

```

1      ;***COPYRIGHT 1969, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.***
2
3
4      ;THIS SUB-PROGRAM ASSEMBLED WITH SYSTEM PARAMETER FILE - S,MAC(V414)
5          XLIST
6          LIST
7      ;THIS SUB-PROGRAM ASSEMBLED WITH CONFIGURATION DEPENDENT FEATURE SWITCHES - FT50SB,MAC(
8      V003)
9          XLIST
10         LIST
11     IF FTDISK+FTRC10+2,<
12     TITLE SCHE0B - SCHEDULING ALGORITHM FOR SWAPPING SYSTEM(10/50)(BURROUGHS DISK)
13     >
14     IF FTDISK+FTRC10+1,<
15     TITLE SCHEDD - SCHEDULING ALGORITHM FOR SWAPPING SYSTEM(10/50)(DATA PRODUCTS DISK)
16     >
17     SUBTTL CLKCSW R,KRASIN/AF/TH/RCC TS 02 JUNE 69 V421
18     XP VSCHED,421+
19             ;PUT VERSION NUMBER IN GLOB LISTING AND LOADER STORAGE MAP
20
21
22     INTERNAL      FTRC10 ;THIS SOURCE FILE MAY BE ASSEMBLED TO USE EITHER THE
23                     ; NEW PDP-10 DISK (MODEL RC-10) OR THE OLD PDP-6 DISK (DATA
24                     ; PRODUCTS DISK FILE) FOR SWAPPING.
25
26
27     IFN           FTRC10, <
28     ENTRY        RCXSKD ;THIS SYMBOL IS SOLELY TO PERMIT SYSTEM
29     RCXSKD:       ; BUILDER TO RETRIEVE THE CORRECT BINARY FILE.
30     INTERNAL      XCKCSW
31     >
32     IF            FTRC10, <
33     ENTRY        XCKCSW
34     >
35
36
37     EXTERNAL JOB,JBTSTS
38     EXTERNAL JRTO,PJBSTS
39     EXTERNAL PJBSTS,TIMEF,MJOBN
40
41     INTERNAL NXTJOR,FTSWAP
42
43     ;ACCUMULATOR DEFINITIONS ---
44     000006 QJ=DEV DAT      ;QJOR WORD
45     000002 SW=TAC1        ;STATUS WORD
46     000004 J=ITEM        ;JOB NO.
47
48     ;INITIALIZE SCHEDULER (CALLED FROM SYSINI BEFORE ALL OTHER
49     ; DEVICES ARE INITIALIZED)
50
51     INTERNAL NXTINI
52
53     000000 XCKCSW:
54     000000 201040 000011 NXTINI: MOVEI TAC,MAYO
55     000001 402001 000244 SETM AVALTB(TAC)
56     000002 476001 000275 SETOM RECTAB(TAC)

```

57
58 000023 365040 000001'
59 000024 402000 000322'
60 000025 402000 000323'
61 000026 402000 000321'

SOJGE TAC.,-2
SETZM QJOB
SETZM XJOB
SETZM JORQUE

I USING DEVICE OTHER THAN INITIALIZATION
ICLEAR NO. OF JOBS NEEDING REQUEING
ICLEAR NO. OF JOBS NEEDING EXPANDING
ICLEAR JOB NO. TO BE REQUEUED

SCHEPB - SCHEDULING ALGORITHM FOR SWAPPING SYSTEM (12/57) (BURROUGHS DISK)
CLKCSW R,KRASIN/AF/1H/RCC TS 02 JUNE 69 V421

MACRO.V36 19:10 4-JUN-69 PAGE 13-1

42 000207 263140 000210 PDPJ PDP,
A3

```

64      ,NXTJOB DECREASES CURRENT JOB'S QUANT, AND PROTECT
65      ,TIMES AND REQUEUES IT IF QUANT, TIME GONE TO 0.
66      ,SERVICES ANY JOB REQUEUING REQUESTED AT OTHER PRIORITY
67      ,LEVELS THEN CALLS SHUFFLER,SWAPPER AND SCHEDULAR.
68      ,MAKES NO ASSUMPTIONS RE, ACS
69      ,RETURNS NEXT JOB TO RUN IN J.
70
71
72      EXTERNAL JBTSWP,POTLST
73
74      000010 336000 000000 NXTJOB: SKIPN TIMEF          ICLOCK TIC?
75      000011 254000 000022' JRST NXTJBI          INO
76      000012 205200 000000 MOVSI J,MJOBN          IYES
77      000013 205100 777777 MOVSI SW,-1          IDECREMENT IN CORE PROTECT TIME
78      000014 205240 000000 MOVSI DAT,SWP
79      000015 612244 000000 TONE DAT,JRTSTS(J)
80      000016 254000 000021' JRST .+3
81      000017 331004 000000 SKIPL JBTSWP(J)
82      000020 272104 000017' ADDM SW,JBTSWP(J)
83      000021 253200 000015' ADRJN J,-4
84
85      000022 336200 000000 NXTJBI: SKIPN J,JOB
86      000023 254000 000041' JRST CKJBI
87      000024 201300 000000 MOVEI QJ,0
88      000025 554104 000015' HLRLZ SW,JRTSTS(J)
89      000026 620100 311404 TRZ SW,RUNMSK+CMWB
90      000027 302100 440000 CAIE SW,RUNABLE
91      000030 254000 000051' JRST CKJBI3
92      000031 336000 000010' SKIPN TIMEF
93      000032 254000 000041' JRST CKJBI
94
95      000033 370104 000025' SOS SW,JRTSTS(J)
96      000034 602100 777777 TRNE SW,-1
97      000035 254000 000041' JRST CKJBI
98      000036 201240 000002' MOVEI DAT,QTIME
99      000037 200040 0000321' MOVE TAC,JOBQUE
100     000040 260140 000162' PUSHJ PDP,QXFER
  
```

```

101 000041 337300 000322' CKJB1: SKIPG QJ,QJOB      ISET QJ NON ZERO IF ANY REQUEUEING TO DO
102 000042 254000 000073'      JRST CKJB5      INO REQUEUEING NECESSARY
103 000043 201200 000000      MOVEI J,JORMAX  ISTART WITH HIGHEST JOB NUMBER ASSIGNED
104 000044 205100 000002 CKJB2: MOVSI SW,JRQ      IJOB NEEDS REQUEUEING BIT
105 000045 616104 000033'      TONN SW,JBTSTS(J) ITHIS JOB?
106 000046 367200 000045'      SOJG J,-1      INO,KEEP LOOKING
107 000047 323200 000073'      JUMPLE J,CKJB5    IYES,LOOKED AT ALL JOBS?
108                                     I (MAY NOT FIND A KJOBED JOB IF HIGHEST
109                                     I GO DECR, COUNT QJOB ANYWAY)
110 000050 412104 000045'      ANDCAM SW,JBTSTS(J) INO,MARK THIS JOB AS DONE
111 000051 200104 000050' CKJB3: MOVE SW,JBTSTS(J) IJOB STATUS WORD
112 000052 201240 000330'      MOVEI DAT,QCMW  IASSUME COMMAND WAIT
113 000053 607100 200000      TLNN SW,CMBW  IIS JOB IN COMMAND WAIT?
114 000054 254000 000057'      JRST CKJB9      INO.
115 000055 603100 002001      TLNE SW,SWP+JXPN  IYES, IS JOB ON DISK, OR TRYING TO EXPAND?
116 000056 254000 000067'      JRST CKJB4A      IYES, PUT JOB IN COMMAND WAIT Q
117 000057 325100 000066' CKJB9: JUMPG SW,CKJB4  INO,WAIT STATUS CODE DETERMINES NEW Q
118 000060 135100 000000      LDB SW,PJBSTS  IYES, GET QUEUE CODE.
119 000061 306100 000001      CAIN SW,WSQ      I*** EXPERIMENTAL ***
120 000062 254000 000132'      JRST CKJB10     I*** EXPERIMENTAL ***
121 000063 306100 000013      CAIN SW,TLOWO  ICURRENT JOB GOING INTO TTY IO WAIT?
122 000064 563004 000020'      HRRS JBTSWP(J) IYES, SET IN CORE PROTECT TIME TO -1,
123                                     I SO HE CAN BE SWAPPED IMMEDIATELY IF SOMEONE
124                                     I ELSE WANTS TO BE SWAPPED IN
125 000065 334242 000256' CKJB4H: SKIPA DAT,QBITS(SW) IGET ADDRESS OF TRANSFER TABLE
126 000066 201240 000326' CKJB4: MOVEI DAT,OSTOP  IIF RUN BIT WAS OFF
127 000067 260140 000162' CKJB4A: PUSHJ PDP,QXFER  IREQUEUE THE JOB
128 000070 322300 000113'      JUMPE QJ,SCHFD  IIF FROM NXTJOB GO DIRECTLY TO SCHED
129                                     I I.F, CURRENT JOB NO LONGER RUNNABLE(IOW)
130                                     I BUT JRO WASN'T SET SO DON'T DECR QJOB
131 000071 373300 000322'      SOSLE QJ,QJOB      IANY MORE JOBS TO REQUEUE?
132 000072 367200 000044'      SOJG J,CKJB2      IYES,BUT LOOK AT EACH JOB ONLY ONCE PER CLOCK TICK
133
134 000073 201300 000011 CKJB5: MOVEI QJ,AVLNUM  ICK AVAL FLAGS FOR SHAR, DEVS.
135 000074 336006 000244' CKJB6: SKIPN AVALTB(QJ) IFLAG=0?
136 000075 367300 000074'      SOJG QJ,CKJB6  IYES - TRY NEXT ONE
137 000076 323300 000112'      JUMPLE QJ,CKJB7  INO - OR FINISHED?
138 000077 544206 000307'      HLR J,AVLQTB(QJ) INO--GET 1ST JOB IN Q
139 000100 570204 000000 CKJB6A: HRRR J,JRT0(J)
140 000101 323200 000111'      JUMPLE J,CKJB8
141 000102 200244 000051'      MOVE DAT,JBTSTS(J) IFINISHED Q? WAIT TILL SWAPPER BRINGS IN JOB
142 000103 603240 002000      TLNE DAT,SWP  IIS JOB IN CORE?
143 000104 254000 000100'      JRST CKJB6A
144                                     INO, LOOK AT NEXT JOB IN THIS QUEUE
145 000105 550246 000307'      HRRZ DAT,AVLQTB(QJ) I TO SEE IF IN CORE,
146 000106 402006 000244'      SETM AVALTB(QJ) INO--GET TRANS, TABLE ADDRESS
147 000107 301300 000003      CAIL QJ,MINQ  ICLEAR AVAL FLAG
148 000110 260140 000162'      PUSHJ PDP,QXFER  ILESS THAN MIN, SHARABLE DEV. Q?
149                                     IREQUEUE THE JOB AND PUT IT IN
150 000111 367300 000074' CKJB8: SOJG QJ,CKJB6  I PROCESSOR Q SO SCHEDULER WILL RUN IT
                                     ICONTINUE IF ANY MORE FLAGS TO LOOK AT

```

```

151 000112          CKJB7:          INONE--GO SHUFFLE AND SWAP
152          IFE FTSWAP,<          EXTERNAL CHKSHF
153          PUSHJ PDP,CHKSHF
154          >
155          IFN FTSWAP,<
156 000112 260140 000465'          PUSHJ PDP,SWAP>
157
158          ;SCHEDULAR--SEARCH THRU QUEUES ACCORDING TO SSCAN TABLE
159          ;FOR 1ST JOB IN CORE--RETURN ITS NO. IN J
160
161 000113 402000 000000' SCHED: SETZM POTLST          ;CLEAR POTENTIALLY LOST TIME FLAG
162 000114 201240 000430'          MOVEI DAT,SSCAN          ;ADDRESS OF SCAN TABLE
163 000115 265040 000225'          JSP TAC,SSCAN          ;BEGIN SCAN
164 000116 254000 000130'          JRST SCHED1          ;NO MORE JOBS--RETURN NULLJOB
165 000117 476000 000113'          SETOM POTLST          ;SET POTENTIALLY LOST TIME FLAG FOR CLOCK1
166 000120 200304 000102'          MOVE QJ,JBTSTS(J)          ;IS THIS JOB SWAPPED OUT
167          TLNE QJ,SWP+SHF+JXPN          ;MONITOR WAITING FOR I/O TO STOP,OR JOB EXPANDING CORE?
168 000121 603300 006001          JRST (TAC1)          ;YES--CONTINUE SCAN,JOB CANNOT BE RUN
169 000122 254002 000200
170
171 000123 576100 000321'          HLREM TAC1,JOBQUE          ;YES--SAVE ITS Q
172 000124 205300 000370'          MOVSI QJ,WTMASK          ;CLEAR WAIT CODE
173 000125 412304 000120'          ANDCAM QJ,JBTSTS(J)
174 000126 402000 000117'          SETZM POTLST          ;CLEAR POTENTIALLY LOST TIME AS A USER IS TO BE RUN
175 000127 263140 000000'          POPJ PDP,          ;RETURN
176 000130 400200 000200' SCHED1: SETZ J,          ;RETURN NULL JOB
177 000131 263140 000000'          POPJ PDP,
178
179
180
181          ;TEMPORARY EXPERIMENTAL SCHEDULING CHANGE TO PERMIT TTY-I/O-WAIT-SATISFIED JOBS ON
182          ; THE DISK TO DISPLACE I/O BOUND JOBS IN CORE,... R.CLEMENTS/D.PLUMER 9 MAY 68
183 000132 332000 000141' CKJB10: SKIPE INFLG
184 000133 331004 000064'          SKIPL JBTSWP(J)
185 000134 254000 000065'          JRST CKJB4B
186 000135 201240 000137'          MOVEI DAT,CKJBT
187 000136 254000 000067'          JRST CKJB4A
188 000137 400000 000205' CKJBT: EXP EQFIX
189 000140 000006 777760          XWD QUTTY,-PQ2          ;MAKE JOB(LPT) COMPETE WITH CPU BOUND JOBS
190 000141 000000 000000' INFLG: 0          ;NON-ZERO MEANS AT LEAST ONE JOB ON DISK WAITING TO COME IN.

```

```
191          SUBTTL  QCSS R, KRASIN/AF TS3,17 22 MAR 68 V000
192
193          ,THIS ROUTINE MUST BE ASSEMBLED WITH THE CONFIGURATION
194          ,TAPE TO DEFINE NUMBER OF JOBS
195          ,THIS SECTION CONTAINS 2 ROUTINES FOR Q MANIPULATION
196          ,AND NECESSARY TABLES FOR SPECIFYING OPERATIONS PERFORMED
197          ,BY THEM.
198
199          EXTERNAL IMGIN,JRTSTS,JRTADR,PJBSTS
200          INTERNAL QXFER,QSCAN,FTSWAP,FTDISK
201
202          ,STORAGE:
203          ,EACH Q IS A RING STRUCTURED, FORWARD AND BACKWARD
204          ,LINKED STRING LIST, THE "FIRST" LINK IN A Q IS
205          ,A Q-HEADER POINTING TO THE FIRST AND LAST MEMBERS OF THE Q,
206          ,A NULL Q HAS ONE LINK--THE Q-HEADER ITSELF, THE LINKS MAKING
207          ,UP THE QS ARE CONTAINED IN A TABLE (JBTO) WITH NEGATIVE
208          ,INDICES (ADDRESSES LESS THAN JRTQ) USED FOR Q-HEADERS AND
209          ,POSITIVE INDICES USED FOR MEMBERS (JOBS), THUS ONLY ONE WORD
210          ,PER LINK IS NECESSARY--ITS ADDRESS RELATIVE TO JBTO GIVES THE
211          ,JOB NO. (OR Q NO. IF NEGATIVE) WHICH IT REPRESENTS WHILE
212          ,ITS CONTENTS CONTAINS THE LINKING POINTERS, THESE
213          ,POINTERS ARE ALSO INDICES RELATIVE TO JBTO RATHER THAN
214          ,ABSOLUTE ADDRESSES--RH(LINK)=FORWARD POINTER;
215          ,LH(LINK)=BACKWARD POINTER.
216          ,A JOB IS ASSUMED TO BE IN NO MORE THAN ONE Q AT A TIME, AND
217          ,THE NULL JOB (JOB 0) DOES NOT APPEAR IN THE QS (I.E. JBTO
218          ,ITSELF IS THE Q-HEADER FOR Q 0).
219
220          ,ROUTINES:
221          ,BOTH ROUTINES ARE "TABLE DRIVEN" IN THE SENSE THAT THE
222          ,CALLING ROUTINE PROVIDES THE ADDRESS OF A TABLE WHICH
223          ,DEFINES THE SPECIFIC OPERATIONS TO BE PERFORMED.
```

```

224
225
226
227
228
229
230
231
232
233
234
235 000142 211040 000000 QINI: MOVN1 TAC,MXQUE      ;MAKE ALL QUEUE HEADERS POINT TO THEMSELVES
236 000143 504040 000001      HRL TAC,TAC          ;BACKWARD POINTERS TOO
237 000144 202041 000100      MOVEM TAC,JBTQ(TAC)
238 000145 253040 000144      ADRJN TAC,.-1
239 000146 201040 777763      MOVEI TAC,-NULO      ;PUT JOBS ALL IN NULO QUEUE
240 000147 206040 000000      MOVSM TAC,JBTQ1      ;BACK POINTER FOR JOB 1
241 000150 201200 000043      MOVEI ITEM,JOBMAX      ;MAX. JOB NO.
242 000151 202044 000144      MOVEM TAC,JBTQ(ITEM)    ;FOR. POINTER OF JOBMAX JOB NO.

243 000152 506200 000000      HRLM ITEM,JBTQMN      ;SET NULO HEADER TO POINT TO JOB1
244 000153 201040 000001      MOVEI TAC,1          ;AND JOBMAX
245 000154 542040 000152      HRRM TAC,JBTQMN ;FORWARD POINTER
246 000155 542204 000000 QINI1: HRRM ITEM,JBTQMN(ITEM) ;JOB I-1 POINT TO JOB I
247 000156 402000 000151      SETPM JBTQ
248 000157 363200 000000      SOULF ITEM,CPOPJ      ;FINISHED?
249 000160 506204 000147      HRLM ITEM,JBTQ1(ITEM) ;BACK POINTER JOB I+1 POINTS TO JOB I
250 000161 254000 000155      JRST QINI1

```

```
251      ,DELETES A JOB FROM ITS "SOURCE-Q", DETERMINES A "DEST-Q"  
252      ,ACCORDING TO ONE OF 3 FUNCTIONS, AND INSERTS THE JOB AT  
253      ,THE BEGINNING OR END OF THIS DEST-Q. IN ADDITION IT MAY  
254      ,RESET THE JOB'S QUANTUM TIME (RW JBTST).  
255      ,THE DRIVING TABLES ARE "TRANSFER TABLES":  
256      ,  
257      ,T, TABLE:      EXP <CODE>  
258      ,                  XWD <QUANT-TAB>, <Q-TAB>  
259      ,  
260      ,  
261      ,DEPENDING ON <CODE>, THE SECOND WORD IS EITHER DATA OR THE  
262      ,ADDRESSES OF "CORRESPONDANCE TABLES",  
263      ,  
264      ,THE PREFIX OF <CODE> SPECIFIES WHETHER THE JOB IS TO BE  
265      ,INSERTED AT THE BEGINNING OR END OF THE DEST-Q. THE SUFFIX  
266      ,DETERMINES THE FUNCTION USED TO SELECT THE DEST-Q.  
267      ,THE FOLLOWING ARE THE SIX CODES AND THEIR TABLE FORMATS:  
268      ,  
269      ,  
270      ,DEST-Q AS A FIXED (PREDETERMINED) Q:  
271      ,BOFIX: INSERT AT BEG OF DEST-Q  
272      ,EQFIX: INSERT AT END  
273      ,  
274      , THE JOB IS TRANSFERED TO THE END OR BEG. OF THE Q <Q-TAB>  
275      , IF <QUANT-TAB> = -1, QUANT. TIME IS NOT RESET,  
276      , IF <QUANT-TAB> .G. 0 , QUANT. TIME IS RESET TO <QUANT-TAB>.  
277      , SINCE THIS FUNCTION IS FULLY DEFINED BY THE SECOND WORD  
278      , ALONE, NO CORRES. TABLE IS NECESSARY.
```

```

279      ,DEST-Q AS A FUNCTION OF SOURCE-Q
280      ,BQLINK:      INSRT AT BEG OF DEST-Q
281      ,EQLINK:      INSERT AT END
282      ,
283      , <Q-TAB>=ADDRES OF A CORRES. TABLE "LINKING" SOURCE-QS TO
284      , DEST-QS,
285      , IF <QUANT-TAB> = -1, QUANT. TIME IS NOT RESET,
286      , OTHERWISE <QUANT-TAB> IS TAKEN AS THE ADDRESS OF A
287      , TABLE OF QUANT. TIMES CORRESPONDING TO THE Q-LINKING TABLE.
288      , FORMAT OF THE TABLES ARE:
289      ,
290      , <Q-TAB>:      XWD <SQ1>,<DQ1> 11ST SOURCE-Q1DEST-Q PAIR
291      ,
292      ,      ...
293      ,      XWD <SQN>,<DQN> INTH ...
294      ,      ?      ZERO TERMINATES TABLE
295      ,
296      , <QUANT-TAB>:  EXP <QUANT1>      1CORRES. TO <Q-TAB>+0
297      ,      ...
298      ,      EXP <QUANTN>      1CORRES. TO <Q-TAB>+N-1
299      ,
300      ,      ?
301      ,
302      , UPON A CALL TO QXFER FOR THESE 2 CODES, AC T2 CONTAINES
303      , THE SOURCE-Q (CURRENT Q) OF THE JOB, THE LH OF THE
304      , <Q-TAB> ENTRIES ARE SEARCHED FOR A MATCH.,IF FOUND, THE
305      , RH IS TAKEN AS THE DEST-Q AND THE QUANT. TIME IS RESET
306      , (IF <QUANT-TAB> NOT -1) TO THE CORRRES. ENTRY IN THE
307      , <QUANT-TAB> TABLE.
308      , IF NO MATCH FOUND.,NO TRANSFER TAKES PLACE.
309
310      ,DEST-Q AS A FUNCTION OF JOB SIZE
311      ,BOJSIZ INSERT AT BEG OF DEST-Q
312      ,EQJSIZ INSERT AT END
313      ,
314      , <Q-TAB>=ADDRESS OF A TABLE ASSOCIATING JOB SIZE
315      , (IN 1K BLOCKS) TO DEST-QS.
316      , <QUANT-TAB> HAS SAME MEANING AS FOR B-EQLINK
317      ,
318      , <Q-TAB>:      XWD <JSIZ1>,<DQ1>
319      ,      ...
320      ,      XWD <JSIZN>,<DQN>
321      ,      ?
322      ,
323      , <QUANT-TAB>: SIMILAR TO THAT FOR B-EQLINK
324      ,
325      , THE <JSIZ>'S MUST BE IN INCREASING ORDER.
326      , THE TABLE IS SEARCHED UNTIL <JSIZ> IS LESS THAN OR
327      , EQUAL TO THE JOB SIZE, THEN THE CORRES. <DQ> IS
328      , TAKEN AS THE DEST-Q. IF THE TABLE IS EXAUSTED, NO
329      , TRANSFER TAKES PLACE,
330      , QUANT. TIME IS HANDLED AS IN B-EQLINK.

```

```

330      ,CALLING SEQUENCE:
331      ,      MOVE J,(JOB NUMBR)
332      ,      MOVE T2,(CURRENT Q)      ;BQLINK AND FQLINK ONLY
333      ,      MOVE T1,TRANS TABLE ADDRESS
334      ,      PUSHJ PDP,QXFER
335      ,      ...      ;RETURN
336      ,      ON RETURN J IS UNALTERED; LH(Q)=-1 IF QUANT. TIME NOT
337      ,      RESET; REQUANT. TIME IF RESET;R4(Q)=DEST.Q
338      ,
339      ,ACS:
340      TT=DAT      ;POINTER TO TRANSFER TABLE
341      J=ITEM      ;JOB NO.
342      Q=PROG      ;DEST-Q AND QUANT. TIME ON RETURN
343      T1=TAC1      ;TEMP
344      T2=TAC      ;TEMP AND SOURCE-Q ON CALL TO B,EGLINK
345
346      EXTERNAL ERROR
347
348      000162  000345  000001  QXFER:  MOVE Q,1(TT)      ;GET TRANSFER TABLE ADDRESS
349      000163  254025  000000      JRST Q(TT)      ;DISPATCH
350
351      ,DEST-Q AS FUNCTION OF SOURCE-Q
352      000164  336107  000000  QLINK:  SKIPN T1,(Q)      ;END OF TABLE?
353      000165  263140  000000      POPJ PDP,      ;YES
354      000166  574100  000002      HLRE T1,T1
355      000167  312100  000001      CAME T1,T2
356      000170  252340  000164'  AOBJP Q,QLINK      ;NO--SOURCE-Q=LH(TABLE ENTRY)?
357      000171  254000  000201'  JRST QX2      ;NO--CONTINUE SEARCH
358      ;YES
359
360      ,DEST-Q AS FUNCTION OF JOB SIZE
361      000172  554044  000000  QJSIZ:  HLRE T2,JBTADR(J)      ;HIGHEST REL. LOC. OF JOB
362      000173  240040  777766      ASH T2,-#D10      ;CONVERT TO NO. OF 1K BLOCKS - 1.
363      000174  205041  000001      MOVSI T2,1(T2)      ;NO. OF 1K BLOCKS TO LH,
364      000175  336107  000000  QX1:  SKIPN T1,(Q)      ;END OF TABLE?
365      000176  265240  000000      JSP DAT,ERROR
366      000177  313040  000002      CAMLE T2,T1
367      000200  252340  000175'  AOBJP Q,QX1      ;JOB SIZE, LE. LH(TABLE ENTRY)?
368      000201  204040  000007  QX2:  MOVSI T2,Q      ;NO--CONTINUE SEARCH, JUMP ALWAYS,
369      000202  560347  000000      HRRO Q,(Q)
370      000203  331005  000001      SKIPL 1(TT)
371      000204  504341  000000      HRL Q,(T2)      ;T2 IS ADDR. OF QUANT.TIME(IF REQUESTED)
372      ;RH(Q)=DEST-Q;LH=-1(NO QUANT.TIME REQ.)
373      ;WAS QUANT. TIME REQUESTED?
374      ;YES--GET IT

```

```

372
373 ,FIXED DEST-Q
374 000205 200104 000156' QFIX: MOVE T1,JBTQ(J) ;DELETE JOB FROM SOURCE-Q
375 000206 004040 000102' MOVES T2,T1 ;T1=FORW. LINK, T2=BACK LINK
376 000207 542101 000205' HRRV T1,JBTQ(T2) ;FORW. LINK PAST JOB
377 000210 506042 000207' HRLM T2,JBTQ(T1) ;BACK LINK PAST JOB
378
379 000211 335005 000000' SKIPCE (TT) ;END OR REG. OF Q?
380 000212 544347 000210' HLR Q,JBTQ(Q) ;END--THIS WILL LEAVE Q=IDX OF
381 ; CURRENT LAST LINK;T2=IDX OF Q-HEADER
382 000213 000047 000212' MOVE T2,JBTQ(Q) ;BEG--T2=IDX OF CURRENT 1ST LINK
383 ; Q=IDX OF Q-HEADER
384 000214 542207 000213' HRRM J,JBTQ(Q) ;INSERT JOB IN DEST-Q
385 000215 506201 000214' HRLM J,JBTQ(T2)
386 000216 542044 000215' HRRM T2,JBTQ(J)
387 000217 506344 000216' HRLM Q,JBTQ(J)
388
389 000220 321340 000224' JUMPL Q,QX3 ;RETURN IF QUANT. TIME NOT REQ.
390 000221 546344 000125' HLRM Q,JBTSTS(J) ;SFT QUANT. TIME

391 000222 201240 000000' MOVEI TT,RNQ ;SET JOB STATUS WAIT
392 000223 137240 000060' DPR TT,PJBTSTS ;CODE TO RUN QUEUE (X),
393 000224 263140 000000' QX3: POPJ PDP,
394
395 000205' BQFIX=QFIX
396 400000 000205' EQFIX=QFIX+100
397 000164' BQLINK=QLINK
398 400000 000164' EQLINK=QLINK+100
399 000172' BQJSIZ=QJSIZ
400 400000 000172' EQJSIZ=QJSIZ+100

```

```

401      ,SCANS THE QS RETURNING THE NUMBERS OF THE JOBS IN THE QS.
402      ,THE ORDER AND MANNER IN WHICH THE QS ARE SEARCHED IS
403      ,DETERMINED BY A "SCAN TABLE" ADDRESSED IN THE CALLING SEQ.
404      ,THE SCAN TABLE HAS THE FORM:
405      ,
406      ,SCANTAB:      XWD <Q1>,<CODE1>      ;SCN Q1 ACCORDING TO CODE1
407      ,              ...
408      ,              XWD <QN>,<CODEN>      ;QN ACCORDING TO CODEN
409      ,              Z                      ;ZERO TERMINATES TABLE
410      ,
411      ,EACH Q MAY BE SCANNED IN ONE OF FOUR WAYS SPECIFIED BY <CODE>
412      ,THE CODES ARE:
413      ,
414      ,QFOR  SCAN WHOLE Q FORWARD
415      ,QFOR1 SCAN FOR ONLY THE 1ST MEMBER (IF ANY)
416      ,QRAK  SCAN WHOLE Q BACKWARD
417      ,QRAK1 SCAN BACKWARD FOR ALL MEMBERS EXCEPT THE 1ST
418      ,
419      ,CALLING SEQ.
420      ,
421      ,      MOVEI ST,SCAN TABLE ADDRESS
422      ,      JSP PC,QSCAN      ;SFT UP PC FOR REPEATED RETURNS
423      ,      ...              ;RETURN HERE WHEN NO MORE JOBS
424      ,      ...              ;RETURN HERE WITH NEXT JOB IN AC J
425      ,      ...              ; AND ITS Q IN LH(QR)
426      ,
427      ,      PERFORM ANY NECESSARY TESTING OF THIS JOB
428      ,      J,ST,PC,QR MUST BE PRESERVED
429      ,
430      ,      JRST (QR)          ;RETURN TO QSCAN TO GET NEXT JOB
431      ,      ; IF THIS ONE NOT ACCEPTABLE
432      ,
433      ,ACS:
434      000004 J=ITEM ;JOB NO.
435      000005 ST=DAT ;POINTER TO SCAN TABLE
436      000001 PC=TAC ;RETURN ADDRESS
437      000002 QR=TAC1 ;ITERATED RETURN ADDRESS TO QSCAN

```

```

438 000225 336125 000000 QSCAN: SKIP A QR,(ST) IEND OF SCAN TABLE?
439 000226 254001 000000 JRST (PC) IYES--RETURN TO CALL+1
440 000227 574200 000002 HLRE J,QR INO--GET NO. OF Q
441 000230 254002 000000 JRST (QR) IDISPATCH
442
443 000231 201100 000234 QFOR1: MOVE1 QR,QFOR2 IONLY THE FIRST JOB
444
445 000232 570204 000217 QFOR: HLRE J,JBTQ(J) ISCAN FOWARD ALL JOBS
446 000233 327201 000001 JUMPG J,1(PC) IRETURN THIS JOB NO. CALL+2 UNLESS--
447 000234 344240 000225 QFOR2: AOJA ST,QSCAN IEND OF THIS Q--GET NEXT Q
448
449 000235 574204 000232 QBAK1: HLRE J,JBTQ(J) ISCAN BACKWARD ALL JOBS EXCEPT 1ST
450 000236 333004 000235 SKIPLE JBTQ(J) IIS THIS THE FIRST MEMBR?
451 000237 254001 000001 JRST 1(PC) INO--RETURN CALL+2
452 000240 344240 000225 AOJA ST,QSCAN IYES--GET NEXT Q
453
454 000241 574204 000236 QBAK: HLRE J,JBTQ(J) ISCAN BACKWARD ALL JOBS
455 000242 327201 000001 JUMPG J,1(PC) IRETURN CALL+2 WITH JOB NO. UNLESS
456 000243 344240 000225 AOJA ST,QSCAN IBEG OF THIS Q--GET NEXT Q

```

```

457      INTERNAL FTCHECK,FTMONP
458      IFN FTCHECK+FTMONP,<
459      EXTERNAL AVALTB
460      DEFINE X(A,B),<
461      EXTERNAL A'AVAIL
462      INTERNAL A'Q
463      A'Q=ZZ
464      ZZ=ZZ+1
465      >
466      ZZ=0
467      QUEUES
468      LOC=ZZ
469      >
470      IFE FTCHECK+FTMONP,<
471
472      ;SHARABLE DEVICE JUST BECAME AVAILABLE(EXTENDED TO OTHER QUEUEW TOO)
473      ;APPROPRIATE ENTRY IS SET NON-ZERO WHEN SCHEDULER SHOULD LOOK
474      ;AT THAT QUEUE TO FIND A JOB TO RUN
475      ;WSAVAL CONTAINS THE NO. OF JOBS WITH IO WAIT SATISFIED(0=NONE)
476
477      DEFINE X(A,B)
478      <INTERNAL A'AVAIL,A'Q
479      A'Q=-AVALTB
480      A'AVAIL: 0
481      >
482
483      INTERNAL AVALTB
484
485      000244      AVALTB: QUEUES ;GENERATE THE AVAL FLAGS
486      000244      000000 000000 RNAVAL: 0
487      000245      000000 000000 WSAVAL: 0
488      000246      000000 000000 TSAVAL: 0
489      000247      000000 000000 STAVAL: 0
490      000250      000000 000000 AUAVAL: 0
491      000251      000000 000000 MQAVAL: 0
492      000252      000000 000000 DAAVAL: 0
493      000253      000000 000000 DTAVAL: 0
494      000254      000000 000000 DCAVAL: 0
495      000255      000000 000000 MTAVAL: 0
496      000012      LOC=-AVALTB
497      >
498      000012      NQUEUE=LOC
499      XP MAXQ,NQUEUE-1      ;NO. OF QUEUES COUNTING RUN QUEUE
500      XP MINQ,STQ          ;MAX. STATE CODE WHICH HAS AN AVAL FLAG
501      XP AVLNUM,MAXQ        ;MINIMUM SHARABLE DEVICE QUEUE
502                          ;MAX. STATE CODE WHICH HAS AN AVAL FLAG
503
504      ;DEFINE STATE CODES WHICH DO NOT HAVE AVAL AND REQ FLAGS
505
506      DEFINE X(A)
507      <INTERNAL A'Q
508      A'Q=LOC
509      LOC=LOC+1

```

MACRO,V36 19:10 4-JUN-69 PAGE 25-1

```

510                                     >
511                                CODES= X IOW,*INTERNAL IOWQ
512
513                                YP MXCODE,LOC-1 :MAX, JOB STATE CODE
514                                PQ1=LCC
515                                LOC=LCC+1
516                                PQ2=LCC
517                                LOC=LCC+1
518                                PQ3=LCC
519                                LOC=LCC+1
520                                CMC=LCC                               ;COMMAND DELAY QUEUE
```

```

521
522          ;CORRESPONDENCE TABLE BETWEEN JOB STATUS CODES AND QUEUE TRANSFER TABLES
523          ;USED BY SCHEDULER
524          ;RUNCSS SETS JOB STATUS WORD TO NEW STATE CODE.
525          ;SCHEDULER SETS UP QUEUE TRANSFER TABLE ADDRESS FROM
526          ;FOLLOWING TABLE USING NEW STATE CODE AS INDEX
527
528          DEFINE X(A,B)
529          <      EXP Q'A'W
530          >
531
532          INTERNAL BITS
533
534          000256 000000 000332' QBITS: QUEUES+ X RN,7 +      EXP QRNW
535          000257 000000 000334'          X WS,6 +      EXP QWSW
536          000260 000000 000336'          X TS,6 +      EXP QTSW
537          000261 000000 000354'          X ST,6 +      EXP QSTW
538          000262 000000 000344'          X AU,4 +      EXP QAUW
539          000263 000000 000346'          X MQ,4 +      EXP QMQW
540          000264 000000 000350'          X DA,4 +      EXP QDAW
541          000265 000000 000356'          X DT,4 +      EXP QDTW
542          000266 000000 000352'          X DC,4 +      EXP QDCW
543          000267 000000 000360'          X MT,4 +      EXP QMTW
544          000270 000000 000340'          CODES+ X IOW, +      EXP QIOWW
545          000271 000000 000342'          X TIOW, +      EXP QTIOWW
546          000272 000000 000362'          X SLP, +      EXP QSLPW
547          000273 000000 000324'          X NUL, +      EXP QNULW
548          000274 000000 000326'          X STOP, +      EXP QSTOPW

```

```

549      IFN FTCHECK+FTMONP,<
550      DEFINE X(A,B),<
551      EXTERNAL A'REQ
552      >
553      QUEUES
554      EXTERNAL REQTAB
555      >
556      IFN FTCHECK+FTMONP,<
557
558      ;SHARABLE DEVICE REQUEST TABLE(GENERALIZED FOR OTHER QUEUES TOO)
559      ;CONTAINS THE NUMBER OF JOB WAITING TO USE SHARBLE DEVICE
560      ;WSREQ AND RNREQ ARE UNUSED
561
562      DEFINE X(A,B)
563      <A'REQ: 0
564      INTERNAL A'REQ
565      >
566
567      INTERNAL REQTAB
568
569      REQTAB: QUEUES ;GENERATE REQ TABLE
570      000275 0000'0 000000 X RN,7 +RNREQ: 0
571      000276 0000'0 000000 X WS,6 +WSREQ: 0
572      000277 0000'0 000000 X TS,6 +TSREQ: 0
573      000300 0002'0 000000 X ST,6 +STREQ: 0
574      000321 0000'0 000000 X AU,4 +AUREQ: 0
575      000302 0000'0 000000 X MO,4 +MOREQ: 0
576      000323 0000'0 000000 X DA,4 +DAREQ: 0
577      000324 0000'0 000000 X DT,4 +DTREQ: 0
578      000325 0000'0 000000 X DC,4 +DCRFQ: 0
579      000326 0000'0 000000 X MT,4 +MTREQ: 0
580      >

```

```

581
582          ;CORRESPONDENCE TABLE LH=QUEUE CODE, RH=QUEUE TRANSFER TABLE ADR.
583          ;INDEX INTO TABLE ALSO = QUEUE CODE
584          ;FOR SHARABLE DEVICES ONLY
585          ;SCHEDULER TAKES ONE JOB WAITING FOR A SHARABLE DEVICE AND
586          ;PUTS IT IN THE APPROPRIATE PUN QUEUE ACCORDING TO
587          ;QUEUE TRANSFER TABLE AS SPECIFIED BELOW BY THE JOB WAIT
588          ;STATE CODE.
589
590          DEFINE X(A,B)
591          <      XWD -A'Q,Q'A'S
592          >
593
594          000000 GRNS=0      ;NO CORRESPONDENCE TABLES FOR THESE QUEUES
595          000000 QWSS=0
596          000000 QTSS=0
597
598          INTERNAL AVLQTB
599
600          000307 000000 000000 AVLQTB: QUEUES* X RN,7 +      XWD -RNQ,GRNS
601          000310 777777 000000      X WS,6 +      XWD -WSQ,QWSS
602          000311 777776 000000      X TS,6 +      XWD -TSQ,QTSS
603          000312 777775 000372'      X ST,6 +      XWD -STQ,QSTS
604          000313 777774 000470'      X AU,4 +      XWD -AUQ,CAUS
605          000314 777773 000364'      X MQ,4 +      XWD -MQQ,QMQS
606          000315 777772 000366'      X DA,4 +      XWD -DAQ,QDAS
607          000316 777771 000374'      X DT,4 +      XWD -DTQ,QDTS
608          000317 777770 000370'      X DC,4 +      XWD -DCQ,QDCS
609          000320 777767 000376'      X MT,4 +      XWD -MTQ,QMTS

```

```

610          IFN FTCHECK+FTMONP,<
611          EXTERNAL QJQR,JQRQUE          ;JOBQUE WILL CAUSE LOAD OF PROPER SCHDAT
612                                          ; DEPENDING ON FTQC10 IN SCHDAT
613          IFN FTSWAP,<
614          EXTERNAL XJOB
615          >>
616          IF FTCHECK+FTMONP,<
617
618
619          INTERNAL JOBQUE
620 000321 000000 000000 JOBQUE: ?          ;JOBS TO BE REQUEUED ON CLOCK INTERRUPT
621
622          INTERNAL QJQR
623 000322 000000 000000 QJQR: ?          ;NUMBER OF JOBS NEEDING 0 TRANSFERS AT OTHER THAN CLOCK LEVEL
624
625          IFN FTSWAP,<
626 000323 000000 000000 XJOB: ?          ;NUMBER OF JOBS NEEDING CORE EXPANSION BY SWAPOUT-IN
627          INTERNAL XJOB
628          >
629          >

```

```

630          INTERNAL QSTOP,QTIME,SSCAN,QCMW
631
632          000205' BQFIX=QFIX      ;BEGINNING OF QUEUES FIXED QUEUE DISCIPLINE
633          400000 000205' EQFIX=QFIX+1R0 ;END OF QUEUES " " "
634          000164' RQLINK=QLINK
635          400000 000164' EQLINK=QLINK+1R0
636          000172' RQJSIZ=QJSIZ
637          400000 000172' EQJSIZ=QJSIZ+1R0
638          DEFINE TTAB(FCTN,QUEUE,QUANT)
639          <          EXP FCTN
640          XWD QUANT,-QUEUE
641          >
642          DEFINE PTTAB(FCTN,QUEUE,QUANT)
643          <          EXP FCTN
644          XWD QUANT,QUEUE
645          >
646
647          000324          QNULW: TTAB EQFIX,NULQ,-1      ;NULL QUEUE JOB NO. NOT ASSIGNED
648          000324 400000 000205'          EXP EQFIX
649          000325 777777 777763          XWD -1 , -NULQ
650          000326          QSTOP:QSTOPW: TTAB EQFIX,STOPQ,-1 ;UNRUNABLE JOBS TO END OF STOPQ
651          000326 400000 000205'          EXP EQFIX
652          000327 777777 777762          XWD -1 , -STOPQ
653          000330          QCMW: TTAB EQFIX,CMQ,-1      ;COMMAND WAIT TILL JOB IN CORE
654          000330 400000 000205'          EXP EQFIX
655          000331 777777 777756          XWD -1 , -CMQ
656          000332          QRNW: PTTAB EQJSIZ,QSTAB,QOSTAB ;JUST RUNABLE JOBS
657          000332 400000 000172'          EXP EQJSIZ
658          000333 000414' 000404'          XWD QOSTAB ,QSTAB
659          ;WHICH ARE NOT IN SOME WAIT STATE BELOW,ENTER PROCESSOR
660          ;QS AT END AND GET QUANT. TIME ACCORDING TO THEIR SIZE
661
662          000334          QWSW: TTAB BQFIX,PQ1,QTTY      ;IO WAIT SAT.(EXCEPT TTY)
663          000334 400000 000205'          EXP BQFIX
664          000335 000006 777761          XWD QTTY , -PQ1
665          ;ENTER FRONT OF PROCESSOR QS AND GET QUANT. TIME
666          ;ACCORDING TO JOB SIZE
667          000336          QTSW: TTAB BQFIX,PQ1,QTTY      ;TTY IO WAIT SATISFIED(ENTER FRONT OF PQ1)
668          000336 000000 000205'          EXP BQFIX
669          000337 000006 777761          XWD QTTY , -PQ1
670
671          000340          QIOWW: TTAB EQFIX,IOWQ,-1      ;IOW(EXCEPT TTY) HELD IN IOWQ
672          000340 400000 000205'          EXP EQFIX
673          000341 777777 777766          XWD -1 , -IOWQ
674          000342          QTIOWW: TTAB EQFIX,TIOWQ,-1   ;TTY IOW HELD IN TIOWQ
675          000342 400000 000205'          EXP EQFIX
676          000343 777777 777765          XWD -1 , -TIOWQ
677          000344 400000 000205'          QAUW: TTAB EQFIX,AUQ,-1, *          EXP EQFIX
678          000345 777777 777774          XWD -1, -AUQ
679          000346          QMQW: TTAB EQFIX,MQO,-1      ;MON. Q(DISK) WAIT
680          000346 400000 000205'          EXP EQFIX
681          000347 777777 777773          XWD -1 , -MQO
682          000350          QDAW: TTAB EQFIX,DAQ,-1      ;DEV. ALLOC.(DISK)

```

683	000350	400000	000205'	EXP EQFIX	
684	000351	777777	777772	XWD -1 , -DAO	
685	000352			QDCW: TTAB EQFIX,DCQ,-1	DATA CONTROL WAIT
686	000352	400000	000205'	EXP EQFIX	
687	000353	777777	777770	XWD -1 , -DCQ	
688	000354			QSTW: TTAB EQFIX,STQ,-1	SYST TAPE
689	000354	400000	000205'	EXP EQFIX	
690	000355	777777	777775	XWD -1 , -STQ	
691	000356			QDTW: TTAB EQFIX,DTQ,-1	DEC TAPE
692	000356	400000	000205'	EXP EQFIX	
693	000357	777777	777771	XWD -1 , -DTQ	
694	000360			QMTW: TTAB EQFIX,MTQ,-1	MAG TAPE
695	000360	400000	000205'	EXP EQFIX	
696	000361	777777	777767	XWD -1 , -MTQ	
697	000362			QSLPW: TTAB EQFIX,SLPQ,-1	SLEEP UUD
698	000362	400000	000205'	EXP EQFIX	
699	000363	777777	777764	XWD -1 , -SLPQ	

```

720                                     ;TRANSLATION TABLE FROM WAIT STATE TO SATISFIED STATE
721                                     ;DO NOT RESET QUANTUM RUN TIME
722
723          777777  777777  QWSD=-1
724
725  000364      000000  000205' QMDS:  TTAB BQFIX,PQ1,QQSD      ;START MON. Q(DISK) AT PQ1
726  000364      000000  000205'      EXP BQFIX
727  000365      777777  777761      XWD QQSD      ,-PQ1
728  000366      000000  000205' QDAS:  TTAB BQFIX,PQ1,QQSD      ;DEV. ALLOC.(DISK)...
729  000366      000000  000205'      EXP BQFIX
730  000367      777777  777761      XWD QQSD      ,-PQ1
731  000370      000000  000205' QDCS:  TTAB BQFIX,PQ1,QQSD      ;DATA CONTROL...
732  000370      000000  000205'      EXP BQFIX
733  000371      777777  777761      XWD QQSD      ,-PQ1
734  000372      000000  000205' QSTS:  TTAB BQFIX,PQ1,QQSD      ;SYST TAPE
735  000372      000000  000205'      EXP BQFIX
736  000373      777777  777761      XWD QQSD      ,-PQ1
737  000374      000000  000205' QDTS:  TTAB BQFIX,PQ1,QQSD      ;DEC TAPE
738  000374      000000  000205'      EXP BQFIX
739  000375      777777  777761      XWD QQSD      ,-PQ1
740  000376      000000  000205' QMTS:  TTAB BQFIX,PQ1,QQSD      ;MAG TAPE
741  000376      000000  000205'      EXP BQFIX
742  000377      777777  777761      XWD QQSD      ,-PQ1
743  000400      000000  000205' QAMS:  TTAB BQFIX,PQ1,QQSD      ;ALTER UFD
744  000400      000000  000205'      EXP BQFIX
745  000401      777777  777761      XWD QQSD      ,-PQ1
746  000402      000000  000164' QTIME: PTTAB EQLINK,QTTAB,QQSTAR      ;MOVE JOB TO LOWER Q
747  000402      400000  000164'      EXP EQLINK
748  000403      000414'  000410'      XWD QQSTAR      ,QTTAB
749                                     ;WHEN QUANT, TIME EXCEEDED AND RESET QUANT. TIME

```

```

730
731
732 000424 000004 777761 ,ENTER PROCESSOR QS ACCORDING TO JOB SIZE
733 000425 000020 777760 QSTAR: XWD 4,-PQ1 ;PQ1 IF SIZE .LE. 4K
734 000426 000400 777757 XWD +D16,-PQ2 ;PQ2 IF 4K .L. SIZE .LE. 16K
735 000427 000000 000020 XWD +D256,-PQ3 ;PQ3 IF 16 .L. SIZE
736
737
738 000410 777761 777760 ,PUT JOB DOWN A Q IF EXCEEDS QUANT. TIME
739 000411 777760 777757 QTTAR: XWD -PQ1,-PQ2
740 000412 777757 777760 XWD -PQ2,-PQ3
741 000413 000000 000020 XWD -PQ3,-PQ2 ;BACK TO PQ2 TO COMPETE WITH IOWS JOBS
742
743
744 ,QUANTUM TABLES
745 000006 QQSD=6 ;TENTH SEC. INITIAL QUANT. FOR SHAR. DEV. WAITERS
746 000006 QQTTY=6 ;TENTH SEC. INITIAL QUANT. FOR TTY IOWS
747
748 , QUANT. TIMES ACCORDING TO PROCESSOR Q:
749
750 INTERNAL RNQUNT
751
752 000414 RNQUNT:
753 000414 000000 000236 QQSTAR: EXP +D30 ;PQ1: ONE HALF SECOND
754 000415 000020 000170 EXP 2*+D60 ;PQ2: TWO SECONDS
755 000416 000000 000170 EXP 2*+D60 ;PQ3: TWO SECONDS
756 000417 000000 000236

```

```

757      IFN FTSWAP,<
758      INTERNAL ISCAN,OSCAN
759      ISCAN:  JSCAN FOR INPUT
760      000420 777756 000232' XWD -CMQ,QFOR JMONITOR COMMAND WHICH NEEDS CORE IMAGE IN CORE
761      000421 777773 000231' XWD -MQQ,QFOR1 JLOOK FOR 1ST JOBS IN SHAR, DEV QUEUES
762      000422 777772 000231' XWD -DAQ,QFOR1
763      000423 777774 000231' XWD -AUQ,QFOR1
764      000424 777770 000231' XWD -DCQ,QFOR1
765      000425 777767 000231' XWD -MTQ,QFOR1
766      000426 777775 000231' XWD -STQ,QFOR1
767      000427 777771 000231' XWD -DTQ,QFOR1
768      000430 777761 000232' SSCAN: XWD -PQ1,QFOR JSCAN PROCESSOR AS SCHEDULER DOES
769      000431 777760 000232' XWD -PQ2,QFOR
770      000432 777757 000232' XWD -PQ3,QFOR
771      000433 000000 000000 Z JPATCH SPACE
772      000434 000000 000000 Z
773      000435 000000 000000 Z JFINAL ZERO TO FLAG END
774
775      OSCAN: JSCAN FOR OUTPUT
776      000436 777762 000232' XWD -STOPQ,QFOR JUNRUNABLE JOBS FIRST
777      000437 777764 000232' XWD -SLPQ,QFOR
778      000440 777771 000235' XWD -DTQ,QBAK1 JANY SHAR, DEV, WAITERS MORE THAN 1 DEEP
779      000441 777775 000235' XWD -STQ,QBAK1
780      000442 777767 000235' XWD -MTQ,QBAK1
781      000443 777770 000235' XWD -DCQ,QBAK1
782      000444 777774 000235' XWD -AUQ,QBAK1
783      000445 777772 000235' XWD -DAQ,QBAK1
784      000446 777773 000235' XWD -MQQ,QBAK1
785      000447 777765 000232' XWD -TIOWQ,QFOR ITTY IOW
786      000450 777757 000241' XWD -PQ3,QBAK
787      000451 777771 000231' XWD -DTQ,QFOR1
788      000452 777775 000231' XWD -STQ,QFOR1
789      000453 777767 000231' XWD -MTQ,QFOR1 JNOW SCAN FIRST JOB IN QUEUES
790      000454 777770 000231' XWD -DCQ,QFOR1
791      000455 777774 000231' XWD -AUQ,QFOR1
792      000456 777772 000231' XWD -DAQ,QFOR1
793      000457 777773 000231' XWD -MQQ,QFOR1
794      000460 777760 000241' XWD -PQ2,QBAK
795      000461 777761 000241' XWD -PQ1,QBAK
796      000462 000000 000000 Z JPATCH SPACE
797      000463 000000 000000 Z
798      000464 000000 000000 Z JFINAL ZERO TO FLAG END
799
800      >

```

```

821          SUBTTL  SWAP R. KRASIN/AF TS4,34 03 FEB 69 V406
822
823          ,SWAPPER CALLED EVERY CLOCK TIC,
824          ,SINCE MOST OPERATIONS STARTED BY THE SWAPPER REQUIRE SEVERAL
825          ,TICS TO RUN TO COMPLETION, SEVERAL FLAGS(FINISH,FIT,FORCE
826          ,ARE USED TO "REMEMBER" PREVIOUS STATES,
827          ,THE BASIC ALGORITHM:
828          ,IS CORE SHUFFLER WAITING FOR IO TO FINISH FOR SOME JOB?
829          ; YES--TRY AGAIN TO SHUFFLE(WHEN IO STOPS)
830          ,IS CORE SHUFFLER STILL WAITING FOR IO TO FINISH?
831          ; YES--RETURN AND DO NOTHING
832          ,IS SWAPPER STILL BUSY?
833          ; YES--RETURN AND DO NOTHING
834          ,SCAN QS FOR 1ST JOB OUT OF CORE,
835          , IF NONE--RETURN
836          ;A:
837          , IF ONE--WILL LOW(HIGH) SEG FIT IN LARGEST HOLE IN CORE?
838          , YES--START INPUT AND RETURN
839          , NO--IS TOTAL FREE CORE(CORTAL) ENOUGH TO ACCOMMODATE LOW(HIGH) SEG?
840
841          ; YES--CALL CORE SHUFFLER
842          ; IS SHUFFLER WAITING FOR IO TO STOP?
843          ; YES--RETURN AND DO NOTHING
844          ; NO--GO TO A:
845          , NO--"REMBER" THIS JOB FOR INPUT AND LOOK FOR OUTPUT:
846          ,ANY JOBS WAITING TO XPAND CORE BY SWAP OUT/IN?
847          , YES--OUTPUT ONE AND RETURN
848          , NO--SCAN QS BACKWARD FOR JOB IN CORE WHOSE PROTECT TIME
849          , (SET ON INPUT) HAS GONE TO 0.
850          , IF NONE--RETURN
851          , IF ONE--IS IT SWAPPABLE(NO ACTIVE IO AND NOT CURRENT JOB)?
852          , YES--OUTPUT HIGH SEG(IF ANY AND NOT ON DISK) THEN LOW SEGMENT
853          , NO--SET SWP BIT(SO SCHEDULER WILL NOT RUN), IO WILL CONTINUE
854          , IN LOW SEGMENT AS LONG AS IT CAN
855          , IO ROUTINES NO LONGER STOP IF SWP SET, JUST SHF)
856
857          EXTERNAL JBTSTS
858          EXTERNAL BIGHOL,CORTAL,ANYDEV,JBTADR,JBTSWP,KCORE1,TRYSWP
859          EXTERNAL IMGOUT,IMGIN,FINISH,FIT,FORCE
860          EXTERNAL OERROR,CORGET,JBTDAT,JOBDPG,JOBDPO,JOBPC
861          EXTERNAL JRTDAT,SHFWAT,CHKSHF
862          EXTERNAL FULCNT,ERRPNT,EXCALP,PCSTOP,PCORSZ,VIRTUAL
863
864          INTERNAL SWAP
865          INTERNAL XPAND,FT2REL
866
867          000006 T=DEV DAT
868          000002 T1=TAC1
869          000001 T2=TAC
870          000004 J=ITEM
871
872          ,ALL DEVICE DEPENDENT CODE MARKED WITH A "*"

```

```

852 000465 332000 000000 SWAP:  SKIPN SHFWAT      ;IS CORE SHUFFLER WAITING FOR IO TO STOP
853                                ; FOR SOME JOB?
854 000466 260140 000000      PUSHJ PDP,CHKSHF    ;YES, CALL CORE SHUFFLER TO SEE IF
855                                ; IO STOPPED YET
856 000467 336000 000465'     SKIPN SHFWAT      ;IS SHUFFLER STILL WAITING?
857 000470 332000 001340'     SKIPE SQREQ      ;NO--IS SWAP SERV. ROUT. STILL BUSY WITH LAST JOB?
858 000471 263140 000000      POPJ PDP,        ;YES--RETURN
859 000472 402000 000141'     SETZM INFLC      ;*** EXPERIMENTAL ***
860 000473 336200 000000      SKIPN J,FINISH    ;NO--ANY IN/OUTPUT TO FINISH?
861 000474 254000 000547'     JRST SWP?        ;NO-
862 000475 321200 000537'     JUMPL J,FINOUT    ;YES--INPUT OR OUTPUT?
863 000476 332000 001342'     SKIPE SERA      ;INPUT, ANY INPUT ERRORS?
864 000477 254000 000526'     JRST INERR      ;YES
865 000500                                FININ0:  ;HERE IF NOTHING TO SWAP IN(HIGH OR LOW SEG EXPANDING FROM 0)
866                                IFN FT2REL,<
867                                EXTERN FININ
868 000500 260140 000000      PUSHJ PDP,FININ    ;IS THERE A HIGH SEG WHICH MUST BE SWAPPED IN?
869 000501 254000 000561'     JRST FIT1        ;YES, GO SWAP IT IN(J SET TO HIGH SEG NO.JOB # IN INPJOB)
870                                ; NO, EITHER HIGH SEG ALREADY IN FOR ANOTHER USER
871                                ; OR THERE IS NONE, J STILL JOB NO,(IE LOW SEG)
872                                ; OR J IS HIGH SEG WHICH EXPANDED FROM NOTHING(XPANDH)
873                                ; IN WHICH CASE IT HAS NO DISK SPACE AND DIDING ACS
874                                ; AND SETTING PROTECT TIME WON'T MATTER EITHER,
875                                >
876 000502 135300 000000      LDB T,IMGIN      ;NEW CORE SIZE
877 000503 135100 000000      LDR T1,IMGOUT    ;OLD SIZE WHEN ON DISK
878 000504 274100 000006      SUB T1,T         ;OLD-NEW=DECREASE
879                                ; HAS USER DECREASED VIRTUAL MEMORY FROM M TO N(N GR 0)
880
881                                ; WHILE OUT ON DISK(R,RUN,GET,KJOB) TO 140 WORDS?
882                                ; CORE COMMAND ALWAYS FORCES SWAP IN BEFORE
883                                ; CORE REASSIGNMENT SO NOT IN THIS CATEGORY
884                                ; FRAGMENTED USER TOO HARD TO PARTIALLY RECLAIM DISK SP
885                                ACE
886                                ; ON REDUCTION WHICH DOES NOT GO TO 0
887 000505 333000 000002      SKIPL T1        ;DECREASED?
888 000506 272100 000000      ADDM T1,VIRTAL    ;YES, NOW INCREASE VIRTUAL MEMORY AVAILABLE BY
889                                ; AMOUNT OF DECREASE IN HIGH OR LOW SEG
890 000507 260140 001034'     PUSHJ PDP,ZERSWP   ;RETURN LOW SEG DISK SPACE, SET IMGOUT,IMGIN
891                                ; AND SWP!SHF(JBTSTS) TO 0
892 000510 135300 000000      LDR T,PCORSZ      ;COMPUTE AND SET IN CORE IN CORE PROTECT TIME FROM
893                                ; SIZE OF JOB(1K BLOCKS-1)
894 000511 220300 000000      IMUL T,PROT      ;ADD VARIABLE AMOUNT DEPENDING ON CORE SIZE
895 000512 270300 000000      ADD T,PROT0      ;ADD FIXED AMOUNT INDEPENDENT OF CORE SIZE
896 000513 506304 000133'     WRLM T,JRTSWP(J)
897 000514 200344 000000      MOVE JDAT,JBTDAT(J) ;SETUP LOW SEG PROTECTION,RELOCATION

```

```

898
899      IFN JDAT-PROG,<
900      MOVE PROG,JBTADR(J)
901      >
902      MOVE T,JOBCPC(JDAT)      ;JOB STOPPED IN EXEC MODF?
903      TLNE T,USRMCD            ;TEST PD FLAG
904      JRST SWP1                ;NO
905      HRRZ T,JOBDPG(JDAT)      ;YES, ADJUST PROG AND PDP IN DUMP AC AREA
906      SUBI T,(PROG)            ;OLD RELOC-NEW RELOC
907      MOVNS T                  ;NEW RELOC-OLD RELOC
908      ADDM T,JOBDPD(JDAT)      ;ADJUST DUMP PDP
909      MOVEM PROG,JOBDPG(JDAT) ;STORE NEW AC PROG
910      JRST SWP1
911      000515 202307 202200
912      000516 603300 012000
913      000517 254000 000546'
914      000520 550307 000000
915      000521 275307 002000
916      000522 213000 000006
917      000523 272307 000000
918      000524 202347 000520'
919      000525 254000 000546'
920
921      000526 402000 000473' INERR: SETZM FINISH      ;CLEAR FINISH FLAG SO SWAPPING CAN CONTINUE
922      000527 200344 000172' MOVE PROG,JBTADR(J)      ;SETUP RELOC,PROTECTION FOR HIGH OR LOW SEG
923      IFN PROG-JDAT,<MOVE JDAT,JBTADR(J)>
924      PUSHJ PDP,KCORE1        ;RETURN CORE
925      JSP TAC,ERRPNT          ;PRINT ON USER CONSOLE
926      000532 516570 150100
927      000533 512130 142100
928      000534 426452 247644
929      000535 000000 000000
930      000536 254000 000000
931      000537 213000 000004
932      000540 332000 001342'
933      000541 254000 000575'
934
935      FINOUT: MOVNS J          ;STOP JOB AND FORCE RESCHEDULING
936      SKIPE SERR              ;FINISH OUTPUT, -FINISH=JOB NO.
937      JRST SWPREC            ;ANY ERRORS
938      ;YES, RECORD ERROR AND TRY AGAIN,
939      ; IN A DIFFERENT PLACE ON DISK
940      ;XWD PROTECT,,RELOC, FOR LOW SEG
941
942      MOVE PROG,JBTADR(J)
943      IFN PROG-JDAT,<
944      MOVE JDAT,JBTADR(J)
945      >
946      000543 260140 000530' PUSHJ PDP,KCORE1      ;RETURN CORE FOR LOW OR HIGH SEG JUST SWAPPED OUT
947      ; EVEN IF
948      ; ANOTHER JOB STARTED TO SHARE HIGH SEG DURING
949      ; SWAP OUT (GET) SINCE JOB IS MARKED WITH
950      ; SWP BIT ON AND CANNOT RUN UNTIL HIGH SEG IS SWAPPED B
951
952      ACK IN
953      IFN FT2REL,<
954      EXTERN FINOT
955      PUSHJ PDP,FINOT
956      JRST FORCEL
957
958      ;IS THIS A HIGH SEG WHICH WAS JUST SWAPPED OUT?
959      ;YES, J SET TO LOW SEG NO, GO TRY SWAP IT OUT
960      ; NO, THIS WAS A LOW SEG, ALL SWAPPING FOR THIS USER
961      ; IS FINISHED.
962
963      >
964      SWP1: SETZM FINISH      ;CLEAR FINISH FLAG
965      SWP2: SKIPE J,FORCE     ;WAITING FOR JOB TO BECOME SWAPPABLE?
966      JRST FORCE1            ;YES
967      FIT0: SKIPE J,FIT      ;NO-- WAITING TO FIT JOB IN CORE?
968      JRST FIT1              ;YES
969      000546 402000 000526'
970      000547 332200 000000
971      000550 254000 000663'
972      000551 332200 000000
973      000552 254000 000561'
  
```

```

944      ,SCAN FOR INPUT
945      000553 201240 000420'  MOVEI DAT,ISCAN
946      000554 265040 000225'  JSP TAC,OSCAN
947      000555 254070 000606'  JRST CHKXPN
948      000556 200304 000221'  MOVE T,JRTSTS(J)
949      000557 607370 002000'  TLNN T,SWP
950
951
952      000560 254072 000000'  JRST (TAC1)
953
954      000561 202270 000551'  FIT1:  MOVEM J,FIT
955      000562 135640 000502'  LDR AC1,IMGIN
956
957      IFB FT2REL,<
958      CAMLE AC1,CORTAL
959      >
960      IFB FT2REL,<
961      EXTERN FITSIZ
962      000563 260140 000000'  PUSHJ PDP,FITSIZ
963
964
965
966
967      >
968      000564 254070 000610'  JRST SCNOUT
969
970      000565 317640 000000'  CAMG AC1,BIGHOL
971
972
973
974      000566 254070 000772'  JRST SWAPI
975      IFB FT2REL,<
976      EXTERN FRECR1,HOLEF
977      000567 336000 000000'  SKIPN HOLEF
978
979      000570 260140 000000'  PUSHJ PDP,FRECR1
980
981
982      >
983      000571 260140 000466'  PUSHJ PDP,CHKSHF
984      000572 336000 000467'  SKIPN SHFWAT
985      000573 254070 000551'  JRST FIT0
986      000574 263140 000000'  POPJ PDP,

```

;NO INPUT TO DO--CK FOR EXPANDING JOBS
 ;THIS JOB OUT OF CORE?
 ;SWP ON IF HIGH SEG SWAPPED OUT FOR THIS USER
 ; OR BOTH SEGS SWAPPED OUT

 ;NO--CONTINUE SCAN

 ;REMEMBER JOB(OR HIGH SEG) TRYING TO FIT IN
 ;CORE SIZE NEEDED FOR THIS SEG(0 IF LOW SEG
 ; OR HIGH SEG WITH UWP OFF ALREADY IN CORE)

 ;WILL LOW SEG FIT IN FREE+DORMANT CORE?

 ;COMPUTE AMOUNT OF CORE NEEDED TO BRING IN
 ; 1. THIS JOBS LOW SEG AND HIGH SEG
 ; 2. THIS JOBS LOW SEG(HIGH ALREADY IN OR NONE)
 ; 3. THIS HIGH SEG BECAUSE LOW SEG ALREADY IN
 ;WILL LOW SEG FIT IN FREE+DORMANT+IDLE CORE?

 ;NO,WILL NOT FIT EVEN IF ALL DORMANT SEGS DELETED
 ; AC1=TOTAL CORE NEEDED(IN K)
 ;YES, WILL THIS SEG FIT IN BIGGEST HOLE OF FREE CORE
 ; WITHOUT DELETING ANY DORMANT OR IDLE SEGS?
 ; (AC1 RESTORED TO SIZE FOR JUST THIS LOW OR HIGH SEG)

 ;YES, GO SWAP IN THIS LOW OR HIGH SEG

 ;NO, ARE THERE ANY HOLES IN CORE WHICH THE SHUFFLER
 ; COULD ELIMINATE(NOT COUNTING ONE AT TOP)?
 ;NO, GO DELETE ONE DORMANT SEG IN CORE
 ; AND ALWAYS SKIP RETURN(THERE MUST BE AT LEAST
 ; ONE, OTHERWISE CORTAL=BIGHOL)MONITOR ERROR IF NONE

 ;YES, CALL CORE SHUFFLER TO MOVE ONE SEG DOWN
 ;SHUFFLER WAITING FOR IO TO STOP?
 ;NO, SEE IF JOB WILL FIT NOW.
 ;YES, RETURN AND WAIT TILL IO STOPS

```

987          EXTERN VIRTUAL,SWPERC
988
989 000575 200040 001342' SWPREC: MOVE TAC,SERA      IERROR FLAGS
990 000576 436040 000000      IORM TAC,SWPERC      ISAVE FOR POSTERITY
991 000577 135040 000503'      LDR TAC,IMGOUT      IDECREASE TOTAL AMOUNT
992 000600 213000 000001      MOVNS TAC          IOF VIRTUAL CORE IN THE MACHINE
993 000601 272040 000506'      ADDM TAC,VIRTUAL    IBY THE AMOUNT BEING GIVEN UP
994 000602 135040 000577'      LDR TAC,IMGOUT
995 000603 661040 000001      TLO TAC,1
996 000604 272040 000576'      ADDM TAC,SWPERC
997 000605 254000 000670'      JRST SWAPO          IGO TRY AGAIN
998
999
1000          ;NO INPUT TO DO, CHECK FOR EXPANDING JOBS
1001 000606 337000 000323' CHKXPN: SKIPG XJOB      IANY JOBS TO EXPAND?
1002 000607 263140 000000      POPJ PDP,          INO, RETURN FROM SWAPPER, NOTHING TO INPUT OR OUTPUT
1003                                     I YES, FALL INTO SCNOUT WHICH WILL SWAP OUT EXPANDING
1004                                     I JOB SINCE THERE IS ONE
1005          ;INPUT TO DO, CHECK TO SEE IF ANY JOBS JUST HAPPEN TO WANT TO EXPAND

1006          EXTERN HIGHJB,JBTSTS,ERROR,MAXSIZ,MAXJBN,SUMCOR
1007 000610 337000 000323' SCNOUT: SKIPG XJOB      IANY JOBS WAITING TO EXPAND?
1008 000611 254000 000624'      JRST SCNJOB        INO, SCAN ALL JOBS IN PRIORITY ORDER LOOKING
1009                                     I FOR ONE TO SWAP OUT
1010 000612 200200 000000      MOVE J,HIGHJB      IYES, START WITH HIGHEST JOB NUMBER ASSIGNED
1011 000613 205300 000001      MOVSI T,JXPN      ISETUP JOB EXPANDED BIT
1012 000614 616304 000556'      TDNN T,JBTSTS(J)   IIS THIS JOB EXPANDING?
1013 000615 367200 000614'      SOJG J,-1          INO, KEEP LOOKING
1014                                     IFN FTRCHK,<
1015 000616 327200 000621'      JUMPG J,SCNOK
1016 000617 402000 000323'      SETEM XJOB        ICLEAR XJOB SO MESSAGE WILL PRINT
1017 000620 265240 000176'      JSP DAT,ERROR      IERROR IF NONE FOUND
1018                                     >
1019 000621 370000 000323' SCNOK: SOS XJOB          IDECREMENT COUNT OF EXPANDING JOBS
1020 000622 412304 000614'      ANDCAM T,JBTSTS(J) ICLEAR EXPAND BIT IN JOB STATUS WORD
1021 000623 254000 000655'      JRST FORCE0        IGO TRY TO SWAP JOB OUT

```

```

1022          ;SCAN FOR JOB TO OUTPUT IN ORDER TO MAKE ROOM FOR JOB TO COME IN
1023          ;SIZE(IN K) NEEDED TO GET THIS USER IN CORE IS IN AC1(FITSIZE)
1024          ;JUST LOW SEG SIZE IF NO HIGH OR HIGH ALREADY IN, JUST HIGH IF LOW ALREADY IN,
1025          ;OR SUM IF BOTH MUST BE SWAPPED IN
1026
1027 000624 200300 000000 SCANJOB: MOVE T,CORTAL          ;INITIALIZE FREE CORE COUNTER
1028 000625 202300 000000          MOVEM T,SUMCOR
1029 000626 402000 000000          SETZM MAXSIZE
1030 000627 201240 000436          MOVEI DAT,OSCAN
1031 000630 265040 000225          JSP TAC,OSCAN
1032 000631 254000 000714          JRST NOFIT
1033 000632 316200 000561          CAMA J,FIT
1034 000633 254000 000000          JRST(TAC1)
1035 000634 335304 000622          SKIPGE T,JBTSTS(J)
1036 000635 335004 000513          SKIPGE JRTSWP(J)
1037
1038
1039
1040          TLNE T,NSWP+SWP
1041 000636 603300 012000
1042
1043 000637 254000 000000          JRST (TAC1)
1044 000640 554304 000542          HLRZ T,JRTADR(J)
1045
1046 000641 322300 000200          JUMPE T,(TAC1)
1047
1048 000642 240300 777766          ASH T,-12
1049 000643 271300 000001          ADDI T,1
1050
1051          IFN FT2REL,<
1052 000644 260140 000000          FXTNRM FORSIZE
1053          PUSHJ POP,FORSIZE
1054
1055          >
1056 000645 317300 000626          CAMG T,MAXSIZE
1057 000646 254000 000651          JRST FORCE2
1058 000647 202300 000645          MOVEM T,MAXSIZE
1059 000650 202200 000000          MOVEM J,MAXJBN
1060 000651 272300 000625          FORCE2: ADDM T,SUMCOR
1061 000652 313640 000651          CAMLE AC1,SUMCOR
1062 000653 254000 000000          JRST (TAC1)
1063 000654 200200 000650          MOVE J,MAXJBN
  
```

```

1063 000655 260140 000000 FORCEW: PUSHJ PDP,TRYSWP
1064 000656 254072 000000 JRST (TAC1)
1065
1066
1067
1068
1069 IFN FT2REL,<
1070 000657 260140 000000 EXTERN FORHGH
1071 PUSHJ PDP,FORHGH
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087 000660 205300 002000 MOVSI T,SWP:IFE FT2REL,<SHF> ;SET SWAPPED OUT BIT FOR LOW OR HIGH SEG
1088 000661 436324 000634' IORM T,JBTSTS(J) ;SCHEDULER WILL NO LONGER RUN THIS JOB
1089
1090
1091
1092 000662 202200 000547' FORCEL: MOVEM J,FORCE
1093
1094 000663
1095 FORCE1:
1096 IFN JDAT-PROG,<
1097 MOVE JDAT,JBTDAT(J)
1098
1099
1100 000663 336344 000640' SKIPM PROG,JRTADR(J)
1101 000664 254000 000670' JRST SWAPO
1102 000665 312200 000022' CAME J,JOB
1103
1104 000666 260140 000000 PUSHJ PDP,ANYDEV
1105 000667 263140 000000 POPJ PDP,

```

;CAN THIS JOB BE STOPPED IN ORDER TO DO SWAP?
 ;NO, NSWP OR NSHF SET(DISPLAY,REAL TIME) OR
 ;SAVE OR GET IN PROGRESS WITH DEVICE STILL ACTIVE
 ;LOOK FOR AN OTHER JOB TO SWAP

 ;IS THERE A HIGH SEG TO BE WRITTEN BEFORE
 ;TRYING TO SWAP OUT LOW SEGMENT?
 ;WRITE HIGH SEG IF ALL OF THE FOLLOWING ARE TRUE:
 ; 1. JOB HAS A HIGH SEG AND
 ; 2. IT HAS NOT BEEN SWAPPED FOR THIS USER
 ; (SWP=0 FOR JOB)
 ; 3. IT IS IN CORE(NOT XPANDH)
 ; 4. IF IN-CORE COUNT IS EXACTLY 1 MEANING
 ; THIS ONLY USER USING IN CORE
 ; 5. HIGH SEG NOT ON DISK YET
 ; 6. THIS HIGH SEG IS NOT THE SAME ONE AS JOB
 ; BEING FITTED IN IS GOING TO WANT

 ; RETURN HIGH SEG NO. IN J IF YES, OTHERWISE
 ; RETURN LOW SEG NO.
 ; IF JOB JUST HAS LOW SEG, SHF BIT IS SET IN JBTSTS
 ; FOR JOB SO IO WILL STOP NEXT BUFFER

 >
 ;SET SWAPPED OUT BIT FOR LOW OR HIGH SEG
 ;SCHEDULER WILL NO LONGER RUN THIS JOB
 ; SET SHF BIT IF ONE SFG SOFTWARE, SO IO WILL
 ; STOP AFTER NEXT BUFFERFUL.

 ;ASSUME NOT SWAPPABLE--IS IT?

 ;LOC. IN PHYSICAL CORE, IS CORE
 ; ASSIGNED IN MEMORY?
 ;NO, CANNOT HAVE ACTIVE DEVICES
 ;IF THIS IS CURRENT JOB, WAIT UNTIL
 ; PROTECTED AREA IS MOVED BACK TO JOB DATA AREA
 ;ANY ACTIVE DEVICES?(2ND HALF OF ANYACT ROUT.)
 ;YES--RETURN AND WAIT FOR I/O TO STOP.

```

1105                                ;SWAP OUT LOW OR HIGH SEGMENT
1106
1107                                INTERNAL FTTRACK
1108
1109                                SWAPO:
1110                                IFN FTTRACK,<EXTERN LASOUT
1111                                MOVEM J,LASOUT                                ;SAVE LAST SWAP OUT FOR DEBUGGING ONLY
1112                                >
1113                                000670 402030 000662' SETZM FORCE                                ;CLEAR FORCE FLAG
1114                                000671 554324 000663' HLRZ T,JBTADR(J)                                ;COMPUTE CORE IMAGE
1115                                000672 322330 000546' JUMPE T,SWP1                                ;DONT OUTPUT IF 0 CORE(IMGOUT ALREADY SET TO 0
1116                                ; WHEN CORE WAS RETURNED
1117
1118                                000673 550134 000671' HRRZ T1,JBTADR(J)
1119                                000674 212300 000001' MOVNM T,T2                                ;*SAVE COUNT FOR CALL TO SQOUT
1120                                000675 240300 777766' ASH T,-*D10                                ;CONVERT TO 1K BLOCKS
1121                                000676 271300 000001' ADDI T,1
1122                                000677 137330 000602' DPB T,IMGOUT                                ;RECORD AS OUT IMAGE
1123                                000700 505101 777777' HRLI T1,-1(T2)                                ;*BUILD AND SAVE IOWD FOR SQOUT
1124                                000701 261140 000002' PUSH PDP,T1                                ;*
1125                                000702 135240 000562' LDR DAT,IMGIN                                ;HAS SIZE OF CORE NEEDED WHEN NEXT SWAPPED IN
1126                                000703 336000 000005' SKIPN DAT                                ;ALREADY BEEN SET(XPAND)
1127                                000704 137300 000702' DPB T,IMGIN                                ;NO, SO SET TO # 1K BLOCKS OF CORE NEEDED
1128                                000705 200240 000006' MOVE DAT,T                                ;*CONVERT CORE IMAGE TO 128 WD BLOCKS
1129                                000706 260140 001206' PUSHJ PDP,GXSAT                                ;*GET DEVICE STORAGE
1130                                000707 254000 000717' JRST FULL                                ;*NONE AVAILABLE
1131                                000710 506044 000635' HRLM TAC,JBTSWP(J)                                ;*SAVE DEVICE ADDRESS
1132                                000711 212200 000546' OUTP2: MOVNM J,FINISH                                ;DISK SWAP SPACE ASSIGNED, NOW SET FINISH FLAG
1133                                ; SO THAT SWAPPER WILL KNOW WHICH SEG FINISHED
1134                                ; WHEN IO COMPLETED(SQREQ BECOMES ZERO)
1135                                000712 262140 000002' POP PDP,TAC1                                ;*GET IOWD
1136                                000713 254000 001066' JRST SQOUT                                ;*START OUTPUT AND RETURN
1137
1138                                000714 402030 000632' NOFIT: SETZM FIT                                ;FORGET ABOUT FITTING IN A JOB ON DISK
1139                                000715 476000 000141' SETOM INFLG                                ;*** EXPERIMENTAL *** MARK DESIRE TO INPUT
1140                                000716 263140 000000' POPJ PDP,                                ;ALL JOBS IN CORE ARE HIGHER PRIORITY.
1141

```

```

1142                                     ;COME HERE WHEN THE AMOUNT OF SPACE NEEDED ON THE DISK
1143                                     ;IS NOT AVAILABLE IN ONE CONTIGUOUS BLOCK
1144
1145                                     EXTERN GETFCR
1146
1147 000717 506240 000017 FULL:  HRLM  DAT,AC3          ;SAVE DAT (LARGEST AVAILABLE HOLE)
1148 000720 260140 000755'  PUSHJ  PDP,FULCOR        ;GET 4 FREE CORE LOCS
1149 000721 554240 000017  HLR2   DAT,AC3          ;RESTORE DAT
1150 000722 200740 000002  MOVE   AC3,TAC1         ;LOC OF 1ST FREE CELL
1151 000723 505740 777774  HRLI   AC3,-4          ;4 LOCS
1152 000724 660100 400000  TRO    TAC1,FRGSEG      ;LIGHT FRAGMENTED BIT
1153 000725 506104 000710'  HRLM  TAC1,JBTSWP(ITEM) ;SAVE LOC OF TABLE IN JBTSWP
1154 000726 261140 000005  FULL1: PUSH  PDP,DAT      ;SAVE AMOUNT OF SPACE BEING REQUESTED
1155 000727 260140 001206'  FULL1A: PUSHJ PDP,GXSAT    ;GET SOME SWAPPING SPACE
1156 000730 254000 000742'  JRST  FULL2          ;CANT HAVE THAT MUCH
1157 000731 542057 000000  WRRM  TAC,(AC3)        ;SAVE LOC OF THE DISK SPACE
1158 000732 262140 000005  POP    PDP,DAT        ;RESTORE AMT GOTTEN
1159 000733 506257 000000  HRLM  DAT,(AC3)        ;SAVE AMOUNT IN TABLE
1160 000734 274300 000005  SUB    T,DAT          ;AMOUNT STILL NEEDED

1161 000735 322300 000751'  JUMPE  T,FULSET      ;THROUGH IF NEED 0 K NOW
1162 000736 260140 000760'  PUSHJ  PDP,BMPAC3     ;STEP TO NEXT TABLE LOCATION
1163 000737 200240 000006  MOVE   DAT,T          ;TRY TO GET ALL WE NEED NOW IN 1 CHUNK
1164 000740 254000 000726'  JRST  FULL1
1165 000741 200240 000006  FULL1B: MOVE DAT,T      ;RESET AMOUNT OF SPACE NEEDED
1166
1167                                     ;COME HERE WHEN CANT GET THE CHUNK REQUESTED
1168 000742 202243 000000  FULL2: MOVEM DAT,(PDP)    ;DAT HAS LARGEST CHUNK AVAILABLE
1169 000743 327240 000727'  JUMPG  DAT,FULL1A     ;GO GET THAT AMOUNT
1170                                     ; -1=0 MEANS NO MORE LEFT
1171 IFN FT2REL,<
1172     EXTERN FRESWP
1173     PUSHJ PDP,FRESWP
1174     JRST FULL1B
1175                                     ;TRY TO DELETE AN UNUSED HIGH SEG FROM DISK
1176                                     ;FOUND ONE, TRY AGAIN, J PRESERVED
1177                                     ; NONE FOUND, PRINT MONITOR ERROR
    >

```

```

1176                                     FXTERRN CERROR
1177
1178 000746 262140 000001      POP      PDP,TAC      ;WHAT? NONE LEFT?
1179 000747 262140 000001      POP      PDP,TAC      ;SET PDP TO RIGHT VALUE
1180 000750 265240 000000      JSP      DAT,CERROR    ;ERROR IN MONITOR AT .....
1181
1182                                     ;HERE WHEN THE TOTAL AMOUNT OF SPACE NEEDED HAS BEEN OBTAINED
1183 000751 260140 000760' FULSET: PUSHJ PDP,RMPAC3    ;STEP TO NEXT (LAST) TABLE LOCATION
1184 000752 402017 000000      SETZM   (AC3)          ;ZERO MEANS END OF TABLE
1185 000753 554044 000725'      HLRZ     TAC,JBTSWP(ITEM) ;LOC OF TABLE OF FRAGMENTS
1186 000754 254000 000711'      JRST    OUTP2         ;GO START OUTPUT
1187
1188                                     ;HERE TO GET 4 LOCS OF FREE CORE
1189 000755 261140 000004 FULCOR: PUSH PDP,ITEM        ;GETFOR USES ITEM
1190 000756 260140 000000      PUSHJ   PDP,GETFOR      ;GET 4 CELLS
1191 000757 254000 000000      JRST    IPOPJ          ;RETORE ITEM AND RETURN
1192
1193                                     ;STEP AC3 TO NEXT LOC OF TABLE BEING BUILT
1194 000760 253740 000157' RMPAC3: ACRJN AC3,CPOPJ      ;OK IF MORE LOCS OF TABLE
1195 000761 260140 000755'      PUSHJ   PDP,FULCOR      ;GET 4 MORE LOCS
1196 000762 505740 777774      HRLI     AC3,-4
1197 000763 306117 000000      CAIN     TAC1,(AC3)      ;ARE THEY CONTIGUOUS?
1198 000764 263140 000000      POPJ     PDP,           ;YES, RETURN
1199 000765 200057 777777      MOVE     TAC,-1(AC3)     ;NO, CONVERT LAST GOOD LOC
1200 000766 562117 777777      HRPDM    TAC1,-1(AC3)   ;TO A POINTER TO NEXT PART OF TABLE
1201 000767 202042 000000      MOVEM    TAC,(TAC1)     ;STORE GOOD DATA IN 1ST WD OF NEW PART
1202
1203 000770 540740 000002      HRR      AC3,TAC1        ;NEW TABLE LOC
1204 000771 253740 000760'      ACRJN    AC3,CPOPJ      ;COUNT WORD AND RETURN

```

```

1215
1216          ;SWAP IN A JOB OF HIGH SEGMENT
1217
1218 000772      SWAPI:
1219
1220          IFN FTTRACK,<EXTERN LASIN
1221              MOVEM J,LASIN          ;SAVE LAST SWAP IN FOR DEBUGGING ONLY
1222      >
1223
1224 000772 202200 000711'      MOVEM J,FINISH          ;SET FINISH FLAG TO INPUT
1225 000773 402000 000714'      SETM FIT              ;CLEAR FIT FLAG
1226 000774 135040 000704'      LDR TAC,IMGIN          ;SIZE OF CORE TO BE ASSIGNED WHEN SWAPPED IN (INK)
1227 000775 242040 000012'      LSH TAC,*D10          ;CONVERT TO HIGHEST ADR
1228 000776 275040 000001'      SUR1 TAC,1            ;-1 FOR CALL TO CORGET
1229 000777 332344 000673'      SKIPE PROG,JBTADR(J)    ;IS (LOW)SEG ALREADY IN CORE?
1230 001000 254000 000500'      JRST FININ0           ;YES, POSSIBLE IF THIS IS LOW SEG AND ONLY
1231                                ; HIGH SEG WAS SWAPPED OUT.
1232 001001 260140 000000'      PUSHJ PDP,CORGET        ;NO, GET CORE FOR LOW OR HIGH SEG
1233 001002 265240 000000'      JSP DAT,0ERROR        ;NOT AVAILABLE-SHOULD NEVER HAPPEN(TELL OPER)
1234
1235
1236          IFN FT2REL,<EXTERN FITHGH
1237              PUSHJ PDP,FITHGH          ;INCREASE INCORE COUNT FOR THIS JOB'S HIGH SEG.
1238      >
1239
1240 001004 135300 000677'      LDR T,IMGOUT           ;GET OUTPUT IMAGE
1241 001005 322300 000500'      JUMPE T,FININ0         ;DONT INPUT IF OUT IMAGE IS 0
1242 001006 135100 000774'      LDR TAC1,IMGIN        ;IS SIZE OF CORE SMALLER THAN DISK SPACE?
1243 001007 315100 000006'      CAMGE TAC1,T          ;WELL?
1244 001010 200300 000002'      MOVE T,TAC1           ;YES, ONLY INPUT SMALLER AMOUNT(R,RUN,GET,KJOB)
1245 001011 242300 000034'      LSH T,*D18+*D10       ;*BUILD TOWD FOR SQIN
1246 001012 210100 000006'      MOVN TAC1,T          ;*
1247 001013 540104 000777'      HRR TAC1,JBTADR(J)    ;*
1248 001014 554044 000753'      HLRZ TAC,JBTSWP(J)    ;*GET DEVICE ADDRESS
1249 001015 254000 001065'      JRST SQIN            ;*START INPUT

```

```

1238 ;ROUTINE TO CHANGE DISK SWAPPING SPACE ALLOCATION(OR SET TO 0)
1239 ;DIFFERS FROM ZERSWP IN THAT VIRTUAL TALLY FOR SYSTEM IS ALSO CHANGED
1240 ;CALLED FROM CORE0
1241 ;CALL: MOVE ITEM,JOB OR HIGH SEC NO.
1242 ; MOVE TAC,#1K BLOCKS TO BE NEW ASSIGNMENT
1243 ; PUSHJ PDP,CHGSWP
1244 ; ALWAYS RETURN
1245 ;CALLED ONLY FROM VIRTUAL+PHYSICAL CORE ROUTINE CORE0
1246
1247 INTERN CHGSWP
1248 EXTERN JRTSTS,IMGIN,IMGOUT,JRTSWP,VIRTUAL,IPOPJ
1249
1250 001016 135100 001006' CHGSWP: LDB TAC1,IMGIN ;SIZE WHEN SEG NEXT SWAPPED IN
1251 001017 322040 001035' JUMPE TAC,CHG1 ;IS ZERO BEING ASKED FOR?
1252 001020 242040 777766 LSH TAC,-12 ;NO, CONVERT TO 1K BLOCKS
1253 001021 271040 000001 ADDI TAC,1 ;BUT DO NOT ATTEMPT TO RETURN DISK SPACE
1254 ; SINCE IT MIGHT BE FRAGMENTED(SWAPPER WILL
1255 ; RETURN ALL OF DISK SPACE ON NEXT SWAPIN)
1256 ; HAPPENS ONLY ON R,RUN,GET,KJOB
1257 001022 137040 001016' DDB TAC,IMGIN ;STORE NEW SIZE WHEN NEXT SWAPPED IN
1258 001023 261140 000004' PUSH PDP,ITEM ;SAVE AN AC
1259 001024 135200 001004' LDB ITEM,IMGOUT ;GET OLD DISK SIZE OF THIS USER(USES ITEM)
1260 001025 315100 000004' CAMGE TAC1,ITEM ;IS OLD IN-CORE SIZE BIGGER?
1261 001026 200100 000004' MOVE TAC1,ITEM ;NO, USE DISK SIZE AS USER'S OLD VIRTUAL CORE
1262 001027 315040 000004' CAMGE TAC,ITEM ;IS NEW IN-CORE SIZE BIGGER?
1263 001030 200040 000004' MOVE TAC,ITEM ;NO, USE DISK SIZE AS USER NEW
1264 ; VIRTUAL CORE
1265 001031 274100 000001' SUB TAC1,TAC ;DECREASE OF USER VIRT. CORE=OLD-NEW
1266 001032 272100 000001' ADDM TAC1,VIRTUAL ;USER'S DECREASE=SYSTEM'S INCREASE OF VIRTUAL
1267 ; CORE
1268 001033 254000 000757' JRST IPOPJ ;RESTORE ITEM AND RETURN

```

```

1269 ;ROUTINE TO RETURN ALL OF DISK SPACE FOR A LOW OR HIGH SEG
1270 ;THIS IS A PHYSICAL DEALLOCATION ONLY AND HAS NO EFFECT ON A SEGMENTS
1271 ;VIRTUAL CORE ASSIGNMENT
1272 ;CALL: MOVE ITEM, JOB NO. OR HIGH SEG NO.
1273 ; PUSHJ PDP,ZERSWP
1274 ;CALLED FROM SEGCON IN MANY PLACES(5)
1275 ;AND FINISH HERE IN SWAP
1276
1277         INTERN ZERSWP
1278
1279 001034 634040 000001 ZERSWP: TOZA TAC,TAC        ;REQUEST 0 SPACE ON DISK AND ALWAYS SKIP
1280 001035 2721'0 001032' CHG1:  ADDM TAC1,VIRTUAL    ;INCREASE SIZE OF VIRTUAL CORE AVAILABLE IN SYSTEM
1281                                     ; AND THEN RETURN ALL OF DISK SPACE(CHGSWP)
1282 001036 2051'0 006000        MOVSI TAC1,SWP:SHF    ;CLEAR SWAPPED OUT BIT IN JOB OR SEG
1283 001037 4121'4 000661'      ANDCAM TAC1,JBTSTS(ITEM);STATUS WORD(SHF SET IF IO WAS TO BE STOPPED
1284                                     ; FOR SWAP OR CORE SHUFFLE
1285
1286 001040 261140 000005        PUSH PDP,DAT        ;SAVE TTY OUTPUT BYTE POINTER(COMMAND DECODER)
1287 001041 135240 001024'      LDB DAT,IMGOUT        ;*SIZE ON DISK(1K BLOCKS)
1288
1288 001042 322240 001045'      JUMPE DAT,CHG3        ;DID SEG HAVE ANY DISK SPACE?
1289 001043 554044 001014'      HLRZ TAC,JBTSTP(ITEM) ;*YES, LOGICAL DISK BLOCK+FRGSEG BIT
1290 001044 260140 001224'      PUSHJ PDP,FXSAT      ;*FREE THE DISK BLOCKS NO LONGER NEEDED
1291 001045 262140 000005 CHG3: POP PDP,DAT        ;RESTORE TTY OUTPUT BYTE POINTER
1292 001046 201040 000000        MOVEI TAC,0        ;0 IS NEW DISK ASSIGNMENT
1293 001047 137040 001041'      DPB TAC,IMGOUT      ;SET DISK ASSIGNMENT TO 0
1294 001050 137040 001022'      DPB TAC,IMGIN      ;SET NEW CORE IMAGE BLOCK SIZE WHEN NEXT SWAPPED IN
1295                                     ; HERE FROM CHGSWP IF NOT ASKING FOR 0
1296 001051 263140 000000        POPJ PDP,          ;RETURN

```

```

1297          EXTERN PROT0,PROT          ;PROT AND PROT0 OCCUR IN COMMON
1298
1299          IFB      FTRC10, <
1300          XP ICPRT1,3+1*3              ;PROTECT TIME IN CLOCK TICS=
1301          XP ICPROT,*D10              ;((JOBSIZE/1K)*PROT0)*PROT
1302                                      ; PROT0=3,PROT=4 PRODUCE PROTECT TIMES ROUGHLY
1303                                      ; EQUAL TO 270 DISK SWAP(1-WAY) TIMES.
1304          >
1305          IFN      FTRC10, <
1306          ;SIMILAR IN-CORE PROTECT TIME PARAMETERS FOR FASTER RD-10 DISK.....
1307          XP ICPRT1,3+1*3              ;ZERO CORE PLUS K MULTIPLIER
1308          XP ICPROT,3                  ;MULTIPLY BY K-1 OF LOW SEG
1309          >

```

```

1310 ;XPAND SETS CONDITIONS TO GET MORE CORE FOR A JOB BY SWAPPING IN OUT
1311 ;THEN BACK IN TO DESIRED AMOUNT.
1312 ;JOBS POSITION IN OS NOT AFFECTED.
1313 ;CALLED ONLY FROM CORE COMMAND
1314 ;ASSUMES CALL FOR CURRENT JOB IF EXPANDING HIGH SEG, IE ASSUME AT UUD LEVEL
1315 ;THIS IS TRUE SINCE THERE IS NO CORE COMMAND WHICH CAN EXPAND HIGH SEG
1316 ;CALL: MOVE ITEM,[JOB NO.]
1317 ; MOVE TAC,[HIGHEST LEGAL ADDRESS DESIRED]
1318 ; PUSHJ PDP,XPAND
1319 ; RETURN, TAC DESTROYED
1320
1321 001052 242040 777766 XPAND: LSH TAC,-12 ;CONVERT HIGHEST DESIRED ADDRESS
1322 001053 271040 000001 ADDI TAC,1 ;TO 1K BLOCKS
1323 001054 137040 001050' DPH TAC,IMGIN ;STORE, SO SWAPPER WILL KNOW HOW MUCH CORE
1324 ; TO REQUEST WHEN NEXT SWAPPED IN
1325
1326 ;ROUTINE TO FLAG JOB TO BE STOPPED AND SWAPPED OUT
1327 ;BECAUSE IT HAS JUST BEEN CONNECTED TO A HIGH SHARABLE SEG WHICH IS ON DISK
1328 ;OR ON ITS WAY IN OR OUT. THE SIZE OF THE HIGH SEG IS UNCHANGED
1329
1330 ;THE JOB MUST BE STOPPED UNTIL HIGH SEG SWAPPED IN JUST AS IF JOB HAS
1331 ;EXPANDED HIGH SEG(MUST BE CALLED FROM UUD LEVEL FOR CURRENT JOB IF HIGH SEG)
1332 ;CALL: MOVE ITEM,HIGH SEG NUMBER
1333 ; PUSHJ PDP,XPANDH
1334
1335 ; INTERN XPANDH
1336 ; EXTERN IPOPJ
1337 001055 XPANDH:
1338 IFN FT2REL,<
1339 001055 261140 000004 PUSH PDP,ITEM ;SAVE JOB NUMBER
1340 001056 303200 000150' CAILE ITEM,JOBMAX ;IS THIS A LOW OR HIGH SEG?
1341 001057 200200 000665' MOVE ITEM,JOB ;HIGH,SO GET JOB NO.(MUST BE CURRENT JOB)
1342 >
1343 001060 205100 000001 MOVSI TAC1,JXPN ;SET THIS JOB EXPANDING BIT SO IT WILL NOT BE RUN
1344 001061 616104 001037' TDNN TAC1,JBTSTS(ITEM) ;IS IT ALREADY SET FOR THIS JOB?(UNLIKELY)
1345 001062 350000 000323' AOS XJOB ;NO, INCREMENT COUNT ONLY ONCE FOR EACH JOB EXPANDING
1346 001063 436104 001061' IORM TAC1,JBTSTS(ITEM) ;AND SET JOB EXPANDING BIT
1347 IFE FT2REL,<
1348 POPJ PDP, ;RETURN
1349 >
1350 IFN FT2REL,<
1351 001064 254000 201033' JRST IPOPJ ;RESTORE JOB OR HIGH SEG NUMBER (ITEM) AND RETURN
1352 >

```

```

1353          SUBTTL SWPSE R,KRASIN/AF TS4,34 03 FEB 69 V406
1354
1355          INTERNAL SQIN,SQOUT,SQGO,SQGO1
1356          INTERNAL FTSWAP
1357          EXTERNAL DFB'ISY,DFRED,DFWRT,CPOPJ,JORDAC,MJOBCK,CHECK,JBTCHK
1358
1359          ;PUT A REQUEST IN THE SWAPPING QUEUE, ENTER AT SQIN FOR
1360          ; INPUT, SQOUT FOR OUTPUT
1361          ;CALL: MOVE TAC1,XWD -NO, OF WORDS,FIRST CORE LOC.(IE IOWD+1)
1362          ; HRRZ TAC,DISK BLOCK NO.
1363          ; PUSHJ PDP,SQIN/SQOUT
1364          ; RETURN HERE ALWAYS
1365          ; CONTENTS OF TAC,TAC1 LOST
1366
1367 001065 661040 400000 SQIN: TLO TAC,400000 ;SET READ INDICATOR
1368 001066 202040 001342' SQOUT: MOVEM TAC,SERA ;STORE THE BLOCK NUMBER
1369 001067 202100 001340' MOVEM TAC1,SQREQ ;STORE THE IOWD
1370 001070 202100 001341' MOVEM TAC1,ESQREQ ;SAVE IN CASE OF DISK ERROR ON FRAGMENTED JOB
1371 001071 211040 000001 MOVNI TAC,1 ;IS THE DEVICE BUSY?
1372 001072 250040 000000 EXCH TAC,DFBUSY
1373 001073 326040 000771' JUMPN TAC,CPOPJ ;YES IF JUMP
1374
1375
1376          000003 ERATRY=3 ;NO. OF TIMES TO READ AND WRITE ON ERRORS
1377
1378          ;START UP DEVICE WITH SWAPPING REQUEST, THIS ROUTINE
1379          ;IS CALLED FROM DISK INTERRUPT SERVICE, AS WELL AS FROM ABOVE,
1380          ;IF A SWAPPER REQUEST IS WAITING(SQREQ WILL BE NON-ZERO)
1381
1382 001074 201100 000003 SQGO: MOVEI TAC1,ERATRY
1383 001075 202100 001343' MOVEM TAC1,SERACT
1384 001076 205040 400000 MOVSI TAC,400000
1385 001077 612040 001342' TDNE TAC,SERA ;WRITE?
1386 001100 254000 001107' JRST SQGO1 ;NO
1387 001101 550040 001340' HRRZ TAC,SQREQ
1388 001102 271040 000000 ADDI TAC,JOBOAC
1389 001103 505040 000000 HRLI TAC,MJOBCK
1390 001104 260140 000000 PUSHJ PDP,CHECK
1391 001105 214040 000772' MOVN TAC,FINISH
1392 001106 202101 000000 MOVEM TAC1,JBTCHK(TAC)

```

1393	001107	402000	001344'	SQG01:	SETZM SQLEN	I*ZERO AMOUNT TRANSFERRED SO FAR
1394	001110	200100	001340'		MOVE TAC1,SQREQ	I*PUT IOWD INTO TAC1
1395	001111	205040	200000		MOVSI TAC,200000	I*SET "SWAPPER I/O GOING" FLAG ON
1396	001112	437040	001342'		ORR TAC,SERA	
1397	001113	626040	400000		TRZN TAC,FRGSEG	I*FRAGMENTED?
1398	001114	254000	001142'		JRST SQG02	I*NO, READ IN ENTIRE (OR PART) OF SEG
1399						
1400					EXTERN CLCOR1	
1401						
1402	001115	261140	000005	FRGIO:	PUSH POP,DAT	I*AMOUNT PREVIOUSLY TRANSFERRED
1403	001116	210240	001344'		MOVN DAT,SQLEN	I*INCREASE CORE ADDRESS BY WORDCOUNT PREVIOUS
1404	001117	274100	000005		SUR TAC1,DAT	I*NO OF K IN THIS DISK CHUNK
1405	001120	574241	000000	FRGIO1:	HLRE DAT,(TAC)	I*SWAPPING ADDRESS FOR THIS DISK CHUNK
1406	001121	540041	000000		HRR TAC,(TAC)	I*POINTER TO NEW CORE LIST IF NEG.
1407	001122	325240	001127'		JUMPGE DAT,FRGIO2	I*CLEAR OUT ADR(15 BITS)
1408	001123	201240	077777		MOVEI DAT,77777	
1409	001124	412240	001342'		ANDCAM DAT,SERA	I
1410	001125	436040	001342'		ORM TAC,SERA	I*INSERT NEW ADDRESS
1411	001126	254000	001120'		JRST FRGIO1	
1412						
1413	001127	242240	000012	FRGIO2:	LSH DAT,12	I*CONVERT FROM K TO WORDS
1414	001130	272240	001344'		ADDM DAT,SQLEN	I*ADD TO PREVIOUSLY TRANSFERRED AMOUNT
1415	001131	213000	000005		MOVNS DAT	I*-N WORDS
1416	001132	506240	000002		HRLM DAT,TAC1	I*IOWD IN TAC1
1417	001133	524240	000005		HRLO DAT,DAT	I*-NO, OF WORDS FOR THIS DISK TRANSFER TO LH
1418	001134	317240	001340'		CAMG DAT,SQREQ	I*COMPARE WITH - NO. WORDS FOR REST OF SEG
1419	001135	500100	001340'		HLL TAC1,SQREQ	I*SWAPPER ONLY WANTS TO READ A PORTION OF SEG
1420						I*NOT ALL OF IT(R,UN,GET,KJOB COMMAND)
1421	001136	213000	000005		MOVNS DAT	I*-NO. OF WORDS FOR THIS NEXT TRANSFER
1422	001137	271240	777777		ADDI DAT,77777	
1423	001140	272240	001340'		ADDM DAT,SQREQ	I*UPDATE LH OF IOWD FOR ENTIRE SEG, SO IT HAS
1424						I*-NO, OF WORDS LEFT AFTER THIS TRANSFER IS DONE
1425	001141	262140	000005		POP PDP,DAT	
1426	001142	621040	377777	SQG02:	TLZ TAC,377777	I*CLEAR POSSIBLE TRASH IN LH.
1427	001143	241040	000003		ROT TAC,BLKSPK	I*RE-POSITION DISK LOGICAL BLOCK NUMBER.
1428	001144	622040	000004		TRZE TAC,4	I*TEST AND CLEAR READ/WRITE BIT,
1429	001145	364100	000000		SOJA TAC1,DFRED	I*YES
1430	001146	364100	000000		SOJA TAC1,DFWRT	I*NO, WRITE.

```

1431          ;SERVICE A SWAPPING INTERRUPT
1432          EXTERNAL DINT4R,CKSMCT
1433          INTERNAL SWPINT
1434
1435 001147 602000 700000 SWPINT: TRNE IOS,IODTER;IODERR;IOIMPM
1436 001150 254000 001176' JRST SWPERR IERRORS
1437 001151 622040 400000 TRZE TAC,FRGSEG ;*FRAGMENTED?
1438 001152 331100 001340' SKIPL TAC1,SQREQ ;*YES, MORE IOWD TO GO?
1439 001153 254000 001157' JRST DINT8B ;NO, ALL DONE SWAP IN OR OUT
1440 001154 350000 001342' AOS SERA ;YES, FRAGMENTED AND MORE TO GO
1441 001155 332001 000001 SKIPE 1(TAC) ;IS THIS THE END OF SWAP COMMAND LIST?
1442 001156 344040 001115' AOJA TAC,FRAGIO ;NO, BO DO NEXT PIECE OF FRAGMENTED SEG
1443
1444 001157 607040 400000 DINT8B: TLNN TAC,400000 ;*INPUT?
1445 001160 254000 001170' JRST DINT8A ;*NO
1446 001161 550040 001340' HRRZ TAC,SQREQ
1447 001162 271040 001102' ADDI TAC,JOBDA
1448 001163 505040 001103' HRLI TAC,MJOBCK
1449 001164 260140 001104' PUSHJ PDP,CHECK
1450 001165 214040 001105' MOVN TAC,FINISH
1451 001166 312101 001106' CAME TAC1,JBTCHK(TAC)
1452 001167 254000 001173' JRST SWPER1
1453 001170 552000 001342' DINT8A: HRRZM IOS,SERA
1454 001171 402000 001340' SETZM SQREQ
1455 001172 254000 000000 JRST DINT4B
1456
1457 001173 205040 000100 SWPER1: MOVSJ TAC,100
1458 001174 272040 000000 ADDM TAC,CKSMCT
1459 001175 660000 100000 TRO IOS,IODTER
1460 001176 214040 001165' SWPERR: MOVN TAC, FINISH ;*RESET SERA IN CASE OF FRAGMENTED JOB
1461 001177 554041 001043' HLRZ TAC, JBTSWP(TAC) ;*SWAP LOC (DISK ADR OR TABLE ADR)
1462 001200 542040 001342' HRRM TAC, SERA ;*RESTORE SERA
1463 001201 200040 001341' MOVE TAC,ESQREQ ;*RESTORE ESQREQ IN CASE OF FRAGMENTED JOB
1464 001202 202040 001340' MOVN TAC,SQREQ
1465 001203 373000 001343' SOSLE SERACT ;*TRIED ENOUGH?
1466 001204 254000 001107' JRST SOGO1 ;*NO, TRY AGAIN
1467 001205 254000 001170' JRST DINT8A ;*YES, TOUGH.

```

```

1468 IFF FTRC10, <
1469 ;SWPSER LOGIC FOR THE OLD POP-6 (DATA PRODUCTS) DISK FILE ---
1470
1471 ;FIND A SERIES OF BLOCKS ON THE DISK TO SWAP ONTO. CALLED
1472 ;AT CLOCK LEVEL.
1473 ;CALL: MOVEI DAT,NO. OF 1K BLOCKS DESIRED
1474 ;      PUSHJ POP,GXSAT
1475 ;      ERROR EXIT      (DISK IS FULL)
1476 ;      NORMAL EXIT      ;C(TAC) = BLOCK NO.
1477
1478 ;CONTENTS OF ACS TAC,TAC1,DAT WILL BE LOST.
1479
1480 INTERNAL GXSAT
1481 EXTERNAL GETRIT,IPOPJ1
1482
1483 GXSAT: MOVE AC1,XSAT1      ;SAVE AC1, SET IT TO TABLE LOC.
1484        MOVE AC2,XSAT2 ;
1485        LSH DAT,CONVMD      ;CONVERT TO 128 WORD DISK BLOCKS
1486        PUSH POP,ITEM      ;SAVE C(ITEM)
1487
1488        MOVE ITEM,DAT      ;GETRIT EXPECTS PARAMETER IN ITEM
1489        PUSHJ POP,GETBIT    ;FIND A HOLE BIG ENOUGH
1490        JRST IPOPJ         ;NONE, RESTORE ITEM AND ERROR RETURN
1491        MOVEI TAC,-1(TAC1)  ;IS IT ON DISK 17?
1492        CALL TAC,BLOCKS     ;YES
1493        ADDI TAC,DIFF        ;CARRY DISK ADDRESS SHIFTED TO FIT IN JBT SWP.
1494        LSH TAC,-CONVMD      ;SKIP RETURN AND RESTORE JOB NUMBER(ITEM)
1495        JRST IPOPJ1
1496
1497 ;FREE UP A SERIES OF BLOCKS ON THE SWAPPING DEVICE. CALLED
1498 ;AT CLOCK LEVEL.
1499 ;CALL: MOVEI DAT,NO. OF 1K CORE BLOCKS TO FREE
1500 ;      MOVE TAC,BLOCK NO. OF FIRST DISK BLOCK TO FREE
1501 ;      PUSHJ POP,FXSAT
1502 ;      ALWAYS RETURN HERE
1503
1504 ;CONTENTS OF ACS TAC,TAC1 WILL BE LOST.
1505
1506 INTERNAL FXSAT
1507 EXTERNAL CLRIT,IPOPJ
1508
1509 FXSAT: LSH TAC,CONVMD      ;RESTORE SHIFTED DISK ADDRESS.
1510        CALL TAC,HISWAP     ;ON DISK 17?
1511        SURI TAC,DIFF        ;YES
1512        LSH DAT,CONVMD      ;CONVERT TO 128 WORD DISK BLOCKS
1513        MOVE AC1,XSAT1      ;SET UP AC1 WITH POINTER TO SAT BLOCK
1514        MOVE AC2,XSAT2      ;AC2 TOO
1515        PUSH POP,ITEM      ;SAVE C(ITEM)
1516        MOVE ITEM,DAT      ;CLRIT EXPECTS PARAMETER IN DAT
1517        PUSHJ POP,CLRBIT    ;RETURN, AND RESTORE ITEM
1518        JRST IPOPJ
1519
1520 ;INITIALIZE SWAPPER DISK STORAGE TABLE

```

```

1521          INTERNAL SWPINI
1522
1523          SWPINI: MOVE TAC,XSAT2
1524                  MOVEM TAC,XSAT31
1525                  MOVSI TAC,1818
1526                  MOVEM TAC,XSAT3
1527                  SETZM XSAT4
1528                  MOVE TAC,XSAT4P
1529                  RLT TAC,XSAT61
1530                  MOVE TAC,XSAT7
1531                  MOVEM TAC,XSAT5
1532                  MOVE TAC,XSAT8
1533                  MOVEM TAC,XSAT6
1534                  POPJ PDP,
1535
1536          IFN FTICHECK+FTMONP,<
1537          EXTERNAL SREQ,SERA,SERACT,XSAT1,XSAT2,XSAT3,XSAT4,XSAT5,XSAT6,XSAT7
1538          EXTERNAL XSAT8,SWPSIZ,HISWAP,DIFF,CONVMO,BLOCKS,XSAT31,XSAT4P,XSAT61
1539          >
1540          IFE FTICHECK+FTMONP,<
1541
1542          ;DATA AND STORAGE AREA FOR SWAPPING, ON THE 270 DISK, DISKS
1543          ;      0 & 17 ARE USED FOR SWAPPING, EACH DISK CONTAINING
1544          ;      5400 (OCTAL) RECORDS,
1545
1546          INTERNAL          SREQ,SERA,SERACT
1547          SREQ:  ?          ;C(LH)=NEG. OF SIZE OF READ/WRITE
1548                          ; C(RH)=LOC. OF FIRST WORD TO READ/WRITE
1549          ESREQ:  ?          ;COPY OF SREQ, IN CASE OF
1550                          ; ERROR IN FRAGMENTED JOB
1551          SERA:  ?          ;SIGN IS 1 IF A READ
1552                          ; C(RH)=BLOCK NUMBER BEFORE READING,
1553                          ; ERROR BITS AFTER READING,
1554          SERACT: 0          ;COUNTER FOR ERRORS
1555          SQLEN: 0          ;AMOUNT TRANSFERRED SO FAR - FRAG SEG
1556
1557          XSAT1: EXP XSAT3-1          ;POINTER USED BY GETBIT,CLRBIT
1558          XSAT2: XWD -SWPSIZ,XSAT4    ;POINTER TO BIT TABLE
1559          XSAT3: XWD 400000,          ;2-WORD TABLE ALTERED BY GETBIT,CLRBIT
1560                  XWD -SWPSIZ,XSAT4
1561
1562          ; BIT TABLE FOR SWAPPING ALLOCATION
1563          XP BLOCKS,5400          ;NUMBER OF RECORDS/DISK PLATE
1564                  W1=BLOCKS/*D36    ;NUMBER OF ZERO WORDS AT TOP OF TABLE
1565                  E1=BLOCKS-W1*+D36 ;LEADING ZEROES IN W1+1ST WORD
1566                  EX=BLOCKS+E1-+D35
1567                  W2=EX/*D36        ;ZERO WORDS AT BOTTOM OF TABLE
1568                  F2=EX-W2*+D36    ;LEADING ZEROES IN W1+W2+2ND WORD
1569
1570          XP XSAT31,XSAT3+1
1571          XSAT4: BLOCK W1
1572          XSAT5: BLOCK W2+1
1573          XSAT6: BLOCK 1

```

```

1574
1575             REPEAT 1,<
1576                 IFE F1,<X=1B4>
1577                 IFN F1,<X=1
1578                 REPEAT +035-F1,<X=X*2>>
1579                 IFE F2,<Z=-1>
1580                 IFN F2,<Z=2
1581                 Y=1
1582                 REPEAT +036-F2,<Z=Z+Y
1583                 Y=Y*2>
1584             >>
1585             XP XSAT61,XSAT6-1
1586             XSAT7: EXP X
1587             XSAT8: EXP Z
1588
1589             SWPSIZ=XSAT6-XSAT4+1             ;SIZE OF TABLE
1590             HISWAP=17*BLOCKS                 ;LOGICAL BLOCK NUMBER OF FIRST
1591                                             ; BLOCK ON DISK
1592             DIFF=HISWAP-BLOCKS-1
1593
1593             XP CONVMD,3                     ;CONVERSION FROM 1K CORE BLOCKS TO 128 WORD
1594                                             ;DISC BLOCKS(SHIFT COUNT)
1595
1596             XP BLKSPK,CONVMD                 ;NO. OF BLOCKS PER K, SAME AS CONVMD
1597             XSAT4P: YWD XSAT4,XSAT4+1
1598             >
1599             >                               ;END OF SWPSER LOGIC FOR THE OLD PDP-6 DISK.

```

```

1600 IFN FTRC10, <
1601 ;SWPSER LOGIC FOR THE NEW PDP-10 (MODEL PC-10) DISK ---
1602
1603 INTERNAL GXSAT,FXSAT,SWPINI
1604 EXTERNAL GETBIT,CLRBIT
1605 EXTERNAL LBHIGH,IPOPJ,IPOPJ1
1606
1607
1608 ;SUBROUTINE "GXSAT" IS CALLED TO FIND A SERIES OF CONSECUTIVE FREE BLOCKS ON
1609 ; THE DISK TO SWAP SOME JOB OUT ONTO. IT IS CALLED AT CLOCK LEVEL,
1610
1611 ;CALLING SEQUENCE ---
1612 ; PUSHJ PDP,GXSAT
1613 ; ERROR EXIT -- THE DISK IS FULL, NO SWAPPING SPACE AVAILABLE.
1614 ; NORMAL EXIT
1615 ;ENTRY CONDITIONS ---
1616 ; C(DAT) = NUMBER OF 1K BLOCKS OF DISK STORAGE NEEDED.
1617 ;EXIT CONDITIONS ---
1618 ; C(TAC) = LOGICAL BLOCK NUMBER (DIVIDED BY 8) OF THE FIRST DISK BLOCK IN
1619 ; THE SERIES OF CONSECUTIVE BLOCKS WHICH SATISFY THIS REQUEST.
1620 ;ACCUMULATORS DAT, TAC, TAC1, AC1, AND AC2 ARE DESTROYED BY THIS SUBROUTINE.
1621
1622 001206 261140 000024 GXSAT: PUSH PDP,ITEM ;THIS ROUTINE SAVES AND RESTORES ACCUMULATOR "ITEM".
1623 001207 200200 000005 MOVE ITEM,DAT
1624 001210 201640 001301' MOVEI AC1,SWPENT ;SET UP ENTRY CONDITIONS FOR THE "GETBIT" SUBROUTINE.
1625 001211 200700 001301' MOVE AC2,SWPENT
1626 001212 260140 000000 PUSHJ PDP,GETBIT
1627 001213 254000 001364' JRST IPOPJ ;NO ROOM IN THE SWAPPING PART OF THE DISK.
1628 001214 271104 777777 ADDI TAC1,-1(ITEM) ;ROOM FOUND--COMPUTE LOGICAL BLOCK NUMBER OF THIS
1629 001215 313100 001300' CAMLE TAC1,MAXSWP ;FIRST BLOCK IN A SERIES OF CONSECUTIVE FREE DISK
1630 001216 202100 001300' MOVEM TAC1,MAXSWP ;BLOCKS. THE "SWPTAB" TABLE SEARCHED HAS ONE BIT
1631 001217 242100 000003 LSH TAC1,BLKSPK ;FOR EACH 1K OF SWAPPING DISK SPACE, BUT IS STORED
1632 001220 274100 000000 SUB TAC1,LBHIGH ;BACKWARDS (1ST BIT OF SWPTAB = LAST 1K OF DISK).
1633 001221 214040 000002 MOVEM TAC,TAC1 ;ALSO UPDATE "MAXSWP" IF PREVIOUS MAXIMUM EXCEEDED.
1634 001222 242040 777775 LSH TAC,-BLKSPK ;COMPRESS DISK ADDRESS TO 17 BITS FOR LH OF JBTSW.
1635 001223 344040 000000 AOJA TAC,IPOPJ1 ;ADD 1 TO TAC AND RESTORE ITEM AND SKIP RETURN

```

```

1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688

```

```

;SUBROUTINE "FXSAT" IS CALLED TO RETURN A SERIES OF CONSECUTIVE DISK BLOCKS TO
; THE FREE STORAGE POOL, THUS MAKING THEM AVAILABLE FOR RE-USE IN HANDLING
; FUTURE SWAPPING REQUESTS. IT IS CALLED AT CLOCK LEVEL.

;CALLING SEQUENCE ---
;   PUSHJ   POP,FXSAT
;   NORMAL EXIT
;ENTRY CONDITIONS ---
;   C(TAC) = LOGICAL BLOCK NUMBER OF THE FIRST DISK BLOCK IN THE
;   SERIES WHICH IS TO BE MADE AVAILABLE.
;   C(DAT) = NUMBER OF CONSECUTIVE 1K BLOCKS OF DISK SPACE WHICH
;   ARE TO BE MADE AVAILABLE.
;EXIT CONDITIONS ---
;   THE REQUESTED BITS IN THE SWAPPING SPACE AVAILABILITY TABLE
;   (NAMELY, SWPTAB) HAVE BEEN CLEARED TO ZERO.
;ACCUMULATORS DAT, TAC, TAC1, AC1, AND AC2 ARE DESTROYED BY THIS SUBROUTINE.

FXSAT: TRZN      TAC,FRGSEG      ;FRAGMENTED?
JRST    FXSAT1     ;NO. DO IN REGULAR WAY

FRAGRK: HRRZ      AC3,TAC
FRGBK1: HRRZ      TAC,(AC3)
        HLRZ      DAT,(AC3)
        JUMPLE    DAT,FRGBK2
        PUSHJ     POP,FXSAT1
        ALUP      AC3,FRGBK1
FRGBK2: HRRZ      TAC,AC3
        PUSH      POP,ITEM
        HLRZ      ITEM,AC3
        SUB       TAC,ITEM
        LSH       ITEM,-2
        AOS       ITEM
        PUSHJ     POP,CLCOR1
        POP        POP,ITEM
        SKIPE     TAC,(AC3)
        JRST      FRAGRK
        POPJ      POP,
        ;GIVE UP FREE CORE
        ;END OF TABLE?
        ;NO. GO CHASE NEXT PART
        ;YES. DONE

FXSAT1: PUSH      POP,ITEM
        MOVE      ITEM,DAT
        LSH       TAC,RLKSPK
        SUR       TAC,LBHIGH
        MOVMS     TAC
        ADDI      TAC,1
        MOVNI     AC1,RLKSPK
        LSH       TAC,(AC1)
        SUR       TAC,DAT
        MOVEI     AC1,SWPENT
        MOVE      AC2,SWPENT
FXSATC: PUSHJ     POP,CLRBIT
        JRST      IPOPJ

;THIS ROUTINE SAVES AND RESTORES ACCUMULATOR "ITEM".
;REGENERATE THREE LOW ORDER 0 BITS IN DISK ADDRESS.
;PREPARE TO CLEAR BITS IN THE SWAPPING
; AVAILABILITY TABLE BY TRANSFORMING THE LOGICAL
; BLOCK NUMBER TO AN EQUIVALENT BIT NUMBER IN THIS
; TABLE (WHICH IS STORED IN REVERSE ORDER AND HAS
; ONE BIT PER 1K OF SWAPPING SPACE ON THE DISK).

;SET UP ENTRY CONDITIONS FOR THE "CLRBIT" SUBROUTINE.
;THE "CLRBIT" SUBROUTINE IN "DSKSER" ACTUALLY DOES
; THE WORK OF CLEARING THE BITS.
;RESTORE ITEM AND RETURN

```

```

1689
1690          ;ROUTINE TO RE-INITIALIZE THE SWAPPING AVAILABILITY TABLE, CALLED AT
1691          ; SYSTEM INITIALIZATION TIME,
1692
1693          EXTERN VIRTUAL,SWPHGH
1694
1695 001264 200040 001252' SWPINI: MOVE TAC,LBHIGH          ;SET HIGHEST LOGICAL BLOCK
1696 001265 202240 000000          MOVEM TAC,SWPHGH        ;FOR SWAPPING IN COMMON
1697 001266 211040 000001          MOVNI TAC,1             ;SET ENTIRE TABLE TO ONES, I.E., MARK EVERYTHING
1698 001267 200100 001301'          MOVE TAC1,SWPENT        ; AS BEING UNAVAILABLE FOR SWAPPING.
1699 001270 202042 000000          MOVEM TAC,(TAC1)
1700 001271 253100 001270'          AOBUN TAC1,.-1
1701 001272 201640 001301'          MOVEI AC1,SWPENT        ;THEN USE THE "CLRBIT" ROUTINE TO CLEAR OUT AS MANY
1702 001273 200700 001301'          MOVE AC2,SWPENT        ; BITS AT THE FRONT OF THE TABLE AS THERE ARE 1K DISK
1703 001274 200200 000000          MOVE ITEM,K4SWAP        ; AREAS ALLOCATED FOR SWAPPING USE, THE PARAMETER
1704 001275 202200 001035'          MOVEM ITEM,VIRTUAL      ; TOTAL AMOUNT OF VIRTUAL CORE ALLOWED
1705                                     ; IS JUST EQUAL TO AMOUNT OF SWAPPING SPACE
1706                                     ; EVEN IF USER CAN NOT HAVE ALL OF PHYSICAL USER CORE

```

SCHEDB - SCHEDULING ALGORITHM FOR SWAPPING SYSTEM(10/50)(BURPOUGHS DISK)
SWPSER R,KRASIN/AF TS4,34 03 FEB 69 V426

MACRO.V36 19:10 4-JUN-69 PAGE 61

1727	001276	201040	002300	MOVEI	TAC,0	:	"K4SWAP" IS SET UP DURING DISK REFRESHING, AND
1728						:	RECALLED FROM THE DISK AT EACH SYSTEM-INITIALIZE.
1729	001277	254000	001262	JRST	CLRBIT	:	IN LIEU OF "PUSHJ CLRBIT" FOLLOWED BY "POPJ".

```

1710      IFN      FTD4FCK+FTMONP, <
1711      EXTERNAL  RCYSDT ;DUMMY GLOBAL SYMBOL TO PERMIT THE SYSTEM BUILDER TO
1712                  ; RETRIEVE THE CORRECT BINARY COPY OF "SCHDAT".
1713      EXTERNAL  SREQ,SERA,SFRACT
1714      EXTERNAL  LBHIGH,BLKSPK,MXX2SWP,SWPENT,SWPTAB,MAXSWP
1715      >
1716
1717      IF      FTD4FCK+FTMONP, <
1718      ;DATA ASSOCIATED WITH THE SWPSER LOGIC FOR THE NEW PDP-10 DISK ---
1719
1720          INTERN  MXX2SWP,CONVMD,BLKSPK,SWPTAB,MAXSWP
1721          EXTERN  K4SWAP
1722      ;THE ABOVE ARE REFERENCED OR INITIALIZED BY THE "ONCE" ROUTINE.
1723
1724      ;C(LCWSWP)=LOWEST LOGICAL BLOCK NUMBER TO BE USED FOR SWAPPING
1725      ;DETERMINED BY ONCE ONLY REFRESH DIALOG
1726      ;C(K4SWAP)=MAX. NO. OF K DISK WORDS ALLOCATED FOR SWAPPING, ALSO
1727      ;DETERMINED BY ONCE ONLY REFRESH DIALOG
1728      001300  000000  000000  MAXSWP:  ?      ;MAX. NO. OF K FROM TOP TO LOWEST JOB
1729                  ; USED SINCE SYSTEM REGAN
1730      ;C(MAXSWP)=MAX. NO. OF K EVER ASSIGNED COUNTING FROM TOP DOWN TO
1731      ;LOWEST EXCURSION, INCLUDING HOLES. WHEN = C(K4SWAP), THEN FRAGMENTATION HAD
1732      ;HAPPENED SOMETIME DURING THE PAST
1733      000003  BLKSPK=J      ;SHIFT FACTOR TO CONVERT K OF CORE TO DISK BLOCKS, I.E.,
1734                  ; EIGHT DISK BLOCKS PER K OF CORE.
1735      000003  CONVMD=BLKSPK ;ONLY FOR CONSISTENCY WITH OLDER PDP-6 ROUTINE.
1736      001750  MXX2SWP=+01200 ;MAXIMUM NUMBER OF 1K DISK BLOCKS WHICH MIGHT BE ALLOCATED
1737                  ; FOR SWAPPING (UPPER BOUND ON THE VALUE OF K4SWAP WHICH
1738                  ; MAY BE REQUESTED AT DISK REFRESH TIME). (ONE MILLION WORDS
1739                  ; FOR SWAPPING SEEMS LIKE A NON-RESTRICTIVE ARBITRARY LIMIT.)
1740      000034  SWPSIZ=MXX2SWP/*036+1 ;SIZE OF SWPTAB ALLOCATION TABLE.
1741
1742      001301  777744  001304'  SWPENT:  XWD      -SWPSIZ,SWPTAB ;THREE WORD POINTER TABLE
1743      001302  400000  000000      XWD      400000,0      ; REQUIRED BY THE "GETRIT" AND
1744      001303  777744  001304'      XWD      -SWPSIZ,SWPTAB ; "CLRRIT" SUBROUTINES.
1745
1746      001304      SWPTAB: BLOCK  SWPSIZ ;SWAPPING SPACE AVAILABILITY TABLE.
1747
1748
1749      ;DATA CARRIED OVER FROM THE COMMON PART OF OLD AND NEW SWPSER ROUTINES ---
1750      INTERNAL  SREQ,SERA,SFRACT,ESREQ
1751      001340  000000  000000  SREQ:  ?      ;C(LH)=NEG. OF SIZE OF READ/WRITE
1752                  ; C(RH)=LOC. OF FIRST WORD TO READ/WRITE
1753      ESREQ:  ?      ;COPY OF SREQ IN CASE OF SWAP ERROR ON FRAGMENTED JOB
1754      001341  000000  000000
1755      001342  000000  000000  SERA:  ?      ;SIGN IS 1 IF A READ
1756                  ; C(RH)=BLOCK NUMBER BEFORE READING,
1757                  ; ERROR BITS AFTER READING.
1758      001343  000000  000000  SERACT: ?      ;COUNTER FOR ERRORS
1759      001344  000000  000000  SULEN: ?      ;AMOUNT TRANSFERRED SO FAR FOR FRAGMENTED JOB
1760      >
1761      >      ;END OF SWPSER LOGIC FOR THE NEW PDP-10 DISK.
  
```

SCHEB - SCHEDULING ALGORITHM FOR SWAPPING SYSTEM (12/57) (BURROUGHS DISK)
SWPSFR R.KRASN/AF TS4,34 03 FEB 69 V476

MACRO,V36 19:10 4-JUN-69 PAGE 63

1762 701345

SCHEB: END

NO ERRORS DETECTED

PROGRAM BREAK IS 001345

AC1	000015	INT	AC2	000016	INT	AC3	000017	INT
ANYDEV	000066	EXT	AUAVAL	000250	INT	AJO	000004	INT
AUREO	000031	INT	AVALT8	000244	INT	AVLNUM	000011	INT
AVLQTB	000037	INT	RIGHOL	000565	EXT	BLKSPK	000003	INT
BMPAC3	000076		RQFIX	000225		BQJSIZ	000172	
BQLINK	000164		CEPROR	000750	EXT	CHECK	001164	EXT
CHG1	001035		CHG3	001045		CHGSWP	001016	INT
CHKS4F	000571	EXT	CHKXPN	000606		CKJB1	000041	
CKJB10	000172		CKJB2	000044		CKJB3	000051	
CKJB4	000066		CKJB4A	000067		CKJB4B	000065	
CKJB5	000073		CKJB6	000074		CKJB6A	000100	
CKJB7	000112		CKJB8	000111		CKJB9	000057	
CKJB7	000137		CKSMCT	001174	EXT	CLCOR1	001242	FXT
CLRBIT	001277	EXT	CMQ	000022		CMWB	000000	INT
CONVMD	000003	INT	CORGET	001001	EXT	CORTAL	000624	FXT
CPOPJ	001073	EXT	DAVAL	000252	INT	DAQ	000006	INT
DAREO	000303	INT	DAT	000005	INT	DCAVAL	000254	INT
DCC	000010	INT	DCREQ	000305	INT	DEV DAT	000006	INT
DFRUSY	001072	EXT	DFREQ	001145	EXT	DFWRT	001146	EXT
DINT4B	001172	EXT	DINT8A	001170		DINT8B	001157	
DTAVAL	000253	INT	DTQ	000007	INT	DTREQ	000304	INT
EQFIX	400000	000205	EQJSIZ	400000	000172	EQLINK	400000	000164
ERATRY		000003	ERROR		000620	ERRPNT		000531
ESQREQ		001341	EXCALP		000000	FININ		000500
FININ0		000520	FINISH		001176	FINOT		000544
FINOUT		000537	FIT		000773	FIT0		000551
FIT1		000561	FITHGH		001003	FITSIZ		000563
FORCE		000670	FORCEL		000655	FORCE1		000663
FORCE2		000651	FORCEL		000662	FORHGH		000657
FORSIZ		000644	FRAGRK		001226	FRAGIO		001115
FRECR1		000570	FRESWP		000744	FRGBK1		001227
FRGBK2		001234	FRGIO1		001120	FRGIO2		001127
FRGSFG		400000	FT2REL	777777	777777	FTCCL	777777	777777
FTCHEC		000000	FTDISK	777777	777777	FTLOG1	777777	777777
FTMONP		000000	FTRC10	777777	777777	FTRCBK	777777	777777
FTSWAP	777777	777777	FTTRAC		000000	FULCNT		000200
FULCOR		000755	FULL		000717	FULL1		000726
FULL1A		000727	FULL1B		000741	FULL2		000742
FULSET		000751	FXSAT		001224	FXSAT1		001247
FXSATC		001262	GETBIT		001212	GETFCR		000756
GXSAT		001276	HIGHJB		000612	HOLEF		000567
ICPROT		000003	ICPRT1		000006	IMGIN		001054
IMGOUT		001047	INERR		000526	INFLG		000141
IONERR		000000	IODTER		100000	IOIMPM		400000
IOS		000000	IOWQ		000012	IPOPJ		001263
IPOPJ1		001223	ISCAN		000420	ITEM		000004
J		000004	JBTADR		001013	JBTCHK		001166
JBTDAT		000514	JBTQ		000241	JBTQM1		000155
JBTOMN		000154	JBTGP1		000160	JBTSTS		001063
JBTSWP		001177	JDAT		000007	JOB		001057
JORDAC		001162	JOROPD		000523	JOBOPG		000524
JORMAX		001056	JORPC		000515	JOBQUE		000321
JRO		000002	JXPN		000001	K4SWAP		001274

KCORE1	000543' EXT	LBHIGH	001264' EXT	LOC	000022
MAXJRN	000654' EXT	MAXQ	000011 INT	MAXSIZ	000647' EXT
MAXSWP	001320' INT	MINQ	000003 INT	MJOBCK	001163' EXT
MJOBN	000012' EXT	MCAVAL	000251' INT	MQQ	000005 INT
MQREQ	000322' INT	MTAVAL	000255' INT	MTQ	000011 INT
MTREQ	000326' INT	MXCODE	000016 INT	MXK2SW	001750 INT
MXQUE	000142' EXT	NOFIT	000714' INT	NQUEUE	000012
NSWP	010000 INT	NULQ	000015 INT	NXTINI	000000' INT
NXTJRI	000022' INT	NXTJOB	000010' INT	OERROR	001002' EXT
OSCAN	000436' INT	OUTP2	000711' INT	PC	000001
PCORSZ	000510' EXT	PCSTOP	000536' EXT	PDP	000003 INT
PURSTS	000223' EXT	POTLST	000126' EXT	PQ1	000017
PQ2	000020	PQ3	000021	PROG	000356' INT
PROT	000511' EXT	PROT2	000512' EXT	Q	000007
QAUS	000420' INT	QAUW	000344' INT	QBAK	000241' INT
QBAK1	000235' INT	QBITS	000256' INT	QCMW	000330' INT
QDAS	000366' INT	QDAW	000350' INT	QDCS	000370' INT
QDCW	000352' INT	QDTS	000374' INT	QDTW	000356' INT
QFIX	000225' INT	QFOR	000232' INT	QFOR1	000231' INT
QFOR2	000234' INT	QINI	000142' INT	QINI1	000155' INT
QIOWW	000340' INT	QJ	000006	QJOB	000322' INT
QJSIZ	000172' INT	QLINK	000164' INT	QMQS	000364' INT
QMOW	000346' INT	QMTS	000376' INT	QMTW	000360' INT
QNULW	000324' INT	QQSD	000006	QOSTAB	000414' INT
QTTY	000006	QH	000002	QRNS	000000
QRNW	000332' INT	QSCAN	000225' INT	QSLPW	000362' INT
QSTAR	000424' INT	QSTOP	000326' INT	QSTOPW	000326' INT
QSTS	000372' INT	QSTW	000354' INT	QTIME	000402' INT
QTIONW	000342' INT	QTSS	000000	QTSW	000336' INT
QTTAP	000410' INT	QWSS	000000	QWSW	000334' INT
QX1	000175' INT	QX2	000221' INT	QX3	000224' INT
QXFER	000162' INT	RCXSKD	000000' INT	REQTAB	000275' INT
RNAVAL	000244' INT	RNQ	000000 INT	RNQUNT	000414' INT
RNREQ	000275' INT	RUNARL	440000 INT	RUNMSK	111404 INT
SCHD1	000130' INT	SCHED	000113' INT	SCHEND	001345' INT
SCNJOB	000624' INT	SCNOK	000621' INT	SCNOUT	000610' INT
SERA	001342' INT	SERACT	001343' INT	SHF	004000 INT
SHFWAT	000572' EXT	SLPQ	000014 INT	SQGO	001074' INT
SQG01	001127' INT	SQGO2	001142' INT	SQIN	001065' INT
SQLEN	001344' INT	SQOUT	001066' INT	SQREQ	001340' INT
SSCAN	000430' INT	ST	000005	STAVAL	000247' INT
STOPQ	000016 INT	STQ	000003 INT	STRED	000300' INT
SUMCOR	000652' EXT	SW	000002	SWAP	000465' INT
SWAPI	000772' INT	SWAP0	000670' INT	SWP	002000 INT
SWP1	000546' INT	SWP2	000547' INT	SWPENT	001301' INT
SWPER1	001173' INT	SWPERC	000604' EXT	SWPERR	001176' INT
SWPHGH	001265' EXT	SWPINI	001264' INT	SWPINT	001147' INT
SWPREC	000575' INT	SWPSIZ	000034	SWPTAB	001304' INT
T	000026	T1	000002	T2	000001
TAC	000001 INT	TAC1	000002 INT	TIMEF	000031' EXT
TIONW	000013 INT	TRYSWP	000655' EXT	TAVAL	000246' INT
TSQ	000002 INT	TSREQ	000277' INT	TT	000005
USRMQD	010000 INT	VIRTUAL	001275' EXT	VSCHED	000421 INT

SCHEM - SCHEDULING ALGORITHM FOR SWAPPING SYSTEM (10/50) (BURROUGHS DISK)
SYMBOL TABLE

MACRO.V36 19:10 4-JUN-69 PAGE 64-2

WSAVAL 000245' INT
WTMASK 000370' INT
XPAND 001052' INT

WSC 000021' INT
XCKCSW 000000' INT
XPANDH 001055' INT

WSREQ 000276' INT
XJOB 000323' INT
ZERSWP 001034' INT

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

HOLEF	976	977
HSAMSK	6#	6
HSAPDS	6#	6
HSASIZ	6#	6
HUNGST	6#	6
HUNGST	6#	6
I	6#	6
IB	6#	6
IBUFR	6#	6
ICLOS8	6#	6
ICPROT	1329#	1329
ICPRT1	1328#	1328
ILM	6#	
ILUERR	6#	6
IMGIN	199	838 676 955 1125 1127 1216 1230 1248 1250 1257 1294 1323
IMGOUT	838	877 991 994 1122 1228 1248 1259 1297 1293
INBR	6#	6
INER	864	911#
INFLG	183	190# 859 1139
INITR	6#	6
INPB	6#	6
IO	6#	6
IOACT	6#	6
IOREG	6#	6
IORCTL	6#	6
IOBT	6#	6
IOCON	6#	6
IODEND	6#	6
IODERR	6#	6 1435
IDOTER	6#	6 1435 1459
IDENO	6#	6
IDOST	6#	6
IDIMPM	6#	6 1435
IDNRCK	6#	6
IDPAR	6#	6
IOS	6#	6 1435 1453 1459
IDTEND	6#	6
IDUSE	6#	6
IDW	6#	6
IDWC	6#	6
IDWQ	511	512# 673
IDPOPJ	1191	1248 1268 1335 1351 1625 1627 1686
IDOPPJ1	1625	1635
IDPPERR	6#	6
IDSCAN	758	759# 945
IDTEM	6#	6 45 241 242 243 246 248 249 341 434 849 1153 1185
J	1189	1258 1259 1260 1261 1262 1263 1283 1289 1339 1340 1341 1344 1346
	1622	1623 1628 1662 1663 1664 1665 1666 1668 1673 1674 1703 1704
	45#	76 79 81 82 83 85 88 95 103 105 106 107 110
	111	122 132 138 139 140 141 166 173 176 184 341# 360 374
	384	385 386 387 390 391 404# 440 445 446 449 450 454 455 849#
	860	862 896 897 912 918 922 940 942 948 954 1010 1012 1013
	1015	1020 1033 1035 1036 1044 1058 1062 1088 1092 1098 1101 1114 1118

JACCT	1131	1132	1214	1219	1235	1236										
JBFADR	6#	6														
JBFCTR	6#	6														
JBFPTR	6#	6														
JBTADR	199	360	837	912	922	1044	1008	1114	1118	1219	1235					
JBTCHK	1357	1392	1451													
JBTDAT	839	840	897													
JBTQ	37	139	230	237	242	247	374	376	377	380	382	384	385	386		
	387	445	449	450	454											
JBTQM1	232	246														
JBTQMN	233	243	245													
JBTQP1	231	240	249													
JBTSTS	36	79	88	95	105	110	111	141	166	173	199	390	836	948		
	1076	1012	1020	1035	1088	1248	1283	1344	1346							
JBTSWP	72	81	82	122	184	837	896	1036	1131	1153	1185	1236	1248	1289		
	1461															
JBUF	6#	6														
JDAT	6#	6	897	898	901	904	907	908	913	923	1095					
JERR	6#	6														
JLOG	6#	6														
JNA	6#	6														
JOR	36	85	1101	1341												
JORDAC	1357	1388	1447													
JORDPO	839	907														
JORDPG	839	904	908													
JORMAX	103	230	241	1340												
JOBPC	839	901														
JOBQUE	61	99	171	619	620#											
JRO	6#	6	104													
JWPOS	6#	6														
JWSIZ	6#	6														
JXPN	6#	6	115	167	1011	1343										
K4SWAP	1703	1721														
KCORE1	837	914	926													
LBHIGH	1605	1632	1676	1695												
LISTCN	6															
LOC	496#	498	512	512#	514	515	515#	516	517	517#	518	519	519#	520		
LOOKR	6#	6														
MAXJRN	1006	1058	1062													
MAXQ	53	500#	500	502												
MAXSIZ	1006	1029	1055	1057												
MAXSWP	1629	1630	1720	1728#												
MEDDL	6#	6														
MINQ	147	501#	501													

[illegible]

[illegible]

[illegible]

[illegible]

XJOB	60	626#	627	1001	1007	1016	1019	1345
XPAND	844	1321#						
XPANDH	1334	1337#						
7ERSWP	892	1277	1279#					

CODES	6#	511	544											
DISABL	6#													
FNABLE	6#													
NOSCHE	6#													
NOSHIF	6#													
PTTAB	643#	656	726											
QUEUFS	6#	485	534	569	600									
SCHEDU	6#													
SHUFFL	6#													
STARTD	6#													
TTAB	639#	647	650	653	662	667	671	674	677	679	682	685	688	691
	694	697	705	708	711	714	717	720	723					
X	478#	486	487	488	489	490	491	492	493	494	495	507#	511	512
	512#	529#	534	535	536	537	538	539	540	541	542	543	544	545
	546	547	548	563#	570	571	572	573	574	575	576	577	578	579
	591#	600	601	602	603	604	605	606	607	608	609			
XP	6#	6	18	499	500	501	513#	1307	1308					