C1063.1

SYSTEM-TABLE

```
0         " HEERSENDE SYSTEEM TABEL
1         CPRIML = 211                    " LENGTH OF PRIMARY INPUT PROGRAM
2         UNRGR = 1                       " APPARAATNUMMER GOLDEN READER
3         UG = '72 000'      } constants that transform Mo-addressing into
4         UA = '73 000'      }      MG, MA, MS, MC-addressing when added to instruction (address-part)
5         US = '74 000'      }
6         UC = '75 000'      }
7         UDEAE = '660 060 000'  } these are actual machine-instructions.
8         UHEAR = '760 060 000'  }
9         UCLOCKPERIOD = 200              "                                   MMM
10        UNUMPM = 5                              " NUMBER OF PM'S
11        UNAM = 15                       " NUMBER OF AM'S
12        CONSEQ = 20                     " NUMBER OF CONSECUTIVE LINE PRINTER FORMS
13        UNUMLPSTR = 5                           " TOTAL NUMBER OF LINE PRINTER STREAMS
14        UVECTORTJE = 32                 "                                   MMM
15        ULOADH = '772 531 463'
16        ULOADT = '063 146 315'
17        URLLEN = 24                             " LENGTH PLOTTER BUFFER
18        URLLENTEL = '30 000 000'                " URLLEN * 2 ↑ 18
19        UCONN1 = '1 000 000'
20        URLSUNOUT = 3000
21        URLOWIDTH = 2794
22        URLOWIDTHMIN = 2750
23        UNUMNORMALTR = 2
24        UCR = '27000'
25        UDML = 180                      " LENGTH DRUMMAG
26        UCRC = 11392
27        UCTLEN = 292
28        UDRUMMARGE = 50                 "                                   MMM
29        UMAXERROR = 5                   "                                   MMM
30        UI = -1 048 575
31        CMAGLE = 13                     " LENGTH OF CONSOLEPRINTERBUF
32        CMAXP = 69                      " NUMBER OF PRINT POSITIONS ON CONSOLE PRINTER LINE
33        UOVERSHOOT = 70
34        UPOOLEXTENT = 48
35        USTEEK = 10
36        UDANGERPOINT = 100
37        UINITIALE1 = 191
38        UINITIALEIOR = 1011
39        UINITIALEIO = 255
40        UINITIALEIR = 947
41        UMAXPOR =5      " MAXIMUM PORTION SIZE IN SEGMENTS      MMM
42        UINITIALERAS = UINITIALEIOI - UINITIALE1 - UMAXPOR)
43        UNUMTRPMSTR = 2                 " NUMBER OF TAPEREADER STREAMS PER PM   MMM
44        UNUMLRPMSTR = 1                 "                                       MMM
45        UNUMTRPMSTR = 2                 "                                       MMM
46        UNUMRLPMSTR = 1                 "                                       MMM
47        UNUMTR = 3                              " NUMBER OF TAPE READERS
48        UTRLOC = 18                             " NUMBER OF LOCALS IN TAPEREADER STACKBOTTOM
49        UTRLEN = 24                     " NUMBER OF HEPTADES READ BY TAPEREADER MMM
50        UTRLENTEL = '30 000 000'        " UTRLEN * 2 ↑ 18                       MMM
51        UTRMAXLEN = 503

52        UNRMT = 10
53        JMTREWRR = 20
54        JOPENTAPERR = 500
55        JCLOSETAPR = 500
56        JPOSPRICE = 1000
```

Introduction of constants and assembly parameters

```
57          COCTRRICE = 15
58          CHARERRICE = 19
59          CNUMINSRRICE = 25
60          CNUMRRICE = 50
61          CSYMRRICE = 21
62          CSYMINSRRICE = 20
63          UNUMPLUSTR = 15                 " NUMBER OF PLUNCHER STREAMS
64          UNUMTRSTR = 10                  " NUMBER OF PUNCH STREAMS
65          UNUMPLSTR = 5                   " NUMBER OF PLOTTER STREAMS
66          UNUMPLU = 4                     " NUMBER OF PLUNCHERS
67          UNUMTR = 3                      " NUMBER OF PUNCHERS
68          UNUMPL = 1                  " NUMBER OF PLOTTERS
69          UNUMSVLIB = 100
70          URUNLENGTH = 4095
71          URUNSIGNAL = 3900

72          UNAMEWORDS = 7                  " MAXIMUM NAMELENGHT OF A PROGRAM IN WORDS

73          UTRLEN = 12                     " LENGTH OF TR BUFFER
74          UMAXHERSEG = 1524           " MAX NUMBER OF OCTADES IN PUNCH SEGMENT
75          UHERRLENTY = 22750              " NUMBER OF OCTADES ON 1/4 REEL
76          UHERREEL = 91000                " NUMBER OF OCTADES ON A REEL
77          UNUMBLANK1 = 20
78          UNUMBLANK2 = 40
79          UNUMBLANK3 = 60
80          UNUMBLANK4 = 40
81          UNUMBLANK5 = 20

82          UNAMELENGTH = 19
83          USTANDARDNEED = 3

84          UENTIERRRICE = 13
85          USORTRRICE = 24
86          UABSRRICE = 11
87          ULNRRICE = 30
88          UEXRRICE = 45
89          UCOSRRICE = 27
90          USINRRICE = 27
91          UARCTANRRICE = 39
92          US.GNRRICE = 11
93          UMODRRICE = 31
94          URERRICE = 11
95          UIMRRICE = 11
96          UCOMRRICE = 12
97          UANDRRICE = 13
98          UORRRICE = 13
99          USHIFTRR.CE = 16
100         UBITRRICE = 15
101         UHEADRRICE = 12
102         UTAILRRICE = 12
103         UCOMPOSERRICE = 14
104         UA1 = '72 000'
105         UA2 = '71 000'
106         RRLOTRRICE = 25
107         RRLOTRRICE1 = 6
108         RRLOTRRICE2 = 4
109         RRLOTRRICE3 = 4
110         CENDRLDOCRRICE = 20
111         RRLOTERAMERRICE = 19
112         RCRLOTCURVERRICE1 = 11
```

```
113         RCRLOTCURVERRICE2 = 16
114         RCRLOTCURVERRICE = 200
115         RTRLOTTEXTRRICE1 = 6
116         RTRLOTTEXTRRICE2 = 10
117         RTRLOTTEXTRRICE = 30
118         RLOTNUMBERRRICE = 75
119         UGECRRICE = 12
120         UMAXWR = 104
121         UMAGLE = 13
122 D3[9]
123         HREGELDEL(1)

124 D1[2]
125         HIL(2)
126         HCL(2)
127         HNL(4)
128         HRL(4)
129         HOBTEXT(2)

130 D0[16]                          " DYNAMIC ADDRESSES IN PLOTTERTEXT
131         UEIRSTRL(1)
132         UINVRLADRES(1)
133         URACKEDRLWORD(2)
134 D0[23]
135         URLIN(1)

136 D0[3]                           " DYNAMIC ADDRESSES IN PUNCH TEXT
137         URLUSEG(1)
138         URLUTYRE(1)
139         URLUDISC(2)
140         UHOKRLU(3)
141         UCYCLE(1)
142         UM28(1)
143         ULAST(1)
144         UEXNUM(1)
145         UEXNUMDOC(1)
146         ULENGTHSEG(1)
147         UCUTVAR(1)
148         UAVAILTAPE(1)
149         UTRLETTER(1)
150         URUNARROW(1)
151         UM20(1)
152         UM23(1)
153         USRECIALRLU(1)

154 D0[1]                           " DYNAMIC ADDRESSES IN LINEPRINTERTEXT
155         CURHDL(1)
156         CLRLAB(3)
157         CRRHDL(1)
158         CEORMNUM(1)
159         CLRSV(5)
160         CLEE(1)
161         CEEED(1)
162         CLRMES(5)
163         CLRBUE(1)
164         CSVADR(1)                " FROM HERE INITIALISATION IRRELEVANT
165         CSEGLW(1)
166         CRORTIONSIZE(1)
167         CRARE(1)

168 D0[1]                           " DYNAMIC ADDRESSES IN TAPEREADERTEXT
```

dynamic addresses of the translater that also play a role in the dying rites of programs

identif. list
name list
linenumber list
objectcode

dynamic addresses

```
169         UCURSTR(1)
170         ULALAB(1)
171         UTRSEG(1)
172         UTRSLOT(1)
173         UDISC(1)
174         UM12(1)
175         UHOKHAR(6)
176         UNUMVSEG(1)
177         UM1(1)
178         UM8(1)
179         UM14(1)
180         USPECIALTR(1)

181 D0[1]
182         UHOKDRUM(1)          " DYNAMIC ADDRESSES IN DRUMTEXT
183         USCREAD(2)
184 D0[4]
185         GL23(1)
186         GL24(1)

187         GL25(1)
188         GL16(1)
189         USP(1)
190         UWARNINGPOINT(1)
191         UM34(1)
192         UM35(1)
193         UM36(1)
194         UKILLSEM(3)
195         UWHO(1)              " INVARIANT ADDRESS PARTLY RECOGNIZED NAME
196         UPROGID(1)           " INVARIANT ADDRESS PROGRAM IDENTIFICATION
197         URUNNUM(1)           " RUNNUMBER
198         UBLOCKSTR(1)         " HANDLE BLOCKING STREAM
199         UIRSTREAMS(1)        " BASIC ADDRESS LIST OF OWN TAPE READER STREAMS
200         ULRSTREAMS(1)
201         UTRSTREAMS(1)
202         URLSTREAMS(1)
203         UREQUEST(1)          " NUMBER OF REQUESTED OUTPUT SEGMENTS

204         UFINISHDOUBTFUL(1)
205         UTRRMVAR(0)
206         ULOAMRUN(1)
207         UCLAIMRUN(1)
208         UDOCRUN(1)
209         URLRMVAR(0)
210         ULOAMRLO(1)
211         UCLAIMRLO(1)
212         UDOCRLO(1)
213         GL0(1)
214         GL1(1)
215         GL3(1)
216         GL4(1)
217         GL5(1)
218         GL6(1)
219         GL7(1)
220         GIR(2)
221         ELV(1)
222         ERV(1)
223         ECV(1)
224         ELVM(1)
225         ERVM(1)
```

dynamic addresses in stackbottoms of PMs

a PM is a "programmable machine" (the executor of a user program)

```
226           ECVM(1)
227           GL17(1)
228           GL18(1)
229           GL19(1)
230           GL20(1)
231           GL21(1)
232           GL26(1)
233           CGL1(1)
234           GBILL(1)
235           UNITS(1)
236           CSTEEK(1)
237           UTRANSTIME(1)
238           CGL2(1)

239           ULIST(1)
240           UKIND(1)
241           USPECIALPM(1)
242           UFIRSTENTRANCE(1)
243           USTATE1(1)
244           USTATE2(1)
245           USTATE4(1)
246           USTATE8(1)
247           USTATE16(1)
248           USTATE32(1)
249           USTATE64(1)
250           RXLAST(1)
251           RYLAST(1)
252           RXMAX(1)
253           RYMAX(1)
254           RPEN(1)
255           RSCX(2)
256           RSCY(2)
257           RXMIN(2)
258           RYMIN(2)
259           RCY1R(2)
260           RCSCALEE(2)
261           RCX1(2)
262           RCX2(2)
263           RCY1(2)
264           RCY2(2)
265           RCCOUNT(1)
266           RCDERIV(1)
267           RCXISX(1)
268           RTX0(2)
269           RTY0(2)
270           RTSHIFT(1)
271           JMTCLAIMED(1)
272           GLELOC(1)
273           GLABCO(1)
274           GLTYPE(1)
275           GLRETURN(1)
276           GL27(1)
277           CGLSLOW(1)
278           CMTUSED(1)
279           JCURUNIT(1)
280           JMTEINDR(1)
281           CGLSTV(3)

282 DI[ - 21)
283           PPLOT(2)
```

own variables of hand coded plotter procedures

```
284          RX(4)
285          RY(4)
286          RIREN(4)
287 D2[2]
288          RLX(1)
289          RLY(1)
290          RXY(1)
291          RDX(1)
292          RDY(1)
293          RDX2(1)
294          RDY2(1)
295          RI(1)
296          RDEV(1)
297          RABSIREN(1)
298          RRIGHT(1)
299 D1[- 31]
300          REXMIN(4)
301          REYMIN(4)
302          REXMAX(4)
303          REYMAX(4)
304          RELXMAX(4)
305          RELYMAX(4)
306 D1[- 21]
307          RCPLOTCURVE(2)
308          RCX(4)
309          RCY(4)
310          RCI(4)
311 D2[2]
312          RCTEMP(2)
313          RCTEMP2(2)
314          RCA(2)
315          RCB(2)
316          RCSC1(2)
317          RCSC2(2)
318          RCXS1(2)
319          RCXS2(2)
320          RCYS1(2)
321          RCYS2(2)
322          RCMAX1(2)
323          RCMAX2(2)
324          RCYOR(2)
325          RCXO(2)
326          RCYO(2)
327          RCLX(1)
328          RCLY(1)
329          RCWHICH(1)
330          RCWHICH1(1)
331          RCWHICH2(1)
332          RCWHICH3(1)
333          RCDIF(1)
334          RCABSDIF(1)
335          RCINCRX(1)
336          RCE(1)
337          RCDELTA(1)
338          RCDELTA2(1)
339          RCDELTA3(1)
340          RCXOI(1)
341          RCYOI(1)
342          RCXSI(1)
343          RCYSI(1)
```

Variables and formals of plotter-procedures (hand coded)

PCTEMP is the symbolic name for dyn. address M2[2]

```
344           RCLN1(1)
345           RCLN2(1)
346           RCLN3(1)
347           RCISC1(1)
348           RCISC2(1)
349           RCISC3(1)
350           RCISC4(1)
351           RCISC5(1)
352           RCISC6(1)
353           RCISC7(1)
354           RCABSXS1(1)
355           RCXSMIN(1)
356           RCYSMIN(1)
357           RCXSMAX(1)
358           RCYSMAX(1)
359           RCSMALL(1)
360           RCHUGE(1)
361           RCFIRSTTRY(1)
362           RCOKSOFAR(1)
363 Q1[- 39]
364           RTX(4)
365           RTY(4)
366           RTANGLE(4)
367           RTHEIGHT(4)
368           RTITALICITY(4)
369           RTFIRST(4)
370           RTI(4)
371           RTTEXT(4)
372 Q2[2]
373           RTIREN(1)
374           RTJ(1)
375           RTK(1)
376           RTN(1)
377           RTABSI(1)
378           RTXKI(1)
379           RTHECOSAN(2)
380           RTHESINAN(2)
381           RTLX(2)
382           RTLY(2)
383           RTXK(2)
384           RTYK(2)
385           RTAN(2)
386           RTHE(2)
387           RTIT(2)
388           RTABSHE(2)
389           RTTANIT(2)
390           RTCOSIT(2)
391 Q1[- 47]
392           RNX(4)
393           RNY(4)
394           RNANGLE(4)
395           RNHEIGHT(4)
396           RNITALICITY(4)
397           RNFIRST(4)
398           RNI(4)
399           RNN(4)
400           RNM(4)
401           RNNUMBER(4)

402 Q0[0]
```

```
403             M0(512)
404             M1(512)
405             M2(512)
406             M3(512)
407             M4(512)
408             M5(512)
409             M6(512)
410             M7(512)
411             M8(512)
412             M9(512)
413             M10(512)
414 D58[0]
415             MG(512)
416             MA(512)
417             MS(512)
418             MC(512)
419             MT(512)
420             MD(512)
421 D0[-26]
422             UKILLCOUNTER(1)
423             UKILLVAR(1)
424             UAMTELBEQ(1)
425             UAMVAR(1)
426             UAMSEM(3)
427             USCAMSEM(3)
428             UHAMR(13)

429 C 0
430             M(24)
431             UINTERRUPTJUMP(33)
432             E(1)
433             G(1)
434             A(1)
435             S(1)
436             B(1)
437             T(1)
438             D(1)
439             UARTRA(4)
440             AROGR(0)
441             UARTRA(4)
442             UARTRB(4)
443             UARTRB(4)
444             UARRLA(4)
445 C 96
446             UARCR(4)
447             UARCK(4)
448             UARMT(4)
449             UARDR(4)
450 C 128
451             UARLRA(4)
452             UARTRC(4)
453             UARTRC(4)
454             UARCRB(4)
455             UARCKB(4)
456 C 218
457             USTADRUM(1)
458             USRILTIME(1)
459             UARCLOCK(4)
460 C 256
461             UINT(5)
```

here starts the introduction of core addresses

the introduction of M[57] – M[63]

→ comm. channels with peripherals Tape Punch A

→ Tape Reader A

→ console printer

→ console keyboard

etc.

here the actual text of the system starts

```
462         WINT1(3)
463         WINT2(15)
464         WNEXTINT(37)
465         WCHOICE(8)
466         WCURRENT(6)
467         WRESTORE(7)
468         WEXPINT(28)
469         WROR(12)
470         WRROOD = '563 400 000'
471         WP = '563 400 003'
472         WROR1(4)
473         WROR2(5)
474         WROR3(3)
475         WRORW(18)
476         WRH = '563 400 000'
477         WVOR(43)
478         WV = '563 400 002'
479         WVWII = '563 400 000'
480         WHAM(1)
481         WCURRR1(1)
482         WBELSOM(2)
483         WKLB(1)
484         WEXPINTW(2)
485         WCON1(1)
486         WCON2(1)
487         WRSEL(1)
488         WSEL(1)
489         WRR1(WNAM)
```

WV : derived constant

So much memory used for the implementation of multiprogramming, P and V-operations

```
490         SRQ(57)
491         SRQ1(7)
492         SRQ2(11)
493         SRQ3(15)
494         WHERSTELTERUG(2)
495         WDRUMAET(24)
496         WTEXTDRUM(10)
497         WRIETDRUM(45)
498         WDRUMMAG(WDML)
499         WSCWRITE(1)
500         WIETDR(10)
501         WSCS(3)
502         WVICTIM(1)
503         WEREE(1)
504         WSCWORK(1)
505         WDRUMPOSITION(1)
506         WDRUMAAN(1)
507         WDRUMCOM(1)
508         WDRUMERROR(1)
509         WTOTALERROR(1)
510         WERSTERROR(1)
511         WSECONDERROR(1)
512         WCON3(1)
513         WRAGECLOCK(1)
514         WZEROHOUR(12)
515         WRROEANELINK(13)
516         WRROEANATION(4)
517         WKILLSEG(6)
518         MECR(8)
519         WREDDEN(4)
520         WHERSTEL(5)
```

Segment controller

```
521         MEDR(10)
522         RSE50(2)
523         RSE52(3)
524         RSE49(1)
525         R49E1(9)
526         CEILL(18)
527         RSE59(1)
528         RSE39(1)
529         SE39 = '563 400 000'
530         R39E2(2)
531         RSE61(1)
532         RSE41(3)
533         R39E1(30)
534         DURT(40)
535         CMI(33)

536         UINSTV(6)
537         UDNSS(25)
538         UEXBEAM(15)
539         UCONEBEE(5)
540         URBINTNLCR(4)
541         UCONVTAIL(12)
542         CMAKEIETZERO(6)
543         CNOTCASE(7)
544         CAESL(7)
545         CNKC(29)
546         CNNRC(15)
547         CREADDEC(14)
548         CDNONS(4)
549         CPRINT(58)
550         CPRINTDEC(29)
551         CLEESHERKEN(55)
552         CWRONG(19)
553         CSELCON(7)
554         CLCON(13)
555         CPRINTINT(5)
556         CONVINTCK(5)
557         CONVCKINT(3)
558         CPRINTSTRING(10)
559         COMSEMPADRES(2)
560          COMSEMP = '563 400 000'
561         COMSEMVADRES(2)
562          COMSENV = '563 400 000'
563         CTABINTCK(34)
564         CTABCKINT(21)
565         WIETCK(10)
566         UWORK1(1)
567         VLISTAM(UNAM(1))
568         CASETB(1)
569         CLNRC(1)
570         CPC(1)
571         CBUR(1)
572         CNCHAR(1)
573         CTWNR(1)
574         CHLW(1)
575         UMIREAC(10)
576         COMSEM(3)
577         CWORDLIST(39)
578         CERIS(15)
579         VIETCR(10)
```

message interpreter

```
580         CGARR(1)
581         CASEDW(1)
582         CKLDW(1)
583         CKLACH(1)
584         CMAGADRES(1)
585         CMAGAZYN(CMAGLE)
586          CMAXMAG = 9
587         CHVR(1)
588         CRLABEL(3)
589         CBVR(1)
590         UCLAIMCON(7)
591         UCONCON(9)
592         UROWNSEMADRES(2)
593          UROWNSEM = '563 400 000'
594         CREADOCT(15)
595         CPROCT(14)
596         CREQSS(4)
597         CRMMES(19)
598         CSTACKMES(7)

599         SRR(30)
600         UCLAIMCONRED(3)
601         UCONFREEWHITE(5)
602         UMES30(23)
603         UENDMES(6)
604         USWANSONG(5)
605         UIMMORTAL(4)
606         UMORTAL(7)
607         USTORREN(6)
608         UMES32(1)
609         UMES33(13)
610         UMES35(8)
611         UMES34(20)
612         UMES36(30)
613         CMNA30(1)
614         CM31(1)
615         CMNA32(1)
616         CMROOL(1)
617         CMINPUT(1)
618         CMEREE(1)
619         ULISTRM(UNUMRM(2))
620         UTREX(3)
621         UTREXRADRES(2)
622         UTREXR = '563 400 000'
623         UTREXVADRES(2)
624         UTREXV = '563 400 000'
625         UTREXRROODADRES(2)
626         UTREXRROOD = '563 400 000'
627         UTREXVWITADRES(2)
628         UTREXVWIT = '563 400 000'
629         UEI(1)
630         UEIR(1)
631         UEIO(1)
632         UEIOR(1)
633         UROOL(1)
634         UEINDRITEN(2)
635         CMNN(1)
636         USGAAN(23)
637         USGAE(27)
638         USGLONG(14)
```

```
639         UWORK3(1)
640         UTRUTREEE(1)
641         UTRUTLIST(86)
642         UTRUTADR(255)
643         UTRSTRL1(UNUMTRRMSTR)
644         UTRSTRL2(UNUMTRRMSTR)
645         UTRSTRL3(UNUMTRRMSTR)
646         UTRSTRL4(UNUMTRRMSTR)
647         UTRSTRL5(UNUMTRRMSTR)
648         ULRSTRL1(UNUMLRRMSTR)
649         ULRSTRL2(UNUMLRRMSTR)
650         ULRSTRL3(UNUMLRRMSTR)
651         ULRSTRL4(UNUMLRRMSTR)
652         ULRSTRL5(UNUMLRRMSTR)
653         UTPSTRL1(UNUMTPRMSTR)
654         UTPSTRL2(UNUMTPRMSTR)
655         UTPSTRL3(UNUMTPRMSTR)
656         UTPSTRL4(UNUMTPRMSTR)
657         UTPSTRL5(UNUMTPRMSTR)
658         URLSTRL1(UNUMPLRMSTR)
659         URLSTRL2(UNUMPLRMSTR)
660         URLSTRL3(UNUMPLRMSTR)
661         URLSTRL4(UNUMPLRMSTR)
662         URLSTRL5(UNUMPLRMSTR)
663         UTRACT(26)
664         UEXTRVAR(2)
665         UEXPMTRVAR(3)
666         UWORK2(1)
667         UPENREQ(1)
```

lists of Tape reader, puncher, lineprinter streams, one for each PM (5 PMs)

```
668         ULISTTR(UNUMTR[2])
669         UEIOEIOR(40)
670         UWORKB(1)
671         USTRLEN(2)
672         UHAR(12)
673         UHAR1(7)
674         UNONHAR(21)
675         UDISCONNECT(7)
676         UTEXTTR(36)
677         UTEXTTR1(10)
678         UTEXTTR2(74)
679         CVAETRIET(10)
680         UTEXTART(11)
681         UVULL(3)
682         UIETTRA(10)
683         UIETTRB(10)
684         UIETTRC(10)
685         USCHAKELTRA(6)
686         USCHAKELTRB(6)
687         USCHAKELTRC(6)
688         UWIGA(UTRLEN[3])
689         UWIGB(UTRLEN[3])
690         UWIGC(UTRLEN[3])
691         UWAGA(UTRLEN[3])
692         UWAGB(UTRLEN[3])
693         UWAGC(UTRLEN[3])
694         UMES13(10)
695         CMNA13(1)
696         UMES9(11)
697         CMNA9(1)
698         UMES14(5)
```

text of the tape-reader CM (constant machine)

```
699         UANS15(5)
700         UANS16(4)
701         UMES5(18)
702         CMMA6(1)
703         CMMA7(1)
704         UMES12(9)
705         UANS2(10)
706         USTAN(20)
707         UNONSTAN(23)
708         URERM(4)
709         UMES1(19)
710         UANS3(21)
711         UMES8(6)
712         CQCT(53)
713         CREMES(10)
714         CDATE(4)
715         CSERIAL(4)
716         UDATE(1)
717         UCURRUNNUM(1)
718         COLIM1(2)
719         COLIM14(2)
720         UKILLSEGR(7)

721         CLRSEL(39)
722         CVULMAG(13)
723         CFORME(6)
724         CLRSTART(44)
725         CLRTXT(26)
726         CNEWFORM(43)
727         CENDOFFORM(25)
728         CLRAMAG(38)
729         CLRCON1(1)
730         CLRCON2(1)
731         CXXXXX(4)
732         UIETLRA(10)
733         USUMR(1)
734         UERAS(1)
735         UNED(1)
736         CLRQUT(9)

737         UEIOBARRIER(17)
738         UCLOCKINCR(2)
739         URORTIONCLOCK(2)
740         RSE73(9)
741         SE73 = '563 400 000'
742         RSE47(22)
743         RSE18(26)
744         SE18 = '563 400 000'
745         RSE1(14)
746         SE1 = '563 400 000'
747         RSE6(9)
748         SE6 = '563 400 000'
749         RSE20(4)
750         R20E1(1)
751         R20E2(3)
752         R20E3(5)
753         RSE21(7)
754         RSE22(13)
755         RSE23(5)
756         RSE24(7)
```

} text of the lineprinter CM

PSE --- stands for System Entry
they perform a lot of tasks for algol programs
like blockentry, exit, procedure call,
handing over actuals, type checking etc, etc.

```
757         RSE25(5)
758         RSE26(5)
759         RSE27(5)
760         RSE28(5)
761         RSE29(5)
762         RSE30(11)
763         RSE31(13)
764         RSE32(7)
765         CHEENADRES(8)
766         CHEEN = '563 400 000'
767         CTERUGADRES(8)
768         CTERUG = '563 400 000'
769         SCL(17)
770         SCL1(6)
771         RSE75(3)
772         SE75 = '571 400 000'
773         RSE76(3)
774         RSE89(3)
775         R89E1(6)
776         RSE77(19)

777         RSE94(5)
778         RSE78(14)
779         SE78      = '562 400 000'
780         RSE79(4)
781         R79E1(8)
782         RSE80(4)
783         RSE81(4)
784         RSE82(8)
785         R82E1(2)
786         R82E2(2)
787         R82E3(2)
788         RSE83(9)
789         RSE84(10)
790         RSE85(6)
791         RSE86(13)
792         RSE95(4)
793         RSE112(4)
794         SE112 = '562 400 000'
795         RSE33(20)
796         SE33 = '563 400 000'
797         RSE34(11)
798         SE34 = '563 400 000'
799         RSE72(2)
800         RSE71(28)
801         SE71 = '563 400 000'
802         RSE46(12)
803         R46E4(2)
804         R46E2(10)
805         R46E3(16)
806         RSE90(12)
807         R90E21(11)
808         R90E31(7)
809         R90E34(4)
810         R90E32(7)
811         R90E35(11)
812         R90E33(7)
813         R90E36(3)
814         R90E0(34)
815         R91E3(26)
```

Subscription

```
816         RSE92(27)
817         R92E1(10)

818         CAANLR(31)
819         CTAB9(3)
820         CTABINTLR(23)

821         RSE115(21)
822           SE115 = '563 600 000'          " Y,  SUBCD(:   )
823         CALARM(2)
824         RSE88(6)
825         RSE19(2)
826         CHARETAB(43)
827         CNUMTAB(22)
828         USTACKRM1(28)                    " RUIMTE VOOR 16 EXTRA GLOBALEN
829         USBRM1(119)
830         UD1SRRM1(8)
831         UWRRM1(357)
832         USTACKRM2(28)
833         USBRM2(119)
834         UD1SRRM2(8)
835         UWRRM2(357)
836         USTACKRM3(28)
837         USBRM3(119)
838         UD1SRRM3(8)
839         UWRRM3(357)
840         USTACKRM4(28)
841         USBRM4(119)
842         UD1SRRM4(8)
843         UWRRM4(357)
844         USTACKRM5(28)
845         USBRM5(119)
846         UD1SRRM5(8)
847         UWRRM5(357)
848         CSYMTAB1(13)
849         CSYMTAB2(54)
850         CSYMTAB3(18)
851         URROCTINE(1)

852         RSE10(42)
853         RSE87(3)
854         RSE8(3)
855         RSE9(4)
856         R9E1(1)
857         R9E2(18)
858         RSE53(1)
859         R53E1(5)
860         RSE63(1)
861         RSE43(5)
862         RSE60(1)
863         RSE40(1)
864         R40E1(14)
865         RSE58(1)
866         RSE38(4)
867         SE38 = '563 400 000'
868         RSE62(1)
869         RSE42(4)
870         RSE65(3)
871         RSE66(3)
872         RSE67(6)
```

Space for the stackbottoms of the PMS

```
873         R67E1(7)
874         RSE68(9)
875         R62E1(3)
876         R58E1(7)
877         RSE54(2)
878         RSE105(14)
879         RSE110(1)
880         RSE107(6)
881         RSE109(1)
882         RSE106(7)
883         RSE108(3)
884         RSE111(14)
885         RSE35(1)
886         SE35 = '570 400 000'
887         RSE36(3)
888         RSE51(3)
889         RSE37(3)
890         RSE48(3)
891         RSE55(10)
892         RSE57(8)

893         UL!STRLU(UNUMRLU)
894         UTRVAR(0)
895         URUNER(1)
896         URUNACCERT(1)
897         URUNCASH(1)
898         URUNEREE(1)
899         URUNIDLE(1)
900         UTRVAR5(1)
901         UTRVAR6(1)
902         UTRVAR7(1)
903         UTRVAR8(1)
904         UTRVAR9(1)
905         UTRVAR10(1)
906         URLVAR(0)
907         URLOER(1)
908         URLOACCERT(1)
909         URLOCASH(1)
910         URLOEREE(1)
911         URLOIDLE(1)
912         URLVAR5(1)
913         URLVAR6(1)
914         URLVAR7(1)
915         URLVAR8(1)
916         URLVAR9(1)
917         URLVAR10(1)
918         UBANKERSALLOWANCE(32)
919         UINCERASLUUED(6)
920         UERASWAK!NG(13)
921         URROCBANKER(32)
922         UCHANGECLA!M(6)
923         URLUSEL(25)
924         UD!SCRLU(21)
925         UCASHINCREASE(6)

926         CTENRO(10)
927         CTABVDL(50)
928         CLRB11(38)
929         CLRB21(38)
930         CLRB31(38)
```

punches and plotter texts

```
931          CLRB41(38)
932          CLRB51(38)

933          UEINDOCSEL(23)
934          UHOLYRMSEG(2)
935          UHOLYCMSEG(7)
936          CRLUMES(20)
937          RSE100(9)
938          RSE101(6)
939          RSE102(4)
940          RSE104(7)
941          RSE103(3)
942          R103E1(3)
943          R103E2(2)
944          R103E3(4)
945          RSE0(21)
946          SE0 = '563 400 000'
947          RSE44(6)
948          RSE74(3)
949          R50E1(44)
950          UCLEAR(4)
951          UTEXTRLU(28)
952          UTEXTRLU1(38)
953          UBEGINTRDOC(25)
954          UTRSEGMENT(4)
955          UENDTRDOC(6)
956          UBEGINRLDOC(12)
957          URLSEGMENT(35)
958          UENDRLDOC(14)
959          UBLANKTAPE(7)
960          URUNCHSTRING(68)
961          UMOKRLUNCHER(5)
962          UMES20(7)
963          UANS21(13)
964          UMES23(5)
965          UMES25(13)
966          UMES26(5)
967          UMES27(13)
968          UMES28(6)
969          UFILLSEGMENT(22)
970          UTRFILL(11)
971          URURM(13)
972          ULISTTR(UNUMTR(2))
973          URLCODE(8)
974          URENUR(4)
975          URENLINKS(4)
976          UANSRL21(10)
977          USH4(23)
978          USH8(23)
979          UIETTRA(10)
980          UIETTRB(10)
981          UIETTRC(10)
982          UWIGTRA(UTRLEN)
983          UWIGTRB(UTRLEN)
984          UWIGTRC(UTRLEN)
985          UWAGTRA(UTRLEN)
986          UWAGTRB(UTRLEN)
987          UWAGTRC(UTRLEN)
988          USCHAKELTRA(6)
989          USCHAKELTRB(6)
```

```
990         USCHAKELTRC(6)
991         USCHAKELRLA(6)
992         UIETRLA(10)
993         UWIGRLA(URLLEN)
994         UWAGRLA(URLLEN)
995         UBLANK(1)
996         UARROW1(1)
997         UARROW2(1)
998         USTMAXHERSEG(1)
999         USTHERRLENTY(1)
1000        USTHERREEL(1)
1001        UALLHOLE(1)
1002        UALLHOLE4(1)
1003        CMNA21(1)
1004        CNUMLRTAB(5)
1005        USH1(25)

1006        UTEXTRM(5)
1007        CORSYM(5)

1008        UDYNERR(21)
1009        USTERR(2)
1010        UEXSEM(3)
1011        UWSEXIT(14)
1012        URUNSEM(3)
1013        URUNAAN(1)
1014        UD200(5)
1015        CORY(35)
1016        CTIMEMES(17)
1017        UKALESRATIE(1)

1018        CODEEIN(20)
1019        UROLINEE(1)
1020        UROLINE(8)
1021        UENTIER(1)
1022        UENTIER1(12)
1023        UEXPNONNEG(3)
1024        USQRT(47)
1025        ULN(77)
1026        UEIDI(14)
1027        UDIDI(18)
1028        UEXR(1)
1029        UEXR1(60)
1030        UCOS(2)
1031        USIN(58)
1032        UARCTAN(58)
1033        UETTR(6)
1034        UTTR(42)
1035        UCTTR(77)
1036        UMOD(17)
1037        UOBSOLEET1(9)
1038        UCOMPOSE(10)
1039        UBIT(15)
1040        USHIET(26)
1041        UOR(6)
1042        UAND(6)
1043        UTAIL(3)
1044        UHEAD(3)
1045        URLSTAR(15)
```

arithmetic subroutines

logical subroutines

```
1046          URACKEDCODE(10)
1047          UMOVE(55)
1048          UEILLRLSEG(26)
1049          UGREER25(1)
1050          UGREER26(1)
1051          UGREER27(1)
1052          UGREER28(1)
1053          DRT(20)
1054          UWS(8)
1055          QNZINN1(2)
1056          QNZINN3(0)
1057          QNZINN4(10)
1058          UDISASTER1(21)
1059          UDISASTER2(24)
1060          QNZINN9(2)
1061          QNZINN11(1)
1062          QNZINN5(0)
1063          QNZINN8(0)
1064          QNZINN10(2)
1065          QNZINN12(2)
1066          QNZINN6(0)
1067          QNZINN7(3)
1068          RSE56(2)
1069          RSE64(2)
1070          RSE116(2)
1071          RSE70(2)
1072          USVLIB(0)
1073    S     MVERTALER1(1)
1074          HALGOLBLOCK(511) I
1075          UVERTALER(38)
1076          ULIBRARY(20)
1077          URUIMTE(8)
1078    S     UINVARVALUE(1)
1079          UARVALUE(190)
1080          UOUTARRAY(19)
1081          UINVSE117(22)
1082          KARAKTERUIT(82)
1083          KARAKTERIN(160)
1084          UTIJD(13) I

1085    S     JMTRROC4(1)
1086          JREADBACK(20)
1087          CREADBACK1(68)
1088          UTELE5(20)
1089          UE267(12)
1090          UE268(22)
1091          UTELE0(5)
1092          UTELE1(14)
1093          UTELE2(10)
1094          UTELE3(7)
1095          UINVMTELE(2)
1096          UTELE4(22)
1097          INAL(20)
1098          UTELESTRING(31)
1099          CINVCOR(5)
1100          INAL2(7)
1101          CSETSTRING(20)
1102          CSTRINGSYM(14)  I
```

} emergency reaction for interrupts on M[25] ... [28] should not occur

} introduction of a segment HALGOLBLOCK is the point where the translator will be activated by a PM

" ACTUAL LENGTH = 43

→ segment containing some mag. tape procedures.

```
1103      S   CRUNSEG(1)
1104          UJOB3(6)
1105          CRUNEMPTY(7)
1106          CINVRUNSTAT(5)
1107          CAFLOOR(20)
1108          CRUNKOP(24)
1109          CRUNABS(30)
1110          CRUNSTAT(192)
1111          CLINENUMBER(14)
1112          CORRERR(43) I
1113      S   JMTPROC3(1)
1114          JMTNBKMES(4)
1115          JMTMESOOC(7)
1116          JMTMESREW(9)
1117          JLOADREEL(4)
1118          JONUNIT(3)
1119          JEINDMES(11)
1120          JTAPEVERSLAG(43)
1121          JINVM1(7)
1122          JINVM2(5)
1123          JMTNBK(30)
1124          JMTVERSLAGKOP(26)
1125          JMTVERSLAGMES(19)
1126          JMTMODELEES(4)
1127          JMTMODESCHRIJF(4)
1128          JMTERRORMES(7)
1129          UF249(22)
1130          UF250(16)
1131          UF251(13)
1132          UF252(14)
1133          UF253(19)
1134          UF254(11)
1135          UF255(25)
1136          UF256(24)
1137          UF257(25)
1138          UF258(16)
1139          UF259(12)
1140          UF260(24)
1141          UF261(16)
1142          UF262(16)
1143          UF263(15)
1144          UF264(19)
1145          UF265(21)
1146          UF266(13)
1147          JINVM3(6) I

1148      S   JMTPROC1(1)
1149          JINSRTAPE(17)
1150          JWRITETAPE(25)
1151          CWRITETAPE1(9)
1152          JREADTAPE(16)
1153          CREADTAPE1(19)
1154          JWRITEINFOBL(36)
1155          JREADINFOBL(19)
1156          JREADBL(8)
1157          JBLOCKCHECK(21)
1158          JSROELE(2)
1159          JSROELB(2)
1160          JREWIND(27)
1161          JSLOTVOORDMT(78)
```

mag. tape. proc.

```
1162          JEINTARE(9)
1163          JSTARTMT(7)
1164          JWRITETELBL(16)
1165          JWRITETAREM(12)
1166          JSROELFORW(5)
1167          JREADTELBL(8)
1168          JSROELBACKW(29)
1169          JOUTOFCONMES(3)
1170          JREWINDMES(23)
1171          JREADB(5)
1172          JNEWREEL(26)
1173          JMOVETARE(86)
1174          JALJUMP(3) I

1175      S   JMTPROC2(1)
1176          JORENTARE(133)
1177          JCLOSETARE(100)
1178          JTARESERR(2)
1179          JTARESERE(86)
1180          JWRITEPOSITION(127)
1181          JREADPOS1(45)
1182          JREADPOSITION(2)
1183          JALARM(16) I

1184      S   UINVSESEG4(1)
1185          UINVAND(26)
1186          UINVOR(26)
1187          UINVSHIFT(33)
1188          UINVBIT(26)
1189          UINVHEAD(17)
1190          UINVTAIL(17)
1191          UINVCOMPOSE(26)
1192          UE235(13)
1193          UE236(9)
1194          UE237(9)
1195          UE238(12)
1196          UE239(12)
1197          UE240(12)
1198          UE241(12)
1199          UE242(12)
1200          UE243(12)
1201          UE244(7)
1202          UE245(11)
1203          UE246(12)
1204          UE247(19)
1205          UE248(15)
1206          UEOUTREGEL(12)  I
1207      S   UINVSESEG(1)
1208          UINVRLOT(397)
1209          UINVRLOTFRAME(102) I
1210      S   UINVSESEG3(1)
1211          UINVRLOTTEXT(319)
1212          UINVFIXRLOT(1)
1213          UINVABSFIXRLOT(1)
1214          UINVFLOPPLOT(121) I
1215      S   UINVSESEG5(1)
1216          UINVRLOTCURVE(511) I
1217      S   UINVSESEG6(1)
1218          UINVRLOTCURVE1(411) I
```

logical procedures

plotter procedures

```
1219     S   UINVSESEG7(1)
1220         UINVENTRANCE(143)
1221         UR0(4)
1222         UR1(2)
1223         UR2(4)
1224         UR3(5)
1225         UR4(2)
1226         UR5(4)
1227         UR6(5)
1228         UR7(2)
1229         UR8(6)
1230         UR9(5)
1231         URA(4)
1232         URB(4)
1233         URC(3)
1234         URD(4)
1235         URE(4)
1236         URF(3)
1237         URG(5)
1238         URH(3)
1239         URI(2)
1240         URJ(3)
1241         URK(3)
1242         URL(2)
1243         URM(4)
1244         URN(3)
1245         URO(4)
1246         URP(4)
1247         URQ(4)
1248         URR(3)
1249         URS(4)
1250         URT(4)
1251         URU(3)
1252         URV(2)
1253         URW(4)
1254         URX(2)
1255         URY(4)
1256         URZ(2)
1257         URA(4)
1258         URB(5)
1259         URC(4)
1260         URD(3)
1261         URE(3)
1262         URF(3)
1263         URG(4)
1264         URHH(3)
1265         URI(3)
1266         URJ(2)
1267         URK(3)
1268         URL(2)
1269         URM(2)
1270         URN(2)
1271         URO(2)
1272         URP(3)
1273         URQ(5)
1274         URR(4)
1275         URS(5)
1276         URT(2)
1277         URU(3)
1278         URV(2)
```

```
1279          URW(2)
1280          URX(3)
1281          URY(2)
1282          URZ(2)
1283          URPLUS(2)
1284          URMINUS(1)
1285          URTIMES(2)
1286          URDIV(1)
1287          URINTDIV(5)
1288          URHUR(2)
1289          UREQUALS(2)
1290          URUNEQUALS(3)
1291          URSMALLER(2)
1292          URATMOST(3)
1293          URGREATER(2)
1294          URATLEAST(3)
1295          URNON(2)
1296          UREQUIVALENT(3)
1297          URIMPLIES(3)
1298          UROR(2)
1299          URAND(2)
1300          URCOMMA(3)
1301          URPERIOD(2)
1302          URDRORREDTEN(4)
1303          URCOLON(4)
1304          URSEMICOLON(5)
1305          UROPEN(2)
1306          URCLOSE(2)
1307          URSUB(2)
1308          URBUS(2)
1309          URSTRINGOPEN(3)
1310          URSTRINGCLOSE(3)
1311          URAPOS(3)
1312          URDITOS(5)
1313          URQUESTIONMARK(5)
1314          URUNDER(2)
1315          URSTROKE(1)
1316          URMARK2(3)
1317          URMARK3(3)
1318          URMARK4(4)
1319          URMARK5(5)
1320          URMARK6(3)
1321          URMARK7(3)
1322          URMARK8(2)
1323          URMARK9(3)
1324          URMARK10(3)
1325          URMARK11(4)
1326          URMARK12(3)
1327          URBLANK(1) I

1328    S     UINVARITHMSEG(1)
1329          UINVENTIER(17)
1330          UINVSQRT(17)
1331          UINVABS(18)
1332          UINVLN(17)
1333          UINVEXP(17)
1334          UINVCOS(17)
1335          UINVSIN(17)
1336          UINVARCTAN(17)
1337          UINVSIGN(19)
```

plotter

arithmetic procedures

```
1338          UINVMOD(19)
1339          UINVRE(17)
1340          UINVIN(19)
1341          UINVCOM(23)
1342          UINVGEC(22)
1343          UINVSE2(5)
1344          UINVSE3(5)
1345          UINVSE4(4)
1346          UINVSE5(219)  I

1347      S   CTRRROC1(1)
1348          CRUTEXT2(2)
1349          CRUTEXT1(45)
1350          CRUTEXT1(11)
1351          CONRUNUM(28)
1352          CRUSPACE1(2)
1353          CRUSPACE2(2)
1354          CRUSPACE1(16)
1355          CRUSYM1(2)
1356          CRUSYM2(2)
1357          CRUSYM1(9)
1358          CRUCHARE1(2)
1359          CRUCHARE2(2)
1360          CRUCHARE1(9)
1361          CRUNLCR1(2)
1362          CRUNLCR2(2)
1363          CRUNLCR1(4)
1364          CRUTAB1(2)
1365          CRUTAB2(2)
1366          CRUTAB1(173)
1367          RUCHARE(177)
1368          CRUNOUT(11)  I
1369      S   CTRRROC2(55)
1370          CRUOCT1(2)
1371          CRUOCT2(2)
1372          CRUOCT1(23)
1373          CRURUNOUT1(2)
1374          CRURUNOUT2(2)
1375          CRURUNOUT1(17)
1376          CENDTRDOC2(2)
1377          CENDTRDOC1(7)
1378          CENDTRDOC1(46)
1379          CBEGINTRDOC(101)
1380          CVISRU(147)
1381          CINVFORMMES(8)
1382          CLRS1(15)  I

1383      S   UINVSESEG1(1)
1384          UINVEXIT(103)
1385          UNORMALEXIT(247)
1386          URRINTREGEL(89)
1387          UIDPRINT(69)  I
1388      S   UINVSESEG2(1)
1389          URRINTERR(183)
1390          ULASTFORM(249)
1391          UCALCULATION(14)
1392          UINVEXECUTION(12)
1393          ORSMUK2(10)  I

1394      S   UINVTOTALETIJD(5)
```

array declaration

puncher procedures

→ final rites

```
1395          UINVVERTAALTIJD(5)
1396          UINVEXECUTIETIJD(5)
1397          UINVAANTALOVER(7)
1398          UINVGETALLEN(6)
1399          UINVLAATSTOVER(7)
1400          UINVGETAL(5)
1401          UINVPROGRAMMADOOR(13)
1402          UINVSTAPEL(8)
1403          UINVSEGMENTEN(6)
1404          UINVKETTING(7)
1405          UINVLEEG(2)
1406          UINVPARAMETER(8)
1407          UINVPROCEDURE(4)
1408          UINVSWITCH(4)
1409          UINVDECLARATIE(5)
1410          UINVCALL(4)
1411          UINVVERTALER(6)
1412          UINVBIBLIOTHEEK(6)
1413          UINVREGEL(3)
1414          UFAILL(100)
1415          UF99(8)
1416          UF200(27)
1417          UF201(11)
1418          UF202(14)
1419          UF203(24)
1420          UF204(31)
1421          UF205(20)
1422          UF206(20)
1423          UF207(22)
1424          UF208(21)
1425          UF209(21)
1426          UF210(24)
1427          UF211(23)
1428          UF212(23) I
1429     S    UF213(21)
1430          UF214(25)
1431          UF215(24)
1432          UF216(24)
1433          UF217(20)
1434          UF218(20)
1435          UF219(20)
1436          UF220(36)
1437          UF221(30)
1438          UF222(42)
1439          UF223(18)
1440          UF224(19)
1441          UF225(36)
1442          UF226(31)
1443          UF227(31)
1444          UF228(12)
1445          UF229(13)
1446          UF230(27)
1447          UF231(13)
1448          UF232(18)
1449          UF233(13)
1450          UF234(11) I

1451     S    UINVRUN1(0) I
1452     S    UINVRUN2(0) I
1453     S    UINVRUN3(0) I
```

message strings

dynamic error strings

```
1454      S   UINVRUN4(0) I
1455      S   UINVRUN5(0) I
1456      S   UINVRUN6(0) I
1457      S   UINVRUN7(0) I
1458      S   UINVRUN8(0) I

1459      S   UINVBSEG(1)
1460          UJOB1(159)
1461          USRCR(5)
1462          USR(5)
1463          UFILLSTRING(17)
1464          UNEXTSYM(17)
1465          UNINT(12)
1466          UNAC(23)
1467          UTARESKIP(19)
1468          UJOB2(6)
1469          UINVBEGINRLDOC(218)
1470          CENDRLDOC(30) I

1471      S   CLRRROC2(1)
1472          CABSFIXT(2)
1473          CFIXT(2)
1474          CFLOT(42)
1475          CRRNTX(28)
1476          CRNCH2(2)
1477          CRNCH1(36)
1478          CRNCH(5)
1479          CRAGEHEADING(59)
1480          CRUVISNUM(4)
1481          CONVRT(64)
1482          CFIXFLO(166)
1483          CABSFIXR2(2)
1484          CABSFIXR1(3)
1485          CFIXR2(2)
1486          CFIXR1(3)
1487          CFLOR2(2)
1488          CFLOR1(63)
1489          CABSFIXR(2)
1490          CFIXR(2)
1491          CFLOR(7)
1492          CONVERTNUM(9) I
1493      S   CLRRROC1(54)
1494          CRRINTTEXT(22)
1495          CHEADN(8)
1496          CTAB(15)
1497          CONNUM(6)
1498          CENDLRDOC(8)
1499          CNEWPAGE(34)
1500          CRRSYM(50)
1501          CNLCR(3)
1502          CHEADS(15)
1503          CRRCHAR(78)
1504          CSPACE(13)
1505          CARRIAGE(13)
1506          CLRRNT(191)  I

1507      S   CTARERROC1(145)
1508          COCT2(2)
1509          COCT1(10)
1510          COCT(2)
```

initial rites

number conversion printing & punching

printer procedures

```
1511          CHARE2(2)
1512          CHARE1(4)
1513          CHARE(2)
1514          CNUMINSR(28)
1515          CNUM2(2)
1516          CNUM1(8)
1517          CNUM(113)
1518          CSYMINSR(69)
1519          CSYM2(2)
1520          CSYM1(5)
1521          SYM(116)
1522          CSYM(2) I
1523      S   CTARERROC2(1)
1524          CODEDEF(51)
1525          CBACKCHARE(6)
1526          CBACKSYM(4)
1527          CBACKOCT(13)
1528          CINICHARE(48)
1529          CINIPATA(4)
1530          CSKIPPATA(78)
1531          CORRECTION(177)
1532          CRUNCHLINE(105)
1533          CNUMCOR(20) I
1534      S   CINVM30(6)
1535          CINVM31(3)
1536          CINVM32(5)
1537          CINVMROOL(2)
1538          CINVMINPUT(3)
1539          CINVMFREE(3)
1540          UINVM361(2)
1541          UINVM362(2)
1542          UINVM363(2)
1543          UINVM364(2)
1544          UINVM365(2)
1545          UINVM351(6)
1546          UINVM352(6)
1547          UINVM353(6)
1548          UINVM354(6)
1549          UINVM355(6)
1550          UINVM341(7)
1551          UINVM342(7)
1552          UINVM343(7)
1553          UINVM344(7)
1554          UINVM345(7)
1555          CINVMNN(2)
1556          CINVM13(3)
1557          CINVM9(8)
1558          CINVM7(5)
1559          CINVM6(5)
1560          UINVM12A(7)
1561          UINVM12B(7)
1562          UINVM12C(7)
1563          UINVM8A(4)
1564          UINVM8B(4)
1565          UINVM8C(4)
1566          UINVM14A(8)
1567          UINVM14B(8)
1568          UINVM14C(8)
1569          UINVM1A(2)
1570          UINVM1B(2)
```

tape reader procedures

messages

```
1571          UINVM1C(2)
1572          UINVWHO1(10)
1573          UINVWHO2(10)
1574          UINVWHO3(10)
1575          UINVWHO4(10)
1576          UINVWHO5(10)
1577          CRAPERTORN(7)
1578          CRARERLOW(7)
1579          CYOKEOREN(7)
1580          CRARITY(6)
1581          CNBKMES(6)
1582          UINVTIME(6)

1583          CINVM21(7)
1584          CINVPLM21(4)
1585          UINVM20A(6)
1586          UINVM20B(6)
1587          UINVM20C(6)
1588          UINVPLM20A(5)
1589          UINVM23A(5)
1590          UINVM23B(5)
1591          UINVM23C(5)
1592          UINVPLM23A(11)
1593          UINVTRM26A(7)
1594          UINVTRM26B(7)
1595          UINVTRM26C(7)
1596          UINVPLM26A(7)
1597          UINVTRM28A(12)
1598          UINVTRM28B(12)
1599          UINVTRM28C(12)
1600          UINVPLM28A(12)
1601          UINVBLANK(1)
1602          UINVTRA(5)
1603          UINVTRB(5)
1604          UINVTRC(5)
1605          UINVALLHOLE(1)
1606          UINVALLHOLE4(2)
1607          UINVARROW1(3)
1608          UINVARROW2(2)

1609          UINVMRU(2)
1610          UINVMPLO(2)
1611          UINVMT(2)
1612          UINVMNONSENS(4)
1613          UINVMINACC(5)
1614          UINVMBEGIN(3)
1615          UINVDESTROYED(9)
1616          UINVRUNSTAT(6)
1617          UINVPRTIME(2)
1618          UINVPREND(2) I
1619          USTACKPLA(26)
1620          USBPLA(24)
1621          UWPPLA(29)
1622          UEXTRA1(10)
1623          USTACKTRA(26)
1624          USBTRA(23)
1625          UWPTRA(37)
1626          UEXTRA2(10)
1627          USTACKTRB(26)
1628          USBTRB(23)
```

end of corespace for library segment variables

```
1629          UWRTRB(37)
1630          UEXTRA3(10)
1631          USTACKTRC(26)
1632          USBTRC(23)
1633          UWRTRC(37)
1634          UEXTRA4(10)
1635          USTACKLRA(26)
1636          USBLRA(24)
1637          UWRLRA(30)
1638          UEXTRA5(10)
1639          USTACKTRA(26)
1640          USBTRA(18)
1641          UWRTRA(28)
1642          UEXTRA6(10)
1643          USTACKTRB(26)
1644          USBTRB(18)
1645          UWRTRB(28)
1646          UEXTRA7(10)
1647          USTACKTRC(26)
1648          USBTRC(18)
1649          UWRTRC(28)
1650          UEXTRA8(10)
1651          USTACKDR(26)
1652          USBDR(4)
1653          UWRDR(3)
1654          UEXTRA9(10)
1655          USTACKMI(26)
1656          USBMI(5)
1657          UWRMI(30)
1658          UEXTRA10(10)
1659          UTRHANDLE11(26)
1660          UTRHANDLE12(26)
1661          UTRHANDLE21(26)
1662          UTRHANDLE22(26)
1663          UTRHANDLE31(26)
1664          UTRHANDLE32(26)
1665          UTRHANDLE41(26)
1666          UTRHANDLE42(26)
1667          UTRHANDLE51(26)
1668          UTRHANDLE52(26)
1669          ULRHANDLE11(30)
1670          ULRHANDLE21(30)
1671          ULRHANDLE31(30)
1672          ULRHANDLE41(30)
1673          ULRHANDLE51(30)
1674          UTRHANDLE11(24)
1675          UTRHANDLE12(24)
1676          UTRHANDLE21(24)
1677          UTRHANDLE22(24)
1678          UTRHANDLE31(24)
1679          UTRHANDLE32(24)
1680          UTRHANDLE41(24)
1681          UTRHANDLE42(24)
1682          UTRHANDLE51(24)
1683          UTRHANDLE52(24)
1684          URLHANDLE11(25)
1685          URLHANDLE21(25)
1686          URLHANDLE31(25)
1687          URLHANDLE41(25)
1688          URLHANDLE51(25)
```

Stack bottoms and permanent information input/output CM's

" TWEE EXTRA PLAATSEN IN ELK HANDLE

own variables of information stream through which documents are built up and broken down.

```
1689          WSTACK(22)
1690          WUNSTACK(22)
1691          WGENERATE(50)
1692          WAEREKENING(15)
1693          WBERGR(25)
1694          JLABELMT(6)
1695          JMTEREE(1)
1696          UIETMT(10)
1697          JUNITLIST(5)                    " ONE PLACE RESERVED FOR EXTRA TAPE UNIT
1698          JTELBLOCKWRITE(3)
1699          JUNITMES(46)
1700          CSERONG(9)
1701          CMTRADRES(4)
1702            CMTR = '56 34 00000'
1703          CMTVADRES(3)
1704            CMTV = '56 34 00000'
1705          CWRITETAPE(37)
1706          CINSRTAPE(9)
1707          WRAG(1)
1708          URROGNAME(22)
1709          CCOCT(10)
1710          JOBSOLEET(1)
1711          JTELBLOCKREAD(3)
1712          UNEVARROW1(6)
1713          CLRRED(12)
1714          WRESTIT(12)
1715          JMTEXSEM(3)
1716          JTAPEMWRITE(3)
1717          ORSMUK1(16)
1718          UOBSOLEET2(2)

1719          CSCALE(10)
1720          UTELDOOR(12)
1721          UDRUMTIME(6)
1722          UIETRC(10)
1723          UIETTT(10)
1724          CASERC(1)
1725          UTWNR(1)
1726          UGARR(1)
1727          CASEDVTT(1)
1728          UKLDW(1)
1729          UKLACW(1)
1730          UMAGADRES(1)
1731          UMAGAZYN(UMAGLE)
1732          UMAXMAG = 9
1733          UMWR(1)
1734          TTLABEL(3)
1735          CONSOLECLAIMED(1)
1736          FORSEM(3)
1737          UEXRINT2(7)
1738          CSLOWMES(8)
1739          CFORMSMES(9)
1740          JUNITADM1(23)
1741          JUNITADM2(23)
1742          JUNITADM3(23)
1743          CREADBACK(6)
1744          CREADKOR(26)
1745          CREADTAPE(14)
1746          CREELCHECK(14)
1747          CFASTREAD(3)
```

mag. tape mutual exclusion semaphore

odd collection of later extensions

corefilling magnetic tape

```
1748 C '26 333'
1749          UCSTOR(1)
1750          UCT(UCTLEN)
1751 C '74 000'
1752          CINBRO(86)
1753          CVIER(22)
1754          CEEN(4)
1755          CDATA(5)
1756          CPASTA(15)
1757          CHEP(58)
1758          CSTART(10)
1759          CHBUF(2)
1760          CEB(1)
1761          CEINDE(1)
1762          CHBUFADR(1)
1763          CHOK(3)
1764          CBUFA(355)
1765          CBUFB(355)
1766          CWERK(35)
1767          CINIT(29)
1768          USYSTEMSTART(38)
1769          USVINBRO(4)
```

} core table containing 4 words for each core-page of 512 that may be occupied by a segment.

} temporary space for the input program.
will be overwritten once system has come into action.

```
 0 P :UINTERRUPTJUMP(5)
 1          GOTO(:CINPRO)
 2          - 0
 3          - 0
 4          - 0
 5          - 0
 6 P :UARMT(3)
 7          :ULABELMT[1]
 8          1
 9          0

10 P :CINPRO(CPRIML)
11          B = :CINPRO[CPRIML]
12          A = MC
13          A + MC
14     U,   B - :CINPRO[- 1], P
15     Y,   JUMP(-3)
16     U,   A + 0, Z                    " SUMCHECK OK?
17     N,   A = 1
18     N,   JUMP(- 1)
19          A = - 0
20          B = 0
21          MC = A
22     U,   B - 23, P
23     N,   JUMP(- 3)
24          B = 29
25          MC = A
26     U,   B - 56, P
27     N,   JUMP(- 3)
28          B = 256
29          MC = A
30     U,   B - :CINPRO[- 1], P
31     N,   JUMP(- 3)
32          B = :CINPRO[CPRIML]
33     U,   B - CDATA[0], P
34     N,   MC = A
35     N,   JUMP(- 3)                   " NOW REMAINING MEMORY HAS BEEN CLEARED TO - 0
36          B = :CPASTA
37          A = M[B], Z                  " BRUSHING TEETH FINISHED?
38     Y,   JUMP(8)
39          M[216] = A                   " ARO VAN DE TANDENPOETSER
40          '760 070 000'[38]            " AF TANDENPOETSER := TRUE
41          '660 075 000'                " IF-WOORD IN A
42          LCA(2), P                    " IF TANDENPOETSER NOG FALSE?
43     Y,   JUMP(- 3)
44          '660 071 000'[38]            " ZET IF TANDENPOETSER WEER FALSE
45          B + 1
46          JUMP(- 10)                   " BRUSH NEXT APPARATUS
47          S = 0
48          CHBUF = S
49          B = :CWERK                   " INITIALISE WORKING SPACE POINTER
50          A = :CBUFA[1]
51          CBUFB[1] = A
52          A - CBUFA[2]
53          A - :CBUFB[1]
54          CBUFB[2] = - A
55          SUBCD(:CEEN)                 " GET OCTAD # 0
56     U,   A - 42, Z                    " CORE FILLING?
57     N,   JUMP(13)
```

primary input program

initialization peripherals

```
58          SUBCD(:CVIER)                   " CHECKING PARITY OF QUARTET, VALUE DELIVERED IN A
59          MC = A                          " STACK STARTING ADDRESS
60          SUBCD(:CVIER)                   " READ NUMBER OF WORDS
61          MC = A, Z                       " STACK NUMBER OF WORDS TO BE READ; FINISHED?
62      Y,  B - 2                           " RESTORE STACK POINTER
63      Y,  JUMP(- 9)
64          SUBCD(:CVIER)
65          S = M[B - 2]
66          MS = A                          " FILL CORE LOCATION
67          A = - 1
68          M[B - 2] - A                    " INCREASE CORE ADDRESS
69          A + MC[- 1]                     " DECREASE NUMBER UNSTACKING)
70          JUMP(- 10)
71          SUBC(:CINIT)                    " INITIALISE FLIPFLOPS AS SYSTEM REQUIRES
72      U,  A - 28, Z                       " SEGMENT FILLING?
73      N,  JUMP(16)
74          B + 1                           " MAKE ROOM FOR NUMBER
75          SUBCD(:CVIER)                   " READ INVARIANT ADDRESS
76          MC = A                          " STACK
77          SUBCD(:CVIER)                   " READ NUMBER OF SEGMENT WORDS
78          M[B - 2] = A . Z                " NO WORDS ON SEGMENT?
79      Y,  JUMP(7)
80          SUBCD(:CVIER)                   " GET FIRST WORD
81          G = A                           " TRANSFER TO E FOR THE BENEFIT OF PSE50
82          SUBCD(:PSE50)                   " ASSIGN TO INTEGER ARRAY ELEMENT, STACK POINTER WILL BE DECREASED BY 1
83          S = 1
84          MC + S                          " INCREASE INVARIANT ADDRESS, RESTORING B
85          M[B-2] - S, Z                   " DECREASE NUMBER, FINISHED?
86      N,  JUMP(-7)
87          B - 2                           " RESET STACK POINTER
88          SUBCD(:CEEN)
89          JUMP(- 18)
90      U,  A - 65, Z                       " END OF SYSTEM?
91      N,  A = MA[ - 256]
92      N,  GOTO(:CINRRO[36])
93          CEINDE = B
94          SUBC(:CHEP)                     " READ UNTIL END OF TAPE
95      N,  JUMP( - 2)
96          GOTO(:USYSTEMSTART)             ⟶ to initialize system

97  "CVIER(22)
98          E = 4
99          MC = E                          " M[B - 2] = PARITY CHECK, M[B - 1] = 4
100         A = 0
101         MC = A                          " SAVE PARTLY BUILT QUARTET
102         SUBC(:CHEP)
103         MC = A
104         A = MC[ - 1]                    " TO FIND PARITY
105         CLR   set condition to last parity
106         A '+' MC[ - 1]
107     N,  JUMP(2)
108         G = M[B - 2]
109         M[B - 2] = - G                  " INVERT PARITY CHECK
110         S = 1
111         M[B - 1] - S, Z                 " 4 HEPTADS DONE ?
112     N,  LCA(7)
113     N,  A '*' - 127
114     N,  JUMP( - 14)
115         S = M[B - 2], R                 " EVEN PARITY ?
116     N,  B - 2
```

```
117      N,  GOTOR(MC[ - 1])
118      U,  S = MS[ - 1]
119          GOTO(:CINRRQ[36])
120  "CEEN(4)
121          SUBC(:CHEP)
122          A + 0, Z
123      N,  GOTOR(MC[ - 1])
124          JUMP( - 4)
125          '137 777'
126          '777 777'
127          0
128          '776 000 010'                  " FOR AR1, SETTING LET TO - 2
129          '001 000 000'                  " = 2 ↑ 18
130          '000 004 000'[18]              " THIS IS ADDRESS CPASTA
131          '000 004 000'[3]               " LVAF = TRUE; LVCA = FALSE
132          '000 004 000'[8]               " CONSOLE PRINTER
133          '000 004 000'[19]
134          '000 004 000'[0]               " PUNCHERS
135          '000 004 000'[2]
136          '000 004 000'[17]
137          '000 004 000'[16]              " LINE PRINTER
138          '000 004 000'[11]              " DRUM
139          '100 004 000'[9]               " CONSOLE KEYBOARD; LVCA = TRUE
140          '100 004 000'[20]
141          '000 004 000'[4]               " PLOTTER
142          '000 004 000'[10]              " MAGNETIC TAPE
143          '000 004 000'[39]              " REAL TIME CLOCK TO REMAIN DORMANT
144          - 0

145  "CHEP(58)
146          S = CHBUF, Z                   " NO BUFFERS USED YET ?
147      Y,  JUMP(16)
148          S + 353
149          A = CHBUFADR
150      U,  A = S, P
151      Y,  JUMP(26)
152          A = MA
153          S = 1
154          PLUSS(CHBUFADR)
155      U,  S = CEB, P
156      N,  GOTO(MC[- 1])
157      U,  S = CEINDE, P                  " SKIPPING REMAINDER OF SYSTEM TAPE ?
158      Y,  GOTO(MC[ - 1])                 " RETURN WITH Y CONDITION
159          A = AROGR[3]
160          A'*' CDATA[1]                  " MAKE ADDRESS OF LAST FILLED WORD
161      U,  S - A, P                       " LAST WORD BEFORE EOT ?
162      N,  A = MS[ - 1]                   " FETCH HEPTAD ONCE MORE
163      N,  GOTO(MC[ - 1])
164          CEB = - S                      " NOTE OCCURRENCE OF EOT
165          S = :CHOK[1]
166          S - AROGR
167          AROGR + S
168          S = CDATA[3]
169          AROGR[1] = S
170          S = CDATA[4]
171          AROGR[2] = S                   " AET := 1
172          '760 070 000'[UNRGR]           " AF := TRUE
173          A = AROGR[1]
174          RUA(18), Z
175      N,  JUMP( - 3)                     " WAIT FOR HOK TO BE FINISHED
```

subroutine to deliver one heptad from paper tape read in two-buffer mode.

```
176           S = :CBUFA[1]
177           A = :CBUFB[1], Z                  " TO MAKE N COND.
178      Y,   S = CHBUF[1]
179      Y,   A = CHBUF
180           CHBUF = S
181           CHBUF[1] = A
182      N,   SUBC(:CSTART)
183           S = CHBUF[1]
184           SUBC(:CSTART)
185           A = AROGR[1]
186           LCA(9)
187           A '*' 2, Z
188      Y,   JUMP( - 4)                        " WAITING FOR AT LEAST ONE IET INCREASE
189           A = CHBUF
190           A + 2
191           CHBUFADR = A
192           S = MA[ - 3]                       " LABEL[ - 1]
193           LCS(9)
194           S '*' 511, Z                       " OK?
195      Y,   JUMP( - 44)                        " TO FETCH HEPTAD
196           S - 510, Z                         " EOT ?
197      Y,   CEB = - S                          " NOTE OCCURRENCE OF EOT
198      N,   A = 3
199      N,   JUMP( - 1)                         " NBK
200           S = AROGR[1]
201           RUS(18), Z                         " CHARON READY, I. E. IET =  - 1 ?
202      N,   JUMP( - 3)
203           JUMP( - 52)
204 "CSTART(10)
205           A = CDATA[4]                       " 2 + 1
206           AROGR[1] - A                       " IET := IET - 1
207           RLUSA(AROGR[2])                    " AET := AET + 1
208           LCA(8), P                          " AET ≠ 1 ?
209      N,   A = CDATA[1]
210      N,   A '*' AROGR
211      N,   S - :MA
212      N,   AROGR + S
213           '760 370 000'[UNRGR]               " N, AE := IBUE
214           GOTOR(MC[ - 1])
215           - 0                                " CHBUF
216           - 0                                " CHBUF[1]
217           - 0                                " CEB, EALSE
218           - 0                                " CEINDE, EALSE
219           - 0                                " CHBUFADR
220 "CHOK(3)
221           0
222           0
223           0
224 "CBUFA(3)
225           0
226           :CBUFB[1]
227           '540 000 000'[:CBUFA + 2]

228 CINIT(29)
229           '660 072 000'                      " LVIE PUNCH A FALSE
230           '660 072 001'                      " LVIE READER A FALSE
231           '660 072 002'                      " LVIE PUNCH B FALSE
232           '660 072 003'                      " LVIE READER B FALSE
233           '660 072 004'                      " LVIE PLOTTER A FALSE
234           '660 072 010'                      " LVIE CONSOLE PRINTER FALSE
```

} original value of masking bits of peripherals

```
235          '660 072 011'                  " LVIE CONSOLE KEYBOARD FALSE
236          '660 072 012'                  " LVIE MAGNETIC TAPE FALSE
237          '660 072 013'                  " LVIE DRUM FALSE
238          '660 072 020'                  " LVIE LINE PRINTER FALSE
239          '660 072 021'                  " LVIE PUNCH C FALSE
240          '660 072 022'                  " LVIE READER C FALSE
241          '660 072 023'                  " LVIE CONSOLE PRINTER B FALSE
242          '660 072 024'                  " LVIE CONSOLE KEYBOARD B FALSE
243          '760 070 011'                  " AE CONSOLE KEYBOARD TRUE
244          '760 070 024'
245          '660 071 001'                  " IE GOLDEN READER FALSE
246          $ = CINIT[28]
247          CINIT[8] = $
248          $ = CINIT[27]
249          M[216] = $                     " FILL ARO TANDENPOETSER
250          '760 070 000'[38]              " AE TANDENPOETSER TRUE
251          '760 075 000'                  " IE WORD IN $
252          LCS(2), P                      " BRUSHING NOT FINSHED?
253     Y,   JUMP(- 3)
254          '660 071 000'[38]              " IE TANDENPOETSER FALSE
255          GOTOR(MC[- 11)
256          '100 000 000'[39]              " LVCA CLOCK TRUE
257     U,   $ = 0
258 USYSTEMSTART(38)
259          $ = 1
260          B = :UPR[UNAM]
261          G = MC[- 1]
262     U,   $ + UHAMR[UG], Z               " AM BLOCKED DURING SEGMENT FILLING?
263     Y,   UHAMR[UG] + $                  " AM BECOMES SELECTABLE
264     Y,   USEL + $                       " UPDATE NUMBER OF SELECTABLE AM'S
265     Y,   A = UHAMR[UG + 1]
266     Y,   URSEL + A                      " UPDATE NUMBER OF RED SELECTABLE AM'S
267     U,   B - :UPR[, Z
268     N,   JUMP(- 8)
269          UARTRA[1] = $                  " SET GOLDEN READER := 0
270          A = MT[25]
271          UHAMR[2] = A                   " SET T IN HAMROOM RM1
272          A = - 0
273          UHAMR[3] = A                   " SET A IN HAMROOM RM1
274          A = - 0
275          UHAMR[4] = A                   " SET S IN HAMROOM RM1
276          F = - 0
277          UHAMR[5] = F                   " SET E IN HAMROOM RM1
278          B = :UWRPM1
279          UHAMR[7] = B                   " SET B IN HAMROOM RM1
280          F = 1
281          UBELSOM = F                    " TOTAL LOAD := 1
282          F/UNUMRM
283          UHAMR[9] = F                   " LOAD RM1 := 1/UNUMRM
284          A = - 0
285          GL24 = A                       " NUMBER OF R-SEGMENTS RM1 := - 0
286          UWARNINGPOINT = A              " WARNINGPOINT RM1 := - 0
287          $ = :USVINPRO[3]
288          SUBCD(:UKILLSEG)
289          $ = :USVINPRO[1]
290          SUBCD(:UKILLSEG)
291          $ = :USVINPRO                  " ADDRESS SV INPUT PROGRAM SEGMENT
292          SUBCD(:UKILLSEG)
293          $ = :USVINPRO[2]
294          SUBCD(:UKILLSEG)
```

make PM that has performed reading of system look like all other PM's.

free the 4 core pages that contain input program and systemstart

} → segment containing this text killed last.

```
295             GOTO(:UNEXTINT)
296             :UTEXTRM[1]                     " VALUE T IN HAMROOM RM
297     USVINRRQ(4)
298             :UCT[UCTLEN - 132]
299             :UCT[UCTLEN - 136]
300             :UCT[UCTLEN - 140]
301             :UCT[UCTLEN - 144]

302     UINTERRUPTJUMP(5)
303             '563 000 000'[:UINT]            " LINK ON M[17]
304             '560 000 000'[:UGREER25]        " LINK ON M[8]
305             '560 000 000'[:UGREER26]
306             '560 000 000'[:UGREER27]
307             '560 000 000'[:UGREER28]
308     UARTRA(3)
309             + 0
310             + 1                             " IRT = 0
311             + 0                             " ART = 0
312     UARTRB(3)
313             + 0
314             + 1
315             + 0
316     UARTRC(3)
317             + 0
318             + 1
319             + 0
320     UARTRB(3)
321             + 0
322             + 1
323             + 0
324     UARTRC(3)
325             + 0
326             + 1
327             + 0
328     UARCR(3)
329             + 0
330             + 1
331             + 0
332     UARCRB(3)
333             + 0
334             + 1
335             + 0
336     UARCK(2)
337             + 0
338             + 1
339     UARCKB(2)
340             + 0
341             + 1
342     UARDR(3)
343             + 0
344             + 1
345             + 0
346     UARLRA(3)
347             + 0
348             + 1
349             + 0
350     UARRLA(3)
351             + 0
352             + 1
353             + 0
```

this instruction will still be executed in "deafness"

comm. channels

```
354 UARCLOCK(3)
355         + 0
356         UCLOCKPERIOD
357         + 0
358                                                                  " COORD
359                                                                  " INTERRUPT-2
360         " INTERRUPT PROGRAM
361 UINT(5)
362         M[18] = A                       " SAVE A FOR THE TIME BEING
363         A = UHAM                        " :USBRMI
364         UHAMR[UA + 4] = S               " SAVE S
365         UHAMR[UA + 5] = E               " SAVE E
366         E = M[17]                       " LINK IN E=HEAD; A IN E=TAIL
367 UINT1(3)                               " ENTRY POINT UR
368         UHAMR[UA + 2] = E               " SAVE A AND T
369         UHAMR[UA + 7] = B               " SAVE B
370         UCURRI = - A                    " INDICATION FREE CHOICE
371 UINT2(15)                              " ENTRY POINT UV
372         S = M[63]
373         UHAMR[UA + 8] = S               " SAVE D
374         G = M[220]                      " NEW CLOCK IN G
375         G - UKLB                        " -OLD CLOCK
376         E + 1
377         E * UHAMR[UA + 11]              " * WEIGHT
378         E * UBELSOM                     " * TOTAL LOAD
379         M[17] = E                       " SAVE INCREMENT
380         E + UBELSOM
381         UBELSOM = E                     " INCREASE TOTAL LOAD
382         E = M[17]
383         E + UHAMR[UA + 9]               " INCREASE LOAD
384         UHAMR[UA + 9] = E
385         B = UCURRI, P                   " NO FREE CHOISE?
386     Y,  GOTO(:UCURRENT)
387 UNEXTINT(37)                           " ENTRY POINT NEXT INTERRUPT
388         B = :URRI[UNAM]                 " B JUST ABOVE PRIORITY LIST
389         '761 075 000'                   " S = FIRST FLIPFLOP WORD, Z
390         '761 376 100'                   " N, S '*' FIRST LVIE WORD, Z
391     Y,  JUMP(8)
392     U,  S '*' UEXPINTW, Z               " NO EXCEPTIONAL INTERRUPTS?
393     N,  GOTO(:UEXPINT)                  " DEAL WITH EXCEPT. INTERRUPTS AND
394                                         " BACK TO UNEXTINT
395                                                                  " COORD
396                                                                  " INTERRUPT - 3
397         G = MC[- 1]                     " G = URRI[J] = :USBK
398         A = UHAMR[UG], P                " HARDWARE BLOCKING?
399     N,  JUMP(- 3)
400     U,  S '*' MA[8], Z                  " AM WAITING FOR OTHER INTERRUPTS?
401     Y,  JUMP(- 5)
402         JUMP(10)
403         '761 075 001'                   " S = SECOND FLIPFLOPWORD, Z
404         '761 376 101'                   " N, S '*' SECOND LVIE, Z
405     Y,  GOTO(:UCHOICE)                  " INTERRUPTS EXHAUSTED?
406     U,  S '*' UEXPINTW[1], Z            " NO EXCEPTIONAL INTERRUPTS
407     N,  JUMP( - 1)
408         G = MC[- 1]                     " G = URRI[J] = :USBK
409         A = UHAMR[UG], P                " HARDWARE BLOCKING?
410     N,  JUMP(- 3)
411     U,  S '*' MA[9], Z                  " AM WAITING FOR OTHER INTERRUPTS?
412     Y,  JUMP(- 5)
```

```
413           UHAMR[UG] = A                    " HAM[0] := - 0, AM SELECTABLE
414           S = UHAMR[UG + 1], R             " AM RED?
415           S = 1
416       Y,  URSEL + S                        " RED SELECTABLE AM'S + 1
417           USEL + S                         " SELECTABLE AM'S + 1
418       Y,  MA[1] - S                        " QUEUING REDS - 1
419           MA[2] - S, Z                     " QUEUING AM'S - 1; NO WAITERS?
420       Y,  DO(MA[6])                        " IF NO WAITERS THEN LVIF := FALSE
421           S = UCON1                        " S = 2 ↑ 18
422           G = MA[0]                        " G = ADDRES ARD
423           MG[1] = S                        " LET := LET - 1
424           DO(MA[4])                        " IF := FALSE
425       U,  MG[1] = S, R                     " LET > 0?
426       Y,  DO(MA[3])                        " IF := TRUE
427           GOTO(:UNEXTINT)                  " TRY AGAIN

428 UCHOICE(8)
429           S = USEL, R
430       N,  GOTO(:UTELDOOR)                  " NO SELECTABLE AM'S
431           S = URSEL
432           A = NC[- 1]                      " A = URR![J] = :USBK
433       U,  S = UHAMR[UA], Z                 " AM SELECTABLE?
434       Y,  S '+' - UHAMR[UA + 1], E         " REJECT WHITE AM IF RED
435       N,  JUMP(- 4)                        " SELECTABLE
436           UCURRR! = B                      " POINTER ON SELECTED AM

437                                                                  " COORD
438                                                                  " INTERRUPT - 4
439 UCURRENT(6)
440           A = M[B]                         " A = :USB OF SELECTED AM
441           UHAM = A                         " SET UHAM
442           S = M[220]
443           UKLB = S                         " SET CLOCK
444           S = UHAMR[UA + 8]
445           M[63] = S                        " RESET D
446 URESTORE(7)
447           S = UHAMR[UA + 2]
448           M[17] = S                        " PREPARE LINK
449           B = UHAMR[UA + 7]                " RESET B
450           E = UHAMR[UA + 5]                " RESET E
451           S = UHAMR[UA + 4]                " RESET S
452           A = UHAMR[UA + 3]                " RESET A
453           GOTOR(M[17])                     " GO ON WITH NEW AM

454                                                                  " COORD
455                                                                  " UEXPINT - 0
456           " UEXPINT(DEAL WITH EXCEPTIONAL INTERRUPT)
457 UEXPINT(28)
458           LCS(1), R                        " NO CLOCK INTERRUPT?
459       Y,  JUMP(- 1)                        " EMERGENCY, IMPOSSIBLE SITUATION, INTERRUPT CAUSED BY "NOBODY".
460           A = :URR!                        "                INTERRUPTS
461           M[63] = A
462           A + 1
463           B = :URR![UNUMRM]
464       U,  B - A, Z                         " PRIORITY LIST REARRANGED?
465       Y,  JUMP(7)
466           S = MC[- 1]                      " :USBRMI
467           E = UHAMR[US + 9]                " LOAD RMI
468           GOTO(:UEXPINT2)
469       Y,  M[63] = B
```

enough comment

} reshuffeling of priority list every 2 seconds on account of CPU-time used recently.

```
470       U,  B = A, R
471       Y,  JUMP(- 6)
472           B = MA[- 1]
473           S = MD[0]
474       N,  MA[- 1] = S                    " INTERCHANGE
475       N,  MD[0] = B
476           E = UHAMR[US + 9]              " MAX. LOAD
477           E/UBELSOM
478           UHAMR[US + 9] = E              " STORE NEW LOAD
479       N,  JUMP(- 19)
480           E = 1
481           UBELSOM = E                    " TOTAL LOAD := 1
482           '660 071 047'                  " IE := FALSE
483           S = UCLOCKPERIOD               " SET CLOCK PERIOD
484           M[221] = S
485           GOTO(:UNEXTINT)
486 UEXPINT2(7)
487       U,  A = CGLSLOW[US], R
488       U,  A = CGLSLOW, E
489       N,  JUMP(2)
490           E = UHAMR[9], R
491           GOTO(:UEXPINT[11])
492       U,  A = CGLSLOW[US], R
493           GOTO(:UEXPINT[11])
494                                                  " COORD
495                                                  " SOFTW. POP = 1
496           " SOFTWARE P-OPERATION
497 UPOR(12)                                 " ENTRY UPROOD
498           MC[0] = A                      " SAVE A
499           A = UHAM, R                    " :USB IN A, SET YES CONDITION
500           JUMP(2)
501           MC[0] = A                      " SAVE A, ENTRY UP
502           A = UHAM, Z                    " :USB IN A, SET NO CONDITION
503           UHAMR[UA + 4] = S              " SAVE S
504           S = 1
505       Y,  UHAMR[UA + 1] = S              " MAKE AM RED
506       Y,  URSEL + S                      " INCREASE NUMBER OF RED SELECTABLE AM'S
507       U,  S - MG[0], R                   " VALUE SEMAPHORE LESS THAN 1?
508       Y,  JUMP(4)                        " BLOCKADE
509           MG[0] - S                      " DECREASE VALUE SEMAPHORE
510 UPOP1(4)                                 " ENTRY UPM AND UV
511           S = UHAMR[UA + 4]              " RESTORE S
512           A = MC[- 1]                    " RESTORE A
513           GOTOP(MC[- 1])                 " RETURN
514           UHAMR[UA] = - G                " SOFTWARE BLOCKING
515 UPOP2(5)                                 " ENTRY UPM
516           USEL - S                       " DECREASE USEL
517           MG[2] + S                      " INCREASE NUMBER OF QUEUING AM'S
518       U,  S = UHAMR[UA + 1], R           " AM RED?
519       Y,  MG[1] + S                      " INCREASE QUEUING REDS
520       Y,  URSEL - S                      " DECREASE URSEL
521 UPOP3(3)                                 " ENTRY UV
522           UHAMR[UA + 5] = E              " SAVE E
523           E = MC[- 2]                    " TRANSPORT T AND A
524           GOTO(:UINT1)                   " MERGE WITH INTERRUPT
525                                                  " COORD
526                                                  " HARDWARE POR = 0
527           " HARDWARE P-OPERATION
528 UPOPH(18)
```

P-operation

```
529           MC[0] = A                          " SAVE A
530           A = UHAM                           " :USB IN A
531           UHAMR[UA + 4] = S                  " SAVE S
532           A = MC[0]                          " ADDRES ARO IN A
533           S = UCON1                          " S = 2 ↑ 18
534      U,   MA[1] - S, R                       " LFT > 0?
535      N,   JUMP(6)                            " BLOCKADE
536           MA[1] - S                          " LFT - 1
537           DO(MC[4])                          " LE := FALSE
538      U,   MA[1] - S, R                       " LFT > 0?
539      Y,   DO(MC[3])                          " LE := TRUE
540           A = UHAM
541           GOTO(:UROR1)                       " MERGE WITH UROR, RETURN
542           DO(MC[5])                          " LVIF := TRUE
543           A = UHAM
544           UHAMR[UA] = G                      " HARDWARE BLOCKING
545           S = 1
546           GOTO(:UROR2)                       " MERGE WITH UROR AND INTERRUPT

547                                                                  " COORD
548                                                                  " V-OPERATION - 1
549           " V-OPERATION
550 UVOR(43)                                    " ENTRY UVWIT
551           UCURRR1 = - B, R                   " INDICATION FREE CHOICE; SET CONDITION NO
552           JUMP(1)
553      U,   S = 1, R                           " ENTRY UV; SET CONDITION YES
554           MC[0] = A                          " SAVE A
555           A = UHAM
556           UHAMR[UA + 4] = S                  " SAVE S
557           S = 1
558      N,   UHAMR[UA + 1] - S                  " MAKE AM WHITE
559      N,   URSEL - S                          " DECREASE NUMBER OF RED SELECTABLE AM'S
560           MC[0] + S, R                       " INCREASE VALUE SEMAPHORE; POSITIVE?
561      Y,   S = MC[2], R                       " QUEUING AM'S PRESENT?
562      Y,   JUMP(3)
563           S = UCURRR1, R                     " NORMAL V-OPERATION?
564      Y,   GOTO(:UROR1)                       " MERGE WITH UROR; RETURN
565           GOTO(:UROR3)                       " MERGE WITH UROR AND INTERRUPT
566           UHAMR[UA + 5] = E                  " SAVE E
567           E = MC[- 2]
568           UHAMR[UA + 2] = E                  " SAVE T AND A
569           UHAMR[UA + 7] = B                  " SAVE B
570           B = :URR1[UNAM]                    " B JUST ABOVE PRIORITY LIST
571           S = UHAMR[UA + 6]                  " ADDRES SEMAPHORE IN S
572           G = MC[- 1]                        " G = :USB
573      U,   S + UHAMR[UG], Z                   " AM WAITING ON THIS SEMAPHORE?
574      N,   JUMP(- 3)
575      U,   A = MS[1], E                       " REJECT WHITE AM IF RED AM
576      U,   A = UHAMR[UG + 1], E               " IS PRESENT
577      N,   JUMP(- 6)
578           UHAMR[UG] + S                      " MAKE CHOSEN AM SELECTABLE
579           A = 1, E                           " THIS AM RED?
580      Y,   URSEL + A                          " INCREASE URSEL
581      Y,   MS[1] - A                          " DECREASE NUMBER OF RED QUEUING AM'S
582           MS[2] - A                          " DECREASE NUMBER OF QUEUING AM'S
583           USEL + A                           " INCREASE USEL
584           MS[0] - A                          " DECREASE VALUE SEMAPHORE

585                                                                  " COORD
586                                                                  " V-OPERATION - 2
```

```
587           "V-OPERATION
588           A = UHAM                          " A = :USB OF CURRENT AM
589           $ = UCURRI, P                     " NORMAL V-OPERATION?
590      N,   GOTO(:UINT2)                      " MERGE WITH INTERRUPT, FREE CHOICE
591           $ = UHAMR[UA + 1], P              " CURRENT AM RED?
592      N,   B - UCURRI, E                     " CURRENT WHITE AM HIGHEST PRIORITY?
593      Y,   $ '*' - UHAMR[UG + 1], P          " CURRENT AM RED OR CHOSEN AM WHITE?
594      Y,   GOTO(:URESTORE)                   " STICK TO CURRENT AM
595           UCURRI + B                        " SWITCH TO CHOSEN AM
596           GOTO(:UINT2)                      " MERGE WITH INTERRUPT, SWITCH
597                                             " TO CHOSEN AM

598                                                                  " DRUM
599                                                                  " SRQ - 0
600           " SRQ(SEGMENT REQUEST)
601 SRQ(57)
602           SUBCD(:URETDEN)                   " SAVE REGISTERS
603           S = MS[0], P                      " SEGMENT IN CORE?
604           A = DRT[2]                        " 2 ↑ 20
605      Y,   MS[3] + A                         " INCREASE HOLYNESS COUNTER
606      Y,   GOTO(:UHERSTELTERUG)              " RESTORE REGISTERS AND BACK
607           G = USCWRITE                      " SEGM. CONTR. WRITE POINTER
608           MG[3] = - B                       " SET FIRST HALF MESSAGE SYMBOLIC
609           MG[9] = - B                       " SET SECOND HALF MESSAGE SYMBOLIC
610           LCS(1), Z                         " SV EMPTY?
611      N,   BUS(1), P                         " SEGMENT ON DRUM?
612      N,   $ = - $                           " :CTE[0] COMING SEGMENT
613      N,   MS[3] + A                         " INCREASE HOLYNESS COUNTER
614      N,   GOTO(:SRQ3)                       " SEGMENT COMING
615           A = UVICTIM                       " :CTE[0] VICTIM
616           MA[1] = S, P
617      Y,   SUBCD(:UDRUMAFT)                  " MAKE STARTING COMMAND V(AFT)
618           A = UEREE, P                      " :CTE[0] FREE SEGMENT OR - 0
619      Y,   S = MA[0]                         " :CTE[0] NEXT FREE SEGMENT OR - 0
620      Y,   UEREE = S                         " CHAIN OFF
621      Y,   GOTO(:SRQ2)                       " FREE PAGE BECOMES VICTIM
622           S = !UCT[UCTLEN]                  " JUST ABOVE CTE TABLE
623           B = :MS[0]                        " B = S; ADDRESS MINIMUM
624           $ - 4                             " TRY NEXT CTE ENTRY
625      U,   A - MS[3], P                      " NEW MINIMUM?
626      N,   JUMP(- 3)
627           A = MS[3], P                      " CTE TABLE SCANNED? UCT[- 1] = 2 ↑ 26 - 1
628      N,   JUMP(- 6)
629           S = N[B + 1], P                   " DRB COPY OR - 0

630                                                                  " DRUM
631                                                                  " SRQ - 1
632           USCWORK = B                       " SAVE :CTE NEW VICTIM
633      Y,   GOTO(:SRQ1)                       " COPY BECOMES VICTIM
634           S = UDRUMPOSITION                 " DRB LAST HARDWARE COMMAND
635           S + UDRUMMARGE[512]               " SEE EWD 86, EWD113, ANM51
636           S '*' 1023                        " S - IDEAAL
637           A = 0
638           DIVAS(26)                         " S := I, A := J
639           S + :DURT[0]                      " :DURT[I]
640           B = A                             " B := J
641           A = MS[0]                         " DURT[I]
642           LUA(B), Z                         " ALL SUITABLE DRUMPAGES OCCUPIED?
643      N,   RUA(B)
644      N,   JUMP(5)
```

Segment controller.

```
645          S + 1                          " TRY NEXT WORD IN DURT
646      U,  S - :DURT[40], Z                " END OF DRUMPAGE TABLE
647      Y,  S = :DURT[0]
648          A = MS[0], Z                    " ALL SUITABLE PAGES OCCUPIED
649      Y,  JUMR(- 5)
650          NORA                           " B := J
651          A = DRT[15]                    " 2 ↑ 25
652          RUA(B)                         " 2 ↑ (25 - J)
653          MS[0] + A                      " DRUMPAGE := OCCUPIED
654          S - :DURT[- 1]                 " I + 1
655          MUL$(26)                       " 26(I + 1)
656          B + :MS[- 26]                  " 26I + J (+ 0 PREFERENCE)
657          S = B                          " S := S := 26I + J
658          LUS(10)                        " 1024 * S
659          S + B                          " 1025 * S
660          S '*' DRT[18]                  " 2 ↑ 19 - 1, DRB IN S, NO CONDITION
661 SRQ1(7)
662          B = - MG[9]                    " RESTORE B
663          A = USCWORK                    " :CTE[0] NEW VICTIM
664      N,  SUBCD(:UDRUMAET)               " MAKE STARTING COMMAND, V(AET)
665      Y,  MA[1] - S                      " CTE[1] := - 0 (ORIGINAL)
666          S '+' - UCON3                  " SV := DRB + 2 ↑ 26
667          G = MA[0]                      " :SV COUPLED TO VICTIM

668                                                                 " DRUM
669                                                                 " SRQ - 2
670          MG[0] = S                      " SV := DRB + 2 ↑ 26
671 SRQ2(11)
672          S = 1                          " LOWEST INTEREST NUMBER + 2 ↑ 26
673          MA[3] = S                      " CTE[3] := 1
674          G = M[B - 3]                   " ADDRESS SV UPON CALL
675      Y,  S = MG[0], Z                   " NO ORIGINAL CHOSEN; SV EMPTY?
676      Y,  S = A                          " :CTE[0] CHOSEN COPY OR FREE PAGE
677      N,  S = UVICTIM                    " :CTE[0] OLD VICTIM
678          MS[0] = G                      " CTE[0] := :SV
679      Y,  MG[0] = S                      " SV := :CTE[0]; OCCUPIED
680      Y,  GOTO(:UHERSTELTERUG)           " RESTORE REGISTERS AND RETURN
681          UVICTIM = A                    " NEW VICTIM
682          MG[0] = - S                    " SV := - :CTE[0]; COMING
683 SRQ3(15)
684          A = USCWRITE                   " SEGMENT CONTROLE WRITE POINTER
685          F = :USCAMSEM                  " SEGMENT CONTROLE AM SEMAPHORE
686          MA[- 2] = G                    " FILL LABEL [- 2]
687          MA[4] = S                      " FILL LABEL[4] WITH :CTE[0]
688          A + 12                         " CYCLIC INCREASE WRITE POINTER
689      U,  A - :UDRUMMAG[UDML + 2], Z
690      Y,  A = :UDRUMMAG[2]
691          USCWRITE = A
692          SUBCD(:UIMMORTAL)
693          F = :USCS
694          UV                             " V(SCS)
695          F = :USCAMSEM
696          UP                             " P(SCAMSEM)
697          SUBCD(:UMORTAL)                " IN THE CASE OF APPARENT DEATH
698          GOTO(:UHERSTELTERUG)           "       A BREAK MAY OCCUR

699          " UHERSTELTERUG
700 UHERSTELTERUG(2)
701          SUBCD(:UHERSTEL)
702          GOTOR(MC[- 1])                 " RETURN
```

```
703
704                                                                  " DRUM
                                                                     " UDRUMART = 0
705         " UDRUMART
706 UDRUMART(24)
707         UDRUMPOSITION = $                " STORE POSITION DRUM
708     N,  F + 6                            " F POINTS TO HARDWARE LABEL[0]
709         $ '+' = UCON3                    " DRB + 2 ↑ 26
710         MG[1] = $                        " FILL LABEL[1] RESP. [7]
711         $ = MA[2]
712     N,  $ + UCON1                        " ADD 2 ↑ 18 (DIRECTION)
713         MG[2] = $                        " FILL LABEL[2] RESP. [8]
714         $ = 512
715         MG[3] = $                        " FILL LABEL[3], [9]
716         $ = UDRUMAAN                     " LABEL LAST CHAINED COMMAND
717         MS[0] = G                        " CHAIN ON
718         UDRUMAAN = G                     " NEW HARDWARE LABEL
719         $ = UCON1                        " 2 ↑ 18
720         PLUS$(UARDR[2])                  " AFT := AFT + 1
721         BUS(19), Z                       " AFT = 1?
722     Y,  $ = UARDR[0]                     " AR0
723     Y,  $ '+' UCON2                      " ISOLATE ADDRESS PART
724     Y,  $ - G                            " OLD ADDRESS PART = NEW ADDRESS PART
725     Y,  UARDR[0] = $                     " NEW ADDRESS PART IN AR0
726     Y,  DO(UIETDR[7])                    " AE := TRUE
727         $ = 1
728         UDRUMCOM + $                     " INC(NUMBER OF DRUM COMMANDS)
729         $ = UDRUMPOSITION                " RESTORE $
730         GOTOR(MC[- 1])                   " RETURN
```

V-operation to get drum going (physical)

```
731
732                                                                  " DRUM
                                                                     " UBIETDRUM = 0
733 UBIETDRUM(45)
734         $ = - M4[3], R                   " PORTION SYMBOLIC ?
735     Y,  JUMR(37)
736         F = :M2[0]
737         UBH                              " R(IET)
738         $ = M4[- 1], R                   " OK?
739         A = 1
740     Y,  UDRUMCOM - A                     " DEC(NUMBER OF DRUMCOMMANDS)
741     Y,  JUMR(23)
742         UDRUMERROR - A, R                " NUMBER OF ALLOWED ERRORS STILL POSITIV?
743     N,  GOTO(:ONZINI)                    " DISASTER
744         $ = - 0                          " COUNT NUMBER OF
745         $ + A                            " SYMBOLIC PROCESSED TRANSPORTS
746         UDRUMCOM - A, Z                  " CHAIN DEAD?
747     N,  SUBCD(:UPORH)                    " R(IET)
748     N,  JUMR(- 4)
749         A = '000 004 000'[11]            " START OPDRACHT TANDENPOETSER
750         M[216] = A                       " FILL AR0 TANDENPOETSER
751         '760 070 000'[38]                " AE := TRUE
752         '660 075 000'                    " IE WORD IN A
753         LCA(2), R                        " IE TANDENPOETSER NOG FALSE ?
754     Y,  JUMR(- 3)
755         '660 071 000'[38]                " IE TANDENPOETSER := FALSE
756         UDRUMCOM = $                     " RESET NUMBER OF DRUMCOMMANDS
757         A = :UWOKDRUM[1]                 " LABEL WOK COMMAND
758         M1[0] = A                        " FILL AR0
759         $ + 1                            " NUMBER OF COMMANDS, WOK INCLUDED
760         LUS(18)
```

P-operation on hardware semaphore of drum. (waiting for drum to complete V on it)

```
761           N1[2] = S                          " SET ART
762           DO(M2[7])                          " AE := TRUE
763           URH                                " R(IET) HOK COMMAND
764           JUMR(-28)                          " TRY AGAIN
765           S = - UDRUMERROR
766           S + UMAXERROR, P                   " S := NUMBER OF TIMES IT WAS WRONG
767       Y,  UDRUMERROR + S                     " RESET NUMBER OF ALLOWED ERRORS
768       Y,  UTOTALERROR + A                    " NUMBER OF NOT IMMEDIATELY
769                                              " CORRECT TRANSPORTS

770                                                                          " DRUM
771                                                                          " URIETDRUM - 1
772           S - 1, Z
773       Y,  UEIRSTERROR + A                    " ERRORS REQUIRING ONE REPETITION
774           S - 1, Z
775       Y,  USECONDERROR + A                   " ERRORS REQUIRING TWO REPETITIONS
776           S = USCREAD                        " CYCLIC INCREASE READ POINTER
777           S + 6
778       U,  S - :UDRUMMAG[UDML + 2], Z
779       Y,  S = :UDRUMMAG[2]
780           USCREAD = S
781           GOTOR(MC[- 1])

782                                                                          " COORDINATOR
783                                                                          " UZEROHOUR
784           " UZEROHOUR
785   UZEROHOUR(12)
786           SUBCD(:UREDDEN)                    " SAVE REGISTERS
787           S = :UCT                           " BASIC ADDRESS CTE TABLE
788           A = MS[3], R                       " PAGE HOLY?
789       N,  RUA(2)
790       N,  A '*' - DRT[19]                    " D19 := D18 := 0
791       N,  MS[3] = A
792           S + 4
793       U,  S - :UCT[UCTLEN], Z                " READY?
794       N,  JUMR(- 7)
795           S = - DRT[19]                      " -(2 ↑ 18 + 2 ↑ 19)
796           URAGECLOCK = S                     " SET PAGE CLOCK
797           GOTO(:UHERSTELTERUG)

798                                                                          " COORDINATOR
799                                                                          " URROEANELINK
800                                                                          " URROEANATION
801           " URROEANELINK
802   URROEANELINK(13)
803           S = N[8 - 2]                       " TAKE MACHINE LINK OBJECT TEXT
804           RUS(7)                             " ENTRY POINT DIFFERENT SYSTEMENTRIES
805           S '*' 2044
806           S + UCRC                           " :CTE IN S
807           A = DRT[2]                         " ENTRY POINT URROEANATION
808           MS[3] = A, R                       " DEC(HOLYNESS COUNTER), STILL HOLY?
809       Y,  GOTOR(MC[- 1])
810           A = 1
811           RLUSA(URAGECLOCK), Z               " INC(PAGE CLOCK), OVERFLOW?
812       N,  MS[3] = A                          " SET INTEREST
813       N,  GOTOR(MC[- 1])
814           SUBCD(:UZEROHOUR)                  " SCALE INTEREST NUMBERS
815           JUMR(- 6)

816           " URROEANATION
```

```
817 URROEANATION(4)
818         SUBCD(:UREDDEN)                 " SAVE REGISTERS
819         $ = MS[0]                       " :CTE IN $
820         SUBCD(:URROEANELINK[4])
821         GOTO(:UMERSTELTERUG)

822                                                                 " COORDINATOR
823                                                                 " UKILLSEG
824                                                                 " MECR
825         " UKILLSEG(KILL SEGMENT)
826 UKILLSEG(6)
827         A = MS[0], R                    " SEGMENT ON CORE
828         MS[0] = A                       " SV := - 0
829     N,  $ = A                           " DRB + 2 ↑ 26 IN $
830     Y,  SUBCD(:MECR)                    " MAKE FREE CORE PAGE
831     N,  SUBCD(:MEDR)                    " MAKE FREE DRUM PAGE
832         GOTOR(MC[- 1])                  " RETURN

833         " MECR(MAKE FREE CORE PAGE)
834 MECR(8)
835         $ = UFREE                       " :CTE FIRST FREE PAGE OR - 0
836         UFREE = A
837         MA[0] = $                       " CHAIN ON
838         $ = - DRT[0]                    " -(2 ↑ 20 - 1)
839         MA[3] = $                       " FILL CTE[3]
840         $ = MA[1], R                    " IF COPY DRB ELSE - 0
841         MA[1] = $, E                    " CTE[1] := - 0, ORIGINAL
842         GOTO(MC[- 1])                   " RETURN WITH YES CONDITION IF IT
843                                         " WAS AN ORIGINAL ELSE NO CONDITION

844                                                                 " SE 39, 41, 43 - 1
845         " MAKE PHYSICAL ADDRESS
846 R39E1(30)
847         $ = A                           " INVARIANT ADDRESS
848         $ '*' 511                       " LINE NUMBER
849         RUA(9), R                       " ADDRESS $V
850     N,  A '*' DRT[1]                    " PRECAUTION MEMORY > 32K
851         G = MA[0], R                    " SEGMENT ON CORE?
852     N,  JUMP(10)
853         A = 1
854         PLUSA(UPAGECLOCK), Z            " INC(PAGECLOCK), OVERFLOW?
855     N,  A - MG[3], R                    " PAGE PROFANE?
856     Y,  MG[3] + A, B                    " SET INTEREST, OVERFLOW ^ PROFANE?
857     N,  $ + MG[2], E                    " PHYSICAL ADDRESS, OVERFLOW?
858     N,  DO(MD[- 1])                     " RESTORE A
859     N,  GOTOR(MC[- 1])                  " RETURN
860         $ '*' 511                       " LINE NUMBER IN $
861         SUBCD(:UZEROHOUR)               " SCALE INTEREST NUMBERS
862         JUMP(- 10)
863         F + 1, E                        " SEGMENT NOT EMPTY?
864     N,  $ = :UCLEAR                     " ADDRESS CLEARING CONSTANT
865     N,  GL23 = B                        " READ FROM EMPTY $V := TRUE
866     N,  JUMP(- 9)
867         MC[0] = $                       " SAVE $
868         SUBCD(:UIMMORTAL)
869         $ = A                           " ADDRESS $V
870         SUBCD(:$RQ)                     " A BREAK WILL OCCUR
871         G = M$[0]                       " :CTE
872         $ = DRT[2]
873         MG[3] - $                       " DEC(HOLYNESS COUNTER)
```

reading from segment

```
874         SUBCD(:UMORTAL)                 " A BREAK MAY OCCUR
875         $ = MC[- 1]                     " RESTORE $
876         JUMP(- 26)                      " TRY AGAIN

877                                                                 " COORDINATOR
878                                                                 " MFDR
879         " MAKE FREE DRUM PAGE
880 MFDR(10)
881         $ '*' 1023                      " S := $
882         A = 0
883         DIV4$(26)                       " $ := I; A := J
884         $ + :DURT[0]                    " $ := :DURT[I]
885         A + DRT[14]                     " A := BUA(J)
886         UWS[3] = A
887         A = DRT[15]                     " A := 2 ↑ 25
888         DO(UWS[3])                      " A := 2 ↑ (25 - J)
889         MS[0] = A                       " SET TO ZERO
890         GOTOR(MC[- 1])                  " RETURN

891                                                                 " COORDINATOR
892                                                                 " UREDDEN
893                                                                 " UWERSTEL
894         " UREDDEN ( SAVE REGISTERS ) on stack
895 UREDDEN(4)
896         MC[0]=A
897         MC[0]=$
898         MC[0]=E
899         GOTOR(M[B-5])

900         " UWERSTEL ( RESTORE REGISTERS ) from stack
901 UWERSTEL(5)
902         E=M[B-3]
903         $=M[B-4]
904         A=M[B-5]
905         B=6
906         GOTOR(M[B+5])

907                                                                 " SE 50, 52 - 0
908         " SE 50 ( ASSIGN TO INTEGER ON SEGMENT )
909 RSE50(2)
910         SUBCD(:R50E1)                   " PHYSICAL ADDRESS IN $
911         GOTO(:R49E1)                    " MERGE WITH SE 49

912         " SE 52 ( ASSIGN TO REAL ON SEGMENT )
913 RSE52(3)
914         SUBCD(:R50E1)                   " PHYSICAL ADDRESS IN $
915         MS[0]=E                         " ASSIGN VALUE
916         GOTOR(MC[0])

917                                                                 " SE 49 - 0
918         " SE 49 ( ASSIGN TO STACKED INTEGER )
919 RSE49(1)
920         $=MC[-2]                        " TAKE ADDRESS IN $
921 R49E1(9)
922     U,  $=M[57], Z                      " HEAD E ZERO ?
923         MS[0]=G                         " ASSIGN VALUE
924     Y,  GOTOR(MC[0])                    " RETURN
925         E+DRT[11]                       " E + 3*2↑38
926         E-DRT[11]                       " E - 3*2↑38
927     U,  $=M[57], Z
```

```
928       Y,  JUMP(-6)
929           B + 1
930           GOTO(:RSE88[4])
931                                                                  " SE 59, 39, 61, 41
932           " SE 59 ( FORMAL RIGHT SUBSCRIPTED INTEGER )
933 RSE59(1)
934           SUBCD(:R58E1)
935           " SE 39 ( RIGHT SUBSCRIPTED INTEGER )
936 RSE39(1)
937       N,  SUBCD(:R39E1)                    " PHYSICAL ADDRESS IN S
938 R39E2(2)
939           G=MS[0]                          " VALUE IN G
940           GOTOR(MC[-1])
941           " SE 61 ( FORMAL RIGHT SUBSCRIPTED REAL )
942 RSE61(1)
943           SUBCD(:R58E1)
944           " SE 41 ( RIGHT SUBSCRIPTED REAL )
945 RSE41(3)
946       N,  SUBCD(:R39E1)                    " PHYSICAL ADDRESS IN S
947           F=MS[0]                          " VALUE IN F
948           GOTOR(MC[-1])
949 D(1)
950           :UDISRPM1[2]
951 UHAM(1)
952           :USBPM1
953 UCURPRI(1)
954           :URR1
955 UBELSON(2)
956           + 0
957           + 1
958 UKLB(1)
959           + 0
960 UEXPINTV(2)
961           '377 016 074'
962           '037 777 777'
963 UCON1(1)
964           '001 000 000'
965 UCON2(1)
966           '000 777 777'
967 URSEL(1)
968           1
969 USEL(1)
970           2
971 URR1(UNAM)
972           :USBPM1
973           :USBPM2
974           :USBPM3
975           :USBPM4
976           :USBPM5
977           :USBRLA
978           :USBTRA
979           :USBTRB
980           :USBTRC
981           :USBLRA
982           :USBTRA
983           :USBTRB
```

→ lowest priority

priority list

the addresses in this list point to a base reference-point in the stack bottoms of these AMs (Abstract Machines). The AMs are identified by these addresses.

e.g. for PM3 static address :USBPM3 is the same as dynamic address :M0

```
 984          :USBTRC
 985          :USBDR
 986          :USBMI
 987 USCWRITE(1)
 988          :UDRUMMAG[2]
 989 UIFTDR(10)
 990          :UARDR
 991          - 0
 992          - 0
 993          '760 071 000'[11]
 994          '660 071 000'[11]
 995          '760 072 000'[11]
 996          '660 072 000'[11]
 997          '760 070 000'[11]
 998          '000 000 100'
 999          0
1000 USCS(3)
1001          - 0
1002          - 0
1003          - 0
1004 UVICTIM(1)
1005          :UCT
1006 UFREE(1)
1007          :UCT[4]
1008 UDRUMPOSITION(1)
1009          + 0
1010 UDRUMAAN(1)
1011          :UDRUMMAG[2]
1012 UDRUMCOM(1)
1013          - 0
1014 UDRUMERROR(1)
1015          UMAXERROR
1016 UTOTALERROR(1)
1017          - 0
1018 UFIRSTERROR(1)
1019          - 0
1020 USECONDERROR(1)
1021          - 0
1022 UCON3(1)
1023          '377 777 777'
1024 URAGECLOCK(1)
1025          U1
1026 UGREER25(1)
1027          JUMP(- 1)
1028 UGREER26(1)
1029          JUMP(- 1)
1030 UGREER27(1)
1031          JUMP(- 1)
1032 UGREER28(1)
1033          JUMP(- 1)
1034 DPT(20)
1035          + 1 048 575
1036          + 262 143
1037          + 1 048 576
1038          '002 000 000'
1039          - 58 722 815
1040          A = MD[0]
1041          + 8 388 608
1042          E = M[0]
```

highest

digit pattern table

several useful constants
they should better have been placed with the text where they are used

```
1043            + 16 777 216
1044            DO(MC[- 1])
1045            G = M[0]
1046            + 12 288
1047            + G
1048            DO(M[0])
1049            RUA(0)
1050            + 33 554 432
1051            + 261 632
1052            LCS(0)
1053            + 524 287
1054            + 786 432

1055 DUPT(40)
1056            '400 000 000'
1057            '400 000 000'
1058            '400 000 000'
1059            '400 000 000'
1060            '400 000 000'
1061            '400 000 000'
1062            '400 000 000'
1063            '400 000 000'
1064            '400 000 000'
1065            '400 000 000'
1066            '400 000 000'
1067            '400 000 000'
1068            '400 000 000'
1069            '400 000 000'
1070            '400 000 000'
1071            '400 000 000'
1072            '400 000 000'
1073            '400 000 000'
1074            '400 000 000'
1075            '400 000 000'
1076            '400 000 000'
1077            '400 000 000'
1078            '400 000 000'
1079            '400 000 000'
1080            '400 000 000'
1081            '400 000 000'
1082            '400 000 000'
1083            '400 000 000'
1084            '400 000 000'
1085            '400 000 000'
1086            '400 000 000'
1087            '400 000 000'
1088            '400 000 000'
1089            '400 000 000'
1090            '400 000 000'
1091            '400 000 000'
1092            '400 000 000'
1093            '400 000 000'
1094            '400 000 000'
1095            '400 377 777'

1096 UC$TOP(1)
1097            '377 777 777'
1098 UCT(UCTLEN)
1099            - 0
1100            - 0
```

drum page table

one bit for each halftrack denotes free/occupied drum page.

" VICTIM

core table 4 entries / core page

```
1101        UCR
1102        1
1103        :UCT[8]
1104        - 0
1105        UCR[512]
1106        UI
1107        :UCT[12]
1108        - 0
1109        UCR[1024]
1110        UI
1111        :UCT[16]
1112        - 0
1113        UCR[1536]
1114        UI
1115        :UCT[20]
1116        - 0
1117        UCR[2048]
1118        UI
1119        :UCT[24]
1120        - 0
1121        UCR[2560]
1122        UI
1123        :UCT[28]
1124        - 0
1125        UCR[3072]
1126        UI
1127        :UCT[32]
1128        - 0
1129        UCR[3584]
1130        UI
1131        :UCT[36]
1132        - 0
1133        UCR[4096]
1134        UI
1135        :UCT[40]
1136        - 0
1137        UCR[4608]
1138        UI
1139        :UCT[44]
1140        - 0
1141        UCR[5120]
1142        UI
1143        :UCT[48]
1144        - 0
1145        UCR[5632]
1146        UI
1147        :UCT[52]
1148        - 0
1149        UCR[6144]
1150        UI
1151        :UCT[56]
1152        - 0
1153        UCR[6656]
1154        UI
1155        :UCT[60]
1156        - 0
1157        UCR[7168]
1158        UI
1159        :UCT[64]
1160        - 0
```

"FIRST FREE PAGE"

free pages are chained

→ to contain address of SV if occupied

→ base-address of corresponding core page

→ interest number & holyness counter

```
1161          UCR[7680]
1162          UI
1163          :UCT[68]
1164          - 0
1165          UCR[8192]
1166          UI
1167          :UCT[72]
1168          - 0
1169          UCR[8704]
1170          UI
1171          :UCT[76]
1172          - 0
1173          UCR[9216]
1174          UI
1175          :UCT[80]
1176          - 0
1177          UCR[9728]
1178          UI
1179          :UCT[84]
1180          - 0
1181          UCR[10240]
1182          UI
1183          :UCT[88]
1184          - 0
1185          UCR[10752]
1186          UI
1187          :UCT[92]
1188          - 0
1189          UCR[11264]
1190          UI
1191          :UCT[96]
1192          - 0
1193          UCR[11776]
1194          UI
1195          :UCT[100]
1196          - 0
1197          UCR[12288]
1198          UI
1199          :UCT[104]
1200          - 0
1201          UCR[12800]
1202          UI
1203          :UCT[108]
1204          - 0
1205          UCR[13312]
1206          UI
1207          :UCT[112]
1208          - 0
1209          UCR[13824]
1210          UI
1211          :UCT[116]
1212          - 0
1213          UCR[14336]
1214          UI
1215          :UCT[120]
1216          - 0
1217          UCR[14848]
1218          UI
1219          :UCT[124]
1220          - 0
```

```
1221        UCR[15360]
1222        UI
1223        :UCT[128]
1224        - 0
1225        UCR[15872]
1226        UI
1227        :UCT[132]
1228        - 0
1229        UCR[16384]
1230        UI
1231        :UCT[136]
1232        - 0
1233        UCR[16896]
1234        UI
1235        :UCT[140]
1236        - 0
1237        UCR[17408]
1238        UI
1239        :UCT[144]
1240        - 0
1241        UCR[17920]
1242        UI
1243        :UCT[164]
1244        - 0
1245        UCR[18432]
1246        UI
1247        :USVINRRO[3]
1248        - 0
1249        UCR[18944]
1250        + 1
1251        :USVINRRO[2]
1252        - 0
1253        UCR[19456]
1254        + 1
1255        :USVINRRO[1]
1256        - 0
1257        UCR[19968]
1258        + 1
1259        :USVINRRO
1260        - 0
1261        UCR[20480]
1262        + 1
1263        :UCT[168]
1264        - 0
1265        '100 000'
1266        UI
1267        :UCT[172]
1268        - 0
1269        '101 000'
1270        UI
1271        :UCT[176]
1272        - 0
1273        '102 000'
1274        UI
1275        :UCT[180]
1276        - 0
1277        '103 000'
1278        UI
1279        :UCT[184]
1280        - 0
```

```
1281          '104 000'
1282          UI
1283          :UCT[188]
1284          - 0
1285          '105 000'
1286          UI
1287          :UCT[192]
1288          - 0
1289          '106 000'
1290          UI
1291          :UCT[196]
1292          - 0
1293          '107 000'
1294          UI
1295          :UCT[200]
1296          - 0
1297          '110 000'
1298          UI
1299          :UCT[204]
1300          - 0
1301          '111 000'
1302          UI
1303          :UCT[208]
1304          - 0
1305          '112 000'
1306          UI
1307          :UCT[212]
1308          - 0
1309          '113 000'
1310          UI
1311          :UCT[216]
1312          - 0
1313          '114 000'
1314          UI
1315          :UCT[220]
1316          - 0
1317          '115 000'
1318          UI
1319          :UCT[224]
1320          - 0
1321          '116 000'
1322          UI
1323          :UCT[228]
1324          - 0
1325          '117 000'
1326          UI
1327          :UCT[232]
1328          - 0
1329          '120 000'
1330          UI
1331          :UCT[236]
1332          - 0
1333          '121 000'
1334          UI
1335          :UCT[240]
1336          - 0
1337          '122 000'
1338          UI
1339          :UCT[244]
1340          - 0
```

C1063.1
Part 2

```
1341            '123 000'
1342            UI
1343            :UCT[248]
1344            - 0
1345            '124 000'
1346            UI
1347            :UCT[252]
1348            - 0
1349            '125 000'
1350            UI
1351            :UCT[256]
1352            - 0
1353            '126 000'
1354            UI
1355            :UCT[260]
1356            - 0
1357            '127 000'
1358            UI
1359            :UCT[264]
1360            - 0
1361            '130 000'
1362            UI
1363            :UCT[268]
1364            - 0
1365            '131 000'
1366            UI
1367            :UCT[272]
1368            - 0
1369            '132 000'
1370            UI
1371            :UCT[276]
1372            - 0
1373            '133 000'
1374            UI
1375            :UCT[280]
1376            - 0
1377            '134 000'
1378            UI
1379            :UCT[284]
1380            - 0
1381            '135 000'
1382            UI
1383            :UCT[288]
1384            - 0
1385            '136 000'
1386            UI
1387            - 0
1388            - 0
1389            '137 000'
1390            UI

1391 UTELDOOR(12)
1392            $ = MT[4]
1393            M[24] = $
1394            $ = - M[220]
1395            GOTOR(MT[0])
1396            '2 777 777'
1397            '563 000 000'[:UTELDOOR + 6]
1398            $ + M[220]
1399            M[219] + $
```

```
1400          S = MT[2]
1401          M[24] = S
1402          GOTO(:UNEXTINT)
1403          '563 000 000'[:UINT]

1404 USRILTIME(1)
1405          + 0
```

```
0          " MESSAGE INTERPRETER                                          " TEXTMI = 0
1  CMAKEIETZERO(6)
2          S = 2
3          UARCK[1] = S                    " IETTB := 0
4          DO(UIETCK[4])                   " IETB := EALSE
5       U, S - UARCK[1], Z                 " IE IETTB = 0 IHEN ....
6       N, JUMP( - 4)
7          GOTOR(MC[ - 1])
8  CNOTCASE(7)
9       U, A - 27, Z                       " IS AANSLAG = CY?
10      Y, A = 32
11      Y, JUMP(2)
12      U, A - 31, Z                       " IS AANSLAG = LE?
13      Y, A = 0
14      Y, CASETB = A
15         GOTO(MC[ - 1])
16                                                                        " TEXTMI = 1
17 CAESL(7)
18         S = 1
19         UARCK[0] = - S                  " AR0 := - 1
20         DO(UIETCK[7])                   " AETB := IRUE
21         A = UARCK[3]                    " A := AR3
22         SUBC(:CNOTCASE)                 " IE A = CASE IHEN BOOK
23         UARCK[3] = - S                  " AR3 := - 1
24         GOTOR(MC[ - 1])

25                                                                        " TEXTMI = 2
26 CNKC(29) collect next keyboard character
27         SUBCD(:UREDDEN)
28         E = :UIETCK
29         URH                             " R(IETTB)
30         A = UARCK[1]                    " A := AR1
31         A '*' - UCON2, Z                " IETTB = 0 ?
32      Y, JUMP(6)
33         SUBC(:CMAKEIETZERO)
34         A = UARCK[0], P                 " AR0 ≥ + 0 ?
35      N, JUMP( - 8)
36         SUBC(:CNOTCASE)
37         SUBC(:CAESL)
38         JUMP( - 11)
39         A = UARCK[0], P
40      N, JUMP( - 13)
41      U, A - 31, P                       " A > 31, D.W.Z, CHARACTER INVALID ?
42      N, SUBC(:CNOTCASE)
43      Y, JUMP( - 7)
44      U, A - 8, Z                        " VERWERK CHARACTER
45      Y, A = 2                           " MAAK VAN NR EEN TW
46      U, A - 2, Z                        " TW ?
47      Y, CTWNR = A                       " MESSAGE RECALLED BY OPERATOR := IRUE
48      Y, JUMP(2)
49      U, A - 4, Z                        " SPATIE ?
50      N, A + CASETB                      " MAAK SEMI HARDWARE REPRESENTATIE
51         CGAPR = B                       " GA PRINTEN := IRUE
52         SUBC(:CPRINT)                   " TERUGPRINTEN CHARACTER
53         M[B - 4] = A                    " VERVANG TE REDDEN A DOOR NIEUWE WAARDE
54         SUBC(:CAESL)
55         GOTO(:UHERSTELTERUG)            " NU MET NIEUWE WAARDE VAN A

56                                                                                " TEXTMI = 3
```

collect next non recalled character

```
57 CNNRC(15)
58          A = CLNRC, Z                    " LNRC = EMPTY ?
59      N,  CLNRC - A                       " LNRC := 0, D,W,Z, EMPTY
60      N,  GOTOR(MC! - 1!)
61          $ = CRC                         " PREVIOUS CHARACTER
62      U,  $ - 50, Z                       " RC = $$ ?
63      Y,  JUMR(6)
64      U,  $ - 2, Z                        " RC = TWNR ?
65      Y,  JUMR(4)
66          SUBC(:CNKC)                     " NEXT KEYBOARDCHARACTER IN A
67          CRC = A, Z                      " A = HT (HERROEPINGSTEKEN) ?
68      N,  $ = $, Z                        " WAS RC EMPTY ?
69      Y,  JUMR( - 9)
70      Y,  CRC - $                         " RC := EMPTY
71          A = $                           " NNRC := "RC"
72          GOTOR(MC! - 1!)

73                                                                  " TEXTM! = 4
```

read dec. number from keyboard

```
74 CREADDEC(14)                             " VERSIE 2, DD, 13 - 1 - 67
75          SUBC(:CDNONS)
76          E = 0
77          $ = A                           " COPY SYMBOL # BLANK SPACE IN $
78          SUBC(:CONVCKINT)
79      U,  A - 9, R                        " ¬ DECIMAL DIGIT ?
80      Y,  CLNRC = $                       " BACK SYMBOL FOR NEXT CALL OF CNNRC
81      Y,  $ = G                           " DELIVER VALUE IN $ AND G
82      Y,  GOTOR(MC! - 1!)
83          E * 10
84          G + A
85          A = E, Z                        " HEAD OF E = 0, I.E. NO OVERFLOW ?
86      N,  GOTO(:CWRONG)
87          SUBC(:CNNRC)
88          JUMR( - 12)
```

deliver non-space

```
89 CDNONS(4)
90          SUBC(:CNNRC)
91      U,  A - 4, Z                        " BLANK SPACE ?
92      Y,  JUMR( - 3)                      " SKIP BLANKS
93          GOTOR(MC! - 1!)

94                                                                  " TEXTM! = 5
```

print character on console printer

```
95 CRRINT(58)
96          SUBCD(:UREDDEN)
97      U,  A - 2, Z                        " IS A = TWNR ?
98      Y,  JUMR(28)
99      U,  A - 31, R                       " Y, INDIEN CASE = CY
100     Y,  A - 32                          " MAAK HARDWARE REPRESENTATIE
101         $ = CASEDW, E                   " CASEDW = CASECHAR ?
102     N,  CASEDW = - $                    " CASEDW := CASECHAR
103     N,  JUMR(3)
104         $ = CKLACH, E
105         $ = CKLDW, E                    " KLEURDW = KLEURCHAR ?
106     Y,  JUMR(9)
107         $ = CKLACH
108         CKLDW = $, R                    " KLEURDW := KLEURCHAR, ZWART ?
109     Y,  E = 32                          " MAAK REPRESENTATIE VAN DRUKWERK CASE/KLEUR SHIFT
110     N,  E = 33
111         $ = CASEDW, R
112     N,  E + 4
113         $ = 1
114         RLUSS(CMAGADRES)                " VW := VW + 1
```

```
115          MS[ - 1] = G                    " ZET CASE/KLEUR IN MAGAZIJN
116          A + 0, Z
117      Y,  A = 0                           " MAAK VAN 0   + 0
118          S = 1
119          RLUSS(CMAGADRES)
120          MS[ - 1] = A                    " ZET CHARACTER IN MAGAZIJN
121          A - 8, Z                        " WAS CHARACTER = NR
122      N,  A = 1                           " NWR := NWR + 1
123          RLUSA(CNWR)
124          A - CMAXVR, Z                   " REGELBREEDTE BEREIKT ?
125      N,  JUMR(8)
126          A = 2
127          S = 2
128          RLUSS(CMAGADRES)                " VW := VW + 2
129          MS[ - 2] = A                    " IN MAGAZIJN TW
130          A = 8
131          MS[ - 1] = A                    " IN MAGAZIJN NR
132          A = 0
133          CNVR = A
134          S - CMAXMAG, R          " MAGAZIJN VOL ?
135      N,  A = CGARR, R                    " GA PRINTEN = IRUE ?
136      N,  JUMR(15)
137          S + CMAGLE[ - 4]                " S := AANTAL GEVULDE PLAATSEN
138          A = CRLABEL[2]
139          LCA(9)
140          RUSA(9)
141          CRLABEL[2] = A                  " CODEWOORD IN LABEL[1] IN ORDE
142          A = :CRLABEL[1]
143          UARCR[0] = A                    " ARODW := (:LABEL[0])
144          A = UCON]                       " 2 ↑ 18
145          UARCR[2] = A                    " ARTDW := 1
146          DO(UIETCR[7])                   " ZET AEDW IRUE
147          E = :UIETCR[0]
148          URH
149          S = :CMAGAZYN
150          CMAGADRES = S                   " HERINITIALISERING MAGAZIJNWIJZER
151          CGARR = - B                     " GA PRINTEN := EALSE
152          GOTO(:UHERSTELTERUG)
153          + 10 000 000                    " CRRINT[57], FOR THE BENEFIT OF CRRINTDEC

154                                                                  " TEXTM] = 7
155  CRRINTDEC(29)    print decimal number
156          SUBCD(:UREDDEN) ——————————→     "save registers contents
157          S = A, Z                        " DE NUL KRIJGT EEN SPECIALE BEHANDELING
158      Y,  E = 1                           " AS IF POWER OF TEN EXHAUSTED
159      Y,  S = 0
160      Y,  JUMR(12)
161          S = A, R
162      N,  S = - S
163      N,  A = 56                          " SEMI HARDWARE REPR. - SIGN
164      N,  SUBC(:CRRINT)
165          A = CRRINT[57]
166      U,  A - S, R
167      Y,  RCA(1)
168      Y,  MC = S
169      Y,  LUS(2)
170      Y,  S + MC[ - 1]                    " MULTIPLY BY 5
171      Y,  JUMR( - 6)                      " LEADING ZEROS BEING SKIPPED
172          G = A                           " STORE LAST POWER OF TEN
173          A = 0
```

```
174          DIVA$(G)
175          RCSA(27)                          " IN A RESULT , IN S REMAINDER
176          SUBC(:CPRINTINT)
177          MULS(5)
178          A = G                             " TEN POWER BACK IN A
179          RCA(1), R                         " TEN POWER NOT EXHAUSTED ?
180      Y,  JUMR( - 9)
181          A = 4                             " SPATIE
182          CGARR = A                         " GA PRINTEN := IBUE
183          SUBC(:CPRINT)
184          GOTO(:UHERSTELTERUG) ——→ "restore registers contents and subroutine exit.

185                                                                  " TEXTMI = 8
186 CLEESHERKEN(55)    :collect and recognize message
187          MC = F                            " SAVE F
188          S = 0
189          CBUR = S
190          CNCHAR = S                        " INITIALISATION
191          SUBC(:CNNRC)
192      U,  A - 4, Z                          " SPACE
193      N,  JUMR(4)
194          A = CNCHAR, Z                     " NO WORD UNDER CONSTRUCTION ?
195          A = 1
196      N,  JUMR(6)
197          JUMR( - 7)
198      U,  A - 50, Z                         " CHARACTER = $$ ?
199      Y,  A = 3
200      Y,  JUMR(2)
201      U,  A - 2, Z                          " CHARACTER = TWNR ?
202      N,  JUMR(4)
203          S = CBUR
204          RCA(3)                            " SHIFT TERMINAL INDICATION TO D25 D24
205          S + A                             " TERMINAL INDICATION INCLUDED
206          JUMR(9)
207          S = 1
208          PLUS$(CNCHAR)
209          S - 5, Z                          " FIFTH CHARACTER COMING ?
210          S = CBUR                          " (UNFINISHED) WORD IN S
211      N,  LUS(6)
212      N,  S + A
213      N,  CBUR = S                          " STORE WORD
214      N,  JUMR( - 24)
215          CLNRC = A                         " FILL LAST NON PROCESSED CHARACTER
216          F = 0
217          A = CHLW                          " INITIALISATION RECOGNITION PROCEDURE
218          A = MA[0], R                      " CODEWORD IN A, INVERTED ?
219          MC = A                            " SAVE END OF LIST INDICATION
220      N,  A = - A                           " RESTORE INVERSION
221          LCA(9)                            " TELLINGDEEL IN DB T/M D0
222          A '*' 511                         " A := TELLINGDEEL
223          A + :CWORDLIST
224      U,  S - MA, Z                         " COMPARISON SUCCESSFULL ?
225      Y,  JUMR(6)
226          A = MC[ - 1], R                   " IS THIS NOT THE END OF THE CODELIST
227      N,  GOTO(:CWRONG)                     " ABNORMAL EXIT
228          A = 1
229          PLUSA(CHLW)                       " NEW LISTPOINTER IN A
230          F + 1
231          JUMR( - 14)
232          A = MC[ - 1], R
```

```
233      N,  A = - A                          " RESTORE INVERSION
234          LCA(9), R                        " D17 = 0 ?
235          LCA(2)                           " D15 HAS TO BE = 0
236          RUA(11)                          " ADDRESS IN A
237      Y,  CHLW = A
238      Y,  GOTO(:CLEESHERKEN[1])            " CONTINUE SEARCH FOR MESSAGE
239          S = G                            " SERIAL NUMBER IN S
240          E = MC[ - 2]                     " RESTORE E
241          GOTOR(MC[ - 1])

242                                                                  " TEXTMI = 10
243          " TEXT MESSAGE INTERPRETER PROPER, INITIALISATION ON FIRST INSTRUCTION, I.E. T = :CMI[0] AND
244          " INDICATING THIS AM TO BE READ AND DEAF.
245  CMI(33)                                                  address of semaphore
246          E = QUIETCK[0]                       " R(IETTB)  ---->  the corresponding V is done by the operator by pressing a key on the console keyboard.
247          URH   hardware P-operation           " R(MUTEX)  ---->  this is the point where the message interpreter is usually held up.
248          COMSEMR                              " S := COMVAR
249          S = UAMVAR                           " COMVAR = 1 ?
250      U,  S - 1, Z                             " OPERATOR PRIORITY := IBUE
251      N,  JUMR(7)
252          UAMTELREQ = S                        " A := ARO
253          COMSEMV
254          SUBC(:CMAKEIETZERO)
255          A = UARCK[0]                         " COMVAR = 0
256          SUBC(:CNOTCASE)
257          SUBC(:CAESL)
258          JUMR( - 13)
259          S = S, Z                             " COMVAR := 3
260      Y,  S = 3                                " COMVAR = 2 ?
261      Y,  UAMVAR = S                           " PREINITIALISATIE LIJSTWIJZER
262      U,  S - 2, Z                             " V(MUTEX)
263      N,  S = :CERIS                           " IETTB := IETTB + 1
264      N,  UMIREAC[0] = S                       " IETB := IBUE
265          COMSEMV                              " NOTEER BEGINWAGENPOSITIE
266          A = UCON1                            " MAAK KLAAR VOOR PRINTROUTINE, ROOD
267          RLUSA(UARCK[1])                      " PREVIOUS CHARACTER := LAST NON PROCESSED CHARACTER := EMPTY
268          DO(UIETCK[3])                        " INITIALISEER HEERSENDE LIJSTWIJZER
269          S = CHWR                             return with address of continuation in A (depending on message)
270          CBWR = S                                 at the end return to first instruction of CMI follows
271          CKLACH = - B
272          S = UMIREAC[0]
273          A = 0
274          CLNRC = A
275          CRC = A
276          CHLW = S
277          SUBC(:CLEESHERKEN)
278          GOTO(A)

279  CWRONG(19)                                   reaction on erroneous message
280          S = CTWNR, R                         " MESSAGE RECALLED BY OPERATOR ?
281      Y,  CTWNR = - S                          " MESSAGE RECALLED := FALSE
282          B = :USBNI[5]                        " RESTO STACKPOINTER ANYWAY
283      Y,  JUMR(7)
284          CKLACH = B                           " MAAK KLEUR TIJDELIJK ZWART
285          A = 50
286          SUBC(:CRRINT)
287          SUBC(:CRRINT)                        " PRINT CROSS CROSS
288          A = 2
289          CGARR = A                            " GA PRINTEN := IBUE
290          SUBC(:CRRINT)
```

This is the cyclical process "message interpreter"

```
291          S = CBWR, Z                    " WAGENPOSITIE ZO LATEN ?
292      Y,  GOTO(:CMI[25])
293          A = 4
294          S - 1, Z                       " COUNT NUMBER OF SPACES TO BE GIVEN
295      Y,  CGARR = A
296          SUBC(:CRRINT)                  " PRINT SPACE
297      N,  JUMR( - 4)
298          GOTO(:CMI[25])

299                                                              " TEXTMI = 12
300 CSELCON(7)
301          S = :USBMI
302          A = UAMTELREQ[US], Z           " = OPERATOR PRIORITY ?
303      N,  UAMTELREQ[US] - A               " OPERATOR PRIORITY := FALSE
304      N,  A = 3
305          UAMVAR[US] = A                 " COMVAR := 0 OF COMVAR := 3
306      Y,  SUBCD(:CLCON)
307          GOTOR(MC[ - 1])
308                                                              " TEXTMI = 13
309 CLCON(13)          claim console
310          S = :USBMI
311          A = UAMVAR[US], Z              " COMVAR = 0 ?
312      Y,  A = :UAMTELREQ  → now anyone can inspect the variable uamvar in the stackbottom of the messageinterpreter
313      Y,  E = :ULISTAM
314      Y,  S = 1
315      Y,  SUBCD(:UEXREAM)
316      Y,  A = :USBMI
317      Y,  UAMVAR[UA] = S          " COMVAR := 1
318      Y,  CKLACH = S                     " KLEUR := ZWART
319      Y,  UAMTELREQ[UG] - S              " UAMTELREQ := 0
320      Y,  E = :UAMSEM[UG]
321      Y,  SUBCD(:UVOR[2])                " V(AMSEM)
322          GOTO(MC[ - 1])
323                                                              " TEXTMI = 14
324 CONVINTCK(5)
325          SUBCD(:UREDDEN)
326      U,  A - 36, B
327      Y,  A - 27                         " CONVERT ALL LETTERS TO ONE TYPE
328          E = :CTABINTCK
329          GOTO(:UCONVTAIL)
330 CRRINTINT(5)   print character in internal representation
331          MC[0] = A                      " SAVE A
332          SUBC(:CONVINTCK)
333          SUBC(:CRRINT)
334          A = MC[ - 1]                   " RESTORE A
335          GOTOR(MC[ - 1])

336                                                              " TEXTMI = 15
337 CONVCKINT(3)
338          SUBCD(:UREDDEN)
339          E = :CTABCKINT
340          GOTO(:UCONVTAIL)
341 UCONVTAIL(12)
342          S = A
343          A = 0
344          DIVAS(3), Z                    " A := POSITION OF CODE IN TABLEWORD
345          S + G                          " ADDRESS OF TABLE WORD
346          S = MS                         " TABLE WORD IN S
347      N,  RCS(9)
348      N,  A - 1, Z                       " CODE SHIFTED INTO CORRECT POSITION ?
```

```
349      N,  RCS(9)
350          S '*' 511                          " RETAIN 9 BITS IN S
351          M[B - 4] = S                       " MAKE NEW VALUE OF A TO BE RESTORED
352          SUBCD(:UHERSTEL)
353          GOTOW(MC[ - 1])
354                                                                  " TEXTMI - 16
355 CRRINTSTRING(10)
356          SUBCD(:UREDDEN)
357          SUBCD(:UINSTV)
358          SUBCD(:UDNSS)
359      U,  A - 510, Z                         " END OF STRING ?
360      Y,  B - 3                              " KILL STRING VARIABLE
361      Y,  GOTO(:UHERSTELTERUG)
362          S - 510, Z                         " LOOK AHEAD IS END OF STRING ?
363      Y,  CGARR = B                          " GA PRINTEN := IBUF
364          SUBC(:CRRINTINT)
365          JUMP( - 8)

366 CTABCKINT(21)
367 '167 070 000'
368 '054 135 063'
369 '000 061 062'
370 '053 066 060'
371 '047 064 055'
372 '076 051 072'
373 '067 046 050'
374 '074 052 075'
375 '056 073 045'
376 '065 071 000'
377 '173 000 057'
378 '011 000 005'
379 '127 131 000'
380 '143 000 130'
381 '010 145 004'
382 '106 132 000'
383 '174 100 003'
384 '006 170 102'
385 '101 103 144'
386 '000 133 002'
387 '142 001 007'

388 CTABINICK(34)
389 '071 075 055'
390 '041 052 060'
391 '054 074 065'
392 '023 030 043'
393 '020 022 016'
394 '005 013 026'
395 '036 032 014'
396 '006 007 011'
397 '035 015 003'
398 '001 024 012'
399 '031 017 034'
400 '021 025 027'
401 '070 061 000'
402 '000 067 063'
403 '000 057 000'
404 '000 000 000'
405 '000 000 000'
```

code conversion tables

```
406 '000 000 000'
407 '000 000 000'
408 '000 000 000'
409 '045 047 046'
410 '000 072 056'
411 '000 000 004'
412 '076 000 000'
413 '053 066 051'
414 '000 000 000'
415 '000 000 000'
416 '000 000 000'
417 '000 000 000'
418 '000 000 000'
419 '002 000 000'
420 '000 000 064'
421 '000 062 040'
422 '000 000 000'
423                                                          " COORD
424 UINSTV(6)
425         S = MC[ - 1]                        " SAVE LINK
426         MC = G                              " INITIALISE STV[0]
427         E = - 0
428         MC = E                              " INITIALISE STV[1] AND STV[2]
429         E = :MC[ - 3]                       " :STV[0] IN G
430         GOTOR(S)

431         " DELIVER NEXT STRING SYMBOL
432 UDNSS(25)
433         A = MG[1]                           " LOOK AHEAD SYMBOL
434     U,  A - 510, Z                          " END OF STRING ?
435     Y,  GOTOR(MC[ - 1])
436         A = MG[2], Z
437     Y,  S = - A, R                          " SHIFTED STRING WORD EXHAUSTED ?
438     N,  JUMP(11)
439         A = MG[0]                           " INVARIANT ADDRESS IN A
440         MC = G, Z                           " STORE G, MAKE N CONDITION
441         SUBCD(:RSE41)                 " E := NEXT TWO WORDS
442         A = M[57]                           " NEW STRING WORD IN A
443     U,  A - 511, Z                          " TRANSITION TO NEW SEGMENT ?
444     Y,  A = M[58]                           " INVARIANT ADDRESS IN A
445         G = MC[ - 1]                        " RESTORE G
446     Y,  MG[0] = A                           " STORE NEW INVARIANT ADDRESS
447     Y,  JUMP( - 8)
448         S = 1
449         MG[0] + S                           " INCREASE INVARIANT ADDRESS
450         S = - 0
451         LCSA(9)                             " NEW LOOK AHEAD SYMBOL IN S
452         MG[2] = A                           " STORE FRESHLY SHIFTED WORD
453         S '*' 511                           " ISOLATE NEW LOOK AHEAD SYMBOL
454         A = MG[1], R                        " NEXT SYMBOL IN A; NOT EMPTY ?
455         MG[1] = S                           " STORE NEW LOOK AHEAD SYMBOL
456     N,  GOTO(:UDNSS)
457         GOTOR(MC[ - 1])

458                                                          " COORD
459         " EXIST REQUIRED AM ?
460 UEXREAM(15)
461         A = :M0[0]                          " A := DISTANCE FROM STACK BOTTOM
462         UWORK1 = B                          " SAVE B
463         B = MG[0]                           " CYCLIC POINTER IN B
```

initialise string variable

to be called with : string variable in G

```
464       U,  B - :MG[2], Z
465       Y,  B + MG[1]                    " PREPARE B FOR NEXT AM
466           A + MC[ - 1]                 " ADDRESS STATUS VARIABLE IN A
467       U,  S - MA[0], Z                 " REQUIRED AM ?
468       Y,  MG[0] = B, R                 " SET CYCLIC POINTER
469       Y,  JUMR(3)
470           A - M[B]
471       U,  B - MG[0], Z
472       N,  JUMR( - 9)
473           G = M[B], E                  " ADDRESS STACK BOTTOM IN G; SET CONDITION
474           B = UVORK1                   " a bad decision to use a fysical address as working location.
475           GOTO(MC[ - 1])               " RETURN WITH CONDITION          Only possible on account of deafness during execution of this subroutine.
476 UIETCK(10)
477           :UARCK
478           - 0
479           - 0
480           '760 071 011'                " IE := IBUE
481           '660 071 011'                " IE := EALSE
482           '760 072 011'                " LVIE := IBUE
483           '660 072 011'                " LVIE := EALSE
484           '760 070 011'                " AE := IBUE
485           '400'                        " 1 IN POSITION IN FLIP-FLOP WORD OF APPARATUS NO. 9
486           0

487 UCONEREE(5)
488           SUBCD(:UBRINTNLCR)
489           COMSEMR                      } free console
490           SUBCD(:CSELCOM)
491           COMSEMV
492           GOTOR(MC[ - 1])

493 URRINTNLCR(4)
494           A = 2
495           CGARR = A
496           SUBC(:CRRINT)
497           GOTOR(MC[ - 1])

498 ULISTAM(UNAM[1])
499           :ULISTAM[2]
500           UNAM[ - 1]
501           :USBRM1
502           :USBRM2
503           :USBRM3
504           :USBRM4
505           :USBRM5
506           :USBRLA
507           :USBTRA
508           :USBTRB
509           :USBTRC
510           :USBLRA
511           :USBTRA
512           :USBTRB
513           :USBTRC
514           :USBDR
```

list to be scanned by "exist required AM"

~~priority list of all AM's, always searched~~

```
515           " MESSAGE INTERPRETER, KLEINE VULLINGEN
516 CASETB(1)
517           + 0
518 CTWNR(1)
519           - 0
520 COMSEM(3)
```

```
521           + 1
522           - 0
523           - 0
524 UIETCR(10)
525           :UARCR                    " hardware semaphore of console printer "
526           - 0
527           - 0
528           '760 071 010'             " IEDW := IBUE
529           '660 071 010'             " IEDW := EALSE
530           '760 072 010'             " LVIEDW := IBUE
531           '660 072 010'             " LVIEDW := EALSE
532           '760 070 010'             " AEDW := IBUE
533           '1 000'                   " 1 IN FLIPFLOPWORD IN POSITION OF APPARATUS NO. 8   " mask "
534           0
535 CGARR(8)
536           - 0
537           - 0                       " CASEDW
538           + 0                       " CKLDW
539           - 0                       " CKLACH, INITIALISATION IRRELEVANT
540           :CMAGAZYN[3]              " CMAGADRES
541           36                        " CMAGAZYN, LETTERSHIFT RED
542           2                         " CARRIAGE RETURN
543           8                         " LINE FEED
544 CHWR(4)
545           + 0
546           + 0                       " CRLABEL
547           :UDISASTER2[1]
548           :CMAGAZYN[ - 1]
549 USBMI(1)
550           :UIETCK
551 COMSEMRADRES(2)
552           F = :COMSEM
553           GOTO(:UROR[3])
554 COMSEMVADRES(2)
555           F = :COMSEM
556           GOTO(:UVOR[2])

557 UCLAIMCON(7)
558           COMSEMR                   " R(COMSEM)
559           S = 1
560           UAMTELREQ = S
561           SUBCD(:CLCON)
562           COMSEMV                   " V(COMSEM)
563           UROWNSEM                  " R(OWNSEM)
564           GOTOR(MC[ - 1])
565 UCONCON(9)
566           COMSEMR
567           UMIREAC[0] = S
568           S = :M0[0]
569           UMIREAC[1] = S            " FILL IN :USB
570           F = :USBMI
571           S = 2
572           UAMVAR[UG] = S            " COMVAR := 2
573           COMSEMV
574           GOTOR(MC[ - 1])
575 UROWNSEMADRES(2)
576           F = :UAMSEM
577           GOTO(:UROR[3])

578 CREADOCT(15)
```

identifying apparatus nr 8

communication instruction for this apparatus

number of console variables that are only addressed statically.

read octal from keyb.

```
579           SUBC(:CDNONS)
580           E = 0
581           S = A                          " COPY SYMBOL IN S
582           SUBC(:CONVCKINT)
583       U,  A - 7, R                       " ¬ OCTAL DIGIT ?
584       Y,  CLNRC = S                      " FOR NEXT DELIVERY
585           S = G                          " DELIVER VALUE IN S AND G
586       Y,  GOTOR(MC[ - 1])
587           LCS(3)
588       U,  S '*' 7, Z                     " NON OVERFLOW ?
589       N,  GOTO(:CWRONG)
590           S '+' A
591           G = S                          " STORE PARTIAL VALUE
592           SUBC(:CNURC)
593           JUMP( - 13)
594 CRROCT(14)      print octal.
595           A = 2
596           SUBC(:CRRINT)                  " CARRIAGE RETURN
597           A = 8
598           MC = A                         " STACK COUNTER
599           LCS(3)
600           A = 7
601           A '*' S                        " CREATE OCTAL DIGIT IN A
602           SUBC(:CRRINTINT)
603           A = - 1
604           A + MC[ - 1], Z                " DECREASE COUNTER UNSTACKING, LAST DIGIT TO BE PRINTED ?
605       Y,  CGARB = B
606       U,  A + 1, Z                       " DONE ALTOGETHER ?
607       Y,  GOTOR(MC[ - 1])
608           JUMP( - 11)

609 CREQSS(4)
610           SUBC(:CDNONS)
611       U,  A - 50, Z
612       N,  GOTO(:CWRONG)
613           GOTOR(MC[ - 1])
614 CRMMES(19)
615           SUBC(:CREADDEC)
616       U,  S - UNUMRM, R
617       N,  S - 0, E
618       Y,  GOTO(:CWRONG)                  " NUMBER OF NON EXISTENT RM
619           S + :ULISTRM[1]
620           S = MS                         " ADDRESS OF USBRMI IN S
621           UMIREAC[5] = S
622           S = :CRMMES[9]
623           GOTO(:CM.[30])                 " CONTINUE COLLECTION OF MESSAGE
624           '004 400 000'[:UMES32]         " GOSS
625           '005 400 000'[:UMES33]         " OUTSS
626           '002 400 000'[:UMES30]         " STORSS
627           '017 400 000'[:CTIMEMES]       " TIMESS
628           '012 000 000'[:CRMMES + 17]    " STAC
629           '036 400 000'[:CSLOWMES]       " SLOWSS
630           '037 400 000'[:CSLOWMES + 2]   " RUNSS
631           -'040 000 000'[:CFORMSMES]     " FORM
632           '013 400 000'[:CSTACKMES]      " K+SP
633           - '014 400 000'[:CSTACKMES]    " K-SP

634 CSLOWMES(8)
635           A = 0
636           JUMP(1)
```

reactions to specific messages

```
637           A = - 0
638           $ = UMIREAC[5]
639       U,  A = UKILLVAR[US], Z
640       Y,  A = - 0
641           CGLSLOW[US] = A
642           GOTO(:UENDMES[4])
643   CFORMSMES(9)
644           ='041 400 000'[:CFORMSMES + 1]
645           $ = UMIREAC[5]                          " :M0[0]
646           $ = ULPSTREAMS[US]                      " : LIST OF HANDLES
647           $ = M$                                  " :LPHANDLE
648       U,  $ = M$[6], R                            " DOCUMENT UNDER CONSTRUCTION ?
649           G = M$[15]
650           F + 1
651       N,  F = 0
652           GOTO(:CTIMEMES[11])
653   CSTACKMES(7)
654           MC = $                                  " SAVE SERIAL NUMBER IN CODELIST
655           SUBC(:CREADDEC)
656           A = MC[ - 1], Z
657       Y,  UMIREAC[6] = $
658       N,  UMIREAC[6] = - $
659           SUBC(:CREQSS)
660           GOTO(:UMES34)

661   CORY(35)
662           SUBC(:CREADDEC)
663           $ '*' 63                                " NO MORE THAN 63 WORDS IN A ROW
664           UMIREAC[5] = $
665           $ = :CORY[18]
666           GOTO(:CM![30])                          " CONTINUE COLLECTION OF MESSAGE
667           SUBC(:CREADOCT)
668       U,  $ '*' - UCON2, Z                        " POTENTIAL CORE ADDRESS ?
669           G = $                                   " SAVE ADDRESS IN G
670           SUBC(:CREQSS)
671           CKLACH = B
672       N,  JUMP(8)
673           $ = MG
674           SUBC(:CRROCT)
675           $ = 1
676           UMIREAC[5] - $, R                       " DECREASE COUNTER (AT LEAST ONE WORD WILL BE PRINTED)
677       N,  GOTO(:UENDMES[4])
678           F + 1                                   " INCREASE ADDRESS CONTENTS OF WHICH WILL BE PRINTED
679           JUMP( - 7)
680           - '011 400 000'[:CORY + 5]              " CODELIST
681           $ = G
682           RUS(9)                                  " MAKE :$V
683       U,  $ = M$, Z                               " SEGMENT EMPTY ?
684       Y,  GOTO(:UENDMES[4])
685           A = G
686           MC = A, Z                               " SAVE INVARIANT ADDRESS
687           SUBCD(:RSE39)
688           $ = G
689           SUBC(:CRROCT)
690           A = 1
691           UMIREAC[5] - A, R
692           A + MC[ - 1]
693       N,  GOTO(:UENDMES[4])
694       U,  A '*' 511, Z                            " OVERFLOW INTO NEXT SEGMENT ?
695       N,  JUMP( - 10)
```

```
696           GOTO(:UENDMES[4])
697 CEILL(18)
698           SUBC(:CREADOCT)                    " READ ADDRESS IN G AND $
699           UM:REAC[5] = $
700           $ = :CEILL[11]
701           GOTO(:CM:[30])                     " CONTINUE COLLECTION OF MESSAGE
702           SUBC(:CREADOCT)                    " READ NEW CONTENTS
703           SUBC(:CREQSS)
704           $ = UMIREAC[5]
705      U,   $ '*' - UCON2, Z                          " POTENTIAL CORE ADDRESS ?
706      N,   JUMR(3)
707           MS = G                             " REPLACE BY NEW VALUE
708           GOTO(:UENDMES[4])
709           - '007 400 000'[:CEILL + 4]
710           MC = $                             " STACK INVARIANT ADDRESS
711           RUS(9)
712      U,   $ = MS, Z                          " SV EMPTY ?
713      Y,   B - 1
714      N,   SUBCD(:RSE50)                      " WRITE INTEGER ON INV. ADDRESS
715           GOTO(:UENDMES[4])
716 CTIMEMES(17)
717           $ = UMIREAC[5]
718           G = UNITS[US]
719           F * 120
720           MC = F
721           G = URROCTIME
722           G + GBILL[US]
723           F / 16 667
724           F + MC[ - 2]
725           JUMR(0)
726           F + DRT[11]
727           F - DRT[11]                        " ROUND TO SECONDS
728           CKLACH = B                         " PRINT IN BLACK
729           A = 4
730           SUBC(:CRRINT)                      " PRINT BLANK SPACE
731           A = G
732           SUBC(:CRRINTDEC)
733           GOTO(:UENDMES[4])

734
735 SRR(30)
736           $ + :MC[UOVERSHOOT - 2]            " INITIAL AR + $ + OVERSHOOT - 1
737           $ '+' B
738      U,   $ '*' - 511, Z                     " SPACE AVAILABLE IN CURRENT PAGE?
739           $ '+' B                            " RESTORE $
740           $ - :MC[UOVERSHOOT - 2]
741      Y,   GOTO(MC[- 1])                      " RETURN WITH YES-CONDITION
742           SUBCD(:UREDDEN)                    " SAVE REGISTERS
743           SUBCD(:UIMMORTAL)
744           G = GL16                           " NEED
745           G - GL25, P                        " LOAN < NEED?
746      N,   SUBCD(:UCLAIMCONRED)               " RED CLAIM CONSOLE
747      N,   $ = :M0[0]
748      N,   SUBCD(:UMES35)                     " MESSAGE M35
749      N,   SUBCD(:URRINTNLCR)
750      N,   SUBCD(:USTORREN[1])
751      N,   JUMR(- 8)
752           A = 1
753           GL25 + A                           " INCREASE LOAN
754           $ = B
```

```
"SRR - 0
  " stack page request
```

Our stack is broken in pages of 512 words
On certain instances a test is made for enough stack space available, if not a new stack page is requested and hooked on.

```
755           $ '*' = 511                    " ADDRESS SV NEW STACKPAGE
756           SUBCD(:SRQ)
757           G = MS[0]                      " :CTE NEW STACKPAGE
758           A = MG[2]                      " :CRB NEW STACKPAGE
759           MA[0] = - E                    " SV NEXT STACKPAGE := EMPTY
760           MA[1] = S                      " CHAIN TO PREVIOUS STACKPAGE
761           A + 2                          " ADDRESS FIRST FREE PLACE IN NEW PAGE
762           NSR = A, Z                     " FILL NSR, SET NO CONDITION
763           SUBCD(:UMORTAL)
764           SUBCD(:UHERSTEL)
765           GOTO(MC[- 1])                  " RETURN WITH NO-CONDITION

766                                                                  "COORDINATOR
767                                                                  "UIMMORTAL
768                                                                  "UMORTAL
769                                                                  "USTOPPEN
770 UIMMORTAL(4)
771           SUBCD(:UREDDEN)
772           S = 1
773           UKILLCOUNTER + S               " INC(KILLCOUNTER)
774           GOTO(:UHERSTELTERUG)

775 UMORTAL(7)
776           SUBCD(:UREDDEN)
777           S = 1
778           UKILLCOUNTER - S, Z            " DEC(KILLCOUNTER), ZERO?
779      Y,   S = UKILLVAR
780      Y,   S - 2, Z                       " KILLVAR = 2?
781      Y,   SUBCD(:USTOPPEN)
782           GOTO(:UHERSTELTERUG)

783 USTOPPEN(6)
784           SUBCD(:UCLAIMCONRED)           " RED CLAIM CONSOLE
785           S = : M0[0]                    " ENTRY POINT STACKPAGE REQUEST
786           SUBCD(:USWANSONG)              " SIGNAL APPARENT DEATH
787           SUBCD(:UCONFREEWHITE)          " WHITE FREE CONSOLE
788           R = :UKILLSEM
789           GOTO(:UROR[3])                 " R(KILLSEM) AND RETURN

790                                                                  "COORDINATOR
791                                                                  "UMES36
792 UMES36(30)
793           S = UDANGERPOINT
794           S - UFIP, R                    " FIP < DANGERPOINT
795      Y,   S = UKILLVAR, R                " KILLVAR ≥ 1?
796      Y,   RUS(2), Z                      " KILLVAR ≤ 3?
797      N,   GOTOR(MC[- 1])
798           SUBCD(:UCLAIMCONRED)           " RED CLAIMCONSOLE
799           S = :ULISTRM[2]
800           A = MS[0]                      " :USBRMI
801           G = UM36[UA]                   " RMI
802           SUBCD(:CRRINTSTRING)
803           G = UKALESRATIE
804           SUBCD(:CRRINTSTRING)           " SPACE
805           A = GL24[UA]                   " NUMBER OF P - SEGMENTS RMI
806           SUBCD(:CRRINTDEC)
807           SUBCD(:URRINTNLCR)
808           S + 1
809      U,   S - :ULISTRM[UNUMRM + 2], Z
810      N,   JUMR(- 11)
```

more message reactions

```
811           G = CMINRUT                    " INRUT
812           SUBCD(:CRRINTSTRING)
813           A = - VEI
814           A + UINITIALEI
815           SUBCD(:CRRINTDEC)
816           SUBCD(:URRINTNLCR)
817           G = CMEREE                     " EREE
818           SUBCD(:CRRINTSTRING)
819           A = VEIR
820           SUBCD(:CRRINTDEC)
821           SUBCD(:URRINTNLCR)
822           GOTO(:USTORREN[1])
823 UKALESRATIE(1)
824           UINVM351 5)

825                                                                  "COORDINATOR
826                                                                  "UMES34
827                                                                  "UMES32
828                                                                  "UMES33
829 UMES34(20)
830           S = UMIREAC[5]                 " :USBRMI
831           A = UKILLVAR[US], R
832           A - 3, E                       " KILLVAR = 1 ~ 2 ~ 3?
833           G = CMNA30
834      Y,   GOTO(:UENDMES)                 " NON APPLICABLE
835           G = UMIREAC[6]                 " INCREMENT NEED
836           G - UROOL, R                   " INCREMENT > POOL?
837      Y,   E = 0
838           G + UROOL                      " MINIMUM(INCREMENT, POOL)
839           G + GL16[US]                   " DESIRED NEED
840           A = G
841           G - GL25[US], R                " DESIRED NEED > LOAN?
842      N,   A = GL25[US]
843           A - GL16[US]                   " NEW NEED - OLD NEED
844           GL16[US] + A                   " INCREASE NEED
845           UROOL - A                      " DECREASE POOL
846           CKLACH = B                     " COLOUR BLACK
847           G = UM34[US]                   " RMI STACK GRANTED
848           SUBCD(:UMES35[1])
849           GOTO(:UENDMES[4])

850 UMES32(1)
851           A - 2                          " A := :UMES32[- 2]
852 UMES33(13)
853           S = UMIREAC[5]                 " :USBRMI
854           G = UKILLVAR[US]
855           E - 3, Z                       " KILLVAR = 3?
856           G = CMNA32
857      N,   GOTO(:UENDMES)                 " MESSAGE NOT APPLICABLE
858           A - :UMES32, R                 " A := - 2 OR 1
859           UKILLVAR[US] + A               " KILLVAR := 1 OR 4
860      Y,   A = :UEINDRITEN
861      Y,   UHAMR[US + 2] = A              " CHANGE T IN HAMROOM
862      Y,   A = UHAMR[US], Z               " RM SELECTABLE?
863      N,   E = :UKILLSEM[US]
864      N,   SUBCD(:UVOR[2])                " V(KILLSEM)
865           GOTO(:UENDMES[4])

866                                                                  "COORDINATOR
867                                                                  "UMES35
```

```
868 UMES35(8)
869         G = UM35[US]                    " RMI STACK USED
870         SUBCD(:CRRINTSTRING)            " ENTRY POINT UME$34
871         A = GL16[US]                    " NEED
872         SUBCD(:CRRINTDEC)
873         G = CMPOOL                      " POOL
874         SUBCD(:CRRINTSTRING)
875         A = UPOOL                       " POOL
876         GOTO(:CRRINTDEC)

877                                                                 "COORDINATOR
878                                                                 "USWANSONG
879 USWANSONG(5)
880         A = 3
881         UKILLVAR[US] = A                " KILLVAR := 3
882         SUBCD(:URPROGNAME)
883         G = CM31                        " BLOCKED.
884         GOTO(:CRRINTSTRING)             " MERGE WITH PRINTSTRING AND RETURN

885                                                                 "COORDINATOR
886                                                                 "UMES30
887 UMES30(23)
888         S = UMIREAC[5]                  " :USBRMI
889         A = UKILLVAR[US]
890     U,  A - 3, Z                        " KILLVAR = 3?
891     Y,  JUMP(15)
892     U,  A - 2, Z                        " KILLVAR = 2?
893     Y,  GOTO(:UENDMES[4])
894     U,  A - 1, Z                        " KILLVAR = 1?
895         G = CMNA30
896     N,  GOTO(:UENDMES)                  " MESSAGE NOT APPLICABLE
897         UKILLVAR[US] + A                " KILLVAR := 2
898         A = UKILLCOUNTER[US], Z         " KILLCOUNTER = 0?
899     N,  GOTO(:UENDMES[4])
900         A = UKILLSEM[US], Z             " KILLSEM = 0?
901         A = 1
902     N,  UKILLSEM[US] - A                " DEC(KILLSEM)
903     Y,  UKILLSEM[US + 21 + A            " WAITERS ON KILLSEMAPHORE := 1
904     Y,  USEL - A                        " DEC (NUMBER OF SELECTABLE AM'S)
905     Y,  A = :UKILLSEM[US]
906     Y,  UHAMR[US] = - A                 " SET RMI WAITING
907         CKLACH = B                      " COLOUR BLACK
908         SUBCD(:URRINTNLCR)              " NEW LINE CARRIAGE RETURN
909         SUBCD(:USWANSONG)               " SIGNAL APPARENT DEATH
910         GOTO(:UENDMES[4])

911                                                                 "COORDINATOR
912                                                                 "UCLAIMCONRED
913                                                                 "UCONFREEWHITE
914 UCLAIMCONRED(3)
915         F = :COMSEM
916         URROOD                          " RED R(COMSEM)
917         GOTO(:UCLAIMCON[1])             " MERGE WITH UCLAIMCON

918 UCONFREEWHITE(5)
919         SUBCD(:URRINTNLCR)              " NEW LINE CARRIAGE RETURN
920         COMSEMR                         " R(COMSEM)
921         SUBCD(:CSELCOM)                 " SELECT NEW COMVAR VALUE
922         F = :COMSEM
923         GOTO(:UVOR)                     " WHITE V(COMSEM) AND RETURN
```

```
924                                                          "COORDINATOR
925                                                          "UTREXPROODADRES
926                                                          "UTREXVWITADRES
927 UTREXPROODADRES(2)
928         E = :UTREX
929         GOTO(:UROR)

930 UTREXVWITADRES(2)
931         E = :UTREX
932         GOTO(:UVOR)

933 UENDMES(6)
934         CKLACH = B                    " COLOUR BLACK
935         SUBC(:CPRINTSTRING)
936         JUMP(1)
937         UTREXV
938         SUBCD(:UCONFREE)
939         GOTO(:CM1)
940 CMNA30(1)
941         CINVM30
942 CM31(1)
943         CINVM31
944 CMNA32(1)
945         CINVM32
946 CMPOOL(1)
947         CINVMPOOL
948 CMINPUT(1)
949         CINVMINPUT
950 CMFREE(1)
951         CINVMFREE
952 ULISTPM(UNUMPM[2])
953         :ULISTPM[2]
954         UNUMPM
955         :USBPM1
956         :USBPM2
957         :USBPM3
958         :USBPM4
959         :USBPM5
960 UTREX(3)
961         + 1
962         - 0
963         = 0
964 UTREXPADRES(2)
965         E = :UTREX
966         GOTO(:UROR[3])
967 UTREXVADRES(2)
968         E = :UTREX
969         GOTO(:UVOR[2])
970 UEI(1)
971         UINITIALEI
972 UEIR(1)
973         UINITIALEIR
974 UEIO(1)
975         UINITIALEIO
976 UEIOR(1)
977         UINITIALEIOR
978 UPOOL(1)
979         UPOOLEXTENT
```

P(mutex) / V(mutex) } see "coöp. seq. processes"

strategic variables with regard to the distribution of virtual store among input, output and private PM segments (program pages, arrays)

```
                                                                    " TEXTLP = 0
CLRSEL(39)
        E = :USBLRA
        S = UAMVAR[UG], Z                   " PRIVAR = 0, I. E . PRINTER BUSY ?
    Y,  GOTOR(MC[ - 1])
    U,  S - 1, Z                            " PRIVAR = 1 ?
    N,  JUMR(6)                             " THEN PRIVAR = 2
        A = CURHDL[UG]                      " :HANDLE OF CURRENT (= PREVIOUS) STREAM
    U,  S = MA[0], Z                        " NPOR = 0, I.E. STREAM EMPTY ?
    N,  UAMVAR[UG] - S                      " PRIVAR := 0
    N,  E = :UAMSEM[UG]                     " :PRISEM
    N,  SUBCD(:UVOR[2])                     " V(PRISEM)
    N,  GOTOR(MC[ - 1])                     " PRIVAR = 1 OR 0
        S = G                               " :LINEPRINTER STACK BOTTOM
        G = UCON3                           " 2 ↑ 26 - 1
        M[B] = - E                          " MAX := -(2 ↑ 26 - 1)
        A = :ULRSTRL1[0]                    " INITIALISATION LISTPOINTER
        M[B + 2] = A                        " LISTPOINTER ON STACK
        A = MA[0]                           " TAKE HANDLE ADDRESS
        G = MA[0], Z                        " INSPECTED STREAM EMPTY ?
    Y,  JUMR(7)
        E - MA[7]                           " NPOR - ATTACHING TIME
    U,  S = UNFD, R                         " EXIST( NFD > 0) ?
    U,  S = MA[1], E                        " NFD OF THIS STREAM > 0 ?
    Y,  E - M[B], R
    Y,  E + M[B]
    Y,  M[B] = E                            " NEW VALUE OF MAX
    Y,  M[B + 3] = A
        A = 1
        PLUSA(M[B+ 2])                      " INCREASE LISTPOINTER
    U,  A - :ULRSTRL1[UNUMLRSTR], Z         " ALL STREAMS INSPECTED ?
    N,  JUMR( - 14)                         " REPEAT SEARCH
        G = S                               " STACK BOTTOM BACK IN G
        A = UNFD, R                         " EXIST( NFD > 0) ?
    N,  A = USUMR
    N,  A - URRAS, R                        " SUMP > FPAS ?
    N,  GOTOR(MC[ - 1])                     " NO REASON TO PRINT
        S = UAMVAR[UG], Z                   " TO MAKE CONDITION N
        A = M[B + 3]
        CURHDL[UG] = A
        JUMR( - 32)

                                                                    " TEXTLP = 2
CVULMAG(13)
        M[B] = S                            " SAVE NUMBER
        RGS(9)                              " SHIFT TO "TELLINGDEEL" POSITION
        S + CLRBUF
        CLRLAB[2] = S                       " FILL IN CODEWORD
        S = CLRBUF
        M[B] + S                            " LAST PLACE TO BE FILLED, UPPER LIMIT (RESTORED)
        E = MA                              " GET 2 WORDS IN E
        MS[1] = E                           " PLACE IN BUFFER
        S + 2
        M[B ] - S, R                        " MORE WORDS TO BE TRANSPORTED ?
    Y,  A + 2
    Y,  JUMR( - 7)
        GOTOR(MC[ - 1])
CFORMF(6)
        CFEED = B                           " FORMFEED := IRUE
        S = 1
```

text lineprinter CM

```
1039          A = :CRORME[5]
1040          SUBCD(:CVULMAG)
1041          GOTOR(MC[ - 1])
1042          '400 000 000'                    " INDICATION FIRST CHANNEL CYCLIC TAPE = 32
1043                                                                  " TEXTLR - 3
1044 CLRSTART(44)
1045          A = 6
1046          CRARE = A                        " ALLOW 5 CONSECUTIVE PARITY ERRORS
1047          SUBCD(:CVAETRIET)
1048          A = CLRLAB[0], R                 " CLOSING WORD
1049     Y,   A + 0, Z                         " OK ?
1050     Y,   JUMR(34)
1051          S = 0
1052          LCSA(9)                          " TELLINGDEEL IN S
1053     U,   S - 511, Z                       " NBK ?
1054     Y,   GOTO(:CLROUT)                    " S = 511, CONDITION Y
1055          RCA(11), R                       " NO PARITY ERROR ?
1056     Y,   JUMR(9)
1057          S = 1
1058          MINS(CRARE), R                   " NOT TOO MANY PARITY ERRORS ?
1059     N,   GOTO(:CLROUT)                    " S = - 0; CONDITION N
1060          M1[0] = S                        " HOK BEHIND THE SCENES
1061          A = CLRCON1
1062          S = CLRBUF
1063          A '*' MS[1]                      " REMOVE LINEFEED FROM FIRFT BUFFERWORD
1064          MS[1] = A
1065          JUMR( - 19)
1066          RCA(1), R                        " PAPER NOT TORN ?
1067     N,   S = :CLRMES
1068     N,   JUMR(3)                          " PAPER TORN
1069          LCA(2), R                        " PAPER NOT LOW ?
1070     Y,   S = :CLRMES[2]                   " YOKE OPEN
1071     N,   S = :CLRMES[1]                   " PAPER LOW
1072          SUBCD(:CLRRED)
1073          S = 0
1074          CLRLAB[2] = S                    " PREPARE HOK COMMAND
1075          SUBCD(:CVAETRIET)                " AND GIVE IT
1076          SUBCD(:UCLAIMCON)
1077          S = MC[ - 1]
1078          G = MS[0]                        " INVARIANT ADDRESS OF APPROPRIATE STRING
1079          SUBC(:CRRINTSTRING)
1080          SUBCD(:UCONEREE)                 " FREE CONSOLE
1081          S = - 1
1082          S '*' CFORMNUM                   " MAKE FORMNUMBER EVEN
1083          CFORMNUM = S                     " BY SUBTRACTING 1 IF IT WAS ODD
1084          S = CFEED, R
1085     Y,   CFEED = - B                      " FORMFEED :=EALSE
1086     Y,   GOTOR(MC[ - 1])                  " NORMAL EXIT
1087          B - 1                            " DESTROY LINK
1088          GOTO(:CNEWFORM)                  " REPEAT FORM

1089 CLRRED(12)                               " ADDITION TO MAKE LINEPRINTER RED AFTER ERROR ( 8 - 2 - '68)
1090          A = MC[ - 1]                     " UNSTACK LINK
1091          MC = S                           " ADDRESS OF INVARIANT ADRESS
1092     U,   S - :CLRMES[1], Z                " PAPER LOW ?
1093     Y,   GOTOR(A)                         " IN THIS CASE PRINTER WILL BE MADE RED BY HOK COMMAND
1094          S = '4000'[16]                   " BRUSHING OF LINE PRINTER
1095          M[216] = S                       " FILL ARD DENTIST
1096          '760 070 000'[38]                " AFDENTIST := IBUE
```

```
1097          '760 075 000'                    " IF WORD IN $
1098          LCS(2), R                        " IF FALSE ?
1099      Y,  JUMP( - 3)
1100          '660 071 000'[38]
1101          GOTOR(A)

1102                                                          " TEXTLR = 4
1103  CLRTXT(26)
1104          UTREXR
1105          SUBCD(:CLRSEL)                   " POINT OF CYCLIC RETURN
1106          UTREXV
1107          UROWNSEM                         " R(PRISEM)
1108          A = M3[4]                        " :SV TO BE UNCHAINED FIRST
1109          LCA(9), Z                        " MAKE INVARIANT ADDRESS, COND N
1110          SUBCD(:RSE39)                    " GET FIRST WORD OF THIS SEGMENT
1111          CRORTIONSIZE = G                 " FILL NUMBER OF SEGMENTS IN PORTION
1112          UTREXR
1113          G = CURHDL                       " FOR THE BENEFIT OF USGAE
1114          $ = 0                            " INITIALISE COUNTER
1115          MC[0] = $                        " COUNTER ON STACK
1116      U,  $ - CRORTIONSIZE, Z              " DONE WITH UNCHAINING ?
1117      N,  A = :CLRSV
1118      N,  A + $                            " ADDRESS FOR NEXT SV
1119      N,  SUBCD(:USGAE)
1120          $ = MC[ - 1]                     " UNSTACK COUNTER
1121              N,  $ + 1
1122      N,  JUMP( - 6)
1123          USUMP - $                        " DEC(SUMP, PORTIONSIZE)
1124          UFIO + $                         " INC(FIO, PORTIONSIZE)
1125          UFIOR + $                        " INC(FIOP, PORTIONSIZE)
1126          A = 1
1127          M3[0] - A                        " NPOR[STREAM] := NPOR[STREAM] - 1
1128          SUBCD(:UFIOFIOR)
1129          UTREXV                           " LEAVE CRITICAL SECTION

1130                                                          " TEXTLR = 5
1131  CNEWFORM(43)
1132          $ = CLFF, Z                      " TO FOLLOW CLRTXT
1133      Y,  $ = CPRHDL                       " LAST FORM NOT FINAL ?
1134      Y,  $ - CURHDL, Z                    " STREAM UNCHANGED ?
1135      Y,  JUMP(10)
1136          $ = CFORMNUM
1137          RCS(1), R                        " FORMNUMBER EVEN ?
1138      N,  SUBCD(:CFORMF)                   " EXTRA FORMFEED
1139      N,  SUBCD(:CLRSTART)
1140          A = :CXXXXX                      " ADDRESS IMAGE TEAR OFF LINE
1141          $ = 4                            " NO MORE THAN 15 X'S
1142          SUBCD(:CVULMAG)
1143          SUBCD(:CLRSTART)                 " PRINT TEAR OFF LINE
1144          $ =0
1145          CFORMNUM = $                     " FORMNUMBER := 0
1146          $ = 1
1147          A = :CLRCON2                     " CLRCON2 = 2 + 21
1148          SUBCD(:CVULMAG)
1149          SUBCD(:CLRSTART)                 " LINE FEED
1150          $ = :CLRSV
1151          CSVADR = $                       " ADDRESS OF SV OF SEGMENT TO BE CALLED FOR
1152          A = 0
1153          CSEGLW = A                       " POSITIONED JUST AHEAD OF NUMBER
1154          SUBCD(:SRQ)
```

linedprinter CM

```
1155          A = 1
1156          PLUSA(CSEGLW)
1157          $ = MS[0]
1158          A + MS[2]
1159          $ = MA[0], R
1160     N,   GOTO(:CENDOFFORM)
1161          $ + 0, Z
1162     Y,   $ = CSVADR
1163               Y,  SUBCD(:URROFANATION)
1164     Y,   $ = 1
1165     Y,   $ + CSVADR
1166     Y,   JUMR( - 16)
1167          A + 1
1168          CSEGLW + $
1169          SUBCD(:CVULMAG)
1170          $ = CSVADR
1171          SUBCD(:URROFANATION)
1172          SUBCD(:CLRSTART)
1173          $ = CSVADR
1174          JUMR( - 21)

1175
1176 CENDOFFORM(25)
1177          CLFF = - $
1178          $ = CURHDL
1179          CRRHDL = $
1180          $ = CSVADR
1181          SUBCD(:UKILLSEG)
1182          $ = 1
1183          MIN$(CSVADR)
1184     U,   $ - :CLRSV[ - 1], R
1185     Y,   JUMR( - 5)
1186          $ = 1
1187          CFORMNUM + $
1188          SUBCD(:CFORME)
1189          SUBCD(:CLRSTART)
1190          UTREXP
1191          A = 1
1192     U,   A - CLFF, E
1193     Y,   M3[1] - A
1194     Y,   UNFD - A
1195          $ = CFORMNUM
1196     Y,   JUMR(2)
1197     U,   A - M3[1], R
1198     Y,   $ = CONSEQ[ - 1], R
1199     Y,   A = 2
1200          UANVAR = A
1201          GOTO(:CLRTXT[1])

1202 USBLRA(20)
1203          !UIETLRA
1204          - 0
1205          0
1206          0
1207          0
1208          - 0
1209          0
1210          - 0
1211          - 0
1212          - 0
```

Cyclical process

```
" POSITIONED AT NUMBER
" $ = :CTE
" A + CRB? , I.E. MAKE FYSICAL ADDRESS
" $ := NUMBER, NOT END OF FORM ?

" CONTINUATION ON NEXT SEGMENT ?

        " ONTHEILIG


" FYSICAL ADDRESS OF FIRST WORD TO BE TRANSPORTED
" POSITIONED AT LAST WORD


" PRINT ONE LINE OF OUTPUT

                                                  " TEXTLR - 7
" + 0, IF NOT FINAL; > + 0, IF FINAL
" NOTE DOWN  STREAM OF PREVIOUS FORM
" KILL FINISHED SEGMENTS

" MORE SEGMENTS TO BE KILLED ?


" STEP TILL TOP OF NEXT FORM

" ENTER CRITICAL SECTION
" TO GET Y IF LAST FORM FINAL
" Y, IF LAST FORM FINAL
" NFD := NFD - 1
" TOTAL NUMBER OF FINAL DOCUMENTS DECREASED


" NFD ≤ 0 ?
" NUMBER OF COSECUTIVE FORMS > BATCH SIZE

" PRIVAR := 1 OR 2
```

Stuck bottom of line printer.
Is initialized this way on reading in the system.

```
" CURHDL
" CLRLAB(3) (IRRELEVANT)


" CRRHDL
" CFORMNUM
" CLRSV(5)
```

After a failure of some sort the system can only be reinitialized by once more reading it in from tape (This is a very general remark)
However, we naturally have a bagful of tricks by which we can restart the system after a failure, which was useful in the testing fase.

```
1213          - 0
1214          - 0
1215          1                                  " CLEE
1216          - 0                                " CEEED
1217          CRARERTORN                         " CLRMES(5)
1218          CRARERLOV
1219          CYOKEOREN
1220          CRARITY
1221          CNBKMES
1222          :CLRAMAGI - 1]                     " CLRBUE

1223 CLRCON1(6)
1224          '003 777 777'
1225          '010 000 000'                      " CLRCON2
1226          '003 416 070'                      " CXXXXX(4)
1227          '703 416 070'
1228          '703 416 070'
1229          '703 416 070'
1230 UIRTLRA(10)
1231          :UARLRA
1232          - 0
1233          - 0
1234          '760 071 020'                      " IE := IBUE
1235          '660 071 020'                      " IE := EALSE
1236          '760 072 020'                      " LVIE := IBUE
1237          '660 072 020'                      " LVIE := EALSE
1238          '760 070 020'                      " AE := IBUE
1239          2                                  " 1 IN FLIPFLOPWORD POSITION
1240          0

1241 USUMR(3)
1242          - 0
1243          UINITIALERAS                       " UERAS
1244          - 0                                " UNED
1245 CLROUT(9)
1246      Y,  S = :CLRMES[4]
1247      N,  S = :CLRMES[3]
1248          MC = S                             " SAVE ADDRESS OF INVARIANT ADDRESS DISASTER MESSAGE
1249          SUBCD(:UCLAIMCON)
1250          S = MC[ - 1]
1251          G = MS
1252          SUBCD(:CPRINTSTRING)               " PRINTER DISORDER OR PRINTER PARITY
1253          SUBCD(:UCONEREE)
1254          JUMP(- 1)
```

```
 0 CORSYM(5)
 1          MIB - 51 = S, R
 2      Y,  GOTOR(MC[ - 1])
 3          S = 1
 4          RLUSS(M[B - 5])
 5          GOTOR(MC[ - 1])
 6 CTABVOL(41)
 7          512 487                          " GOIO          0
 8          18 918                 " EOR                1
 9          530 067                          " SIER          2
10          12 898 773                       " UNIIL         3
11          5 645 591                        " WHILE         4
12          484                              " DO            5
13          - 20 433 436                     " COMBL         6
14          773                              " EX
15          201                              " IE            8
16          464 148                          " IHEN          9
17          183 685                          " ELSE          10
18          - 27 871 772                     " COMME         11
19          654                              " NI
20          14 982 306                       " BEGIN         13
21          4 549                            " END           14
22          15 087                           " OWN           15
23          394 418                          " REAL          16
24          - 26 029 622                     " INIEG         17
25          581                              " EB
26          - 27 902 493                     " BOOLE         18
27          449                              " AN
28          - 18 560 364                     " SIRIN         21
29          7                                " G
30          26 266 177                       " ARRAY         23
31          - 28 197 295                     " PROCE         24
32          182 948                          " DURE
33          - 29 743 372                     " SWITC         26
34          8                                " H
35          12 748 844                       " LABEL         28
36          5 943 350                        " VALUE         29
37          185 940                          " IRUE          30
38          5 877 798                        " FALSE
39          - 28 066 223                     " PROCE         32
40          142                              " ND
41          '606 204 454'                    " LIBRA
42          '1462'                           " RY
43          593 294                          " NLCR          36
44          '26 003 215'                     " MIARE         37
45          - 20 325 942                     " INVSL         38
46                 8 801                                     " ASH
47          13 081 985                       " ALGOL         40

48 CTABINTLR(23)
49          ' 015 013 002'                   " -     +       ¢
50          '332 017 012'                    " ¡     /       *
51          '506 035 520'                    " ‡     =       ↑
52          '074 310 073'                    " >     ≤       <
53          '306 077 312'                    " ≡     ¬       ≥
54          '076 075 314'                    " ^     ∨       ⊃
55          '102 101 100'                    " SIER  FOR     GOIO
56          '105 104 103'                    " DO    WHILE   UNIIL
57          '006 016 014'                    " ¤     .       ,
```

tape reader
and tape interpretation
(making basic symbols out of tape)

conversion tables

```
58          '106 004 003'                    " COMBL ;       :
59          '111 110 000'                    " IHEN  IE      SPACE
60          '010 113 112'                    " (     COMME   ELSE
61          '033 032 011'                    " ]     [       )
62          '115 512 510'                    " BEGIN +       ¢
63          '120 117 116'                    " REAL  OWN     END
64          '125 123 121'                    " SIBIN BOOLE   INIEG
65          '132 130 127'                    " SWITC PROCE   ARRAY
66          '136 135 134'                    " IRUE  VALUE   LABEL
67          '100 140 137'                    " (NLCR)PROGE   EALSE
68          '005 007 037'                    " ?     "       '
69          '001 142 145'                    " INVSL LIBRA   MIAPE
70          '144 034 040'                    " NLCR  |       ~
71          '000 000 327'                    "               ←

72 CNUMLRTAB(5)                              "       VALUE
73          11                               " +     10
74          13                               " -     11
75          14                               " .     12
76          6                                " ₁₀    13
77          0                                " SPACE 14
78 CAANLR(31)
79          SUBCD(:U.MMORTAL)
80          A = UBLOCKSTR
81          S = MA[21]
82          MC = G
83          MC = S
84          SUBCD(:RSE50)
85          A = UBLOCKSTR
86          S = MA[26]
87          S = :MA[15]
88          UTREXPROOD
89          SUBCD(:UELOBARRIER)
90          USUMR + S
91          S = :MA[16]
92          MC = S
93          SUBCD(:USGAAN)
94          S = 1
95          S + MC[ - 1]
96          A = UBLOCKSTR
97      U,  S = MA[26], R
98      N,  JUMP( - 7)
99          S = 1
100         MA[0] + S
101         G = MC[ - 1]. Z
102     N,  MA[1] + S
103     N,  UNED + S
104         SUBCD(:CLRSEL)
105         UTREXVWIT
106         G = UBLOCKSTR
107     Y,  MG[24] = - B
108         SUBCD(:UMORTAL)
109         GOTOR(MC[ - 1])

110 CTENRO(10)                               " 10 ↑  M * 2 ↑(26 - N)
111         41 943 040                       "       1           4
112         52 428 800                       "       2           7
113         65 536 000                       "       3          10
114         40 960 000                       "       4          14
115         51 200 000                       "       5          17
```

```
116          64 000 000                    "       6          20
117          40 000 000                    "       7          24
118          50 000 000                    "       8          27
119          62 500 000                    "       9          30
120          39 062 500                    "      10          34
121 CSCALE(10)
122          4
123          7
124          10
125          14
126          17
127          20
128          24
129          27
130          30
131          34
132 CTAB9(3)
133          9
134          99
135          999

136 UKILLSEGR(7)                  kill a "private" segment
137          A = MS[0], Z                  " SV EMPTY?
138      Y,  GOTOR(MC - 1)
139          A = 1
140          GL24 - A                      " DEC(P)
141          UFIOR + A                     " INC(FIOP)
142          UFIR + A                      " INC(FIP)
143          GOTO(:UKILLSEG)
144
145 COCT(53)          read octad from tape, basic subroutine      " BANDLEESROUTINES
146          G = UBLOCKSTR                 " ADRESS OF HANDLE
147          S = MG[12], Z                 " VOORGIFT
148      Y,  A = - S, R
149      N,  MG[12] - S                    " VOORGIFT := - 0, I.E. EMPTY
150      N,  GOTOR(MC - 1)                 " DELIVER VOORGIFT IN S
151          A = MG[8], Z                  " PRIVATE SV INDICATING EMPTY ?
152      N,  JUMR(21)
153          A = MG[7], R                  " STREAM STILL ALIVE ?
154      N,  JUMR(42)                      " TO DELIVER - 1
155          SUBCD(:UIMMORTAL)
156          UR                            " R(STREAMSEM)
157          UTREXRROOD
158          G = UBLOCKSTR                 " RESTORE HANDLE ADDRESS
159          A = :MG[8]                    " : PRIVATE SV
160          SUBCD(:USGAR)
161          S = 1
162          GL24 + S                      " INCREASE NUMBER OF P SEGMENTS
163          UFI + S
164          UFIQ + S
165          SUBCD(:UFIQFIOR)
166          UTREXVWIT
167          G = UBLOCKSTR
168          S = 0
169          A = :MG[8]                    " :SV
170          LCA(9)
171          MG[17] = A                    " RUNNING INVARIANT ADDRESS
172          MG[19] = - S                  " INITIALISE SHIFT INDICATOR
173          SUBCD(:UMORTAL)
```

for bin.-dec. conversion

```
174          $ = MG[19], Z                    " WORD SHIFTED OUT ?
175      N,  JUMP(8)
176          A = MG[17], Z                    " CONDITION N FOR RSE39
177          SUBCD(:RSE39)                    " FETCH WORD FROM STREAM
178          $ = G
179          G = UBLOCKSTR
180          A = 3
181          MG[19] = A                       " INITIALISE SHIFT INDICATOR FOR NEW WORD
182          A = 1, R                         " TO SKIP NEXTNEXT INSTRUCTION
183          MG[17] + A                       " INCREASE INVARIANT ADDRESS
184      N,  $ = MG[18]                       " WORD BEING UNPACKED
185          LCS(9)                           " SHIFT OCTAD INTO TAIL POSITION
186          MG[18] = $                       " STORE SHIFTED WORD
187          $ '*' 511
188      U,  $ - 509, R                       " END OF TAPE OR END OF SEGMENT ?
189          A = 1
190      N,  MG[19] - A                       " DECREASE SHIFT INDICATOR
191      N,  GOTOR(MC[ - 1])
192          $ - 510, Z                       " END OF TAPE ?
193          $ = :MG[8]                       " : PRIVATE SV
194          SUBCD(:UKILLSEGR)
195      N,  JUMP( - 41)                      " TO GET HEPTAD FROM NEXT SEGMENT
196          SUBCD(:ORSMUK1[13])
197          $ = - 1                          " DELIVER EOT
198          GOTOR(MC[ - 1])

199 UDATE(1)
200          + 10411
201 UCURRUNNUM(1)
202          + 1
203 CMNN(1)
204          CINVMNN
205 UTRSTRL1(UNUMTRRMSTR)
206          :UTRHANDLE11
207          :UTRHANDLE12
208 UTRSTRL2(UNUMTRRMSTR)
209          :UTRHANDLE21
210          :UTRHANDLE22
211 UTRSTRL3(UNUMTRRMSTR)
212          :UTRHANDLE31
213          :UTRHANDLE32
214 UTRSTRL4(UNUMTRRMSTR)
215          :UTRHANDLE41
216          :UTRHANDLE42
217 UTRSTRL5(UNUMTRRMSTR)
218          :UTRHANDLE51
219          :UTRHANDLE52
220 ULRSTRL1(UNUMLRRMSTR)
221          :ULRHANDLE11
222 ULRSTRL2(UNUMLRRMSTR)
223          :ULRHANDLE21
224 ULRSTRL3(UNUMLRRMSTR)
225          :ULRHANDLE31
226 ULRSTRL4(UNUMLRRMSTR)
227          :ULRHANDLE41
228 ULRSTRL5(UNUMLRRMSTR)
229          :ULRHANDLE51
230 UTRSTRL1(UNUMTRRMSTR)
231          :UTRHANDLE11
232          :UTRHANDLE12
```

lists with streamhandles.

```
233 UTRSTRL2(UNUMTRRMSTR)
234          :UTRHANDLE21
235          :UTRHANDLE22
236 UTRSTRL3(UNUMTRRMSTR)
237          :UTRHANDLE31
238          :UTRHANDLE32
239 UTRSTRL4(UNUMTRRMSTR)
240          :UTRHANDLE41
241          :UTRHANDLE42
242 UTRSTRL5(UNUMTRRMSTR)
243          :UTRHANDLE51
244          :UTRHANDLE52
245 URLSTRL1(UNUMRLRMSTR)
246          :URLHANDLE11
247 URLSTRL2(UNUMRLRMSTR)
248          :URLHANDLE21
249 URLSTRL3(UNUMRLRMSTR)
250          :URLHANDLE31
251 URLSTRL4(UNUMRLRMSTR)
252          :URLHANDLE41
253 URLSTRL5(UNUMRLRMSTR)
254          :URLHANDLE51
255 URENREQ(1)
256          - 0
257 ULISTTR(UNUMTR[2])
258          :ULISTTR[2]
259          UNUMTR
260          :USBTRA
261          :USBTRB
262          :USBTRC
263 UIETTRA(10)
264          :UARTRA
265          - 0
266          - 0
267          '76 00 71 000'[1]
268          '66 00 71 000'[1]
269          '76 00 72 000'[1]
270          '66 00 72 000'[1]
271          '76 00 70 000'[1]
272          '00 02 00 000'
273          0
274 UIETTRB(10)
275          :UARTRB
276          - 0
277          - 0
278          '76 00 71 000'[3]
279          '66 00 71 000'[3]
280          '76 00 72 000'[3]
281          '66 00 72 000'[3]
282          '76 00 70 000'[3]
283          '00 00 40 000'
284          0
285 UIETTRC(10)
286          :UARTRC
287          -0
288          -0
289          '76 00 71 000'[18]
290          '66 00 71 000'[18]
291          '76 00 72 000'[18]
292          '66 00 72 000'[18]
```

→ this longwinded constant happens to have value 1

layout of three tape reader hardware semaphores

the first three elements of a hardware semaphore are identical in nature to ~~the~~ f. all three of a software semaphore

```
293           '76 00 70 000'[18]
294           0
295           '20 00 00 000'
296 USCHAKELTRA(6)
297           - 0
298           :USCHAKELTRA[4]
299           :UVIGA[UTRLENTEL - 1]
300           - 0
301           :USCHAKELTRA[1]
302           :UVAGA[UTRLENTEL - 1]
303 USCHAKELTRB(6)
304           - 0
305           :USCHAKELTRB[4]
306           :UVIGB[UTRLENTEL - 1]
307           - 0
308           :USCHAKELTRB[1]
309           :UVAGB[UTRLENTEL - 1]
310 USCHAKELTRC(6)
311           -0
312           :USCHAKELTRC[4]
313           :UVIGC[UTRLENTEL - 1]
314           -0
315           :USCHAKELTRC[1]
316           :UVAGC[UTRLENTEL - 1]
317 CMNA13(1)
318           CINVM13
319 CMNA9(1)
320           CINVM9
321 CMNA7(1)
322           CINVM7
323 CMNA6(1)
324           CINVM6

325 UTRHANDLE11(15)
326           - 0                    " STREAMSEMAPHORE
327           - 0
328           - 0
329           - 0                    " AANHAAKLABEL  attaching label
330           - 0                    " AFHAAKLABEL   detaching label
331           :USBRM1['1 000 000']
332           + 0                    " INTEREST
333           - 0                    " IF STREAM ALIVE THEN :USBTRA/B
334           - 0                    " SV FOR RM SEGMENT
335           + 1                    " TAPE NUMBER
336           - 0                    " BASINT
337           + 0                    " NUMINT
338           - 0                    " VOOR OCT
339           - 0                    " VOOR CHARF
340           - 0                    " VOOR SYM

341 UTRHANDLE12(15)
342           - 0                    " STREAMSEMAPHORE
343           - 0
344           - 0
345           - 0                    " AANHAAKLABEL
346           - 0                    " AFHAAKLABEL
347           :USBRM1['2 000 000']
348           + 0                    " INTEREST
349           - 0                    " IF STREAM ALIVE THEN :USBTRA/B
350           - 0                    " SV FOR RM SEGMENT
```

own variables of tapereader streams

segments in i/o streams form chains in one big buffer, called transput area

```
351          + 1                                " TAPE NUMBER
352          - 0                                " BASINT
353          + 0                                " NUMINT
354          - 0                                " VOOR OCT
355          - 0                                " VOOR CHARE
356          - 0                                " VOOR SYM

357 UTRHANDLE21(15)
358          - 0                                " STREAMSEMAPHORE
359          - 0
360          - 0
361          - 0                                " AANHAAKLABEL
362          - 0                                " AFHAAKLABEL
363          :USBRM2['1 000 000']
364          + 0                                " INTEREST
365          - 0                                " IF STREAM ALIVE THEN :USBTRA/B
366          - 0                                " SV FOR RM SEGMENT
367          + 1                                " TAPE NUMBER
368          - 0                                " BASINT
369          + 0                                " NUMINT
370          - 0                                " VOOR OCT
371          - 0                                " VOOR CHARE
372          - 0                                " VOOR SYM

373 UTRHANDLE22(15)
374          - 0                                " STREAMSEMAPHORE
375          - 0
376          - 0
377          - 0                                " AANHAAKLABEL
378          - 0                                " AFHAAKLABEL
379          :USBRM2['2 000 000']
380          + 0                                " INTEREST
381          - 0                                " IF STREAM ALIVE THEN :USBTRA/B
382          - 0                                " SV FOR RM SEGMENT
383          + 1                                " TAPE NUMBER
384          - 0                                " BASINT
385          + 0                                " NUMINT
386          - 0                                " VOOR OCT
387          - 0                                " VOOR CHARE
388          - 0                                " VOOR SYM

389 UTRHANDLE31(15)
390          - 0                                " STREAMSEMAPHORE
391          - 0
392          - 0
393          - 0                                " AANHAAKLABEL
394          - 0                                " AFHAAKLABEL
395          :USBRM3['1 000 000']
396          + 0                                " INTEREST
397          - 0                                " IF STREAM ALIVE THEN :USBTRA/B
398          - 0                                " SV FOR RM SEGMENT
399          + 1                                " TAPE NUMBER
400          - 0                                " BASINT
401          + 0                                " NUMINT
402          - 0                                " VOOR OCT
403          - 0                                " VOOR CHARE
404          - 0                                " VOOR SYM

405 UTRHANDLE32(15)
406          - 0                                " STREAMSEMAPHORE
```

```
407          - 0
408          - 0
409          - 0                                " AANHAAKLABEL
410          - 0                                " AFHAAKLABEL
411          :USBRM3['2 000 000']
412          + 0                                " INTEREST
413          - 0                                " IF STREAM ALIVE THEN :USBTRA/B
414          - 0                                " SV FOR RM SEGMENT
415          + 1                                " TAPE NUMBER
416          - 0                                " BASINT
417          + 0                                " NUMINT
418          - 0                                " VOOR OCT
419          - 0                                " VOOR CHARE
420          - 0                                " VOOR SYM

421 UTRHANDLE41(15)
422          - 0                                " STREAMSEMAPHORE
423          - 0
424          - 0
425          - 0                                " AANHAAKLABEL
426          - 0                                " AFHAAKLABEL
427          :USBRM4['1 000 000']
428          + 0                                " INTEREST
429          - 0                                " IF STREAM ALIVE THEN :USBTRA/B
430          - 0                                " SV FOR RM SEGMENT
431          + 1                                " TAPE NUMBER
432          - 0                                " BASINT
433          + 0                                " NUMINT
434          - 0                                " VOOR OCT
435          - 0                                " VOOR CHARE
436          - 0                                " VOOR SYM

437 UTRHANDLE42(15)
438          - 0                                " STREAMSEMAPHORE
439          - 0
440          - 0
441          - 0                                " AANHAAKLABEL
442          - 0                                " AFHAAKLABEL
443          :USBRM4['2 000 000']
444          + 0                                " INTEREST
445          - 0                                " IF STREAM ALIVE THEN :USBTRA/B
446          - 0                                " SV FOR RM SEGMENT
447          + 1                                " TAPE NUMBER
448          - 0                                " BASINT
449          + 0                                " NUMINT
450          - 0                                " VOOR OCT
451          - 0                                " VOOR CHARE
452          - 0                                " VOOR SYM
453 UTRHANDLE51(15)
454          -0                                 "STREAMSEMAPHORE
455          -0
456          -0
457          -0                                 "AANHAAKLABEL
458          -0                                 "AFHAAKLABEL
459          :USBRM5['1 000 000']
460          +0                                 "INTEREST
461          -0                                 "IF STREAMALIVE THEN :USBTBC
462          -0                                 "SV FOR RM SEGMENT
463          +1                                 "TAPE NUMBER
464          -0                                 "BASINT
```

```
465          +0                                  "NUMINT
466          -0                                  "VOOR OCT
467          -0                                  "VOOR CHARE
468          -0                                  "VOOR SYM
469 UTRHANDLE52(15)
470          -0                                  "STREAMSEMPHORE
471          -0
472          -0
473          -0                                  "AANHAAKLABEL
474          -0                                  "AFHAAKLABEL
475          :USBRM5['2 000 000']
476          +0                                  "INTEREST
477          -0                                  "IF STREAM ALIVE THEN :USBTRC
478          -0                                  "SV FOR RM SEGMENT
479          +1                                  "TAPE NUMBER
480          -0                                  "BASINT
481          +0                                  "NUMINT
482          -0                                  "VOOR OCT
483          -0                                  "VOOR CHARE
484          -0                                  "VOOR SYM

485                                              " TEXTTR
486                                              " UWAR = 0
487 UWAR(12)
488          S = 1
489          SUBCD(:UEXRMTRVAR)                  " EXIST IDLE RM?
490     N,   GOTOR(MC[- 1])
491          MC[0] = G                           " STACK :USB CHOSEN RM
492          S = 3
493          SUBCD(:UEXTRVAR)                    " EXIST IDLE GEHAPTE TAPEREADER?
494          S = MC[- 1]                         " S = :USB CHOSEN RM
495     N,   GOTOR(MC[- 1])
496          A = - 0                             " G = :USB CHOSEN TAPEREADER
497          UAMVAR[US] = A                      " PMTRVAR := 0
498          A = 9
499          UAMVAR[UG] = A, P                   " TRVAR := 9, SET YES CONDITION
500 UWAR1(7)                                     " ENTRY ADDRESS UNONWAR
501          A = UBLOCKSTR[US]                   " A = :UTRHANDLE BLOCKING STREAM
502          UCURSTR[UG] = A                     " SET UCURSTR
503          MA[7] = G                           " SET COUPLED TAPEREADER AM
504     Y,   E = :UAMSEM[UG]                     " ADDRESS TAPEREADER SEMAPHORE
505     Y,   SUBCD(:UVOR[2])                     " UV
506          E = :UAMSEM[US]                     " ADDRESS RM SEMAPHORE
507          GOTO(:UVOR[2])                      " V(RMSEM) AND RETURN

508                                              " TEXTTR
509                                              " UEXRMTRVAR
510                                              " UEXTRVAR
511 UEXRMTRVAR(3)
512          E = :ULISTRM
513          A = :UAMVAR
514          GOTO(:UEXREAM)

515 UEXTRVAR(2)
516          E = :ULISTTR
517          GOTO(:UEXRMTRVAR[1])

518                                              " TEXTTR
519                                              " UNONWAR = 0
520 UNONWAR(21)
```

text of tapereader cyclical process

```
521          S = 5
522          SUBCD(:UEXRMTRVAR)              " RM RESERVED FOR URGENT READING?
523          S = 2
524      N,  SUBCD(:UEXRMTRVAR)              " RM ASKING NON STANDARDTAPE?
525      N,  GOTOR(MC[-1])
526          MC[0] = G                       " STACK :USB CHOSEN RM
527          SUBCD(:UEXTRVAR)                " IDLE NONBITTEN TAPEREADER?
528      N,  S = 3
529      N,  SUBCD(:UEXTRVAR)                " IDLE BITTEN TAPEREADER?
530          S = MC[-1]                      " S = :USB CHOSEN RM
531      N,  GOTOR(MC[-1])
532          A = UAMVAR[US]                  " A := RMTRVAR, G = :USB TAPEREADER
533      U,  A - 5, Z                        " RMTRVAR = 5?
534      Y,  A = 6
535      Y,  UAMVAR[US] = A                  " RMTRVAR := 6
536      Y,  UAMVAR[UG] + A                  " TRVAR := TRVAR + 6
537      N,  UAMVAR[US] - A                  " RMTRVAR := 0
538      N,  UAMVAR[UG] + A                  " TRVAR := TRVAR + 2
539          A = 1, Z                        " MAKE NO CONDITION
540          URENREQ - A                     " DECREASE NUMBER OF PENDING REQUESTS
541          GOTO(:UWAR1)                    " MERGE WITH UWAR

542                                                                  " TEXTTR
543                                                                  " UTRACT - 1
544 UTRACT(26)
545          A = UNUMTR                      "NUMBER OF TAPE READERS
546          UWORK2 = A, R                   "INITIALISATION
547      Y,  S = 10
548      Y,  SUBCD(:UEXTRVAR)                "EXIST TRVAR = 10?
549      N,  GOTOR(MC[-1])
550          S = USPECIALTR[UG], R           "IS THIS A SPECIAL READER ?
551      N,  JUMP(4)
552          MC = G
553          SUBCD(:UEXRMTRVAR)
554          G = MC[-1]
555          JUMP(1)
556          S = URENREQ, R                  "PENDING REQUEST FOR NORMAL READER?
557          S = - 0
558      N,  A = UCURSTR[UG]                 ":UTRHANDLE
559      N,  S - MA[0]                       "- LENGHT STREAM
560      N,  LUS(1)                          "- 2 * LENGHT STREAM
561      U,  S + UEIO, R                     "2 * LENGHT < FIO
562      Y,  S + UEIOR, R                    "2 * LENGHT < FIOP
563          S = 1
564      Y,  UAMVAR[UG] - S                  "TRVAR := 9
565      Y,  UEIO - S                        "DEC(FIO)
566      Y,  UEIOR - S                       "DEC(FIOP)
567      Y,  R = :UAMSEM[UG]
568      Y,  SUBCD(:UVOR[2])                 "V(TRSEM)
569          UWORK2 - S, R                   "TRY AGAIN
570          JUMP(-24)

571                                                                  " TEXTTR
572                                                                  " UDISCONNECT
573 UDISCONNECT(7)
574          S=UAMVAR[UG]                    " S = TRVAR
575      U,  S - 2, Z                        " TRVAR = 2 v 3?
576      N,  S - 3, Z
577      Y,  S = UDISC[UG], R                " DISC = TRUE?
578      Y,  S = 16
```

```
579      Y,  UAMVAR[UG] + S                  " TRVAR := TRVAR + 16
580          GOTO(MC[- 1])

581                                                                  " TEXTTR
582                                                                  " UTEXTAFT - 0
583  UTEXTAFT(11)
584          A = UCON1                       " 2 + 18
585          RLUSA(M1[2])                    " AFT := AFT + 1
586          RUA(19), Z                      " AFT = 1?
587      Y,  A = M1[0]                       " OLD AR0
588      Y,  A '*' UCON2                     " ISOLATE ADDRESS PART
589      Y,  A - S                           " OLD ADDRESS PART - NEW ADDRESS PART
590      Y,  M1[0] - A                       " NEW ADDRESS PART IN AR0
591      Y,  DO(M2[7])                       " AF := TRUE
592          S = MS[0]
593          ULALAB = S
594          GOTOR(MC[- 1])

595                                                                  " TEXTTR
596                                                                  " UTEXTTR - 0
597  UTEXTTR(36)
598          UTREXR
599          S = USRECIALTR, R               "IS THIS A SPECIAL READER?
600      N,  S = 2
601          UAMVAR = S                      " TRVAR := 2 OR 99
602          F = :M0[0]
603          SUBCD(:UDISCONNECT)
604      Y,  SUBCD(:UMES12)
605          SUBCD(:UNONHAR)
606          UTREXV
607          A = :UHOKHAR[1]
608          SUBCD(:CVAETRIFT[1])            " HOK COMMAND
609          A = :UHOKHAR[4]
610          SUBCD(:CVAETRIFT[1])            " HAR COMMAND
611          UTREXR
612          S = UAMVAR
613          S - 18, Z
614      N,  S = UHOKHAR[3]
615      N,  LCS(8), R                       " NOT NBK?
616      N,  UDISC = 8                       " DISC := TRUE
617          F = :M0[0]
618          SUBCD(:UDISCONNECT)
619      Y,  SUBCD(:UMES12)
620          S = 1
621          RLUSS(UAMVAR)
622      U,  S = USRECIALTR, R
623      Y,  SUBCD(:UREPM)
624      N,  SUBCD(:UHAR)
625          UTREXV
626          UROWNSEM                        " R(TRSEM)
627          UTREXR
628          S = UAMVAR
629          S - 20, Z                       " TRVAR = 20?
630      Y,  GOTO(:UTEXTTR[1])                " IN CIRCULATION AGAIN
631          UTREXV
632          S = ULALAB
633          SUBCD(:UTEXTAFT)                " V(AFT)

634                                                                  " TEXTTR
635                                                                  " UTEXTTR - 1
```

```
636 UTEXTTR1(10)
637         UTREXR
638         S = UFI, Z
639     N,  S = UFIP, Z
640     Y,  GOTO(:ONZINT4)                  " DISASTER
641         S = 1
642         UFI - S                         " DEC(FI)
643         UFIP - S                        " DEC(FIP)
644         UAMVAR + S                      " TRVAR := 10
645         SUBCD(:UTRACT)                  " TAPE READER ACTIVATION
646         UTREXV
647 UTEXTTR2(74)
648         S = ULALAB
649         SUBCD(:UTEXTART)                " V(ART)
650         F = :M2[0]
651         URH                             " R(IRT)
652         S = :UTRSEG
653         SUBCD(:UHOLYCMSEG)
654         S = M4[1]                       " CODEWORD
655         S '*' UCON2                     " ADDRESSPART
656         G = S
657         A = UHOKHAR[3], R               " HAP COMMAND OK?
658     Y,  A = M3[6], R                    " INTEREST?
659     N,  JUMP(6)
660         A = M4[- 1], R                  " CURRENT COMMAND OK?
661     Y,  S + UTRLEN[- 1]                 " ADDRESS LAST READ HEPTADE - 1
662     Y,  JUMP(7)
663         A '*' UCON1, Z                  " END OF TAPE?
664     Y,  S = M1[3]                       " AR3
665     Y,  S '*' UCON2                     " ADDRESS LAST READ HEPTADE
666         A = - 510
667         MS[1] = - A                     " FILL END OF TAPE
668         MS[2] = - A
669         MS[3] = - A

670                                                                 " TEXTTR
671                                                                 " UTEXTTR - 2
672         UTRSLOT = A                     " IF END OF TAPE IN STREAM NEGATIV ELSE POSITIV
673         B = M5[2]                       " :GRB PRIVATE SEGMENT
674         B + UNUMWSEG                    " FIRST FREE PLACE IN PRIVATE SEGMENT
675         A = MG[1]
676         LUA(9)
677         A + MG[2]
678         LCA(9)
679         A + MG[3]                       " THREE HEPTADES PACKED
680         MC[0] = A
681         F + 3
682     U,  S - :MG[- 1], R
683     Y,  JUMP(- 9)                       " NEXT THREE HEPTADES
684         A = - UTRSLOT, R                " END OF TAPE IN STREAM?
685         A = - 0
686     N,  M[B] = A                        " CLOSING KARAKTER
687     N,  B - M5[2]                       " NUMBER OF WORDS FILLED IN SEGMENT
688     N,  UNUMWSEG = B
689     N,  B - UTRMAXLEN, R                " SEGMENT FILLED?
690         B = :M0[UTRLOC]                 " RESET B
691         S = :UTRSEG                     " ADDRESS SV PRIVATE SEGMENT
692         SUBCD(:URROFANATION)            " PROFANATION
693     N,  GOTO(:UTEXTTR2)                 " FILL IN SAME SEGMENT
694         UNUMWSEG = A                    " INITIALISATION NUMBER OF WORDS IN NEXT SEGMENT
```

```
695          UROWNSEM
696          UTREXR
697          A = MD[3]
698          SUBCD(:USGAAN)                  " CHAIN ON TO STREAM
699          R = :M3[0]
700          UV                              " V(STRSEM)
701          A = - UTRSLOT, R                " END OF TAPE IN STREAM
702      N,  GOTO(:UTEXTTR1[1])               " FILL NEW SEGMENT
703          UTREXV
704          R = :M2[0]
705          URH                             " R(IET)
706          A = M1[0], R                    " AR0, OK?

707                                                                  " TEXTTR
708                                                                  " UTEXTTR - 3
709      N,  JUMP(12)
710          S = - USRECIALTR, R             "NORMAL READER?
711      Y,  SUBCD(:UMES14)                  "MESSAGE M14 ETC,
712      Y,  SUBCD(:UROWNSEMADRES)           "R(TRSEM)
713          SUBCD(:UTREXRADRES)
714      Y,  S = UAMVAR
715      Y,  S - 22, Z                       "TRVAR = 22?
716      Y,  GOTO(:UTEXTTR[1])
717          UTREXV
718          A = ULALAB
719          SUBCD(:CVARTRIET[1])
720          A = M1[0], R                    " AR0 OK?
721      Y,  JUMP(- 4)
722          A '*' UCON1, Z                  " END OF TAPE?
723          UTREXR
724      N,  UDISC = B                               " DISC := TRUE
725          GOTO(:UTEXTTR[1])

726 UTRUTFREE(1)
727          :UTRUTADR
728 UTRUTLIST(86)
729          0
730          0
731          0
732          0
733          0
734          0
735          0
736          0
737          0
738          0
739          0
740          0
741          0
742          0
743          0
744          0
745          0
746          0
747          0
748          0
749          0
750          0
751          0
752          0
```

first element in chain of free transput area buffer space

chaining information kept in core to administrate transput area

753	0
754	0
755	0
756	0
757	0
758	0
759	0
760	0
761	0
762	0
763	0
764	0
765	0
766	0
767	0
768	0
769	0
770	0
771	0
772	0
773	0
774	0
775	0
776	0
777	0
778	0
779	0
780	0
781	0
782	0
783	0
784	0
785	0
786	0
787	0
788	0
789	0
790	0
791	0
792	0
793	0
794	0
795	0
796	0
797	0
798	0
799	0
800	0
801	0
802	0
803	0
804	0
805	0
806	0
807	0
808	0
809	0
810	0
811	0
812	0

```
813             0
814             0
815 UTRUTADR(255)
816             :UTRUTADR[1]
817             :UTRUTADR[2]
818             :UTRUTADR[3]
819             :UTRUTADR[4]
820             :UTRUTADR[5]
821             :UTRUTADR[6]
822             :UTRUTADR[7]
823             :UTRUTADR[8]
824             :UTRUTADR[9]
825             :UTRUTADR[10]
826             :UTRUTADR[11]
827             :UTRUTADR[12]
828             :UTRUTADR[13]
829             :UTRUTADR[14]
830             :UTRUTADR[15]
831             :UTRUTADR[16]
832             :UTRUTADR[17]
833             :UTRUTADR[18]
834             :UTRUTADR[19]
835             :UTRUTADR[20]
836             :UTRUTADR[21]
837             :UTRUTADR[22]
838             :UTRUTADR[23]
839             :UTRUTADR[24]
840             :UTRUTADR[25]
841             :UTRUTADR[26]
842             :UTRUTADR[27]
843             :UTRUTADR[28]
844             :UTRUTADR[29]
845             :UTRUTADR[30]
846             :UTRUTADR[31]
847             :UTRUTADR[32]
848             :UTRUTADR[33]
849             :UTRUTADR[34]
850             :UTRUTADR[35]
851             :UTRUTADR[36]
852             :UTRUTADR[37]
853             :UTRUTADR[38]
854             :UTRUTADR[39]
855             :UTRUTADR[40]
856             :UTRUTADR[41]
857             :UTRUTADR[42]
858             :UTRUTADR[43]
859             :UTRUTADR[44]
860             :UTRUTADR[45]
861             :UTRUTADR[46]
862             :UTRUTADR[47]
863             :UTRUTADR[48]
864             :UTRUTADR[49]
865             :UTRUTADR[50]
866             :UTRUTADR[51]
867             :UTRUTADR[52]
868             :UTRUTADR[53]
869             :UTRUTADR[54]
870             :UTRUTADR[55]
871             :UTRUTADR[56]
872             :UTRUTADR[57]
```

initial of all transputarea being free.

```
873         :UTRUTADR[58]
874         :UTRUTADR[59]
875         :UTRUTADR[60]
876         :UTRUTADR[61]
877         :UTRUTADR[62]
878         :UTRUTADR[63]
879         :UTRUTADR[64]
880         :UTRUTADR[65]
881         :UTRUTADR[66]
882         :UTRUTADR[67]
883         :UTRUTADR[68]
884         :UTRUTADR[69]
885         :UTRUTADR[70]
886         :UTRUTADR[71]
887         :UTRUTADR[72]
888         :UTRUTADR[73]
889         :UTRUTADR[74]
890         :UTRUTADR[75]
891         :UTRUTADR[76]
892         :UTRUTADR[77]
893         :UTRUTADR[78]
894         :UTRUTADR[79]
895         :UTRUTADR[80]
896         :UTRUTADR[81]
897         :UTRUTADR[82]
898         :UTRUTADR[83]
899         :UTRUTADR[84]
900         :UTRUTADR[85]
901         :UTRUTADR[86]
902         :UTRUTADR[87]
903         :UTRUTADR[88]
904         :UTRUTADR[89]
905         :UTRUTADR[90]
906         :UTRUTADR[91]
907         :UTRUTADR[92]
908         :UTRUTADR[93]
909         :UTRUTADR[94]
910         :UTRUTADR[95]
911         :UTRUTADR[96]
912         :UTRUTADR[97]
913         :UTRUTADR[98]
914         :UTRUTADR[99]
915         :UTRUTADR[100]
916         :UTRUTADR[101]
917         :UTRUTADR[102]
918         :UTRUTADR[103]
919         :UTRUTADR[104]
920         :UTRUTADR[105]
921         :UTRUTADR[106]
922         :UTRUTADR[107]
923         :UTRUTADR[108]
924         :UTRUTADR[109]
925         :UTRUTADR[110]
926         :UTRUTADR[111]
927         :UTRUTADR[112]
928         :UTRUTADR[113]
929         :UTRUTADR[114]
930         :UTRUTADR[115]
931         :UTRUTADR[116]
932         :UTRUTADR[117]
```

```
933           :UTRUTADR[118]
934           :UTRUTADR[119]
935           :UTRUTADR[120]
936           :UTRUTADR[121]
937           :UTRUTADR[122]
938           :UTRUTADR[123]
939           :UTRUTADR[124]
940           :UTRUTADR[125]
941           :UTRUTADR[126]
942           :UTRUTADR[127]
943           :UTRUTADR[128]
944           :UTRUTADR[129]
945           :UTRUTADR[130]
946           :UTRUTADR[131]
947           :UTRUTADR[132]
948           :UTRUTADR[133]
949           :UTRUTADR[134]
950           :UTRUTADR[135]
951           :UTRUTADR[136]
952           :UTRUTADR[137]
953           :UTRUTADR[138]
954           :UTRUTADR[139]
955           :UTRUTADR[140]
956           :UTRUTADR[141]
957           :UTRUTADR[142]
958           :UTRUTADR[143]
959           :UTRUTADR[144]
960           :UTRUTADR[145]
961           :UTRUTADR[146]
962           :UTRUTADR[147]
963           :UTRUTADR[148]
964           :UTRUTADR[149]
965           :UTRUTADR[150]
966           :UTRUTADR[151]
967           :UTRUTADR[152]
968           :UTRUTADR[153]
969           :UTRUTADR[154]
970           :UTRUTADR[155]
971           :UTRUTADR[156]
972           :UTRUTADR[157]
973           :UTRUTADR[158]
974           :UTRUTADR[159]
975           :UTRUTADR[160]
976           :UTRUTADR[161]
977           :UTRUTADR[162]
978           :UTRUTADR[163]
979           :UTRUTADR[164]
980           :UTRUTADR[165]
981           :UTRUTADR[166]
982           :UTRUTADR[167]
983           :UTRUTADR[168]
984           :UTRUTADR[169]
985           :UTRUTADR[170]
986           :UTRUTADR[171]
987           :UTRUTADR[172]
988           :UTRUTADR[173]
989           :UTRUTADR[174]
990           :UTRUTADR[175]
991           :UTRUTADR[176]
992           :UTRUTADR[177]
```

```
 993          :UTRUTADR[178]
 994          :UTRUTADR[179]
 995          :UTRUTADR[180]
 996          :UTRUTADR[181]
 997          :UTRUTADR[182]
 998          :UTRUTADR[183]
 999          :UTRUTADR[184]
1000          :UTRUTADR[185]
1001          :UTRUTADR[186]
1002          :UTRUTADR[187]
1003          :UTRUTADR[188]
1004          :UTRUTADR[189]
1005          :UTRUTADR[190]
1006          :UTRUTADR[191]
1007          :UTRUTADR[192]
1008          :UTRUTADR[193]
1009          :UTRUTADR[194]
1010          :UTRUTADR[195]
1011          :UTRUTADR[196]
1012          :UTRUTADR[197]
1013          :UTRUTADR[198]
1014          :UTRUTADR[199]
1015          :UTRUTADR[200]
1016          :UTRUTADR[201]
1017          :UTRUTADR[202]
1018          :UTRUTADR[203]
1019          :UTRUTADR[204]
1020          :UTRUTADR[205]
1021          :UTRUTADR[206]
1022          :UTRUTADR[207]
1023          :UTRUTADR[208]
1024          :UTRUTADR[209]
1025          :UTRUTADR[210]
1026          :UTRUTADR[211]
1027          :UTRUTADR[212]
1028          :UTRUTADR[213]
1029          :UTRUTADR[214]
1030          :UTRUTADR[215]
1031          :UTRUTADR[216]
1032          :UTRUTADR[217]
1033          :UTRUTADR[218]
1034          :UTRUTADR[219]
1035          :UTRUTADR[220]
1036          :UTRUTADR[221]
1037          :UTRUTADR[222]
1038          :UTRUTADR[223]
1039          :UTRUTADR[224]
1040          :UTRUTADR[225]
1041          :UTRUTADR[226]
1042          :UTRUTADR[227]
1043          :UTRUTADR[228]
1044          :UTRUTADR[229]
1045          :UTRUTADR[230]
1046          :UTRUTADR[231]
1047          :UTRUTADR[232]
1048          :UTRUTADR[233]
1049          :UTRUTADR[234]
1050          :UTRUTADR[235]
1051          :UTRUTADR[236]
1052          :UTRUTADR[237]
```

```
1053          :UTRUTADR[238]
1054          :UTRUTADR[239]
1055          :UTRUTADR[240]
1056          :UTRUTADR[241]
1057          :UTRUTADR[242]
1058          :UTRUTADR[243]
1059          :UTRUTADR[244]
1060          :UTRUTADR[245]
1061          :UTRUTADR[246]
1062          :UTRUTADR[247]
1063          :UTRUTADR[248]
1064          :UTRUTADR[249]
1065          :UTRUTADR[250]
1066          :UTRUTADR[251]
1067          :UTRUTADR[252]
1068          :UTRUTADR[253]
1069          :UTRUTADR[254]
1070          - 0

1071                                                                  " TEXTTR
1072                                                                  " UMES13 - 0
1073 UMES13(10)
1074          UTREXR
1075          $ = UMIREAC[5]                    " :USBTRA/B
1076          A = UAMVAR[US]
1077          A - 19, Z                         " TRVAR = 19?
1078     Y,   A = 20                            " TRVAR := 20
1079     Y,   UAMVAR[US] = A
1080     Y,   GOTO(:UANS2[7])                   " MERGE WITH UANS2
1081          UTREXV
1082          G = CMNA13                        " NOT APPLICABLE
1083          GOTO(:UENDMES)

1084                                                                  " TEXTTR
1085                                                                  " UMES 9 - 0
1086 UMES9(11)
1087          UTREXR
1088          G = UMIREAC[5]                    " :USBTRA/B
1089          UMIREAC[2] = G
1090          A = UAMVAR[UG]
1091          A - 18, Z                         " TRVAR = 18?
1092     N,   A - 1, Z                          " TRVAR = 19?
1093     Y,   A = =UDISC[UG],P                  " DISCONNECT FALSE ?
1094     N,   UDISC[UG] = B                     " DISCONNECT := TRUE
1095     N,   GOTO(:UANS3[11])                  " MERGE WITH UANS3
1096          $ = CMNA9                         " NOT APPLICABLE
1097          GOTO(:UMES5[15])                  " MERGE WITH UMES5

1098                                                                  " TEXTTR
1099                                                                  " UMES14
1100                                                                  " UANS15
1101                                                                  " UANS16
1102 UMES14(5)
1103          SUBCD(:UCLAIMCON)
1104          G = UM14                          " :UINVM14A/B
1105          SUBC(:CPRINTSTRING)
1106          $ = :COLIM14
1107          GOTO(:UCONCON)

1108 UANS15(5)
```

more messages, having to do with the tapereadless

```
1109          UTREXR
1110          G = UMIREAC[1]
1111          S = 22
1112          UAMVAR[UG] = S                    " TRVAR := 22
1113          UTREXV
1114 UANS16(4)
1115          G = UMIREAC[1]
1116          E = :UAMSEM[UG]
1117          UV                                " V(TRSEM)
1118          GOTO(:UENDMES[4])

1119                                                                  " TEXTTR
1120                                                                  " UMES5 - 0
1121 UMES5(18)
1122          UTREXR
1123          S = 5
1124          SUBCD(:UEXRMTRVAR)                " EXIST RMTRVAR = 5?
1125      Y,  JUMP(9)
1126          S = 1
1127          SUBCD(:UEXRMTRVAR)                " EXIST RMTRVAR = 1?
1128      N,  JUMP(6)
1129          URENREQ + S                       " INCREASE URENREQ
1130          S = 5
1131          UAMVAR[UG] = S                    " RMTRVAR := 5
1132          SUBCD(:UNONHAR)
1133          SUBCD(:UTRACT)
1134          GOTO(:UENDMES[3])
1135      Y,  S = CMNA7
1136      N,  S = CMNA6
1137          UTREXV
1138          G = S
1139          GOTO(:UENDMES)

1140                                                                  " TEXTTR
1141                                                                  " UMES12 - 0
1142 UMES12(9)
1143          UTREXV                            " V(UTREX)
1144          SUBCD(:UCLAIMCON)                 " CLAIM CONSOLE
1145          UTREXR
1146          UDISC = - B                       " DISCONNECT:=FALSE
1147          UTREXV
1148          G = UM12
1149          SUBC(:CRRINTSTRING)
1150          SUBCD(:UCONEREE)
1151          GOTO(:UTREXRADRES)

1152                                                                  " TEXTTR
1153                                                                  " UANS2 - 0
1154 UANS2(10)
1155          UTREXR                            " R(UTREX)
1156          G = UMIREAC[1]                    " :USBRMI
1157          S = UMIREAC[2]                    " :USBTRA/B
1158          A = 4
1159          UAMVAR[US] + A                    " TRVAR := TRVAR + 4
1160          E = :UAMSEM[UG]
1161          UV                                " V(RMSEM)
1162          E = :UAMSEM[US]
1163          UV                                " V(TRSEM)
1164          GOTO(:UENDMES[3])
```

```
1165                                                                    " TEXTTR
1166                                                                    " USTAN - 0
1167 USTAN(20)
1168          UTREXRROOD                        " RED R(UTREX)
1169          S = UTRSTREAMS
1170          S = MS[0]                         " :UTRHANDLE!1 IN S
1171          UBLOCKSTR = S                     " FILL BLOCKING STREAM
1172          S = USRECIALRM, R                 " SPECIAL RM ?
1173     N,   S = 1
1174          UAMVAR = S
1175     Y,   SUBCD(:URERM)
1176     N,   SUBCD(:UHAR)
1177          UTREXV                            " V(UTREX)
1178          UROWNSEM                          " R(RMSEM)
1179          UTREXR                            " R(UTREX)
1180          S = UAMVAR, Z                     " RMTRVAR = 0?
1181     N,   UAMVAR - S                        " RMTRVAR := 0
1182     N,   SUBCD(:UVOR[2])                   " V(UTREX)
1183     N,   SUBCD(:UMES8)                     " MESSAGE M8 ETC.
1184     N,   S = UBLOCKSTR                     " :UTRHANDLE BLOCKING STREAM
1185     N,   S = MS[7]                         " :USB COUPLED TAPEREADER
1186     N,   E = :UAMSEM[US]
1187          GOTO(:UVOR)                       " WHITE V(UTREX) OR V(TRSEM)
1188 URROGNAME(22)
1189          SUBCD(:UREDDEN)
1190          G = UM36[US]
1191          SUBCD(!CRRINTSTRING)
1192          G = URROGID[US]
1193          SUBCD(:UINSTV)
1194          S = 5                             " TO SKIP FOUR DIGITS
1195          MC = S                            " STACK COUNTER
1196          E = :MC[- 4]
1197          SUBCD(:UDNSS)
1198     U,   A - 510, Z                        " END OF STRING ?
1199     Y,   B = 4
1200     Y,   S = M[B - 3]                      " RESTORE :M0[0]
1201     Y,   A = URUNNUM[US]
1202     Y,   SUBCD(:CRRINTDEC)
1203     Y,   GOTO(:UHERSTELTERUG)
1204     U,   A - 9, R
1205     U,   A + 1, E                          " ¬ DIGIT ?
1206     N,   S = - 1
1207     Y,   S = - 5
1208          S + MC[- 1], R
1209     N,   SUBCD(:CRRINTINT)
1210          JUMR( -16)
1211 CCOCT(10)
1212          E - 0, R
1213          E - UNUMTRRMSTR, E
1214     Y,   A = 228
1215     Y,   GOTO(:UDYNERR)
1216          G + UTRSTREAMS
1217          G = MG[UNUMTRRMSTR - 1]
1218          UBLOCKSTR = G
1219          SUBCD(:COCT[1])
1220          G = S
1221          GOTO(:RSE6[7])
1222                                                                  " TEXTTR
1223                                                                  " UNONSTAN
```

PM waiting for standard tape (i.e. the beginning of an ALGOL program)
This is what all PM's are doing upon initialization of the system.

```
1224 UNONSTAN(23)
1225          SUBCD(:ORSMUK1)
1226          UTREXRROOD                        " RED R(UTREX)
1227          S = USRECIALRM, R                 " IS THIS A SPECIAL RM ?
1228      N,  A = 1
1229      N,  UBENREQ + A                       "INC(NUMBER PENDING REQUESTS)
1230      N,  S = 2
1231          UAMVAR = S
1232      N,  SUBCD(:UNONHAP)
1233      Y,  SUBCD(:UBERM)
1234          SUBCD(:UTRACT)
1235          UTREXV
1236          UROWNSEM
1237      N,  SUBCD(:UMES1)
1238      N,  SUBCD(:UROWNSEMADRES)
1239          UTREXR                            " R(UTREX)
1240          S = UAMVAR, Z                     " RMTRVAR = 0?
1241      Y,  S = UBLOCKSTR
1242      Y,  A = 1
1243      Y,  MS[9] + A                         " INCREASE TAPE NUMBER
1244          UTREXVWIT                         " WHITE V(UTREX)
1245      Y,  GOTO(:UMORTAL)
1246          SUBCD(:USTORREN)
1247          GOTO(:UNONSTAN[1])

1248 UBERM(4)
1249          SUBCD(:UEXRMTRVAR)
1250          MC = G
1251      Y,  SUBCD(:UEXTRVAR)
1252          GOTO(:UHAP[6])

1253 UMES1(19)
1254          SUBCD(:UCLAIMCON)
1255          S = :M0[0]
1256          SUBCD(:UPROGNAME)
1257          S = UBLOCKSTR
1258          UMIREAC[3] = S                    " UMIREAC[3] := UTRHANDLE!U
1259          G = MS[7]
1260          UMIREAC[2] = G                    " UMIREAC[2] := :USBTRA/B
1261          G = UM1[UG]                       " REA/B
1262          SUBCD(:CPRINTSTRING)
1263          A = MS[5]
1264          RUA(18)                           " STREAMNUMBER
1265          SUBCD(:CPRINTDEC)
1266          A = MS[9]                         " TAPE NUMBER
1267          SUBCD(:CPRINTDEC)
1268          S = :COLIM1                       " ADDRESS PROPER CODE LIST
1269          A = 50                            " BLACK IRON CROSS
1270          CGAPR = A
1271          SUBCD(:CPRINT)
1272          GOTO(:UCONCON)

1273                                                                  " TEXTTR
1274                                                                  " UANS3 = 0
1275 UANS3(21)
1276          UTREXR
1277          G = UMIREAC[1]                    " :USBRMI
1278          S = 3
1279          UAMVAR[UG] = S                    " RMTRVAR := 3
1280          F = :UAMSEM[UG]
```

```
1281          UV                                  " V(RMSEM)
1282          G = UMIREAC[3]
1283          MG[7] = - B                         " DISCOUPLE TAPE READER
1284          G = UMIREAC[2]                      " :USBTRA/B
1285          A = 2
1286          UAMVAR[UG] - A                      " TRVAR := TRVAR - 2
1287          SUBCD(:UDISCONNECT)
1288      N,  SUBCD(:UNONHAR)
1289      N,  SUBCD(:UHAR)
1290      N,  GOTO(:UENDMES[3])
1291          UDISC[UG] = - B                     " DISCONNECT := FALSE
1292          UTREXV
1293          SUBCD(:UPRINTNLCR)
1294          G = UMIREAC[2]                      " :USBTRA/B
1295          G = UM12[UG]                        " UINVM12A/B
1296          GOTO(:UENDMES)

1297                                                                  " TEXTTR
1298                                                                  " UMES8 - 0
1299  UMES8(6)
1300          SUBCD(:UCLAIMCON)                   " CLAIM CONSOLE
1301          $ = UBLOCKSTR                       " :UTRHANDLE BLOCKING STREAM
1302          $ = MS[7]                           " :USB COUPLED TAPE READER
1303          G = UM8[US]                         " UINVM8A/B
1304          SUBC(:CPRINTSTRING)
1305          GOTO(:UCONFREE)

1306                                                                  " TEXTTR
1307                                                                  " GVARTRINT - 0
1308  CVARTRIET(10)
1309          A = :CLPLAB[1]                      "FILL ARG WITH LABEL[1]
1310          A '+' M1[0]
1311          A '+' UCON2
1312          A '+' M1[0]                         "NEW LABEL IN ADDRESS PART
1313          M1[0] = A                           "COUNT PART UNCHANGED
1314          A = UCON1
1315          M1[2] + A                           "INCREASE AET
1316          G = MD[2]
1317          DO(MG[7])                           "AF := TRUE
1318          GOTO(:UPORM)

1319                                                                  " COORDINATOR
1320                                                                  " UEIOEIOR - 1
1321  UEIOEIOR(40)
1322          SUBCD(:UREDDEN)
1323          SUBCD(:UTRACT)                      " TAPE READER ACTIVATION
1324          UWORKB = B
1325          $ = UMAXPOR
1326          $ - UEIO, P                         " FIO < MAXPOR?
1327      Y,  JUMP(24)
1328          B = :ULISTRM[UNUMRM + 1]
1329          UWORK2 = - B                        " RM IDENTIFICATION := NEGATIV
1330          G = UCON3
1331          USTRLEN = F                         " MINIMUM LENGTH := 2 ↑ 26 - 1
1332          A = MC[0]                           " :USBRMI
1333          $ = 12
1334      U,  $ - UAMVAR[UA], Z                   " RMSHR = 12?
1335      Y,  $ = UBLOCKSTR[UA]                   " :UHANDLE BLOCKING STREAM
1336          G = - MS[0]                         " - LENGTH
1337      Y,  F - MS[7]                           " - ATTACHMENTTIME
```

messages

strategical subroutine
on increase or decrease of i or o

```
1338      Y,  E + USTRLEN, P                  " LENGTH + ATT.TIME < MIN. LENGTH
1339      Y,  E - USTRLEN
1340      Y,  USTRLEN = - E                   " NEW MINIMUMLENGTH
1341      Y,  UWORK2 = A                      " SET PM-IDENTIFICATION
1342          A - ULISTRM[2], Z               " ROUND?
1343      N,  JUMP(- 12)
1344          B = UWORKB
1345          G = UWORK2, P                   " RM CHOSEN?
1346      Y,  S = UREQUEST[UG]
1347      Y,  UEIO - S                        " DEC(FIO, REQUEST)
1348      Y,  UAMVAR[UG] = A                  " RMSHR := 0
1349      Y,  E = :UAMSEM[UG]
1350      Y,  SUBCD(:UVOR[2])                 " V(RMSEM)
1351      Y,  JUMP(- 27)                      " TRY AGAIN

1352                                                                  " COORD
1353                                                                  " UEIOFIOP - 2
1354          S = UEIOP, P                    " FIOP > 0?
1355      Y,  S = 13
1356      Y,  SUBCD(:UEXRMTRVAR)              " EXIST(RMSHR = 13)?
1357      N,  GOTO(:UHERSTELTERUG)            " RETURN
1358          UAMVAR[UG] - S                  " RMSHR := 0
1359          S = 1
1360          UEIOP - S                       " DEC(FIOP)
1361          E = :UAMSEM[UG]
1362          UV                              " V(RMSEM)
1363          JUMP(- 10)

1364                                                                  " COORDINATOR
1365                                                                  " USGAAN - 0
1366  USGAAN(23)
1367          UWORK3 = B                      " SAVE B
1368          G = UTRUTEREE                   " ADDRESS FIRST AVAILABLE FREE PLACE
1369          B = MG[0]                       " NEXT AVAILABLE FREE PLACE
1370          UTRUTEREE = B                   " FREE PLACE UNCHAINED
1371          B = MS[0], P                    " SV IN B, SEGMENT ON CORE?
1372      N,  JUMP(1)
1373          M[B] = G                        " NEW :SV IN CTE
1374          MS[0] - B                       " - 0 IN OLD SV
1375          MG[0] = B                       " MOVE SV
1376          S = MA[3], Z                    " STREAM EMPTY?
1377      Y,  MA[3] = E                       " SET :SV FIRST SEGMENT TO BE UNCHAINED
1378          MA[3] = G                       " SET :SV LAST CHAINED SEGMENT
1379      N,  E - :UTRUTADR[- 1]              " NUMBER OF LAST CHAINED SV
1380      N,  S = :UTRUTADR[- 1]              " NUMBER OF PREVIOUS SV
1381      N,  MULS(9)                         " UPDATE UTPUT LIST
1382      N,  DIVAS(27)
1383      N,  S + :UTRUTLIST
1384      N,  B = A
1385      N,  A = G
1386      N,  LCA(B)
1387      N,  MS[0] + A                       " END OF UPDATE UTPUTLIST
1388          B = UWORK3                      " RESTORE B, ENTRY USGAE, USGLONG
1389          GOTOR(MC[- 1])                  " RETURN

1390                                                                  " COORDINATOR
1391                                                                  " USGLONG - 0
1392  USGLONG(14)
1393          UWORK3 = B                      " SAVE B
1394          E = 1                           " INITIALISATION LENGTH IN SEGMENTS
```

attach segment
to ~~tape~~ i/o stream
either by tape reader CM to i-stream,
or by PM to o-stream.

```
1395          $ - :UTRUTADR[- 1]              " NUMBER FIRST SV
1396          MUL$(9)
1397          DIVA$(27)
1398          $ + :UTRUTLIST
1399          B = A
1400          $ = M$[0]
1401          RC$(B)
1402      U,  $ '*' 256, Z                    " NO FINAL SEGMENT
1403      Y,  $ '*' 255                       " NUMBER NEXT SV
1404      Y,  F + 1                           " INCREASE LENGTH IN SEGMENTS
1405      Y,  JUMP(- 10)
1406          GOTO(:USGAAN[21])               " RETURN VIA USGAAN

1407                                                          " COORDINATOR
1408                                                          " USGAF - 0
1409 USGAE(27)
1410          UWORK3 = B                      " SAVE B
1411          $ = MG[4]                       " ADDRESS SV IN STREAM
1412          B = M$[0], R                    " SV IN B, SEGMENT IN CORE?
1413          MA[0] = B                       " MOVE SV
1414      Y,  M[B] = A                        " NEW :SV IN CTE
1415          B = UTRUTEREE                   " ADDRESS FIRST AVAILABLE FREE PLACE, MAYBE - 0
1416          M$[0] = B
1417          UTRUTEREE = $                   " FREE PLACE CHAINED
1418      U,  $ - MG[3], Z                    " UNCHAINING LAST SEGMENT IN STREAM?
1419      Y,  MG[3] - $                       " HANDLE[3] := - 0
1420          $ - :UTRUTADR[- 1]              " NUMBER OF UNCHAINED SV
1421          MUL$(9)
1422          DIVA$(27)
1423          $ + :UTRUTLIST
1424          B = A
1425          A = M$[0]
1426          RCA(B)
1427      U,  A '*' 256, Z                    " NO FINAL SEGMENT?
1428          A '*' 255                       " NUMBER NEXT SV TO BE UNCHAINED
1429          A + :UTRUTADR[- 1]              " :SV NEXT SEGMENT TO BE UNCHAINED
1430          MG[4] = A
1431          A = - 511                       " CLEAR UTPUTLIST
1432          LCA(B)
1433          A '*' M$[0]
1434          M$[0] = A                       " END OF CLEARING UTPUTLIST
1435      Y,  $ = - 0
1436          GOTO(:USGAAN[21])               " RETURN VIA USGAAN

1437 COLIM1(2)
1438          '020 400 000'[:UANS2]           " OKSS
1439          - '021 400 000'[:UANS3]         " NOSS
1440 COLIM14(2)
1441          '020 400 000'[:UANS15]          " OKSS
1442          - '022 400 000'[:UANS16]        " SKIPSS
1443 CREMES(10)
1444          $ + :ULISTTR[2]
1445          $ = M$                          " :USBTRA/B
1446          UMIREAC[5] = $
1447          $ = :CREMES[8]
1448          GOTO(:CMI[30])
1449          '025 400 000'[:CREMES]          " ASP
1450          '026 400 000'[:CREMES]   " BSP
1451          - '034 400 000'[:CREMES]        " CSP
1452          '031 400 000'[:UMES13]          " INSS
```

detach segment from i/o stream.

more messages

```
1453          - '005 400 000'[:UMES9]          " OUTSS
1454 CDATE(4)
1455          SUBC(:CREADDEC)
1456          UDATE = S
1457          SUBC(:CREQSS)
1458          GOTO(:UENDMES[4])
1459 CSERIAL(4)
1460          SUBC(:CREADDEC)
1461          UCURRUNNUM = S
1462          GOTO(:CDATE[2])
1463          - '016 400 000'[:CSERIAL]        " ALSP

1464 RSE115(21)
1465          S = URROCTIME
1466          GBILL - S
1467          S = 1
1468          PLUSS(UNITS)                     " TO BE INITIALISED AT 0
1469          S - CSTEEK, Z                    " TO BE INITIALISED AT 1
1470      N,  GOTOR(MC[ - 1])
1471          SUBCD(:UREDDEN)
1472          SUBCD(:UIMMORTAL)
1473          SUBCD(:UCLAIMCONRED)
1474          S = :M0[0]
1475          SUBCD(:URROGNAME)
1476          A = CSTEEK
1477          SUBCD(:CRRINTDEC)
1478          LCA(1)
1479          CSTEEK = A
1480          G = MT[4]
1481          SUBCD(:CRRINTSTRING)
1482          SUBCD(:UCONEREEWHITE)
1483          SUBCD(:UMORTAL)
1484          GOTO(:UHERSTELTERUG)
1485          UINVTIME
1486 URROCTIME(1)
1487          2 000 000                        " BASE AMOUNT OF PROCESSING TIME IN UNITS OF 60 MMS
1488 RSE19(2)
1489          A = 223
1490          GOTO(:UDYNERR)
1491 CALARM(2)
1492          A = :MS[227]
1493          GOTO(:UDYNERR)
1494 RSE88(6)
1495          F + DRT[11]
1496          F - DRT[11]
1497      U,  S = M[57', Z
1498      Y,  GOTOR(MC[- 1])
1499          A = 222
1500          GOTO(:UDYNERR)

1501 CSYMTAB1(13)                             " CHAR  CHAR - VALUE    SYM - VALUE
1502          '000 107 506'                    " ≠     326             71
1503          '000 111 310'                    " ≤     200             73
1504          '000 113 312'                    " ≥     202             75
1505          '000 105 520'                    " ↑     336             69
1506          '000 146 510'                    " ∉     328             102
1507          '000 147 512'                    " ∋     330             103
1508          '000 104 332'                    " ⊥     218             68
1509          '000 115 306'                    " ⊆     198             77
1510          '000 116 314'                    " ⊇     204             78
```

test on time used by PM and if necessary warning to operator via console keyboard.

rounding reals to integers by first adding and then subtracting $3*2\uparrow38$ (very tricky !?)

more code conversion tables

```
1511          '000 201 327'                  " ↑      215             129
1512          '000 176 335'                  " _      221             126
1513          '000 177 535'                  " |      349             127
1514          - 0                            " ENDMARKER

1515 CSYMTAB2(50)
1516          '000 000 010'                  " H
1517          '777 777 770'                  " G
1518          '000 000 704'                  " ND
1519          '000 000 270'                  " EX
1520          '000 000 262'                  " ER
1521          '000 000 056'                  " AN
1522          '000 000 724'                  " NT
1523          '000 001 131'                  " RY
1524          '000 000 446'                  " IF     INITIALISATION POINT
1525          '000 000 217'                  " DO
1526          '777 777 015'                  " OR
1527          '000 012 704'                  " END
1528          '000 014 762'                  " FOR
1529          '000 002 704'                  " AND
1530          '000 034 756'                  " NON
1531          '000 034 764'                  " NOT
1532          '000 037 356'                  " OWN
1533          '777 774 627'                  " ASH
1534          '002 420 256'                  " THEN
1535          '000 737 217'                  " GOTO
1536          '000 531 145'                  " ELSE
1537          '002 212 054'                  " REAL
1538          '002 445 245'                  " TRUE
1539          '002 350 260'                  " STEP
1540          '001 630 162'                  " NLCR
1541          '777 324 672'                  " DURE
1542          '010 516 456'                  " BEGIN
1543          '006 244 071'                  " ARRAY
1544          '011 736 605'                  " BOOLE
1545          '015 732 645'                  " COMME
1546          '030 131 145'                  " FALSE
1547          '102 236 145'                  " PROCE
1548          '125 650 454'                  " UNTIL
1549          '130 131 245'                  " VALUE
1550          '135 022 605'                  " WHILE
1551          '015 733 014'                  " COMPL
1552          '045 650 247'                  " INTEG
1553          '165 736 605'                  " BOOLE
1554          '005 416 754'                  " ALGOL
1555          '066 403 005'                  " MTAPE
1556          '045 655 154'                  " INVSL
1557          '061 105 101'                  " LIBRA
1558          '060 104 254'                  " LABEL
1559          '102 236 345'                  " PROGE
1560          '116 444 456'                  " STRIN
1561          '116 723 203'                  " SWITC
1562          '035 775 217'                  " GO TO
1563          '015 745 105'                  " CORRE
1564          '016 422 756'                  " CTION
1565          '665 242 613'                  " RUNST

1566 CSYMTAB3(17)
1567          '732 722 725'
1568          '721 720 700'
```

```
1569            '136 740 705'
1570            '151 117 126'
1571            '114 120 122'
1572            '741 152 114'
1573            '140 121 137'
1574            '123 164 153'
1575            '150 724 200'
1576            '541 555 157'
1577            '124 560 165'
1578            '534 125 163'
1579            '202 555 554'
1580            '574 575 173'
1581            '556 566 162'
1582            '603 121 561'
1583            '000 204 747'

1584 CHARETAB(43)
1585            '002 001 000'
1586            '777 004 777'
1587            '010 007 255'
1588            '000 251 252'
1589            '000 177 247'
1590            '377 000 371'
1591            '332 003 377'
1592            '254 006 005'
1593            '000 011 253'
1594            '246 177 250'
1595            '000 177 245'
1596            '035 377 377'
1597            '040 037 372'
1598            '043 264 265'
1599            '177 261 177'
1600            '177 256 257'
1601            '034 110 377'
1602            '377 036 377'
1603            '042 041 266'
1604            '131 262 263'
1605            '000 177 260'
1606            '777 101 314'
1607            '370 025 377'
1608            '275 030 027'
1609            '177 033 274'
1610            '267 177 271'
1611            '377 177 312'
1612            '377 024 023'
1613            '276 377 026'
1614            '273 032 031'
1615            '270 127 272'
1616            '320 177 177'
1617            '013 012 377'
1618            '777 015 377'
1619            '021 020 343'
1620            '130 333 344'
1621            '177 177 303'
1622            '777 100 777'
1623            '777 014 377'
1624            '342 017 016'
1625            '000 022 345'
1626            '302 000 306'
1627            '000 000 317'
```

```
1628 CNUMTAB(22)                        " 10↑N, FLOATING REPRESENTATION
1629         '000 000 000'              " HEAD,     N =1
1630         '000 000 012'              " TAIL
1631         '000 000 000'              "                2
1632         '000 000 144'
1633         '000 000 000'              "                4
1634         '000 023 420'
1635         '000 000 001'              "                8
1636         '175 360 400'
1637         '002 004 341'              "               16
1638         '274 467 701'
1639         '010 323 561'              "               32
1640         '153 324 053'
1641         '025 530 236'              "               64
1642         '007 645 000'
1643         '060 222 356'              "              128
1644         '221 744 601'
1645         '145 325 237'              "              256
1646         '272 775 636'
1647         '317 534 306'              "              512
1648         '150 127 246'

1649         '000 003 146'              " 2 ↑ 40 ↓ 10
1650         '146 314 632'

1651 CODEEIN(20)
1652         SUBCD(:COCT)
1653     U,  S - 127, Z                 " ERASE ?
1654     N,  S - 11, Z                  " STOPCODE ?
1655     N,  S + 11, Z                  " BLANK TAPE ?
1656     Y,  JUMP( - 5)
1657     U,  S - 26, Z                  " NLCR ?
1658     Y,  MG[12] = S                 " BACKOCT
1659         A = 0
1660     N,  S - 122, Z                 " LOWERCASE ?
1661     Y,  MG[15] = A                 " LOWER := IBUE
1662     Y,  JUMP(2)
1663         S - 2, Z                   " UPPERCASE ?
1664     Y,  MG[15] = - A
1665     Y,  S = 2
1666     Y,  MG[10] = S                 " BASINT := 2
1667     Y,  MG[13] = - A               " VOORCHARE
1668     Y,  MG[16] = A                 " POS
1669     Y,  MG[20] = - A               " SPATEL
1670     N,  S + 124                    " RETORE VALUE OF HEPTAD
1671         GOTO(MC[ - 1])             " RETURN WITH CONDITION

1672 ORSMUK1(16)
1673         S = UBLOCKSTR
1674         S = MS[7]
1675         A = :ULISTTR[UNUMTR + 2]
1676     U,  S + MA[-1],Z
1677     Y,  JUMP(4)
1678         A = 1
1679     U,  A - :ULISTTR[2],Z
1680     N,  JUMP(-5)
1681         GOTO(:UIMMORTAL)
1682     U,  A -:ULISTTR[UNUMTR+2],Z
1683     Y,  A=:ULISTTR[2]
```

```
1684          ULISTTR = A
1685          GOTO(:UIMMORTAL)
1686          S = MG[7]
1687          MG[7] = - S
1688          GOTOR(MC[- 1])
```

```
 0 UPORTIONCLOCK(2)
 1          = 0
 2          = 0
 3 UCLOCKINCR(2)
 4          '777 500 000'
 5          '000 000 001'

 6 RSE33(20)                                  " SEGMENT TRANSITION
 7          MC[0] = A                         " SAVE A
 8          MC[0] = S                         " SAVE S
 9          S = M[B - 3]                      " TAKE LINK
10          SUBCD(:UPROFANELINK[1])           " PROFANATION VIA LINK
11          A = M[B - 3]                      " TAKE LINK
12          S = MA[0]                         " INVARIANT TARGET ADDRESS
13          GL26 = S                          " FILL CURRENT PROGRAM SEGMENT
14          A '*' - DRT[1]                    " ISOLATE SMALL REGISTERS
15          RCSA(9)                           " :SV IN S
16          LCA(9)                            " SMALL REGISTERS AND LINE NUMBER IN A
17     U,   S = MS[0], P                      " SEGMENT ON CORE?
18     N,   SUBCD(:SRQ)                       " SEGMENT REQUEST
19          S = MS[0]                         " :CTE[0]
20          A + MS[2]                         " ADD CORE PAGE BASE
21          M[B - 3] = A                      " STORE NEW MACHINE LINK
22     Y,   A = DRT[2]                        " 2 ↑ 20
23     Y,   MS[3] + A                         " INCREASE HOLYNESS COUNTER
24          S = MC[- 1]                       " RESTORE A
25          A = MC[- 1]                       " RESTORE S
26          GOTOR(MC[- 1])                    " RETURN

27 RSE34(11)                                 " GENERAL GOTO
28          G = GL1, P                        " NO SKIP?
29     N,   GOTOR(MC[- 1])                    " RETURN IF SKIP
30          SUBCD(:UPROFANELINK)              " PROFANATION VIA LINK
31          S = MG[- 3]                       " TARGET D
32          M[63] = S                         " SET D TO TARGET VALUE
33          A = MG[1]                         " TARGET BLOCK HEIGHT
34          MS[- 1] = A                       " FILL BLOCK HEIGHT IN DISPLAY
35          B = MG[0]                         " B := TARGET AR(=WR)
36          SUBCD(:SCL)                       " STACK COLLAPS
37          G = GL0                           " INVARIANT TARGET ADDRESS
38          GOTO(:R46E3)                      " MERGE WITH NSIS ENTRY

39 RSE72(2)                                  " IMPLICIT SUBROUTINE EXIT
40          S = 1
41          MD[- 1] - S                       " DECREASE CSW IN PARAMETER DISPLAY
42 RSE71(28)                                 "PROCEDURE EXIT
43          S = M[B - 1]                      "TAKE LINK
44          B = :MA[2]                        "B := WR
45          SUBCD(:UPROFANELINK[1])
46          A = - M[B - 3], P                 "SEMPLE IMPLICIT SUBROUTINE?
47     N,   A = - A
48          S = MA[ - 3]
49          M[63] = S                         "RESET D
50     Y,   B - 4
51     Y,   JUMP(17)
52          SUBCD(:SCL1[ - 4])                "FREE STACKPAGES(SEE STACK COLLAPS)
53          S = M[B - 9]                      " INVARIANT RETURN ADDRESS
54          GL26 = S                          " FILL CURRENT PROGRAM SEGMENT
55          RUS(9), P                         " ADDRESS SV IN S
```

system subroutines

this is the goto virtual address ~~in a paged~~

— profanate old segment

} find new segment

GENERAL GOTO → including the necessary block and/or procedure exits

```
56      N,  $ '*' DRT[1]            " PRECAUTION MEMORY ABOVE 32K
57          A = MS[0], P            " SEGMENT ON CORE?
58      N,  SUBCD(:SRQ)             " SEGMENT REQUEST
59      N,  A = MS[0]               " :CTE[0]
60      Y,  $ = DRT[2]
61      Y,  MA[3] + $               " INCREASE HOLYNESS COUNTER
62          $ = MC[- 9]             " INVARIANT RETURN ADDRESS
63          $ '*' 511               " LINE NUMBER IN $
64          $ + MA[2]               " ADD CRB
65          $ + DRT[3]              " STANDARD FILLING SMALL REGISTERS
66          A = MC[- 2]             " A := TARGET LRR
67          B = M[B - 6]            " B := TARGET AR
68          MC[0] = $               " STACK LINK
69          $ = :DRT[0]             " SEE SE57
70          GOTOR(MC[- 1])          " RETURN

71  RSE46(12)                       " NSIS ENTRY
72          B + 6                   " B := TARGET LRR
73          G = MS[2]               " INV. STARTING ADDRESS AND CONTEXT D
74          $ = 1
75          PLUS$(MG[- 1])          " INCRESE CBW OF TARGET DISPLAY
76          M[B + 1] = $            " STACK CBW
77          $ + DRT[4]              " CHANGE A = MD[CBW] IN MG[CBW] = B
78          DO($)                   " FILL NEW LRR IN DISPLAY
79          MC[- 3] = G             " STACK CURRENT D
80          M[63] = G               " SET NEW D
81          $ = 7
82          MC[- 7] = $             " FILL LRR[- 6]; B := VR
83          M[B - 2] = B            " FILL VR
84  R46E4(2)                        " ENTRY POINT SE74 AND SE44
85          $ = M[B - 9]            " TAKE LINK
86          M[B - 7] = F            " STACK INV. STARTING ADDRESS AND CONTEXT D
87  R46E2(10)                       " ENTRY POINT SE73
88          M[B - 3] = A            " STACK RETURN LRR
89          A = MA[- 2]             " OLD SV=LINK
90          M[B - 4] = A            " STACK SV LINK
91          A = :MS[0]              " COPY WITHOUT SMALL REGISTERS
92          $ '*' - 511             " :CRB[0] IN $(WITH SMALL REG.)
93          A '*' 511               " LINE NUMBER IN A
94          A + MS[0]               " INV. RETURN ADDRESS IN A
95          M[B - 9] = A            " STACK INV. RETURN ADDRESS
96          G = M[B - 7]            " INV. STARTING ADDRESS
97          SUBCD(:UBROFANELINK[1]) " PROFANATION VIA LINK
98  R46E3(16)                       " ENTRY POINT SE34 AND SE104
99          GL26 = G                " FILL CURRENT PROGRAM SEGMENT
100         $ = G                   " INV. STARTING ADDRESS IN $
101         RUS(9), P               " $ := ADDRESS SV
102     N,  $ '*' DRT[1]            " PRECAUTION SIGN BIT
103         A = MS[0], P            " SEGMENT ON CORE?
104     Y,  $ = DRT[2]
105     Y,  MA[3] + $               " INCREASE HOLYNESS COUNTER
106     Y,  $ = G                   " INV. STARTING ADDRESS
107     Y,  $ '*' 511               " LINE NUMBER
108     Y,  $ + MA[2]               " ADD :CRB[0]
109     Y,  $ + DRT[3]              " STANDARD FILLING SMALL REGISTERS
110     Y,  DO(MD[- 1])             " RESTORE A(SE46, SE34, SE104)
111     Y,  GOTOR($)                " RETURN
112         SUBCD(:SRQ)             " SEGMENT REQUEST
113         A = MS[0], P
114         JUMP(- 9)
```

non simple implicit subroutine

actual parameter is a piece of code in itself ("expression")

```
115 CHEENADRES(8)                       " CHEEN
116         MC[1] = A                   " SAVE A
117         A = M[B - 2]                " TAKE LINK
118         MC[- 1] = A                 " COPY LINK
119         A '*' - 511                 " CRB + SMALL REGISTERS
120         A - MA[0]                   " - INV. ADDRESS 0 - TH WORD
121         A - 1
122         M[B - 3] - A                " INVARIANT RETURN ADDRESS
123         GOTO(:RSE33[1])             " MERGE WITH $E33
124 CTERUGADRES(8)                      " CTERUG
125         MC[0] = S                   " SAVE S
126         S = M[B - 2]
127         M[B - 2] = A                " SAVE A
128         MC[0] = S                   " COPY LINK
129         SUBCD(:UPROFANELINK[1])     " PROFANATION VIA LINK
130         A = MC[- 1]                 " LINK IN A
131         S = M[B - 3]                " INVARIANT ADDRESS IN S
132         GOTO(:RSE33[6])             " MERGE WITH $E33
```

these are the equivalent of SUBC and GOTO (MC[-1]) but instead of an ordinary link, an invariant return-address is left on top of the stack

```
133
134
135 UEIOBARRIER(17)                                   " COORD
136         A = UMAXPOR                               " UEIOBARRIER - 0
137         A - UEIO, P                 " FIO < MAXPOR?
138     N,  UEIO - S                    " DEC(FIO, CURRENT PORTIONSIZE)
139     Y,  A = 12                      " PMSHR := 12
140     Y,  UAMVAR = A                  " REQUEST := CURRENT PORTION SIZE
141     Y,  UREQUEST = S                " V(UTREX)→leave critical section
142     Y,  SUBCD(:UTREXVADRES)         " P(PMSEM)
143     Y,  SUBCD(:UPOWNSEMADRES)       " P(UTREX)
144     Y,  SUBCD(:UTREXPADRES)         " INC(FIP, PORTION SIZE)
145         UEIR + S
146         GL24 - S
147         A = UBLOCKSTR               " ATT. TIME := PORTIONCLOCK
148         E = URORTIONCLOCK
149         MA[7] = E                   " INCREASE PORTIONCLOCK
150         E + UCLOCKINCR
151         URORTIONCLOCK = E
152         GOTOR(MC[- 1])

153 SCL(17)                             " STACK COLLAPSE
154         G = MG[- 2]                 " TARGET SV LINK IN G
155         A = MG[- 2], Z              " CHAIN TO NEXT LIST OR - 0
156     Y,  JUMP(10)
157         MG[- 2] - A                 " CLEAR CHAIN ELEMENT
158         G = A                       " ADDRESS FIRST SV IN CURRENT LIST
159         S = MG[- 1], Z              " LIST EMPTY?
160     Y,  JUMP(- 6)                   " EXAMINE NEXT LIST
161         S - 512
162         MG[- 1] = S                 " DEC(NUMBER OF SV * 512)
163         RUS(9)
164         S + G                       " ADDRESS CURRENT SV
165         SUBCD(:UKILLSEGP)           " KILL SEGMENT, P - DECREASE
166         JUMP(- 8)
167         S = :MC[- 1]                " TARGET AP(= TARGET WP)
168         S '*' - 511                 " :CRB[0] TARGET STACK PAGE
169         A = MS[0], Z                " IS THIS THE TOP STACK PAGE?
170     Y,  GOTOR(MC[- 1])              " RETURN
171 SCL1(6)                            " ENTRY POINT SE71, SE72
```

strategic subroutine

reason for blocking

wait until room becomes available. If it was available already then the P operation will be kicking an open door and the process continues, once more entering a critical section.

→ for instance upon completion of a goto to an outer block

```
172           MS[0] = A                    " SV := = 0
173           S = 1
174           GL25 = S                     " DEC(NUMBER OF BORROWED STACKPAGES)
175           SUBCD(:MECP)                 " MAKE FREE CORE PAGE
176           S = MA[2]                    " :CRB[0] FREE CORE PAGE
177           JUMP(= 8)

178 RSE73(9)                              " ACTUAL PROCEDURE ENTRY
179           B + 8                        " B ON TARGET VALUE
180           M[B = 8] = S                 " FILL LRP[6]
181           S = M[63]
182           M[B = 6] = S                 " FILL CONTEXT D
183           S = M[B = 9]                 " TAKE LINK
184           S + 1                        " RETURN ONE ABOVE LINK
185           G = MS[= 1]                  " INVARIANT STARTING ADDRESS
186           M[B = 7] = G                 " STACK INVARIANT STARTING ADDRESS
187           GOTO(:R46E2)                 " MERGE WITH NSIS ENTRY

188 RSE47(22)                             " SIS ENTRY   simple implicit subroutine (simple expression as actual parameter)
189           G = MS[3]                    " CONTEXT D
190           M[63] = G                    " SET CONTEXT D
191           S = MS[2]                    " INVARIANT STARTING ADDRESS
192           MC[0] = S                    " STACK INVARIANT STARTING ADDRESS
193           MC[0] = = A                  " STACK = RETURN LRP
194           A = 1
195           PLUSA(MG[= 1])               " INCREASE CBH
196           A + DRT[4]                   " CHANGE A = MD[CBH] IN MG[CBH] = B
197           DO(A)                        " SET LRP IN CURRENT DISPLAY
198           RUS(9), P                    " ADDRESS SV IN S
199       N,  S '*' DRT[1]                 " PRECAUTION MEMORY ABOVE 32K
200           A = MS[0], P                 " SEGMENT IN CORE?
201       Y,  S = DRT[2]
202       Y,  MA[3] + S                    " INCREASE HOLYNESS COUNTER
203       Y,  S = M[B = 2]                 " INVARIANT STARTING ADDRESS
204       Y,  S '*' 511                    " LINE NUMBER
205       Y,  S + MA[2]                    " ADD :CRB[0]
206       Y,  A = :MC[0]                   " A = :LRP[0]
207       Y,  GOTO(S)                      " JUMP TO SIS
208           SUBCD(:SRQ)
209           A = MS[0], P
210           JUMP(= 8)

211 RSE18(26)                             " CREATE DISPLAY → has to be called after procedure entry
212           G = MC[= 1]                  " STACK PAGE REQUEST   copies display elements belonging to context of
213           SUBCD(:SRP)                  " SAVE LINK OF SE18    procedure body and reserves enough display
214           UWS[0] = G                   " NEW LRP              space for inner blocks that may still
215           F = :MC[= 2]                 " GLOBAL VARIABLE NEW STACK PAGE   be entered.
216       N,  B = NSP                      " IN S TARGET AP = TARGET WP
217           S + :MC[0]                   " STACK WP
218           MG[0] = S                    " COPY FBH
219           S = :MA[0]                   " MAKE A = MD[FBH]
220           A + DRT[5]                   " STORE AS STANDARD LOCAL
221           MG[1] = A                    " STORE FBH
222           MC[0] = A                    " INITIALIZE CBH FOR NEW DISPLAY
223           MC[0] = A                    " CURRENT D IN STACK
224           MG[= 3] = B                  " SWITCH OVER TO CURRENT D
225           M[63] = B                    " S = :MD[FBH]
226           S + :MC[0]                   " CONTEXT D IN A
227           A = MG[= 4]                  " SAVE LRP
228           UWS[1] = G
```

```
229           E = MA[0]                          " TRANSPORTATION
230           MC[0] = E                          " OF TWO ELEMENTS
231           A + 2
232       U,  S - B, R
233       Y,  JUMR(- 5)
234           A = UWS[1]                         " RESTORE LRR
235           MS[0] = A
236           B = MA[0]                          " INITIALIZE AR TO WR
237           GOTOR(UWS[0])
238   RSE1(14)
239           S + :MC[UOVERSHOOT - 2)                "ANONYMOUS RESERVATION
240           S'+' B                                 "INITIAL AR + S + OVERSHOOT - 1
241       U,  S '*' - 511, Z                         "SPACE AVAILABLE IN CURRENT PAGE?
242       Y,  GOTOR(MC[ - 1])
243           S '+' B                                "RESTORE $
244           S - :MC[UOVERSHOOT - 2]
245           S + UOVERSHOOT[ - 510], R
246       Y,  A = 234
247       Y,  GOTO(:UDYNERR)
248           SUBCD(:SRR[6])
249           E = :MC[ - 1]
250           B = NSR
251           MA[0] = B
252           GOTOR(MG[0])

253   RSE6(9)                                    " BLOCK EXIT
254           S = 1
255           MD[- 1] - S                        " DECREASE CURRENT BLOCK HEIGHT
256           S = MC[- 1]                        " TAKE LINK
257           G = MA[- 1]                        " RETURN LRR IN G
258           B = MG[0]                          " B := WR
259           MC[0] = S                          " STACK LINK
260           SUBCD(:SCL)                        " STACK COLLAPS
261           DO(MD[- 1])                        " RESET A TO LRR
262           GOTOR(MC[- 1])                     " RETURN

263   RSE92(27)                                  " MORE DIMENSIONAL ARRAY SUBSCRIPTION
264           G = MC[- 1]                        " TAKE LINK
265           UWS[0] = G                         " SAVE LINK
266           G = MS[1]                          " INCREMENT K * R[N]
267           S = MS[0]                          " :B[0]
268           UWS[1] = S                         " SAVE :B[0]
269       U,  A - MS[- 1], Z                     " DIMENSION CHECKING
270       Y,  JUMR(3)
271           GOTO(:ONZIN11)                     " DIMENSION ERROR
272           S - 2
273           G * MS[- 4]                        " TIMES = RANGE
274           UWS[2] = - G                       " - PARTIAL SUM
275           G = MC[- 1]                        " TAKE = SUBSCRIPT
276           G + MS[- 3], R                     " BOUNDCHECKING
277           G + MS[- 4], E
278       Y,  GOTO(:ONZIN12)                     " INDEX OUT OF BOUNDS
279           G + UWS[2]                         " - PARTIAL SUM(- 0 PREFERENCE)
280           A - 1, Z
281       N,  JUMR(- 10)                         " NEXT SUBSCRIPT
282           A = G                              " - ORDINAL NUMBER(- 0 PREFERENCE)
283           G = UWS[1]                         " RESTORE :B[0]
284           DO(MG[1])                          " TIMES SIZE
285           G = MG[- 2]
286           A + MG[- 1], R
```

reservation of anonymous stack space for evaluation of expressions done upon block entry for the maximum needed within that block.

```
287      N.  JUMP(- 3)
288          A = - A
289          A + MG[- 3]
290          GOTOR(UWS[0])                  " RETURN WITH INVARIANT ADDRESS
291                                         " IN A AND NO CONDITION

292 R92E1(10)
293          S = - A                        " INITIAL ORDINAL NUMBER IN S(+ 0 PREFERENCE)
294          A = 0
295          DIV4S(27)
296          UWS[2] = S                     " SAVE REDUCED ORDINAL NUMBER
297          A + DRT[17]                    " ADD LCS(0)
298          S = 1
299          DO(A)                          " MAKE POSITION IN WORD, SEE SE90 - 2
300          GL17 = S                       " STORE POSITION IN WORD
301          A = - UWS[2]                   " - REDUCED ORDINAL NUMBER
302          GOTOR(MC[- 1])

303 R91E3(26)                               " VECTOR SUBSCRIPTION
304          G + G                          " COMPLEX ARRAY IN STACK
305          G + G                          " REAL DITO
306          S = MS[1]                      " INTEGER OR BOOLEAN DITO, S := ADDRESS FIRST WORD
307          S - G, P                       " PHYSICAL ADDRESS IN S
308          GOTO(MC[- 1])                  " RETURN WITH YES CONDITION
309          G + G                          " COMPLEX ARRAY ON SEGMENT ONE SV LIST
310          G + G                          " REAL DITO
311          A = - G                        " INTEGER DITO
312          A + MS[1], Z                   " INVARIANT ADDRESS IN A
313          GOTO(MC[- 1])                  " RETURN WITH NO CONDITION
314          G + G                          " COMPLEX ARRAY ON SEGMENT MORE SV LISTS
315          G + G                          " REAL DITO
316          A = G                          " INTEGER DITO
317          A - MS[1]                      " ORDINAL NUMBER NEGATIV IN A
318          G = MS[0]                      " :B[0]
319          G = MG[- 2]                    " :R[0] SV LIST
320          A + MG[- 1], P                 " ELEMENT IN THIS LIST?
321      N.  JUMP(- 3)                      " TRY NEXT LIST
322          A = - A
323          A + MG[- 3], Z                 " INVARIANT ADDRESS IN A
324          GOTO(MC[- 1])                  " RETURN WITH NO CONDITION
325          A = G                          " BOOLEAN ARRAY ON SEGMENT
326          A - MS[1]                      " ORDINAL NUMBER NEGATIV IN A
327          G = MS[0]                      " :B[0]
328          SUBCD(:R92E1)                  " MAKE REDUCED NEGATIV ORDINAL NUMBER
329          JUMP(- 11)

330 R90E0(34)
331          S + A                          " * 11
332          S + A                          " * 10
333          S + A                          " * 9
334          LCA(3)
335          GOTO(:R90E0[20])
336          S - A                          " * 6
337          S - A                          " * 7
338          LCA(3)                         " * 8
339          GOTO(:R90E0[20])
340          S + A                          " * 5
341          LCA(2)                         " * 4
342          GOTO(:R90E0[20])
343          S - A                          " * 13
```

don't bother, very complicated without additional information

programmed multiplication

```
344           S - A                          " * 14
345           S - A                          " * 15
346           LCA(4)
347           GOTO(:R90E0[20])
348           LCA(2)                         " * 12
349           S + A                          " * 3
350           LCA(1)                         " * 2
351           A + S
352           DO(MG[2])                      " * SIZE
353           A = - A
354           A + MG[6], Z                   " ADD POSITION BOUNDARY LAYER
355           GOTO(MC[0])                    " RETURN WITH NO CONDITION
356           LUA(1)
357           LUA(1)
358           A - MG[6]                      " - POSITION BOUNDARY LAYER, A := - ORDINAL NUMBER
359           G = MG[- 2]
360           A + MG[- 1], R
361       N,  JUMP(- 3)
362           A = - A
363           A + MG[- 3], Z
364           GOTO(MC[0])

365 R90E33(7)
366           LUA(3)
367           RCSA(3)
368           LCA(5)
369           S + MG[7]
370           A = - A
371           A + MS[- 1], Z
372           GOTO(MC[0])
373 R90E36(3)
374           LUA(3)
375           RCSA(3)
376           GOTO(:R90E35[2])

377 R90E31(7)
378           LUA(4)
379           RCSA(4)
380           LCA(4)                         " A = - ORDINAL NUMBER IN UPPERLAYER, S = - NUMBER UPPERLAYER
381           S + MG[7]                      " ADD BASE MULTIPLICATION TABLE
382           A = - A                        " A := ORDINAL NUMBER IN UPPERLAYER
383           A + MS[- 1], Z                 " ADD CORRECTION UPPERLAYER AND FIRST POSITIONING
384           GOTO(MC[0])                    " RETURN  INV. ADDRESS IN A AND CONDITION NEGATIV
385 R90E34(4)
386           LUA(4)
387           RCSA(4)
388           LCA(4)
389           GOTO(:R90E35[3])

390 R90E32(7)
391           LUA(4)
392           RCSA(4)
393           LCA(5)
394           S + MG[7]
395           A = - A
396           A + MS[- 1], Z
397           GOTO(MC[0])
398 R90E35(11)
399           LUA(4)
400           RCSA(4)
```

```
401          LCA(5)                    " A = - ORDINAL NUMBER IN UPPERLAYER, S = - NUMBER UPPERLAYER
402          S + MG[7]                 " ADD BASE MULTIPLICATION TABLE
403          A - MS[- 1]               " - CORRECTION UPPERLAYER, A := - ORDINAL NUMBER
404          G = MG[- 2]               " :B[0] SV LIST
405          A + MG[- 1], B            " ADD CRITERIUM TO NEGATIV ORDINAL NUMBER
406      N,  JUMP(- 3)
407          A = - A
408          A + MG[- 3], Z            " ADD CRITERIUM AND POSITION
409          GOTO(MC[0])               " RETURN WITH NO CONDITION

410 RSE90(12)                          " SUBSCRIPTION HORIZONTAL MATRIX
411          A = G                     " A := - SECOND SUBSCRIPT = - J
412          G = MS[0]                 " :B[0] REFERENCE ADDRESS MAPPING TABLE
413          A + MG[- 3], B            " ADD UB[J] + 1, REQUIRED LT = +
414          A + MG[- 4], E            " ADD LB[J] - UB[J] - 1, REQUIRED LT = -
415      N,  A - MS[1], B              " - COLUMN CORRECTION, REQUIRED LT = -
416          S = MC[- 2]               " S := - FIRST SUBSCRIPT = - I
417      N,  S + MG[- 5], E            " ADD UB[I] + 1, REQUIRED LT = +
418      N,  S + MG[- 6], E            " ADD LB[I] - UB[I] - 1, REQUIRED LT = -
419      Y,  GOTO(:ONZIN6)             " INDEX OUT OF BOUNDS
420      U,  S + MG[5], B              " ADD NUMBER OF ROWS IN UPPERLAYERS, IC
421      Y,  GOTO(MG[3])               " ELEMENT IN UPPERLAYERS
422          GOTO(MG[4])               " ELEMENT IN BOUNDARYLAYER
423 R90E21(11)                         " SUBSCRIPTION VERTICAL MATRIX
424          A = MC[- 2]               " A := - FIRST SUBSCRIPT, FUTURE J
425          MC[- 1] = G
426          G = MS[0]                 " :B[0] REFERENCE ADDRESS MAPPING TABLE
427          A + MG[- 5], B            " ADD UB[I] + 1
428          A + MG[- 6], E            " ADD LB[I] - UB[I] - 1
429      N,  A - MS[1], B              " - COLUMN CORRECTION
430          S = MC[- 2]               " S := - SECOND SUBSCRIPT, FUTURE I
431      N,  S + MG[- 3], E            " ADD UB[J] + 1
432      N,  S + MG[- 4], E            " ADD LB[J] - UB[J] - 1
433      N,  GOTO(:RSE90[9])           " MERGE WITH RSE90
434          GOTO(:ONZIN7)             " INDEX OUT OF BOUNDS

435 RSE75(3)                           " FILL FORMAL REAL CONSTANT
436          SUBC(:B89E1)
437          E = MS[2]
438          + 130
439 RSE76(3)                           " FILL FORMAL INTEGER CONSTANT
440          SUBC(:B89E1)
441          G = MS[3]
442          + 129
443 RSE89(3)                           " FILL FORMAL BOOLEAN CONSTANT
444          SUBC(:B89E1)
445          G = MS[3]
446          + 136
447 B89E1(6)
448          E = MS[1]                 " TAKE F[0] AND F[1]
449          S = MC[- 1]               " TAKE LINK
450          MC[0] = E                 " STACK F[0] AND F[1]
451          E = MS[0]                 " TAKE F[2] AND F[3]
452          MC[0] = E                 " STACK F[2] AND F[3]
453          GOTO(:MS[2])

454 RSE77(19)                          " FILL FORMAL REAL VARIABLE
455          E = 258                   " PAR, CODE IN F[1]
456          MC[0] = G                 " SAVE LINK
457          G = MC[- 2]
```

handing over actuals before procedure entry.

```
458       U,  S = 32767, R                  " A ≥ 32K
459       Y,  JUMR(7)                       " PRO FORMA
460           S + DRT[7]
461           MC[- 1] = S                   " F[0] := E = M[A]
462           S + DRT[8]                    " MAKE M[A] = E
463           MC[0] = S                     " F[2] := M[A] = E
464           S = DRT[9]
465           MC[0] = S
466           GOTOR(G)                      " RETURN
467           MC[1] = S                     " F[2] := A
468           S = MT[3]
469           MC[- 2] = S                   " F[0] := SE36
470           S = MT[2]
471           JUMR(- 7)
472           SUBCD(:RSE36)
473           SUBCD(:RSE51)

474 RSE94(5)                               " FILL FORMAL BOOLEAN VARIABLE
475           E = 264                       " TAKE PARAMETERCODE
476           MC[0] = G                     " STACK PARAMETERCODE
477           E = - 0                       " TAKE F[3] IN G
478           JUMR(4)
479           SUBCD(:RSE49)
480 RSE78(14)                               " FILL FORMAL INTEGER VARIABLE
481           E = 257
482           MC[0] = G                     " STACK PARAMETERCODE
483           G = MT[- 4]                   " TAKE F[3] IN G
484           MC[0] = S                     " STACK PHYSICAL ADDRESS
485           MC[0] = G                     " STACK F[3]
486           G = M[B - 4]                  " TAKE LINK
487       U,  S = 32767, R                  " A > 32K
488       Y,  JUMR(3)
489           S + DRT[10]                   " MAKE G = M[A]
490           M[B - 4] = S                  " FILL F[0]
491           GOTO(G)
492           S = MT[1]
493           JUMR(- 4)
494           SUBCD(:RSE37)

495 RSE79(4)                               " FILL FORMAL INTEGER ARRAY
496           SUBCD(:R79E1)
497           + 513                         " PARCODE INTEGER ARRAY
498           SUBCD(:RSE38)                 " SE38
499           SUBCD(:RSE39)                 " SE39
500 R79E1(8)
501           E = MS[1]
502           MC[0] = E                     " FILL F[1] AND F[2]
503           S = MS[3]
504           MC[0] = S                     " FILL F[3]
505           S = M[B - 4]                  " SAVE LINK
506           M[B - 4] = A                  " FILL F[0]
507           DO(MD[- 1])                   " RESTORE LRR IN A
508           GOTOR(S)
509 RSE80(4)                               " FILL FORMAL REAL ARRAY
510           SUBCD(:R79E1)
511           + 514                         " PARCODE REAL ARRAY
512           SUBCD(:RSE40)                 " SE40
513           SUBCD(:RSE41)                 " SE41
514 RSE81(4)                               " FILL FORMAL COMPLEX ARRAY
515           SUBCD(:R79E1)
```

These fill formals are created in the object-code by the translator

```
516             + 516                           " PARCODE COMPLEX ARRAY
517             SUBCD(:RSE42)                   " SE42
518             SUBCD(:RSE43)                   " SE43

519 RSE82(8)
520             G = MS[1]                       " I.E. F[0]
521             S = MC[- 1]                     " SAVE LINK
522             MC[0] = G                       " STORE F[0]
523             F = MS[0]                       " TAKE LAST WORDS CALLING SEQUENCE
524             MC[0] = F                       " STORE F[1], F[2]
525             G = M[63]                       " TAKE CONTEXT D
526             MC[0] = G                       " STORE F[3]
527             GOTO(:MS[2])                    " RETURN UNDER CALLING SEQUENCE
528 R82E1(2)                                    " FILL FORMAL PROCEDURE OR SWITCH
529             SUBC(:RSE82)
530             SUBCD(:RSE44)
531 R82E2(2)                                    " FILL FORMAL NS|S
532             SUBC(:RSE82)
533             SUBCD(:RSE46)
534 R82E3(2)                                    " FILL FORMAL S|S
535             SUBC(:RSE82)
536             SUBCD(:RSE47)

537 RSE83(9)                                    " FILL FORMAL LABEL
538             MC[2] = S                       " STORE F[3]
539             S = MC[- 2]                     " SAVE LINK
540             G = MS[0]                       " TAKE INVARIANT TARGET ADDRESS
541             MC[1] = G                       " STORE F[2]
542             F = MT[2]                       " TAKE SE48, + 144
543             MC[- 2] = F                     " STORE F[0] AND F[1]
544             GOTO(:MS[1])
545             SUBCD(:RSE48)
546             + 144

547 RSE84(10)                                   " FILL FORMAL COMPLEX VARIABLE
548             MC[1] = S                       " STORE F[2]
549             S = MC[- 2]                     " SAVE LINK
550             F = MT[4]                       " TAKE SE55, + 260
551             MC[- 1] = F                     " STORE F[0], F[1]
552             G = MT[4]                       " TAKE SE53
553             MC[0] = G                       " STORE F[3]
554             GOTO(S)                         " RETURN
555             SUBC(:RSE55)
556             + 260
557             SUBCD(:RSE53)

558 RSE85(6)                                    " FILL FORMAL FORMAL
559             F = MS[2]
560             MC[1] = F                       " STORE F[2], F[3]
561             F = MS[0]
562             S = MC[- 3]                     " SAVE LINK
563             MC[- 2] = F                     " STORE F[0], F[1]
564             GOTO(S)

565 RSE86(13)                                   " FILL FORMAL FORMAL ARRAY ELEMENT
566             S '+' 768                       " INVERSION D[8] AND D[9], ARRAY BECOMES VARIABLE
567             MC[0] = S                       " STORE F[1]
568             G = MC[- 2]                     " SAVE LINK
569       Y,    S = MT[7]                       " FOR S|S
570       N,    S = MT[7]                       " FOR NS|S
```

```
571           MC[- 1] = S                    " STORE F[0]
572           S = MG[0]                      " INV. STARTING ADDRESS
573           MC[0] = S                      " STORE F[2]
574           S = M[63]                      " CONTEXT D
575           MC[0] = S                      " STORE F[3]
576           GOTO(:MG[1])                   " RETURN
577           SUBCD(:RSE47)                  " FOR $|$
578           SUBCD(:RSE46)                  " FOR N$|S

579 RSE95(4)                                 " FILL FORMAL BOOLEAN ARRAY
580           SUBCD(:R79E1)                  " MERGE WITH SE79
581           + 520                          " PAR.CODE BOOLEAN ARRAY
582           SUBCD(:RSE106)                 " SE106
583           SUBCD(:RSE107)                 " SE107

584 RSE112(4)                                " FILL FORMAL STRING
585           MC[0] = E                      " FILL F[2]
586           S = 2080
587           MC[- 2] = S                    " FILL F[1]
588           GOTOR(M[B - 4])                " RETURN

589 RSE20(4)                                 " CHECK FORMAL ARITMETIC
590           S = MG[1]
591      U,   S '*' 3708, Z
592      Y,   GOTOR(MC[- 1])
593      U,   S '*' 3068, Z
594 R20E1(1)
595      Y,   S - 896
596 R20E2(3)
597      Y,   MG[1] = S
598      Y,   GOTOR(MC[- 1])
599      U,   S '*' 3771, Z
600 R20E3(5)
601      Y,   S '*' - 64
602      Y,   S '+' 64
603      Y,   JUMP(- 6)
604           A = 207
605           GOTO(:UDYNERR)

606 RSE21(7)
607           S = MG[1]                      " CHECK FORMAL BOOLEAN
608      U,   S '*' 3703, Z
609      Y,   GOTOR(MC[- 1])
610      U,   S '*' 3063, Z
611      Y,   GOTO(:R20E1)
612           A = 208
613           GOTO(:UDYNERR)

614 RSE22(13)                                " CHECK FORMAL ARITHMETIC ARRAY
615           S = MG[1]
616      U,   S '*' 3580, Z
617      Y,   GOTOR(MC[- 1])
618      U,   S '*' 3515, Z
619      Y,   S '*' - 64
620      Y,   S '+' 64
621      Y,   MG[1] = S
622      Y,   S = MT[4]
623      Y,   MG[3] = S
624      Y,   GOTOR(MC[- 1])
625           A = 210
```

etc. etc.

dynamic typechecking routines
These are supplied in the beginning of the procedure bodies.

```
626          GOTO(:UDYNERR)
627          SUBCD(:RSE56)

628 RSE23(5)                              " CHECK FORMAL BOOLEAN ARRAY
629          $ = MG[1]
630      U,  $ '*' 3575, Z
631      Y,  GOTOR(MC[- 1])
632          A = 211
633          GOTO(:UDYNERR)

634 RSE24(7)                              " CHECK FORMAL PROCEDURE
635          $ = MG[1]
636      U,  $ '*' 2960, Z
637      Y,  $ '*' - 4095
638      Y,  $ '+' 1056
639      Y,  GOTO(:R20E2)
640          A = 213
641          GOTO(:UDYNERR)

642 RSE25(5)                              " CHECK FORMAL ARITHMETIC PROCEDURE
643          $ = MG[1]
644      U,  $ '*' 3068, Z
645      Y,  GOTOR(MC[- 1])
646          A = 214
647          GOTO(:UDYNERR)

648 RSE26(5)                              " CHECK FORMAL BOOLEAN PROCEDURE
649          $ = MG[1]
650      U,  $ '*' 3063, Z
651      Y,  GOTOR(MC[- 1])
652          A = 215
653          GOTO(:UDYNERR)

654 RSE27(5)                              " CHECK FORMAL LABEL
655          $ = MG[1]
656      U,  $ '*' 3951, Z
657      Y,  GOTOR(MC[- 1])
658          A = 217
659          GOTO(:UDYNERR)

660 RSE28(5)                              " CHECK FORMAL SWITCH
661          $ = MG[1]
662      U,  $ '*' 3055, Z
663      Y,  GOTOR(MC[- 1])
664          A = 218
665          GOTO(:UDYNERR)

666 RSE29(5)                              " CHECK FORMAL STRING
667          $ = MG[1]
668      U,  $ '*' 2015, Z
669      Y,  GOTOR(MC[- 1])
670          A = 219
671          GOTO(:UDYNERR)

672 RSE30(11)                             " CHECK FORMAL COMPLEX
673          $ = MG[1]
674      U,  $ '*' 3707, Z
675      Y,  GOTOR(MC[- 1])
676      U,  $ '*' 3067, Z
677      Y,  GOTO(:R20E1)
```

```
678       U,  $ '*' 2620, Z
679       Y,  $ '*' - 4092
680       Y,  $ + 192
681       Y,  GOTO(:R20E2)
682           A = 209
683           GOTO(:UDYNERR)

684 RSE31(13)                              " CHECK FORMAL COMPLEX ARRAY
685           $ = MG[1]
686       U,  $ '*' 3579, Z
687       Y,  GOTOR(MC[- 1])
688       U,  $ '*' 3516, Z
689       Y,  $ '*' - 64
690       Y,  $ '+' 64
691       Y,  MG[1] = $
692       Y,  $ = MT[4]
693       Y,  MG[2] = $
694       Y,  GOTOR(MC[- 1])
695           A = 212
696           GOTO(:UDYNERR)
697           SUBCD(:RSE70)

698 RSE32(7)
699           $ = MG[1]
700       U,  $ '*' 3067, Z
701       Y,  GOTOR(MC[- 1])
702       U,  $ '*' 3004, Z
703       Y,  GOTO(:R20E3)
704           A = 216
705           GOTO(:UDYNERR)

706 RSE10(42)                              " COMPLEX DIVISION
707       U,  $ = M[B - 2], R              " TAIL OF D POSITIV
708           M[B] = E, E                  " STORE C
709       N,  JUMR(19)
710           E - M[B - 3], E              " E - D
711       Y,  JUMR(19)                     " |C|≥|D|
712           E = M[B], Z                  " |C|≤|D|, E := C
713       N,  E/M[B - 3]                   " E/D
714           M[B + 2] = E                 " STORE C/D
715       N,  E * M[B]                     " E * C
716       N,  E + M[B - 3]                 " E + D
717       N,  M[B - 3] = E                 " D := C, C/D + D
718           E = M[B - 7], Z              " E := B
719           M[B] = E                     " STORE B
720       N,  E * M[B + 2]                 " E * C/D
721           E - M[B - 5], Z              " E - A
722       N,  E/M[B - 3]
723           M[B - 7] = E                 " STACK V
724           E = MC[- 5], Z               " E := A
725       N,  E * M[B + 4]                 " E * C/D
726           E + MC[+ 2], Z               " E + B
727       N,  E/M[B + 1]                   " E := U
728           GOTOR(MC[3])                 " RETURN
729           E - M[B - 3], E
730       N,  JUMR(- 19)                   " |C|≤|D|
731           E = M[B - 3], Z              " |C|≥|D|, E := D
732       N,  E/M[B]                       " E/C
733           M[B + 2] = E                 " STORE D/C
734       N,  E * M[B - 3]                 " E * D
```

```
735       N,  E + M[B]                          " E + C
736       N,  M[B] = E                          " C := C + D. D/C
737       N,  E = M[B - 7]                      " E := B
738       N,  M[B - 3] = E                      " STORE B
739       N,  E = - M[B + 2]                    " E := - D/C
740       N,  E * M[B - 5]                      " E * A
741           E + M[B - 7], Z                   " E + B
742       N,  E/M[B]                            " E := V
743           M[B - 7] = E                      " STACK V
744           E = MC[2], Z                      " E := D/C
745       N,  E * M[B - 1]                      " E * B
746           E + MC[- 3], Z                    " E + A
747       N,  E/M[B + 4]                        " E := U
748           GOTOR(MC[3])                      " RETURN

749 RSE87(3)                                   " CHECK COMPLEX LEFT HAND TYPE
750           SUBCD(:R9E2)
751           SUBCD(:RSE53)                     " ASSIGN TO STACKED COMPLEX
752           SUBCD(:RSE54)                     " ASSIGN TO COMPLEX ON SEGMENT
753 RSE8(3)                                    " CHECK INTEGER LEFT HAND TYPE
754           SUBCD(:R9E2)
755           SUBCD(:RSE49)                     " ASSIGN TO STACKED INTEGER
756           SUBCD(:RSE50)                     " ASSIGN TO INTEGER ON SEGMENT
757 RSE9(4)                                    " CHECK REAL LEFT HAND TYPE
758           SUBCD(:R9E1)
759           DO(MC[- 1])                       " ASSIGN TO REAL IN STACK
760           SUBCD(:RSE52)                     " ASSIGN TO REAL ON SEGMENT
761           SUBCD(:RSE51)                     " ASSIGN TO HIGH STACKED REAL

762 R9E1(1)
763       Y,  UWS[1] = - B                      " INDICATION TYPE UNKNOWN
764 R9E2(18)
765           UWS[0] = B                        " SAVE B
766       N,  UWS[1] = B                        " INDICATION TYPE INTEGER, REAL, COMPLEX
767           A = MC[- 2]                       " TAKE LEFT HAND
768       U,  A - MS[1], Z
769       N,  A - MS[2], Z
770           A = MC[- 1]
771       N,  A - MS[3], Z
772       N,  JUMP(3)                           " INCORRECT TYPE?
773           E - 1, Z                          " LAST LEFT HAND?
774       N,  JUMP(- 9)                         " NEXT LEFT HAND
775           DO(MD[- 1])                       " RESTORE A
776           B = UWS[0]                        " RESTORE B
777       Y,  GOTOR(MC[- 1])                    " RETURN
778       U,  B = UWS[1], P                     " TYPE INTEGER, REAL OR COMPLEX
779       N,  S - 3
780       N,  JUMP(- 14)                        " TRY NEXT TYPE
781           A = 220
782           GOTO(:UDYNERR)

783 RSE36(3)                                   " HIGH STACK REAL
784           G = MS[2]                         " ADRES A IN G
785           E = MG[0]                         " REAL VALUE IN E
786           GOTOR(MC[- 1])                    " RETURN

787 RSE37(3)                                   " HIGH STACK INTEGER AND BOOLEAN
788           G = MS[2]                         " ADRES A IN G
789           G = MG[0]                         " VALUE IN E
790           GOTOR(MC[- 1])                    " RETURN
```

high stack, i.e. stack page > 32K
here static addressing is impossible.

```
791 RSE48(3)                            " ACTUAL LABEL VALUE
792         F = MS[2]                   " TAKE F[2] AND F[3]
793         GL0 = F                     " STORE IN GLOBAL LABEL
794         GOTOR(MC[- 1])              " RETURN

795 RSE51(3)                            " ASSIGN TO HIGH STACKED REAL
796         S = MC[- 2]                 " ADDRESS
797         MS[0] = F                   " ASSIGN VALUE
798         GOTOR(MC[0])

799 RSE53(1)                            " ASSIGN TO STACKED COMPLEX
800         S = MC[- 2]                 " TAKE ADDRESS A
801 R53E1(5)
802         MS[2] = F                   " ASSIGN REAL PART
803         F = GIR                     " TAKE IMAGINAIR PART
804         MS[0] = F                   " ASSIGN IMAGINARY PART
805         F = MS[2]                   " REAL PART IN F
806         GOTOR(MC[0])
807 RSE57(8)
808         F = MS[2]                   " TAKE LEFT HAND VALUE
809         S = MS[1]                   " TAKE PARAMETER CODE IN S
810     U,  S '*' 3712, Z               " EXISTENCE OF LEFT HAND VALUE
811     Y,  S = MC[- 1]                 " TAKE LINK
812     Y,  MC[0] = F                   " STACK LEFT HAND VALUE
813     Y,  GOTOR(S)                    " RETURN
814         A = 224
815         GOTO(:UDYNERR)

816 RSE63(1)                            " FORMAL RIGHT SUBSCRIPTED COMPLEX
817         SUBCD(:R62E1)
818 RSE43(5)                            " RIGHT SUBSCRIPTED COMPLEX
819     N,  SUBCD(:R39E1)               " PHYSICAL ADDRESS IN S
820         F = MS[0]
821         M[B - 3] = F                " STACK IMAGINARY PART
822         F = MS[2]                   " REAL PART IN F
823         GOTOR(MC[- 1])

824 RSE58(1)                            " FORMAL LEFT SUBSCRIPTED INTEGER
825         SUBCD(:R58E1)
826 RSE38(4)                            " LEFT SUBSCRIPTED INTEGER
827         F = :MT[1]
828         GOTO(:R40E1)                " MERGE WITH SE40
829         SUBCD(:RSE50)               " ASSIGN TO INTEGER ON SEGMENT
830         SUBCD(:RSE49)               " ASSIGN TO STACKED INTEGER

831 RSE62(1)                            " FORMAL LEFT SUBSCRIPTED COMPLEX
832         SUBCD(:R62E1)
833 RSE42(4)                            " LEFT SUBSCRIPTED COMPLEX
834         F = :MT[1]
835         GOTO(:R40E1)                " MERGE WITH SE40
836         SUBCD(:RSE54)               " ASSIGN TO COMPLEX ON SEGMENT
837         SUBCD(:RSE53)               " ASSIGN TO STACKED COMPLEX

838 RSE60(1)                            " FORMAL LEFT SUBSCRIPTED REAL
839         SUBCD(:R58E1)
840 RSE40(1)                            " LEFT SUBSCRIPTED REAL
841         F = :MT[12]
842 R40E1(14)
843     Y,  JUMP(6)
```

```
844             G = MG[0]
845             MC[0] = G                       " STACK $UBCD(:RSE52, 50, 54)
846             G = M[B - 2]                    " SAVE LINK
847             M[B - 2] = A                    " STACK INVARIANT ADDRESS
848             DO(MD[- 1])                     " RESTORE A
849             GOTOR(G)
850             G = MG[1]
851             MC[0] = G                       " STACK $UBCD(:RSE51, 49, 53)
852             G = M[B - 2]                    " SAVE LINK
853             M[B - 2] = S                    " STACK PHYSICAL ADDRESS
854             GOTOR(G)
855             SUBCD(:RSE52)                   " ASSIGN TO REAL ON SEGMENT
856             SUBCD(:RSE51)                   " ASSIGN TO HIGH STACKED REAL

857 R62E1(3)
858             S = :ERV
859             S + DRT[13]                     " MAKE DO(ERV)
860             ECV = S                         " RESET ECV STANDARD
861 R58E1(7)
862             S = G, R                        " RESTORE RESULT INDEXING
863         N,  A = M[57]                       " PROCES, AND CONDITION
864             E = MT[2]
865             ELV = E                         " RESET ELV, ERV STANDARD
866             GOTO(MC[- 1])                   " MERGE
867         ,   SUBCD(:RSE57)                   " STANDARD VALUE ELV
868         U,  S = 0                           " STANDARD VALUE ERV

869 RSE65(3)                                   " TRANSMIT SUBSCRIPTED INTEGER
870             SUBCD(:R67E1)
871             SUBCD(:RSE58)                   " FORMAL LEFT SUBSCRIPTED INTEGER
872             SUBCD(:RSE59)                   " FORMAL RIGHT SUBSCRIPTED INTEGER
873 RSE66(3)                                   " TRANSMIT SUBSCRIPTED REAL
874             SUBCD(:R67E1)
875             SUBCD(:RSE60)                   " FORMAL LEFT SUBSCRIPTED REAL
876             SUBCD(:RSE61)                   " FORMAL RIGHT SUBSCRIPTED REAL
877 RSE67(6)                                   " TRANSMIT SUBSCRIPTED COMPLEX
878             G = MT[4]
879             ECV = G                         " SET ECV NON STANDARD
880             SUBCD(:R67E1)
881             SUBCD(:RSE62)                   " FORMAL LEFT SUBSCRIPTED COMPLEX
882             SUBCD(:RSE64)                   " ALARM COMPLEX VALUE IN ARITHMETIC
883             SUBCD(:RSE63)                   " FORMAL RIGHT SUBSCRIPTED COMPLEX
884 R67E1(7)
885             G = MC[- 1]
886             E = MG[0]
887             ELV = E                         " SET ELV, ERV NON STANDARD
888         N,  S = - 0                         " SAVE RESULT OF
889             E = A                           " INDEXING PROCES IN E
890         N,  DO(MD[- 1])                     " RESTORE A
891             GOTOR(MC[- 1])

892 RSE68(9)
893         ,   SUBCD(:MT[5])                   " SAVE CONDITION
894             JUMP(- G)
895             GOTO(:RSE67)                    " MERGE WITH SE67
896             - 0                             " UNUSED
897             GOTO(:RSE66)                    " MERGE WITH SE66
898             GOTO(:RSE65)                    " MERGE WITH SE65
899             E - 516, R                      " E := -3, -2, -0 IF TYPE
900         Y,  E - 64                          " ARRAY INTEGER, REAL, COMPLEX
```

```
901           GOTOR(MC[- 1])                  " RESTORE CONDITION

902 RSE54(2)                                  " ASSIGN TO COMPLEX ON SEGMENT
903           SUBCD(:R50E1)                   " PHYSICALL ADDRESS IN $
904           GOTO(:R53E1)                    " MERGE WITH SE53

905 RSE55(10)                                 " STACK COMPLEX
906           G = M[B - 1], P                 " TAKE LINK
907           G = MG[0]                       " TAKE DO(FLV, FRV, FCV)
908           F + 3                           " MAKE DO(FLVM, FRVM, FCVM)
909           DO(G)
910           F = MS[0]                       " IMAGINARY PART IN F
911           M[B - 3] = F                    " STACK IMAGINARY PART
912           F = MS[2]                       " REAL PART OR LEFT HAND VALUE
913           S = MC[- 1]                     " TAKE LINK
914       Y,  MC[0] = F                       " STACK LEFT HAND VALUE
915           GOTO(:MS[1])                    " RETURN

916 RSE105(14)                                " ASSIGN TO BOOLEAN
917           S = MC[- 2], Z                  " BOOLEAN IN STACK?
918       N,  JUMP(3)
919           S = MC[- 2]                     " PHYSICAL ADDRESS
920           MS[0] = G                       " ASSIGN
921           GOTOR(MC[1])                    " RETURN
922           MC[1] = G, P                    " SAVE BOOLEAN VALUE IN CONDITION
923           SUBCD(:R50E1)                   " MAKE PHYSICAL ADDRESS IN $
924                                           " A BREAK MAY OCCUR
925           G = S                           " PHYSICAL ADDRESS IN G
926           S = MS[0]                       " WORD IN $
927           S '*' - MC[- 2]                 " BOOLEAN BIT := 0(TRUE)
928       N,  S '*' M[B - 1]                  " BOOLEAN BIT := 1(FALSE)
929           MG[0] = S                       " ASSIGN
930       N,  F = - 0                         " RESTORE SIGN F(CONSISTENT)
931           GOTOR(MC[1])

932 RSE110(1)                                 " FORMAL RIGHT SUBSCRIPTED BOOLEAN
933           SUBCD(:R58E1)
934 RSE107(6)                                 " RIGHT SUBSCRIPTED BOOLEAN
935       Y,  GOTO(:R39E2)                    " BOOLEAN IN STACK
936           SUBCD(:R39E1)                   " MAKE PHYSICAL ADDRESS IN $
937                                           " A BREAK MAY OCCUR
938           S = MS[0]                       " WORD IN $
939           S '*' GL17, Z                   " BOOLEAN BIT := 0(TRUE)?
940           G = M[62]                       " BOOLEAN VALUE IN G(CONSISTENT)
941           GOTOR(MC[- 1])                  " RETURN

942 RSE109(1)                                 " FORMAL LEFT SUBSCRIPTED BOOLEAN
943           SUBCD(:R58E1)                   " LEFT SUBSCRIPTED BOOLEAN
944 RSE106(7)
945           G = M[B - 1]                    " TAKE LINK IN G
946       Y,  M[B] = - F                      " - 0 IN LEFT HAND VALUE IN STACK
947       N,  S = GL17                        " S := POSITION IN $
948           MC[- 1] = S                     " STACK PHYSICAL ADDRESS OR POSITION
949       N,  M[B - 1] = A                    " STACK INVARIANT ADDRESS
950       N,  DO(MD[- 1])                     " RESTORE A
951           GOTOR(G)                        " RETURN

952 RSE108(3)                                 " TRANSMIT SUBSCRIPTED BOOLEAN
953           SUBCD(:R67E1)
954           SUBCD(:RSE109)                  " FORMAL LEFT SUBSCRIPTED BOOLEAN
```

```
955           SUBCD(:RSE110)                  " FORMAL RIGHT SUBSCRIPTED BOOLEAN

956 RSE111(14)                                " COMPLEX MULTIPLICATION
957           M[B] = F, Z                     " STACK C
958       N,  F * M[B - 7]                    " C * B
959           M[B + 2] = F                    " STACK BC
960           F = M[B - 3], Z                 " TAKE D
961       N,  F * M[B - 7]                    " D * B
962           M[B + 4] = F                    " STACK BD
963       N,  F = M[B - 3]                    " TAKE D
964       N,  F * M[B - 5]                    " D * A
965           F + M[B + 2]                    " ADD BC
966           M[B - 7] = F                    " STACK V
967           F = MC[-5], Z                   " TAKE A
968       N,  F * M[B + 2]                    " A * C
969           F - MC[6]                       " SUBTRACT BD
970           GOTOR(MC[3])

971 RSE35(1)
972           GL1 = - B

973 UTEXTRN(5)
974           SUBCD(:UBROFANELINK)
975           SUBCD(:USTAN)
976           G = MT[1]
977           GOTO(:R46E3)
978           UJOB1

979 RSE70(2)
980           A = 225
981           GOTO(:UDYNERR)
```

called with one operand and im. part of second op. on stack, real part of second op. in F.

apparently something not catered for
maybe dynamic own array.
alarm as with left hand actual in complex specification