3+2           ;
1680        SL   W0   0              ;   IF CHANNEL <0 OR
1682        SL   W0   24             ;   CHANNEL >23 THEN
1684        JL.       A21.           ;   GOTO ERROR1;
1686        LS   W0   1              ;
1688        RS   W0   X2+D11         ;   CHANNEL :=CHANNEL SHIFT 1;
1690        RS.  W3   B22.           ;
1692        AM        (0)            ;
1694        RL   W1   16             ;
1696        RL   W1   X1+2           ;
1698        SL   W0   6              ;   IF CHANNEL <6
1700        SN   W1   2              ;   OR CHANNEL KIND=2
1702        AM        1              ;   THEN CHANNEL=VITAL:=TRUE
1704        AL   W3   0              ;   ELSE FALSE;
1706        RS   W3   X2+D16         ;
1708        SL   W0   6              ;
1710        SZ   W1   2.111111       ;
1712        JL.       A19.           ;
1714        AM        24<6           ;
1716        SH   W1   24<6           ;
1718        SH   W1   1<7            ;
1720        JL.       A19.           ;   IF ITR SOURCE IS MULTIPLE
1722        JL.       A8.            ;   THEN GOTO MULTIPLE ITR;
1724  A19:  RL   W1   X2+D15         ;
1726        SH   W1   0              ;   IF KIND <=0 THEN
1728        JL.       A38.           ;   THEN GOTO ERROR8;
1730        AM        (0)            ;
1732        AL.  W1   B4.            ;
1734        RL   W0   X1             ;
1736        SE.  W0   (B10.)         ;   IF CHANNEL NO. ALREADY
1738        JL.       A21.           ;   USED THEN GOTO ERROR1;
1740        AL   W0   X2-2           ;   SET DEVICEBUFFER ADDRESS;
1742        RS   W0   X1             ;
1744        DL.  W0   B15.           ;
1746        DS   W0   X2-2           ;
1748        JL.       A13.           ;
1750
```



```
1750
1750  A8:   AM.       (B29.)         ;MULTIPLE ITR:
1752        SN   W1   X1             ;
1754        RS.  W1   B29.           ;
1756        SE.  W1   (B29.)         ;   IF ANOTHER EXPANDER EXISTS
1758        JL.       A27.           ;   THEN GOTO ERROR7;
1760        LS   W1   -5             ;
1762        WA   W1   74             ;
1764        AL   W1   X1+2           ;
1766        RS.  W0   A18.           ;
1768        RL   W0   (X1)           ;   IF KIND=TTW
1770        JL.  W3   A206.          ;   THEN GOTO COMPUTE SUBCHANNEL;
1772        JL.       A10.           ;
1774        AL   W1   X1+2           ;
1776        RL   W0   (X1)           ;   EXPANDER:=EXPANDER+1;
1778        JL.  W3   A206.          ;   IF KIND<>TTW
1780        JL.       A10.           ;   THEN
1782        JL.       A27.           ;   GOTO ERROR7;
1784  A10:  RL.  W0   A18.           ;COMPUTE SUBCHANNEL:
1786        SL   W1   (76)           ;
1788        JL.       A27.           ;
1790        RL   W3   X2+D10         ;
1792        AM        (X1)           ;
1794        WS   W3   +10            ;   SUBCHANNEL:=DEVICENO.(BUFFER)-
1796        SL   W3   -2<6           ;   DEVICENO(FIRST TMX);
1798        SL   W3   24<6           ;   IF SUBCHANNEL < -2 OR
1800        JL.       A27.           ;   SUBCHANNEL >23 THEN GOTO ERROR7;
1802        SH   W3   -1             ;   IF DEVICEKIND=EXPANDER
1804        JL.       A16.           ;   THEN GOTO TREATEXP;
1806        LS   W3   -5             ;
1808        AL   W1   X2-2           ;
1810        RX.  W1   X3+B7.         ;   SET DEVICEBUFFER ADDRESS;
1812        SE.  W1   A104.          ;   IF SUBCHANNEL ALREADY USED
1814        JL.       A21.           ;   THEN GOTO ERROR1;
1816        AL.  W3   A101.          ;
1818        RL.  W1   B11.           ;
1820  A17:  RS   W1   X2-4           ;
1822        RL.  W1   B15.           ;
1824        RS   W1   X2-2           ;
1826        AM        (0)            ;
1828        RX.  W3   B4.            ;   SET MULTIPLE ENTRY;
1830        SH.  W3   A101.          ;
1832        JL.       +6             ;
1834        AM        (0)            ;
1836        RX.  W3   B4.            ;
1838        AM        (0)            ;
1840        RL   W1   16             ;
1842        BL   W3   X1+1           ;   COMPUTE MULTIPLE LINK;
1844        AL   W3   X3+6           ;
1846        RS.  W3   B24.           ;
1848  A13:  AL   W0   D14            ;   C:=1;
1850        RL.  W3   B22.           ;
1852  A9:   BA   W3   X3+1           ;NEXT SKIPPARAM:
1854        RL   W1   X3             ;
1856        SN.  W1   (B9.)          ;   IF TEXT THEN
1858        JL.       A21.           ;   GOTO ERROR1;
1860        SE.  W1   (B8.)          ;   IF INTEGER THEN
1862        JL.       A11.           ;   BEGIN
1864        SL   W0   D15-2          ;     IF C >=LAST PARAM THEN
1866        JL.       A21.           ;     GOTO ERROR1;
1868        RL   W1   X3+2           ;
1870        AM        (0)            ;
1872        RS   W1   X2             ;     SKIPCHAR(C):=INTEGER;
1874        BA.  W0   1              ;
1876        BA.  W0   1              ;     C:=C+1;
1878        JL.       A9.            ;     GOTO NEXT SKIPPARAM;
1880                                 ;   END;
```

```
1882
1882  A16: AL   W3   X2-2        ;TREATEXP:
1884       AM        (0)         ;
1886       AM        (16)        ;
1888       RL   W1   +2          ;    IF DEVICE(CHANNEL)
1890       SE   W1   (X2+D10)    ;    <>DEVICE(BUFFER)
1892       JL.       A38.        ;    THEN GOTO ERROR8;
1894       AL   W1   1<7         ;
1896       RS   W1   X2+D15      ;    KIND(BUFFER):=EXPANDER;
1898       RL.  W1   B14.        ;
1900       JL.       A17.        ;
1902
1902  A11: SL   W0   D15         ;    WHILE -,LAST SKIPCHAR THEN
1904       JL.       A12.        ;    BEGIN
1906       AL   W1   -1          ;
1908       AM        (0)         ;
1910       RS   W1   X2          ;      SKIPCHAR(C):=ALL ONES;
1912       BA.  W0   1           ;      C:=C+1;
1914       BA.  W0   1           ;
1916       JL.       A11.        ;    END;
1918
1918  A12: AL   W0   0           ;    CLEAR OVERFLOW FLAG;
1920       RS   W0   X2+D17      ;
1922       RL   W1   X3          ;
1924       BL   W0   X3          ;
1926       SL   W0   4           ;ENDDEVS:
1928       JL.       A5.         ;    IF SP THEN GOTO NEXTDEVICE
1930       JL.       A112.       ;    ELSE GOTO START2;
1932
1932  A21: AL.  W0   C1.         ;ERROR1:
1934       JL.       A29.        ;    WRITE(<:***CALL:>);
1936
1936  A22: AL.  W0   C2.         ;ERROR2:
1938       JL.       A28.        ;    WRITE(<:***SUM ERROR:>,SUM);
1940
1940  A23: AL.  W0   C3.         ;ERROR3:
1942       JL.       A28.        ;    WRITE(<:***PROTECTION KEY:>,KEY);
1944
1944  A24: AL.  W0   C4.         ;ERROR4:
1946       JL.       A29.        ;    WRITE(<:***CORE AREA TOO SMALL:>);
1948
1948  A25: AL.  W0   C9.         ;ERROR5:
1950       JL.       A29.        ;    WRITE(<:***PRIMARY INPUT TROUBLE:>);
1952
1952  A26: AL.  W0   C34.        ;ERROR6:
1954       JL.       A29.        ;    WRITE(<:***SLASLA:>);
1956
1956  A27: AL.  W0   C11.        ;ERROR7:
1958       JL.       A29.        ;    WRITE(<:***MULTIPLE ITR TROUBLE:>);
1960
1960  A38: AL.  W0   C10.        ;ERROR8:
1962       JL.       A29.        ;    WRITE(<:***EXPANDER :>);
1964
1964  A28: JL.  W3   A202.       ;    WRITETEXT;
1966       AL   W0   X2          ;
1968       JL.  W3   A201.       ;    WRITE INTEGER;
1970       JL.       A30.        ;    GOTO NO SUCCESS;
1972  A29: JL.  W3   A202.       ;    WRITETEXT;
1974
1974  A30: AL   W2   10          ;NO SUCCESS:
1976       AM.       (B1.)       ;
1978       JL   W3   H33-2       ;
1980       AL   W2   1           ;
1982       AM.       (B1.)       ;
1984       JL   W3   H7          ;    END PROGRAM;
```



```
1986
1986  A51: RL   W1   74          ;TMXSTART:
1988       AL.  W2   A62.        ;
1990  A52: SL   W1   (76)        ;    IF DEVICE >=TOPDEVICE THEN
1992       JL        X2          ;    GOTO NO TMX;
1994       RL   W0   (X1)        ;    GET KIND(DEVICE);
1996       JL.  W3   A206.       ;    IF KIND=TMX THEN
1998       JL.       A54.        ;    GOTO GENERATE ITR;
2000       AL   W1   X1+2        ;    DEVICE:=DEVICE+2;
2002       JL.       A52.        ;    GOTO SET NEXT KIND;
2004  A54: RS.  W1   A66.        ;
2006       AM        (X1)        ;
2008       RL   W1   +10         ;
2010       SH   W1   5<6         ;    IF DEVICE < 6 THEN
2012       JL.       A62.        ;    GOTO NO TMX;
2014       AL   W0   1           ;    FOR EXPANDER:=1,0 DO
2016  A56: AL   W1   X1-64       ;    BEGIN DEVICE:=DEVICE-64;
2018       AL   W2   0           ;      FOR ITR:=0 STEP 2 UNTIL 46 DO
2020  A57: RL   W3   X2+16       ;      BEGIN
2022       RL   W3   X3+2        ;        IF DEVICE(ITR)=DEVICE(EXPANDER)
2024       SN   W1   X3          ;        THEN GOTO DISABLED
2026       JD.       A59.        ;        GENERATE ITR;
2028       AL   W2   X2+2        ;
2030       SH   W2   46          ;
2032       JL.       A57.        ;      END;
2034       JL.       A62.        ;      GOTO NO TMX;
2036  A59: RS   W2   8           ;DISABLED GENERATE ITR:
2038       RL.  W3   B0.         ;      IF EXPANDER=FINISH ITR
2040       SE   W0   1           ;      AND ENTRY=CLEANTMX
2042       SO   W3   1           ;      THEN
2044       JL.       A61.        ;      BEGIN
2046       RL   W3   0           ;
2048       JL.  W1   A109.       ;        TEST SLA;
2050       AL   W0   X3          ;
2052       AM        (X2+16)     ;
2054       BZ   W3   +1          ;        COMPUTE LINKPOINT
2056       AL   W3   X3+6        ;        IN MONITOR;
2058       RS.  W3   B24.        ;
2060       DL.  W3   (B24.)      ;        SWAP MONITOR
2062       DS.  W3   B26.        ;        MULTIPLE ITR RESPONSE
2064       DL.  W3   B25.        ;        CODE;
2066       DS.  W3   (B24.)      ;
2068       AL   W3   -1          ;        SUBCHANNEL(0:23):=ITR;
2070       RS.  W3   B27.        ;      END;
2072  A61: AL.  W3   A60.        ;
2074       RS   W3   10          ;      RETURN:=A60.;
2076       JL        (12)        ;      GOTO MONITOR;
2078  A60: BS.  W0   1           ;
2080       SL   W0   0           ;
2082       JL.       A56.        ;    END EXPANDER;
2084       AL.  W2   A33.        ;    LOOK FOR ANOTHER TMX;
2086       RL.  W1   A66.        ;
2088       AL   W1   X1+48       ;
2090       JL.       A52.        ;
2092  A62: AL.  W0   C5.         ;NO TMX:
2094       JL.       A29.        ;    WRITE(<:***NO TMX:>);
2096  A66:           0           ;
2098
```

```
2098
2098
2098  A70: AC   W2  (0)          ;DISABLED MULTIITR:
2100       AL   W0  1            ;
2102       LS   W0  X2+23        ;
2104       SL   W1  3            ;   IF ITR <3 OR
2106       SN   W0  0            ;   SUBCHANNEL<0 OR >23
2108       JE.      A21.         ;   THEN GOTO ENABLED ERROR1;
2110       RS.  W0  B27.         ;
2112       LD   W2  -23          ;
2114       RL   W1  X2+16        ;
2116       AL   W0  6            ;   COMPUTE MULTIPLE LINK;
2118       BA   W0  X1+1         ;
2120       RS.  W0  B24.         ;
2122       RL   W1  X1+2         ;   IF KIND(EXPANDER)<=1<7
2124       AM       24<6         ;
2126       SH   W1  24<6         ;   OR >2047 THEN GOTO
2128       SH   W1  1<7          ;   ENABLED WRITE(<:***NO TMX:>);
2130       JE.      A62.         ;
2132       SZ   W1  2.111111     ;
2134       JE.      A62.         ;
2136       JL.  W1  A109.        ;   TEST SLA;
2138       DL.  W1  (B24.)       ;
2140       DS.  W1  B26.         ;   SWAP MONITOR
2142       DL.  W1  B25.         ;   MULTIPLE ITR RESPONSE CODE;
2144       DS.  W1  (B24.)       ;
2146       LD   W2  23           ;
2148       BA   W3  X3+1         ;   NEXTCOMMAND;
2150
2150  A71: LS   W1  1            ;DISABLED START ITR:
2152  A72: RS   W1  8            ;   SET ITR NO.;
2154       BL   W2  X3           ;
2156       SL   W2  6            ;   IF -,END COMMAND
2158       JE.      A21.         ;   THEN GOTO ENABLED ERROR1;
2160       SL   W2  4            ;   IF -,LAST COMMAND
2162       AM       A4-A33       ;   THEN RETURN:=NEXTCOMMAND
2164       AL.  W0  A33.         ;   ELSE RETURN:=TERMINATION;
2166       RS   W0  10           ;
2168       JL       (12)         ;   GOTO MONITOR;
2170
2170  A111:AL.  W1  A200.        ;START1:
2172       SL   W1  X3-(:D0+D1:)
2174       JL.      A24.         ;   IF CORE AREA TOO SMALL
2176       AL   W0  1            ;   THEN GOTO ERROR4;
2178       AM       -D1          ;
2180       RS.  W0  B3.+D1       ;   DEVICES:=1;
2182       AL   W0  2<6          ;
2184       AL   W1  4            ;   SEARCH ITR NO. OF MAIN CONSOLE;
2186  A110:AL   W1  X1+2         ;
2188       AM       (X1+16)      ;
2190       SE   W0  (+12)        ;
2192       JL.      A110.        ;
2194       RS.  W1  A200.+D11    ;
2196       AL.  W0  A200.-2      ;
2198       AM       -D1          ;
2200       RS.  W0  X1+84.+D1    ;
2202       AL.  W0  A200.+D49    ;   WRITEPOINTER:=
2204       RS.  W0  A200.+D12    ;   READPOINTER:=
2206       RS.  W0  A200.+D13    ;   FIRST BUFFER;
2208
```



```
2208
2208  A112:AM       -D1         ;START2:
2210       RL.  W1  B1.+D1       ;
2212       RL   W2  X1+H17-2     ;   GET DEVICENO. OF
2214       RL   W3  X2+10        ;   PRIMARY INPUT;
2216       AM       -D1          ;
2218       RS.  W3  B18.+D1      ;
2220       RL   W0  X2           ;
2222       SE   W0  8            ;   IF KIND(PRIMARY)<>8 THEN
2224       JL.      A115.        ;   GOTO TEST TMX;
2226       AL.  W0  A124.        ;
2228       AL   W1  0            ;
2230       AL   W2  X2-2         ;   SEARCH ITRNO. OF PRIMARY;
2232  A113:SN   W2  (X1+16)      ;   IF ADDR(ITR)=PRIMARY
2234       JD.      A118.        ;   THEN GOTO DISABLED STORE ITR;
2236       AL   W1  X1+2         ;   ITR:=ITR+2;
2238       SL   W1  48           ;   IF ITR >46 THEN
2240       JL.      A25.         ;   GOTO ERROR5;
2242       JL.      A113.        ;
2244
2244  A115:AL   W1  X3           ;TEST IF PRIMARY=TMX:
2246       JL.  W3  A206.        ;   IF KIND<>TMX THEN
2248       JL.      A116.        ;
2250       JL.      A25.         ;   GOTO ERROR5;
2252  A116:LS   W1  -5           ;PREVIOUS:
2254  A117:AL   W1  X1-2         ;   DEVICE:=DEVICE-2;
2256       SH   W1  8            ;
2258       JL.      A25.         ;   IF DEVICE <=8 THEN
2260       AM       (74)         ;   GOTO ERROR5;
2262       RL   W0  (X1)         ;   IF KIND=TMX THEN
2264       JL.  W3  A206.        ;   GOTO PREVIOUS;
2266       JL.      A117.        ;
2268       AL   W1  X1-2         ;   DEVICE:=DEVICE-2;
2270       LS   W1  5            ;
2272       AM       -D1          ;
2274       AM.      (B29.+D1)    ;   IF ANY EXPANDER
2276       SE   W3  X3           ;   EXISTS THEN
2278       JL.      A27.         ;   GOTO ERROR7;
2280       AM       -D1          ;
2282       RS.  W1  B29.+D1      ;
2284       AM       -D1          ;
2286       WS.  W1  B18.+D1      ;
2288       AC   W1  X1+128       ;
2290       LS   W1  -5           ;   SUBCHANNEL:=
2292       AL.  W0  A125.        ;     DEVICENO(PRIMARY)-
2294       AM       -D1          ;
2296       RS.  W0  X1+87. +D1   ;     DEVICENO(FIRST TMX);
2298       AM       -D1          ;
2300       RL.  W3  B29.+D1      ;
2302       AL   W1  0            ;   ITR:=0;
2304  A119:AM       (X1+16)      ;
2306       SN   W3  (+2)         ;   IF DEVICE(ITR)=
2308       JD.      A120.        ;   DEVICE THEN GOTO DISABLED STORE ITR;
2310       AL   W1  X1+2         ;   ITR:=ITR+2;
2312       SL   W1  48           ;   IF ITR >=48 THEN
2314       JL.      A25.         ;   GOTO ERROR5;
2316       JL.      A119.        ;
2318
```

```
2318
2318  A120:RL    W2    X1+16        ;   COMPUTE LINKPOINT IN MONITOR;
2320       BL    W3    X2+1         ;
2322       AL    W3    X3+6         ;
2324       AM          -D1          ;
2326       RS.   W3    B24.+D1      ;
2328
2328       AL.   W0    A101.        ;
2330  A118:AL    W1    X1-D1        ;   STORE ITR;
2332       RX.   W0    X1+B4.+D1    ;
2334       AL    W3    -D1          ;   IF CHANNEL ALREADY USED
2336       SE.   W0  (X3+B10.+D1)   ;   THEN GOTO ENABLED ERROR5;
2338       SN.   W0  (X1+B4.+D1)    ;
2340       JL.         +4           ;
2342       JE.         A25.         ;
2344       AL.   W1    A208.        ;
2346  A109:RS.   W1    A209.        ;   PROCEDURE TEST SLA;
2348       RL    W1    78           ;
2350       RL.   W0    X3+C6.+D1    ;
2352  A114:AM         (X1)         ;   IF PROC SLA ALREADY
2354       SN    W0    (+2)         ;   EXISTS THEN GOTO ENABLED ERROR6;
2356       JE.         A26.         ;
2358       AL    W1    X1+2         ;
2360       SE    W1    (80)         ;
2362       JL.         A114.        ;
2364       JL.       (A209.)        ;   RETURN;
2366
2366  A206:SE    W0    36           ;   PROCEDURE TESTTMX;
2368       SN    W0    46           ;
2370       JL          X3           ;
2372       SE    W0    48           ;
2374       SN    W0    52           ;
2376       JL          X3           ;
2378       SE    W0    56           ;
2380       SN    W0    58           ;
2382       JL          X3           ;
2384       JL          X3+2         ;
2386
2386  A208:DL.   W1  (X3+B24.+D1)   ;   SAVE MONITOR
2388       DS.   W1    X3+B26.+D1   ;   MULTI RESPONSE CODE;
2390       AL.   W0    A100.        ;
2392       RS    W0    12           ;   SET NEW ITR ADDRESS;
2394       RL.   W1    X3+B1.+D1    ;
2396       RL    W2    X1+H16       ;   SWAP(PROCNAME,<:<32>SLA:>);
2398       DL    W1    X2+4         ;
2400       DS.   W1    X3+B12.+D1   ;
2402       AM          -D1          ;
2404       DL.   W1    C6.+D1+2     ;
2406       DS    W1    X2+4         ;
2408       JE.         A150.        ;   GOTO ENABLED WAIT;
2410  A209:             0           ;   LINK;
2412
```



```
2412
2412  C.    K-A207-D1
2412  M.    *** OVERLAP ***
2412  Z.
2412
2412  A211:LS    W1    6            ;RESET:
2414       AL    W2    14           ;
2416  A215:AL    W2    X2+2         ;   FOR ITR:=0 STEP 1 UNTIL 24 DO
2418       SL    W2    64           ;   BEGIN IF ITR=24 THEN
2420       JL.         A212.        ;     GOTO NO ITR;
2422       AM        (X2)           ;
2424       RL    W0    2            ;
2426       SL    W0    1<7          ;     IF DEVICE(ITR)=
2428       SE    W0    X1           ;     PARAM THEN
2430       JL.         +4           ;     GOTO DISABLED START ITR;
2432       JL.         A216.        ;
2434       AM        (X2)           ;
2436       SE    W1    (12)         ;
2438       JL.         A215.        ;   END;
2440  A216:AL    W1    X2-16        ;
2442       JD.         A72.         ;
2444
2444  A212:AL.   W0    C35.         ;NO ITR:
2446       JL.         A29.         ;   WRITE(OUT,<:***NO ITR:>);
2448
2448  A210:             0           ;   SUMCOMPLEMENT;
2450
2450  A240:AL    W0    0            ;   FORM CHECKSUM;
2452       AM          -D1          ;
2454       AL.   W1    A0.+D1       ;
2456  A241:WS    W0    X1           ;
2458       AL    W1    X1+2         ;
2460       SE.   W1    A210.        ;
2462       JL.         A241.        ;
2464       RS.   W0    A210.        ;
2466       AL    W0    0            ;
2468       AL    W2    0            ;
2470       JL          X3           ;
2472       JL.         A240.        ;
2474  J.
2474
2474
2474  I.
2474  E.
2474  E.
SLANG OK 1/2474/5
```

RCSL: 44-RT 183
Author: Per Hansen
Edited: December, 197o.

RC 4000

Troubleshooting scheme

Keywords: RC 4000, Hardware, Description

ABSTRACT: This paper gives some hints intended to aid the work of locating and elimination of more commonplace hardware faults in the RC 4000.

A/S REGNECENTRALEN
Falkoner Alle 1
2ooo Copenhagen F.

RC 4000 Troubleshooting

Contents

1.1 RC 4000 CPU and I/O-bus

After autoloading of monitor or loader the "reset" indicator is lit. (NB: ändrsorg strimmel for laser)	Reset the CPU by activating: power ok. kun teknikkøte (Short circuit testpoint A in pos 32 to ground). andre, slukk-tænd cpu
Error still present or undesired "reset" during run.	Check that all registers (incl. MAR) can be set and reset. Note: You cannot set the IR from the TCP. Reset of IR is performed by MAR := x12y13 with SB as a mask. Adjust the timer interval to maximum (1,6 sec) to check that also this bit can be reset. Set and reset of the IM is performed by MAR := x12y12 and SB := contents wanted. If all registers are ok then run the CPU-test which should give informations of errors in any logic element of the CPU.
Trouble during loading the CPU-test	Check whether some device controller clamps the bus. Disconnect bus cables lo61, lo62, lo71, lo72. Note: When one of the buses have been opened or closed the CPU must be reset by short circuiting testpoint A, pos 32 to ground.
The "RESET" button has no effect.	Check that the key on the operator's panel is in operator mode and that the key on the maintenance panel (TCP) is not in technical mode.
Still no effect.	Reset the CPU by shortcircuiting testpoint A on the print pos 32 in the CPU to ground. stop start cpu

RESET FEJL ?

1.2. RC 4000 CPU intermittent errors

The RC 4000 runs for a long time and then suddenly terminates in "reset".

Let the RC 4000 cycle in a test program (e.g. CPU test or a test loading the I/O buses) and check for connector faults by vibrating the Printcards: let a finger ripple gently along the cardrows and take care that the testpoints do not touch the cards beside, or better, use a special vibration tool.

If possible connect the program error messages to the punch to provide a quick response on any error.

Also try to make the error permanent by raising/lowering of the internal dc voltages.

First of all press "MAR manual controlled" and record the contents of all registers.
Afterwards try to determine the reason for the reset. There is three possible reasons:

1. Power ok may have failed for a moment. The register FR=0. Other registers are undefined.

   If the restart signal on the power ok card is set into the "disable mode" the situation will be "locked" next time the CPU power fails. Note: The power ok signal also comprehends the core store power supervision.

2. An AW instruction has been executed in monitormode due to program error: FR=0; SC=3 (or if the tapereader was not quite empty then 2 or 1 or 0); SB=IC; BR=48; AE=0; EX(22:23)=0; SE=1 con 23 ext 0. The other registers are undefined.

3. The signal "Main power key on and -, RESET" has been false:
   SB=IC; FR=0; Other registers undefined.

In the cases 2 and 3 IC will always point to the word after the last instruction executed, even if this was a jump. The only exception is the situation right after interrupt (only relevant in case 3) where IC points to the first word of the interrupt response routine, i.e. the address stored in word 12.

1.3 Core store

"Core store parity" indicator is lit.	Run the program: "clear core store for parity error".
Parity error is still present.	Disable the core store parity check and run the core store tests.
Address selection error.	Determine the module selected and interchange it (if possible) to determine whether the error is located in- or outside the module.
Error in the module. Error outside the module.	Interchange the address selection cards. Check the STC-bus multiplexer.
Error in bitpattern, Superfluos ones, - " - zeroes	Raise the treshold voltage in the read amplifier. Lower " " " " " " "
Error still present, one fixed bit.	Check read amplifier and inhibit driver for the bit in question.

1.4 Digital Clock (TIM)

The software funtion "date" yields a wrong time or the clock process provides incorrect intervals.

1. Check the timer interrupt interval at the proper IR-bit. Turn the timer switch on the TCP to check minimum and maximum intervals.

2. If the intervals are wrong then start the sladrehank testprogram connected to the device- and channelnumber of the clock and check that the increment resembles the interval switch position. Note that the increment is given with a unit of o.1 ms.

1.5 Lineprinter

One hammer is printing too low.

Change the hammer in question.

One hammer is printing too high.

The flighttime may be too short due to a too high current. If the position moves when the hammerdrivers are interchanged the fault is due to an error on the driver card. Change the Q3 transistor.

If Q3 burns again then change the Q1 transistor too.

If the bad position does not follow the hammerdriver the error may be due to dust in the hammer assembly.

Note: Hammerdrivers do not work properly when mounted on extendercard.

1.6 Drum

Drumtest gives dataerror but not statuserror.	Check the highspeed bus transmitters and receivers and the internal data- and address registers of the DRC. Note: The current core store address is also transferred via the highspeed bus.
Statuserror but no dataerror.	Check lowspeed bus transmitters.
Dataerrors and statuserrors, intermittent and on various segments.	Usually due to electrical noise in the I/O cabinet. Check or turnoff the fans (only for a short time). Also check the peak detector and the data shift register.
Intermittent dataerror, confined to a fixed group of segments.	Error in the head selection circuits or loose taper pins inside the drum cover. Note: DO NOT OPEN DRUM UNLESS YOU ARE SPECIALLY TRAINED.
After reading from drum (or disc) the program is destroyed.	Place the program at another location in the core store by means of the relocatable loader.
Program is destroyed in various places in the core store.	May be due to an incomplete or erroneous address transfer on the lowspeed-bus.

1.7 System lock

The operating system "s" will not accept any messages from the consoletypewriter.

Usually because "s" is in the "wait answer" situation. This means that the monitor waits for an interrupt from a highspeed device which has been "locked" or has lost an interrupt. If the device is a tapeunit this can be remedied by pressing the "Remote" button with the tape at loadpoint.

If the device is a drum/disc then try to disconnect and re-connect the power of the IO-cabinet.

Further error location must be carried out by means of a loader controlled testprogram.

2.1 The logical statusword.

After any transfer of a block of data between two processes (usually an internal and a peripheral process) the logical statusword is formed to enable the fileprocessor or the ALGOL running system to check the transfer.

If there is a "hard error" i.e. the automatic error recovery could not succeed the logical statusword is output on the operators console, bit by bit and every one represented by a name. The peripheral process in question is not identified by the devicenumber but with the document name assigned to it.

The logical statusword is composed by adding the 12 leftmost bits of the hardware statusword to some softwaregenerated statusbits placed in the rightmost 12 bits. However, in some cases one or two of the leftmost bits may be softwaregenerated.

2.2 Significance of the statusbits.

0. Intervention.
   The device was set in local mode during the operation, presumably because the operator changed the paper or the like.

1. Parity error.
   A parity error was detected during the block transfer.

2. Timer.
   The operation was not completed within a certain time defined in the hardware.

3. Data overrun.
   The high speed channel was overloaded and could not transfer the data.

4. Block length.
   A block input from magnetic tape was longer than the buffer area defined for it.

5. End of document.
   Depends on the devicekind: The tape reader was empty, paper low on printer or punch, EOT on magnetic tape, paper out on typewriter.

   If reading or writing outside a backing storage area is attemted the software generates this bit.

6. Load point.
   Load point was sensed after an operation on magnetic tape.

7. Tape mark.
   A tape mark was sensed or written on the magnetic tape.

   The attention key was pressed on the typewriter. Software generated.

8. Write enable.
   A write enable ring is mounted on the magnetic tape.

9. High density.
   The magnetic tape is in high density mode.

10. ~~Unused.~~

11. Unused.
    The following bits are all software generated.

12. Unused.

13. Unused.

14. Unused.

15. Output lost or stopped.
Generated by the check routine when less than wanted was output to a document of any kind or less than wanted was input from a backing storage area.

16. Word defect.
Generated by the check routine when the number of characters transferred to or from a magnetic tape is not divisible by the number of words transferred, i.e. when only a part of the last word was transferred.

17. Position error.
Generated by the check routine after magnetic tape operations when monitors count of file and block number differs from the expected value in the zone descriptor.

18. Unknown.
The document is unknown by the monitor, e.g. a tapestation has been set to local.

19. Malfunction.
The device has been busy or disconnected or has remained busy after interrupt.

2o. Illegal.
The operation attemted is illegal for that device, e.g. input from a printer.

21. Reservation.
The program must not use the document, or it should be reserved first.

22. Normal answer.
The device has attemted to execute the operation.

23. Harderror.
The standard error action has classified the transfer as a hard error, i.e. error recovery could not succeed.

The bits 18 - 22 resembles the result from the monitor. These bits exclude each other.

2.3 Document name.

The document name (names) assigned to a device needs the following explanation:

The printer, punch, typewriter usually have a fixed name, e.g. printer, punch, consolel, console2 .....etc.

The tapestations are named after the name of the tape roll mounted, e.g. mt 12o37.

During transfers to or from backing-storage the area is named after the area to which the transfer takes place. Thus the following error message

```
xxxdevice status algol
   malfunction
```

means that there has been trouble during reading or writing in the area algol.

If the backing storage is consisting of more devices, e.g. one drum and two discs it is possible to determine the device number and segment number of a bad area by means of the fp utility program lookup:

```
lookup cat.yes algol
```

This may give the following output:

```
algol 4 o 7 lol5
   52 o o o o o o
```

Here lol5 is the logical number of the first segment, and it is seen that the size is 52 segments.

The meaning of the logical segment number is that the backing storage devices alltogether are considered as one big area. If the drum has a size of 128 K = 512 segments, the logical segment number = lol5 - 512 gives segment number 5o3 on the first disc drive unit.

Note: If the message:

Parity error on <document name>

appears in the editor this only means that a SUB character is met. This is inserted every time a parity error occurs in tapereader or typewriter input and will remain in the text string even if this is stored on drum or tape.

Example

parity error on mt014711

means that the editor has read a text containing a SUB character from the tape mt014711, not that a parity error has occured on the tape-station.

RCSL: 44-D15
Author: Per Hansen
Edited: November 1971
Type: ALGOL6 program

RC4000

TIMESHARED TESTPROGRAM LIBRARY

BACKING STORAGE

KEYWORDS: RC4000, Diagnostic program, ALGOL program, ISO tape

ABSTRACT: This program contains 3 routines intended for error location and maintenance purposes on the RC4000 drum and disc. It is organized as a block incorporated in the 'TEST' system.

A/S REGNECENTRALEN
Falkoner alle 1
2000 Copenhagen F.

Contents:

1. Call of program

The backing storage test may be used for simultaneous testing of the backing storage devices of the RC4000. It must be run in a process area of at least 14000 bytes with the catalog key 0 and 3. The process may be created in this way:

```
att s
new bstest size 14000 catalog 0 3 run
```

The program is started in this way:

<outfile> =test bs. <dev1> . <dev2> .....

for example

lp=test bs.4.13

means that the test will be executed on device 4 and device 13 which may be a drum and a disc respectively. The lineprinter will be used for output of errormessages.

2. Reservation of test areas

When the testprogram is started it tries to reserve a test-area on each of the devices specified in the call.

The areas reserved are the maximum available. They are named

bsxxx

where xxx stands for the devicenumber. For example, a test-area on device 13 will be named:

bs013

If areas of these names already exist they will be used instead. In this case it will not be checked that the area is located on the corresponding device, i.e. bs013 actually might be reserved on the device 4.

The areas are permanented with catalog key 3.

When the test-areas have been created, a listing of areas is performed. The listing has the following format:

device <d>: bsxxx yyyy zzzz

where d=devicenumber, bsxxx=area name, yyyy=number of the first (physical) segment of the area and zzzz=the area size (in segments).

If no area is available on the device in question, the message:

permanent bsxxx no area

is output.

When the testareas are created, the operator is requested to select the actual testprogram as usual.

After testing, the test-areas may be removed by means of testprogram y: remove and terminate. The test-areas may, of course, be removed by means of the clear-command, too.

If program termination without removal of the test-areas is desired, test-program z may be selected.

3. Write and read test

This program is able to perform writing and reading of testdata in the test-areas. Further, it is possible to specify checking of the status-word and the data.

First the program writes:

blocksize (segments) =

and the operator is expected to type the blocksize desired. If only a <NL> is typed or the blocksize specified is too big, the message:

max. blocksize = <blocksize>

is output. Next the program asks:

mode=

and the operator may specify various modes.

The possible modes are the following:

write
read
monitor
statuscheck
checkall
print
data <datakind>
try <tries>

Only the first letter of the word in question needs to be typed. <datakind> and <tries> are numbers. The words must be separated by spaces.

If only a NL is typed, mode is initialized to:

write,read,statuscheck,data0,try0

The effect of the various modes is

write In each run the entire testareas are written in sequence from the beginning to the end.

read In each run the entire testareas are read in sequence from the beginning to the end. If both write and read are specified, all the write operations will be executed at first.

monitor After completion of read or write of the entire test-areas, the relative number of statuserrors per device is printed on <outfile>. The printing is suppressed, if no errors have occured.

statuscheck Has the same effect as monitor. Further, the statusword is checked, and if an error is detected, the bad statusword is printed in text form on <outfile> together with cylinder,- head,- and sector no., if the device was a discfile, or bar,- head,- and sector no., if the device was a drum.

checkall Has the same effect as described for monitor and statuscheck. Furthermore, the entire data contents of the block is checked, and the inputbuffer is cleared before every inputoperation.
The bad statusword is printed as mentioned above. Additionally the message

dataerror

is output.

print Has the same effect as described for monitor, statuscheck and checkall. Further, the contents of the block is printed in case of dataerror.

data This parameter must be followed by a number which indicates the testdata desired. One of the following numbers must be used:

0 worst-case bit-pattern
1 all zeroes
2 all ones
3 two zeroes, two ones, two zeroes, etc.
4 two ones, two zeroes, two ones, etc.

If data is not specified, the bitpattern corresponding to data 0 is used.

try This parameter must be followed by a number which indicates the number of rereadings or rewritings to be executed in case of status-error.

No errormessage will occur in case of statuserror, except if the number of tries specified is exceeded.

If try is not specified, it is set to zero.

4. Random head move test

This test-program performs a number of random head movements and after each movement the position may be checked.

First the program ask:

check,tries=

and the operator is expected to type either yes or no followed by a number which defines the number of rewritings or rereadings to be performed in case of statuserror. If only a NL is typed, check is set to yes, and tries is set to zero. Next the physical segment number is written in the beginning of each segment. The rest of the segment is cleared.

In each run a number of read-operations is now performed,the total number of which equals ten times the number of segments in the testarea. The read-operations are, however, performed on random segments inside the area.

If check was set to yes, the segment number is checked against the contents of the segment read. If the contents are wrong, an errormessage will be output together with the cylinderhead- and sector no.

After the end of each run the relative number of statuserrors is output (independent of check being yes or no). If the number of error in the run was zero, the message is suppressed.

5. Head step test

This testprogram is intended for use during the adjustment procedures of the disc-file where a continous stepping between two cylinders is required. The operator may specify the first cylinder and the number of cylinders to be stepped. The cylinders must be inside the testareareserved.

First the program asks:

first cylinder =

and the operator should type the number of the first cylinder. If it is outside the test-area, it is automatically set to the first cylinder of the area. Next the program asks:

step (cylinders)=

and the operator types the step size desired. If the step is too big, it is automatically set to the maximum available.

In the case of automatic correction of the cylinder number a message defining the actual cylinder no. is output.

In each run two read operations is performed corresponding to a complete step. The statusword is not checked.

6. Routine-test,(testprogram x)

In each run of the routinetest the following is performed:

a. write and read test with the parameters:
write,read,checkall,data0,try0 and the maximal blocksize possible.

b. random head move test with parameters:
check,tries=yes0

7.Examples

Example 1: creation of testareas (☐ denotes input)

```
att s
new peh size 20000 catalog 0 3 run
ready

to peh
test  bs.4.13.14

RC 4000 backing storage

device  4:  bs004   629    15
device 13:  bs013  7583   256
device 14:  bs014  2406  5714
testprogram: z




end

   ; comment testprogram z terminates 'test' without
   ;         removing the testareas;
```

Example 2:

```
   ; comment bs014 is too large and may be shortened
   ;         in this way:

bs014=set 200
test  bs.4.13.14

RC 4000 backing storage

device  4:  bs004   629    15
device 13:  bs013  7583   256
device 14:  bs014  2406   200
testprogram: a
number of runs = 11

blocksize (segments) = 5

max blocksize =  2 segments
mode =

run no.       1

run no.      11


test end
```

```
testprogram: y



end

   ; comment testprogram y removes the testareas  corresponding
   ;         to the devicenumbers specified when 'test' was started;
```

Example 3:

```
test  bs.13

RC 4000 backing storage

device 13:  bs013  7583   256
testprogram: z



end
bs013=set 50
test  bs.13

RC 4000 backing storage

device 13:  bs013  7583    50
testprogram: b
number of runs = 1

check,tries = y3

run no.       1


test end

testprogram: y



end
```

Example 4:

```
test  bs.14

RC 4000 backing storage

device 14:  bs014  2406  5714
testprogram: c
number of runs = 111

device  14
first cylinder = 99

step (cylinders) = 3

run no.       1

run no.     101


test end

testprogram: y




end
```

Example 5:

```
test  bs.13

RC 4000 backing storage

device 13:  bs013  7583   256
testprogram: a
number of runs = 9

blocksize (segments) = 3

mode = read print monitor try0
```

```
run no.       1

*13  1536 bytes
   1'run segment 7583 reading cylinder 23 head 5 sector 3 dataerror
words:     0:                          1:
received: ...... .....1 11.11. .11111 ...... ...... ...... ......     7583        0
expected: ....11 .....1 1..... 11.... .11... ..11.. ...11. ....1.   792624  6340994

words:     2:                          3:
received: ...... ...... ...... ...... ...... ...... ...... ......        0        0
expected: .....1 1..... 11.... .11... ..11.. ...11. ....11 .....1   396312  3170497

words:     4:                          5:
received: ...... ...... ...... ...... ...... ...... ...... ......        0        0
expected: .1.... 1.1... .1.1.. ..1.1. ...1.1 ....1. 1....1 ..1...  4359434  1321032

  etc.
```

8. Errormessages

*<number>,<status><number> bytes

During test of a device there has been a statuserror. The first <number> is a devicenumber, the second is the number of bytes transferred. <status> is the statusword printed as texts.

This errormessage will usually be followed by a message determining cylinder-head-and sector no. (or headbar,head and sector no. in case of drum). If the blocksize is bigger than 1 segment, this message points to the first segment of this block.

***core area too small

The process area is too small for creating even 1-segment buffers. A minimum of 14000 bytes is needed.

*input parity

parity error on the typewriter.

**lookup <areaname> caterror

statuserror on the catalog device

***permanent <areaname> area reserved

The area is used by another process

***permanent <areaname> cat protect

Catalog protection. The process does not own the catalog key 3.

***permanent <areaname> no area

Not even a single segment is available on the corresponding backing device.

***test call

TEST has been called in a wrong way

***test device <number> no <devicekind>

is a warning about a wrong devicekind.

TEST terminates.

9. Program text

```
RC 4000 BACKING STORAGE TEST                                    PAGE 1
RCTS 2.5/0      1.8.71  PEH
VERSION         1.8.71
NAME IN CATALOG: TESTBS;


BEGIN REAL ARRAY BS_NAME(1:DEVS);
    INTEGER ARRAY FIRST_SEGM(1:DEVS),AREA_SIZE(1:DEVS),
    ERRORS(1:DEVS),OPERATIONS(1:DEVS);
    INTEGER SEGMENT,SEGMENTS,I,BUF_SIZE,CHECKB,MODE,DATA,TRY,SAVEBUF;
    BOOLEAN BSE,DTE,WRITING;

    INTEGER PROCEDURE FUNCTION(F,NAME,INDEX,SIZE);
    INTEGER F,INDEX,SIZE; REAL ARRAY NAME;
    BEGIN COMMENT THIS PROCEDURE EXECUTES ONE OF THESE FUNCTIONS:

          CALL VALUES                              RETURN VALUES

    F=1   CREATE (NAME,INDEX)                FUNCTION=SIZE OR -RESULT
    F=2   LOOKUP (NAME,INDEX)                FUNCTION=DEVICENUMBER OR -RESULT
                                             INDEX=FIRST SEGMENT, SIZE=AREASIZE
    F=3   CHANGE (NAME,INDEX)                FUNCTION=-RESULT
    F=4   RENAME (NAME,INDEX),(BSNAME(SIZE))  DO.
    F=5   REMOVE (NAME,INDEX)                FUNCTION=-RESULT
    F=6   PERMANENT (NAME,INDEX)  KEY=SIZE     DO;

       INTEGER I,RESULT;
       ZONE Z(1,1,STDERROR);
       INTEGER ARRAY IA(1:20);
       I:=INDEX;
       IF F=1 THEN NAME(INDEX):=0.0;
       OPEN(Z,0,STRING NAME(INCREASE(I)),0);
       CASE F OF
       BEGIN IA(1):=SIZE;;;
          BEGIN TDR:=4;
             IA.TDR:=6SNAME(SIZE);
             IA(3):=IA(4):=0;
          END;;
          I:=SIZE;
       END;
       RESULT:=MONITOR(F*2+38,Z,I,IA);
       IF F=1 THEN
       BEGIN GETZONE(Z,IA);
          FOR TDR:=6,10 DO NAME(INDEX+TDR SHIFT (-3)):=IA.TDR;
       END;
       I:=INDEX;
       IF RESULT >0 AND RESULT<3 OR
       RESULT >=3 AND F>2 AND -,(F=5 AND RESULT=3) THEN
       WRITE(OUTC,NL,1,<:***:>,CASE F OF
       (<:CREATE_:>,<:LOOKUP_:>,<:CHANGE_:>,<:RENAME_:>,
       <:REMOVE_:>,<:PERMANENT_:>),STRING NAME(INCREASE(I)),
       CASE RESULT OF (<:_FUNCTION NOT ALLOWED:>,
       <:_CAT ERROR:>,<:_NO AREA:>,<:_CAT PROTECT:>,<:_AREA RESERVED:>,
       <:_NAME FORMAT:>));
       IF F=2 THEN RESULT:=RESULT+MONITOR(52,Z,I,IA) SHIFT 3;
       COMMENT CREATE AREA PROCESS;


COMMENT
```

```
RC 4000 BACKING STORAGE TEST                                    PAGE 2

       IF (RESULT=0 OR RESULT=24) AND F<3 THEN
       BEGIN IF F=1 THEN FUNCTION:=SIZE ELSE
          BEGIN SYSTEM(5,MONITOR(4,Z,I,IA),IA);
             COMMENT GET PROCESS DESCRIPTION;
             FUNCTION:=IA(6) SHIFT (-13);
             COMMENT DEVICENUMBER;
             INDEX:= IA(9); COMMENT FIRST SEGMENT;
             SIZE:=IA(10);
             MONITOR(64,Z,I,IA);
             COMMENT REMOVE AREA PROCESS;
          END;
       END ELSE FUNCTION:=-RESULT;
RETURN:
    END PROCEDURE FUNCTION;


    PROCEDURE BS_ERROR(Z,S,B);
    ZONE Z; INTEGER S,B;
    BEGIN OWN INTEGER TRIES;
       INTEGER PHYSSEG;
       BSE:=S SHIFT (-19) > 0 OR S SHIFT (-9) EXTRACT 9 >0
       OR S SHIFT (-2) EXTRACT 6 >0 OR S EXTRACT 1 >0;
       OPERATIONS(D):=OPERATIONS(D)+1;
       IF S SHIFT (-18) EXTRACT 1=1 THEN
       BEGIN IF SEGMENT+BUF_SIZE >AREA_SIZE(D) OR SEGMENT< 0 THEN
          BEGIN IF -,WRITING THEN SETPOSITION(Z,0,0);
          END ELSE BSE:=TRUE;
       END;
       IF BSE THEN
       BEGIN ERRORS(D):=ERRORS(D)+1;
          IF INCREASE(TRIES)< TRY THEN
          BEGIN GETZONE(Z,IA);
             MONITOR(16,Z,IA(16),IA); COMMENT REPEAT MESSAGE;
             CHECK(Z);
             GOTO RETURN;
          END;
       END;
       TRIES:=0;
       IF -,(DTE OR BSE) OR CHECKB<2 THEN GOTO RETURN;
       ERROR(Z,S,B);
       PHYSSEG:=FIRST_SEGM(D)+SEGMENT;
       WRITE(OUTL,<<_DDD>,RUNNO,<:'RUN SEGMENT:>,
       PHYSSEG, IF WRITING THEN <:_WRITING:> ELSE <:_READING:>);
       IF FALSE ADD DKIND(D) THEN
       WRITE(OUTL,<:_CYLINDER:>,PHYSSEG MOD 40,
          <:_HEAD:>, PHYSSEG SHIFT (-2) MOD 10) ELSE
       WRITE(OUTL,<:_HEADBAR_:>,FALSE ADD (PHYSSEG SHIFT (-8)+65),1,
          <:_HEAD:>,PHYSSEG SHIFT (-2) EXTRACT 6);
       WRITE(OUTL,<:_SECTOR:>,PHYSSEG EXTRACT 2); UDL;
RETURN:
    END PROCEDURE BS_ERROR;


    PROCEDURE BS_END(Z);
    ZONE ARRAY Z;
    BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO SETPOSITION(Z(D),0,0);
       FOR D:=1 STEP 1 UNTIL DEVS DO IF ERRORS(D) >0 AND CHECKB >0 THEN
       BEGIN WRITE(OUTL,NL,1,<:*:>,<<DD>,DEVICENUMBER(DEVICE(D)),
          <:_MONITOR_:>,IF WRITING THEN <:WRITE::> ELSE <:READ::>,
          <<_BD.DOO>,IF OPERATIONS(D)=0 THEN 0.0 ELSE
          100*ERRORS(D)/OPERATIONS(D),<:_%<10>:>); UDL;
       END;
       RESMON;
    END PROCEDURE BS_END;
```

```
547
547
547     PROCEDURE RESMON;
548     FOR D:=1 STEP 1 UNTIL DEVS DO OPERATIONS(D):=ERRORS(D):=0;
549
549
549     PROCEDURE WRITE_AND_READ(DATA,MODE);
550     INTEGER DATA,MODE;
551     BEGIN ZONE ARRAY BS(DEVS,BUF_SIZE*256,2,BS_ERROR);
552         REAL ARRAY TESTDATA(1:BUF_SIZE*128);
553         INTEGER I,J;
554         OPEN_BS(BS);
555         IF RUNNO=1 THEN
556         BEGIN IF DATA >0 THEN
557             BEGIN FOR TDI:=2 STEP 2 UNTIL BUF_SIZE*512 DO
558                 TESTDATA.TDI:=CASE DATA OF (0,-1,3355443,-3355444);
559             END ELSE INITDATA(TESTDATA);
560         END;
561
561         IF FALSE ADD MODE THEN
562         BEGIN WRITING:= TRUE;
563             FOR SEGMENT:= (-2*BUF_SIZE) STEP BUF_SIZE UNTIL
564                 SEGMENTS, -2000, -3000 DO
565             FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
566             BEGIN OUTREC6(BS(D), BUF_SIZE*512);
567                 IF SEGMENT<0 THEN FOR TDR:=4 STEP 4 UNTIL BUF_SIZE*512 DO
568                 BS(D).TDR:=IF SEGMENT < -1024 THEN
569                   0.0 SHIFT 48 ELSE TESTDATA.TDR;
570             END;
571             BS_END(BS);
572         END WRITING;
573
573         IF MODE >1 THEN
574         BEGIN WRITING:=DTE:=FALSE;
575             FOR SEGMENT:=0 STEP BUF_SIZE UNTIL SEGMENTS-1 DO
576             FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
577             BEGIN DTE:=FALSE;
578 REP:            J:=INREC6(BS(D),0);
579             IF J=0 THEN GOTO IF BSE THEN SKIP ELSE REP;
580             INREC6(BS(D),J);
581             IF CHECKB >2 THEN
582             FOR TDR:=4 STEP 4 UNTIL J  DO
583             BEGIN IF BS(D).TDR<>TESTDATA.TDR THEN
584                 BEGIN IF -,DTE THEN
585                     BEGIN DTE:=TRUE;
586                         IF -,BSE THEN BS_ERROR(BS(D),0,J);
587                         WRITE(OUTL,<:_DATAERROR:>);
588                     END;
589                     IF CHECKB >3 THEN
590                     BEGIN WRITE(OUTL,<:<10>WORDS::>,
591                         <<DDDDDD>,TDR SHIFT (-1)-2,
592                         <:::>,FALSE ADD 32,21,TDR SHIFT (-1)-1,<:::>);
593                         PRINTBITS(0,BS(D).TDR,TESTDATA.TDR,6,48);
594                     END;
595                 END;
596                 BS(D).TDR:=0.0 SHIFT 48;
597             END;
598 SKIP:       IF BSE AND -,DTE THEN
599             BEGIN WRITE(OUTL,NL,1); UDL;
600             END;
601         END DEVS;
602         BS_END(BS);
603     END READING;
604 END PROCEDURE WRITE_AND_READ;
605
605
605 COMMENT
```



```
606
607
607
607     PROCEDURE MAXBLOCK;
608     BEGIN INTEGER FREEBYTES;
609         FREEBYTES:=0.7*SYSTEM(2,1,PARAM)-2800;
610         IF BUF_SIZE*(DEVS*2+1)*512 >FREEBYTES THEN
611         BEGIN BUF_SIZE:=FREEBYTES//(DEVS*2+1)//512;
612             IF BUF_SIZE<=0 THEN
613             BEGIN WRITE(OUTC,NL,1,<:***CORE AREA TOO SMALL:>);
614                 GOTO TESTEND;
615             END;
616             WRITE(OUTC,NL,1,<:MAX BLOCKSIZE = :>,
617                 BUF_SIZE, <:_SEGMENTS:>);
618         END;
619     END PROCEDURE MAXBLOCK;
620
620     PROCEDURE MOVETEST(X); BOOLEAN X;
621     BEGIN INTEGER TEST,I;
622         IF RUNNO=1 OR X THEN
623         BEGIN ZONE ARRAY BS(DEVS,256,2,BS_ERROR);
624             OPEN_BS(BS);
625             WRITING:=TRUE;
626             FOR SEGMENT:=-2  STEP 1 UNTIL SEGMENTS DO
627             FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
628             BEGIN OUTREC6(BS(D),512);
629                 BS(D,1):=0.0  SHIFT 24 ADD (SEGMENT+FIRST_SEGM(D)+2) SHIFT 24
630                 IF SEGMENT< 0 THEN FOR I:=2 STEP 1 UNTIL 128 DO
631                 BS(D,I):=REAL<::>;
632             END;
633             BS_END(BS);
634         END;
635         BEGIN ZONE ARRAY BS(DEVS,128,1,BS_ERROR);
636             OPEN_BS(BS);
637             WRITING:=FALSE;
638             I:=0;
639             TDI:=2;
640             FOR TEST:=1 STEP 1 UNTIL SEGMENTS*10 DO
641             FOR D:=1 STEP 1 UNTIL DEVS DO IF RESERVED(D) THEN
642             BEGIN SEGMENT:=RANDOM(1)*(AREA_SIZE(D)-1);
643                 SETPOSITION(BS(D),0,SEGMENT);
644                 IF INREC6(BS(D),0) >0 THEN
645                 BEGIN INREC6(BS(D),512);
646                     IF BS(D).TDI<>SEGMENT+FIRST_SEGM(D) AND CHECKB >1 THEN
647                     BEGIN IF -,BSE THEN BS_ERROR(BS(D),0,512);
648                         WRITE(OUTL,<:SEEK ERROR:>);
649                         PRINTBITS(10,BS(D,1),
650                             0,0 SHIFT 24 ADD
651                           (SEGMENT+FIRST_SEGM(D)) SHIFT 24,8,24); UDL;
652                     END;
653                 END;
654             END;
655             BS_END(BS);
656         END;
657     END PROCEDURE MOVETEST;
658
658     PROCEDURE OPEN_BS(Z); ZONE ARRAY Z;
659     FOR D:=1 STEP 1 UNTIL DEVS DO OPEN(Z(D),MODEKIND(DEVICE(D)),
660         STRING BS_NAME(D), -1 SHIFT 9+3 SHIFT 6+3 SHIFT 3+3);
661
661
661
661 COMMENT
662
```

```
663
663 BSTEST_START:
664
664     BEGIN COMMENT THIS BLOCK CATERS FOR THE CREATION
665         OF AN AREA OF MAXIMUM SIZE ON EACH OF
666         THE BACKING DEVICES IN QUESTION;
667         INTEGER ARRAY IA(0:TOPDEV);
668         REAL ARRAY NAME(1:512);
669         INTEGER I,J,INDEX;
670         BOOLEAN RES;
671
671         INTEGER PROCEDURE CREATEMAX(NAME,INDEX);
672         REAL ARRAY NAME; INTEGER INDEX;
673         COMMENT THIS PROCEDURE CREATES THE MAXIMUM
674         AREA POSSIBLE ON THE BACKING STORAGE BY
675         MEANS OF THE BI-SECTION METHOD;
676         BEGIN INTEGER SIZECREMENT,RESULT,SIZE;
677             SIZE:=2**12;
678             SIZECREMENT:=2**11;
679 CM_START:RESULT:=FUNCTION(1,NAME,INDEX,SIZE);
680             COMMENT SET;
681             IF SIZECREMENT=0 THEN GOTO TERM;
682             IF RESULT>=0 THEN
683             BEGIN SIZE:=SIZE+SIZECREMENT;
684                 FUNCTION(5,NAME,INDEX,SIZE);
685                 COMMENT CLEAR;
686             END ELSE SIZE:=SIZE-SIZECREMENT;
687             SIZECREMENT:=SIZECREMENT SHIFT (-1);
688             GOTO CM_START;
689 TERM:       CREATEMAX:=IF RESULT>=0 THEN RESULT ELSE
690             FUNCTION(1,NAME,INDEX,SIZE-1); COMMENT SET;
691         END PROCEDURE CREATEMAX;
692
692 CREATE_START:
693
693         WRITE(OUTC,NL,1,<:RC 4000 BACKING STORAGE:>,NL,1); UD;
694         FOR D:=1 STEP 1 UNTIL DEVS DO
695         BEGIN I:=DEVICENUMBER(DEVICE(D));
696             BS_NAME(D):=REAL(<:___BS:>) ADD (I//100+48)
697             SHIFT 8 ADD ((I MOD 100)//10+48) SHIFT 8
698             ADD((I MOD 100) MOD 10+48) SHIFT 8;
699         END;
700
700         RESMON;
701         PARITY:=2;
702         SYSTEM(5,74,IA);
703         SYSTEM(5,IA(0),IA);
704         FOR D:=1 STEP 1 UNTIL DEVS DO
705         BEGIN I:=D;
706             J:=FUNCTION(2,BS_NAME,I,J);
707             COMMENT LOOKUP DEVICENUMBER;
708             IF J>=0 THEN
709             BEGIN RESERVED(D):=TRUE;
710                 DEVICE(D):=IA(J);
711             END;
712             IF J=-32 THEN FUNCTION(5,BS_NAME,D,I);
713             COMMENT REMOVE;
714         END;
715
715
715
715 COMMENT
716
```



```
716
717
717
717
717         SYSTEM(5,116,IA); COMMENT THIS PIECE OF
718         CODE STORES INFORMATION IN DKIND BIT 11
719         ABOUT DISC OR DRUM KIND. A MAXIMUM
720         OF FIVE BACKING DEVICES MAY BE INVESTIGATED;
721         SYSTEM(5,IA(0)-8,IA);
722         SYSTEM(5,IA(0),IA);
723         FOR I:=0 STEP 3 UNTIL 16 DO IF IA(I) >0 THEN
724         BEGIN IF IA(I+4)-IA(I+1) >4000 THEN
725             FOR D:=1 STEP 1 UNTIL DEVS DO IF
726             DEVICENUMBER(DEVICE(D))=IA(I) SHIFT (-13) THEN
727             DKIND(D):=DKIND(D) SHIFT(-1) SHIFT 1 ADD 1;
728         END;
729
729         RES:=FALSE;
730         FOR I:=1 STEP 2 UNTIL 510 DO
731         BEGIN IF CREATEMAX(NAME,I)< 1 OR RES THEN GOTO EXIT;
732             RES:=TRUE;
733             FOR D:=1 STEP 1 UNTIL DEVS DO IF -,RESERVED(D) THEN
734             BEGIN INDEX:=I;
735                 RES:=FALSE;
736                 IF FUNCTION(2,NAME,INDEX,J)=DEVICENUMBER(DEVICE(D)) THEN
737                 BEGIN FUNCTION(4,NAME,I,D); COMMENT RENAME;
738                     RESERVED(D):=TRUE;
739                 END;
740             END;
741         END;
742 EXIT:
743         FOR J:=1 STEP 2 UNTIL I DO FUNCTION(5,NAME,J,I);
744         COMMENT CLEAR;
745         SEGMENTS:=0;
746         FOR D:=1 STEP 1 UNTIL DEVS DO
747         BEGIN RESERVED(D):=FUNCTION(6,BS_NAME,D,3)=0;
748             COMMENT PERMANENT;
749             FIRST_SEGM(D):=D;
750             IF RESERVED(D) THEN
751             BEGIN WRITE(OUTC,NL,1,<<DD>,<:DEVICE_:>,
752                 FUNCTION(2,BS_NAME,FIRST_SEGM(D),AREA_SIZE(D)),
753                 <:  :>,STRING BS_NAME(D),
754                 <<DDDDDD>,FIRST_SEGM(D),AREA_SIZE(D));
755                 IF SEGMENTS<AREA_SIZE(D) THEN SEGMENTS:=AREA_SIZE(D);
756             END;
757         END;
758     END CREATION BLOCK;
759
759
759 RTP:
760     GOTO CASE TESTPROG OF (DIRECTORY,A,B,C,RTP,RTP,RTP,RTP,RTP,RTP,
761     RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,X,Y,Z);
762
762
762 DIRECTORY:
763     WRITE(OUTC,<:
764     A    WRITE AND READ
765     B    RANDOM HEAD MOVE
766     C    HEAD STEP
767     X    ROUTINE TEST
768     Y    REMOVE AREAS AND TERMINATE
769     Z    TERMINATE WITHOUT REMOVE
770 :>); GOTO RTP;
771
771
771
771 COMMENT
```

```
772 RC 4000 BACKING STORAGE TEST
773
773
773
773 A: WRITE(OUTC,NL,1,<:BLOCKSIZE (SEGMENTS) = :>); UD; IND;
774     IF INCHAR=10 THEN BUF_SIZE:= 512 ELSE
775     BEGIN REPEATCHAR(INC);
776        RD(A,BUF_SIZE);
777     END;
778     MAXBLOCK;
779 A1:WRITE(OUTC,NL,1,<:MODE = :>); UD; IND;
780     CHECKB:=MODE:=DATA:=TRY:=0;
781     IF INCHAR=10 THEN
782     BEGIN CHECKB:=2;
783        MODE:=3;
784        GOTO A3;
785     END;
786     REPEATCHAR(INC);
787 A2:I:=INCHAR;
788     IF I= 99 THEN BEGIN IF CHECKB< 3 THEN CHECKB:=3 END ELSE
789     IF I=100 THEN BEGIN RD(A1,DATA); REPEATCHAR(INC) END ELSE
790     IF I=109 THEN BEGIN IF CHECKB<1 THEN CHECKB:=1 END ELSE
791     IF I=112 THEN CHECKB:=4 ELSE
792     IF I=114 THEN MODE:=MODE ADD 2 ELSE
793     IF I=115 THEN BEGIN IF CHECKB<2 THEN CHECKB:= 2 END ELSE
794     IF I=116 THEN BEGIN RD(A1,TRY); REPEATCHAR(INC) END ELSE
795     IF I=119 THEN MODE:=MODE ADD 1 ELSE GOTO A1;
796 A4:I:=INCHAR;
797     GOTO IF I=10 THEN A3 ELSE IF I=32 THEN A2 ELSE A4;
798 A3:IF MODE=0 OR MODE>3 OR DATA >4 THEN GOTO A1;
799 AA:RUNADM(RTP,RESMON);
800     WRITE_AND_READ(DATA,MODE);
801     GOTO AA;
802
802 B: WRITE(OUTC,NL,1,<:CHECK,TRIES = :>); UD; IND;
803     CHECKB:=2;
804     TRY:=1;
805     I:=INCHAR;
806     IF I<>10 THEN
807     BEGIN CHECKB:=IF I=110 THEN 1 ELSE 2;
808        RD(B,TRY);
809     END;
810     BUF_SIZE:=1;
811 BB:RUNADM(RTP,RESMON);
812     MOVETEST(FALSE);
813     GOTO BB;
814
814
814
814 COMMENT
815
```



```
815 RC 4000 BACKING STORAGE TEST
816
816 C: BEGIN INTEGER A,B;
817    INTEGER ARRAY FIRST_CYL(1:DEVS),LAST_CYL(1:DEVS);
818
818        ZONE ARRAY BS(DEVS,128,1,BS_ERROR);
819        TRY:=CHECKB:=0;
820        FOR D:=1 STEP 1 UNTIL DEVS DO
821        BEGIN
822 C1:        WRITE(OUTC,NL,1,<:DEVICE :>,DEVICENUMBER(DEVICE(D)),
823              NL,1,<:FIRST CYLINDER = :>); UD; IND;
824           RD(C1,FIRST_CYL(D));
825           A:=FIRST_SEGM(D)//40;
826           B:=(FIRST_SEGM(D)+AREA_SIZE(D)-1)//40;
827           IF FIRST_CYL(D)< A OR FIRST_CYL(D) >=B THEN
828           BEGIN FIRST_CYL(D):=A;
829              WRITE(OUTC,NL,1,<:FIRST CYLINDER = :>,FIRST_CYL(D));
830           END;
831 C2:       WRITE(OUTC,NL,1,<:STEP (CYLINDERS) = :>); UD; IND;
832           RD(C2,LAST_CYL(D));
833           LAST_CYL(D):=FIRST_CYL(D)+LAST_CYL(D);
834           IF LAST_CYL(D) >B THEN
835           BEGIN LAST_CYL(D):=B;
836              WRITE(OUTC,NL,1,<:STEP = :>,LAST_CYL(D)-FIRST_CYL(D));
837           END;
838        END;
839        OPEN_BS(BS);
840
840 CC:    RUNADM(RTP,UD);
841        SETPOSITION(BS(D),0,(FIRST_CYL(D)-FIRST_SEGM(D)//40)*40);
842        INREC6(BS(D),512);
843        SETPOSITION(BS(D),0,LAST_CYL(D)*40-FIRST_SEGM(D));
844        INREC6(BS(D),512);
845        GOTO CC;
846    END;
847
847
847 X: BUF_SIZE:=512;
848    MAXBLOCK;
849    SAVEBUF:=BUF_SIZE;
850    MODE:=3;
851    CHECKB:=3;
852    DATA:=TRY:=0;
853 XX:RUNADM(RTP,RESMON);
854    BUF_SIZE:=SAVEBUF;
855    WRITE_AND_READ(0,3);
856    BUF_SIZE:=1;
857    MOVETEST(TRUE);
858    GOTO XX;
859
859
859 Y: FOR D:=1 STEP 1 UNTIL DEVS DO FUNCTION(5,BS_NAME,D,I);
860    COMMENT REMOVE TEST AREAS;
861
861 Z:
862
862 END TESTBS;
863
863 COMMENT
864
```

RCSL: 44-D16
Author: Per Hansen
Edited: April 1972
Type: ALGOL Program

RC4000
TIMESHARED TESTPROGRAM LIBRARY
PAPER TAPE
(READER AND PUNCH)

Keywords: RC4000, diagnostic program, ALGOL program, ISO tape

ABSTRACT: This program contains 4 routines intended for error location and maintenance purposes on the RC2000 paper tape reader and the RC150 paper tape punch. It is organized as a block incorporated in the 'TEST' system.

A/S REGNECENTRALEN
Falkoner alle 1
2000 Copenhagen F.

Contents:

1. Main characteristics

The pt-test is intended for testing the paper-tape devices, i.e. the RC2000 reader and the RC150 punch. In most cases a simultaneous test of both devices is practical.

The test is called in this way:

<outfile>= test pt.<device1>.<device2>

where <outfile> may be lp or empty.
<device> should be reader and punch respectively.

It should be noted that the RC2000 reader should not be loaded before the message

load reader

appears. If the reader was reserved by another process the message

wait for reader

will be output, and the process is stopped, until the reader becomes free.

The process area must be at least 10000 bytes but to obtain a reasonable speed about 20000 bytes is required.

2. Worst case bitpattern

First the program asks:

parity =

and the operator may type either o,e or n for odd, even and no parity mode respectively. Next, if both a reader and a punch is to be tested simultaneously, the test asks:

length(m)=

and the operator is expected to type a number of meters. Now, a tape of this length containing test information is punched out. This is to enable the tape to be loaded in the reader, (but this should not be done until requested for).

If the reader is to be tested the testprogram tries to reserve the reader, and if it succeeds, the message

load reader

is output, and the operator loads the tape. (It should be noted that this might be a roll of paper tape punched earlier. In this case only the reader should be specified in the call of the testprogram).

In each run the following happens:
A block of worst-case characters is output. The blocklength is 192 characters, and the blocks are separated by a piece of blank tape (which is ignored by the standard reader). Next, the first block on the tape generated after length which has the same contents, is input, and the information and parity is checked.

In case of parityerror, an errormessage is output but the actual bitpattern read cannot be displayed, as it is deleted by the monitor.

In case of dataerror the received and expected bitpatterns are displayed.

In case of paper-out the statusword, stating "end document" is printed, and the current run is terminated. If the testtape is loaded again in a blockgap, the test will continue.

2.1 Synchronization

One cause of dataerror may be loss of characters during output or double reading during input. If five consecutive characters are wrong, the check-program now tries to synchronize on the information in the following way:

First it is checked whether the previously read character is equal to the current reference character, and the currently read character is equal to the next reference character. In this case the message

***synchronize 1 char. missing

is output.

If this is not the case, it is checked, if one of the next 9 characters on the tape is equal to the reference character. If successful the message:

***synchronize <n> char. superflous

is output where <n> is the number of superflous characters.

If this synchronization also fails the message

***synchronize give up

is output, and the test is terminated.
In case of successful synchronization the reference characters are, of course, adjusted according to the message given.

3. Worstcase bitpattern, parity error

The test works just like the worst case bitpattern test mentioned in section 2. The only difference is that the punch and the reader are operated in inverse parity.

The question

parity=

should be answered with either e (for even) or o (for odd). The parity determines the parity of the punch, i.e. the parity of the actual tape punched.

During read all parity errors are converted by the monitor to the value of 26 (SUB). This is checked by the readprocedure, and if the character value differs the message

***not parityerror

is output. Now the character actually read is displayed (as received) together with the expected value. The synchronization mechanism does not work in this test as the normal characters (i.e. the parityerrors) are not accessible.

This means that the dataerror displayed might have been caused by double reading or the like.

4. Read anything

First the test asks:

parity=

Next it starts reading in the parity mode specified without checking of neither parity nor data.

When paper out is reached, the statusword is printed until either a tape is loaded or the last run has been executed. Only one character is read in each run.

5. Punch sequence

First the test asks:

parity=

and odd, even or no parity may be specified. Next the program asks:

type sequence:

and a sequence of characters may be specified according to the rule described in 'test'. (RCSL: 44-D10).

In each run this sequence will now be output on the punch.

6. Routine test

In the routine test the testprogram: worst case bitpattern, parity error, is executed with the punch in odd and the reader in even parity.

7. Release and terminate

If testprogram 'y' is selected the reservation of the devices is cancelled, and the test returns to the fileprocessor. If testprogram 'z' is selected 'test' returns without cancelation of the reservation(s).

8. Examples

Example 1:

```
test pt.0.1

RC 4000 reader/punch
testprogram:  a
number of runs = 9

parity = o

length (m) = 4

from s
message peh load  reader

from peh

run no.       1

* 0  1' run dataerror:
character no. 192
          received: .... ....        0
          expected: .111 111.      126

run no.       2

* 0  2' run dataerror:
character no. 2
          received: ..1. ....       32
          expected: .... ....        0
character no. 3
          received: .1.. ....       64
          expected: ..1. ....       32
character no. 4
          received: .... ...1        1
          expected: .1.. ....       64
character no. 5
          received: .... ..1.        2
          expected: .... ...1        1
character no. 6
          received: .... .11.        6
          expected: .... ..1.        2

***synchronize 1 char. missing

run no.       3
```

```
run no.       4

run no.       5

* 0  parity 32 bytes

run no.       6

run no.       7

* 0  7' run dataerror:
character no. 1
          received: .111 111.      126
          expected: .... ....        0
character no. 2
          received: ..11 1111       63
          expected: .... ....        0
character no. 3
          received: ..11 1111       63
          expected: ..1. ....       32
character no. 4
          received: .111 111.      126
          expected: .1.. ....       64
character no. 5
          received: .... ....        0
          expected: .... ...1        1

***synchronize 4 char. superfluous

run no.       8

* 0  doc_end 100 bytes

* 0  doc_end 0 bytes

run no.       9

* 0  doc_end 0 bytes


test end

testprogram:
```

Example 2:

```
testprogram:  b
number of runs = 9

parity = e

length (m) = 7

from s
message peh load  reader
```

```
from peh

run no.       1

run no.       2

run no.       3

* 0  3'run not parity error
character no. 154
          received: .11. 11.1      109
          expected: .11. 11..      108

run no.       4

run no.       5

run no.       6

run no.       7

run no.       8

run no.       9


test end

testprogram:  y




end
; comment  now the units under test are released for reservation
; by other users. If this is undesired testprogram "z" will termi-
; nate the TEST without giving up the reservation.
```

Example 3:

```
testprogram:
number of runs = 9

parity = n

type sequence:
;0;255,100

:::::full
run no.       1

run no.       2

run no.       3

run no.       4

etc.
```

; Now a sequence of charaters is punched. The character valuse
; are altered between zero (blank) and 255 (all holes).

Example 4:

```
testprogram:  x
number of runs = 9

length (m) = 4

from s
message peh load  reader

from peh

run no.       1

run no.       2

run no.       3

run no.       4

run no.       5

run no.       6

run no.       7

run no.       8

run no.       9

test end

testprogram:
```

; The routine test writes (punches) in odd parity but reads in
; even parity thus checking the hardware parity circuits.

9. Errormessages

*<number>,<status><number> bytes

During test of a device there has been a statuserror. The first <number> is a devicenumber, the second <number> is the number of 12-bit bytes transferred. <status> is the statusword printed as texts.

*<device>,<runno>'run,dataerror:

This message will be followed by a character number defining the position in the block and by the bitpatterns and decimal values of the characters received and expected.

*<device>,<runno>'run not parity error

In the inverse parity test a character without parity error has been found.

*input parity

Parity error on the typewriter.

***stack

The process area is too small.

***synchronize,1 char. missing

The dataerror(s) was probably caused by a missing character.

***synchronize,<n>char. superflous.

The dataerror(s) was probably caused by superflous characters (double reading). <n> lies in the range 1-9.

***synchronize give up

Synchronization was impossible. The test is terminated.

***test call

TEST has been called in a wrong way.

***test device <number> no <devicekind>

is a warning about a wrong devicekind. TEST terminates.

10. Program text

```
1143 RC 4000 PAPERTAPE TEST
1144 RCTS  2.4/0    1.7.71   PEH
1145 VERSION        1.7.71
1146 NAME IN CATALOG: TESTPT;
1147
1147 BEGIN ZONE ARRAY PT(DEVS,32,1,P_ERROR);
1148    INTEGER ARRAY TABLE(0:255),READ_ERRORS(1:DEVS),
1149       CHAR(1:DEVS),CHARACTERS(1:DEVS),LASTCH(1:DEVS);
1150    REAL ARRAY TESTDATA(1:32);
1151    INTEGER LENGTH,I,J,CH;
1152    BOOLEAN REA,PUN,INVPAR;
1153    BOOLEAN ARRAY PARITY_P(1:DEVS),PAPER_OUT(1:DEVS),P(1:DEVS);
1154
1154    PROCEDURE INITPUNCH;
1155    BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO
1156       BEGIN IF DKIND(D)<>12 THEN REA:=TRUE;
1157          IF DKIND(D)=12 THEN PUN:=TRUE;
1158          READ_ERRORS(D):=0;
1159       END;
1160       INITDATA(TESTDATA);
1161       PT_END;
1162    END PROCEDURE INITPUNCH;
1163
1163    PROCEDURE PT_END;
1164    BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO IF DKIND(D)=12 THEN
1165       BEGIN CLOSE(PT(D),FALSE);
1166          SYSTEM(5,DEVICE(D)+2,PARAM);
1167          I:=0;
1168          OPEN(PT(D),2 SHIFT 12+12,STRING PARAM(INCREASE(I)),0);
1169          RESERVED(D):=MONITOR (6,PT(D),0,IA)=0;
1170          IF RESERVED(D) THEN
1171          BEGIN WRITE(PT(D),FALSE,50);
1172             CLOSE(PT(D),FALSE);
1173             I:=0;
1174             OPEN(PT(D),MODEKIND(DEVICE(D)),STRING PARAM(INCREASE(I)),M);
1175             SETPOSITION(PT(D),0,0);
1176          END ELSE IF ¬,RESERVED(D) THEN ERROR (PT(D),4,0);
1177       END;
1178    END PROCEDURE PT_END;
1179
1179    PROCEDURE INITREADER(GIVEUP);
1180    INTEGER GIVEUP;
1181    BEGIN FOR D:=1 STEP 1 UNTIL DEVS DO IF DKIND(D)<>12 THEN
1182       BEGIN SYSTEM(5,DEVICE(D)+2,PARAM);
1183          RESERVED(D):=TRUE;
1184          I:=0;
1185          CLOSE(PT(D),FALSE);
1186          OPEN(PT(D),MODEKIND(DEVICE(D)),
1187             STRING PARAM(INCREASE(I)),GIVEUP);
1188       END;
1189    END PROCEDURE INITREADER;
1190
1190    PROCEDURE PUNCHOUT;
1191    BEGIN INTEGER E;
1192       IF DKIND(D)<>12 THEN GOTO RETURN;
1193       OUTREC6(PT(D),128);
1194       FOR E:= 1 STEP 1 UNTIL 32 DO
1195          PT(D,E):=TESTDATA(E);
1196 RETURN:
1197    END PROCEDURE PUNCHOUT;
1198
1198 COMMENT
1199
```

```
1199 RC 4000 PAPERTAPE TEST
1200
1200     PROCEDURE GETLENGTH;
1201     BEGIN INTEGER I;
1202 RLE:   WRITE(OUTC,NL,1,<:LENGTH (M) = :>); UD; IND;
1203         RD(RLE,LENGTH);
1204         FOR I:=1 STEP 1 UNTIL LENGTH*1.7 DO
1205         BEGIN PT_END;
1206            FOR D:=1 STEP 1 UNTIL DEVS DO PUNCHOUT;
1207         END;
1208     END PROCEDURE GETLENGTH;
1209
1209     PROCEDURE P_ERROR(Z,S,B);
1210     ZONE Z; INTEGER S,B;
1211     BEGIN IF DKIND(D)<>12 THEN
1212        BEGIN CHARACTERS(D):=8 SHIFT (-1)*3;
1213           CHAR(D):=0;
1214           P(D):=FALSE;
1215        END;
1216        PARITY_P(D):=FALSE ADD (S SHIFT (-22));
1217        PAPER_OUT(D):=FALSE ADD (S SHIFT (-18));
1218        IF S<>2 AND -,(PARITY_P(D) AND INVPAR AND DKIND(D)<>12)
1219           THEN ERROR(Z,S,B);
1220        IF PAPER_OUT(D) AND B=0 THEN B:=2;
1221     END PROCEDURE P_ERROR;
1222
1222     INTEGER PROCEDURE TESTCHAR(I); VALUE I; INTEGER I;
1223     BEGIN TDI:=(I MOD 192//3) SHIFT 1+2;
1224        TESTCHAR:=TESTDATA.TDI SHIFT (I MOD 3*8-16)
1225        EXTRACT (IF PARITY >2 THEN 8 ELSE 7);
1226     END PROCEDURE TESTCHAR;
1227
1227     PROCEDURE READ_AND_CHECK;
1228     BEGIN INTEGER I;
1229        BOOLEAN DATAERROR;
1230        IF DKIND(D)=12 THEN GOTO RETURN;
1231        DATAERROR:=FALSE;
1232        FOR I:=0 STEP 1 UNTIL 191 DO
1233        BEGIN
1234 REP:      READCHAR(PT(D),CH);
1235           IF PARITY_P(D) OR PAPER_OUT(D) THEN
1236           BEGIN CHAR(D):=CHAR(D)+1;
1237              IF CHAR(D) >CHARACTERS(D)-3 THEN
1238              BEGIN IF PARITY_P(D) AND CH=26 THEN P(D):=TRUE;
1239                 IF PAPER_OUT(D)  THEN GOTO IF
1240                 CHAR(D)=3 THEN RETURN ELSE REP;
1241                 IF P(D) AND CH=26 THEN GOTO NEXT_I;
1242                 IF P(D) AND CH=0 THEN GOTO REP;
1243              END;
1244           END;
1245           IF CH<>(IF INVPAR THEN 26 ELSE TESTCHAR(I)) THEN
1246           BEGIN IF -,DATAERROR OR INVPAR THEN WRITE(OUTL,NL,1,<:*:>,<<DD>,
1247              DEVICENUMBER(DEVICE(D)),<<DDD>,RUNNO,IF INVPAR THEN
1248              <:'RUN NOT PARITY ERROR<10>:> ELSE <:' RUN DATAERROR:<10>:>);
1249              DATAERROR:=TRUE;
1250              WRITE(OUTL,<:CHARACTER NO.:>,I+1);
1251              PRINTBITS(10,0,0 ADD CH SHIFT 40,
1252                0,0 ADD TESTCHAR(I) SHIFT 40,4,8); UDL;
1253              READ_ERRORS(D):=READ_ERRORS(D)+1;
1254           END ELSE READ_ERRORS(D):=0;
1255
1255
1255
1255
1255 COMMENT
1256
```

```
1256
1257
1257            IF READ_ERRORS(D)>=5 THEN
1258            BEGIN COMMENT SYNCHRONIZE;
1259               READ_ERRORS(D):=0;
1260               WRITE(OUTL,NL,1,<:***SYNCHRONIZE :>);
1261               IF LASTCH(D)=TESTCHAR(I) AND CH=TESTCHAR(I+1) THEN
1262               BEGIN WRITE(OUTL,<:1 CHAR. MISSING<10>:>); UDL;
1263                  REPEATCHAR(PT(D));
1264               END ELSE
1265               BEGIN FOR J:=1 STEP 1 UNTIL 9 DO
1266                  BEGIN LASTCH(D):=CH;
1267                     READCHAR(PT(D),CH);
1268                     IF LASTCH(D)=TESTCHAR(I-1) AND CH=TESTCHAR(I) THEN
1269                     BEGIN WRITE(OUTL,<<D>,J,<:_CHAR. SUPERFLUOUS<10>:>); UDL
1270                        GOTO NEXT_I;
1271                     END;
1272                  END;
1273                  WRITE(OUTL,<:__GIVE UP<10>:>); UDL;
1274                  GOTO RTP;
1275               END;
1276            END;
1277 NEXT_I:    LASTCH(D):=CH;
1278         END;
1279 RETURN:
1280      END PROCEDURE READ_AND_CHECK;
1281
1281
1281 PTTEST_START:
1282
1282      FOR I:=0 STEP 1 UNTIL 255 DO TABLE(I):=7 SHIFT 12+I;
1283
1283      WRITE(OUTC,<:<10>RC 4000 READER/PUNCH:>);
1284 RTP:
1285      INTABLE(0);
1286      REA:=PUN:=INVPAR:=FALSE;
1287      GOTO CASE TESTPROG OF (DIRECTORY, A,B,C,D,RTP,RTP,RTP,RTP,RTP,
1288         RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,RTP,X,Y,Z);
1289
1289 DIRECTORY:
1290         WRITE(OUTC,<:
1291
1291      A    WORSTCASE BITPATTERN
1292      B    WORSTCASE BITPATTERN,PARITY ERROR
1293      C    READ ANYTHING
1294      D    PUNCH SEQUENCE
1295      X    ROUTINE TEST
1296      Y    RELEASE AND TERMINATE
1297      Z    TERMINATE WITHOUT RELEASE
1298
1298 :>);    GOTO RTP;
1299
1299 COMMENT
1300
```

```
1300 RC 4000 PAPERTAPE TEST
1301
1301
1301 A: GETPARITY;
1302    INITPUNCH;
1303    IF PUN AND REA THEN GETLENGTH;
1304    INITREADER(-1);
1305 A1:INTABLE(TABLE);
1306 AA:RUNADM(RTP,PT_END);
1307    PUNCHOUT;
1308    READ_AND_CHECK;
1309    GOTO AA;
1310 B: GETPARITY;
1311    IF PARITY>2 THEN GOTO B;
1312 B1:INITPUNCH;
1313    IF PUN AND REA THEN GETLENGTH;
1314    PARITY:=(PARITY+2) MOD 4;
1315    INITREADER(-1);
1316    PARITY:=(PARITY+2) MOD 4;
1317    INVPAR:=TRUE;
1318    GOTO A1;
1319
1319 C: GETPARITY;
1320    INITREADER(1 SHIFT 18);
1321 CC:RUNADM(RTP, UD);
1322    PAPER_OUT(D):=FALSE;
1323    IF DKIND(D)=12 THEN GOTO CC;
1324 CCC:
1325    READ(PT(D),CH);
1326    GOTO IF PAPER_OUT(D) THEN CC ELSE CCC;
1327
1327 D: GETPARITY;
1328    LENGTH:=(PACK(TESTDATA,256,8));
1329 DD:RUNADM(RTP,PT_END);
1330    OUTREC6(PT(D), LENGTH*4);
1331    FOR I:=1 STEP 1 UNTIL LENGTH DO
1332    PT(D,I):=TESTDATA(I);
1333    GOTO DD;
1334
1334 X: PARITY:=0;
1335    GOTO B1;
1336
1336
1336 Y: FOR D:=1 STEP 1 UNTIL DEVS DO CLOSE(PT(D),TRUE);
1337
1337 Z:
1338
1338
1338 END TESTPT;
```