142 MONIT
/PAC/

```
PAC     IDENT   H19 3/7/68
*
*       SCHEDULER AND SWAPPER
*
PACDMB  ZRO     *
PRMSK   DATA    7777B
PLMSK   DATA    77770000B
ADMSK   DATA    37777B
SECMSK  DATA    NSEC-1
SGCI    OPD     16500000B,1,1
SDST    OPD     15100000B,1,1
LDRQ    EQU     20

* ENTRY POINTS

        ENTRY   PLMSK,PRMSK,ADMSK,SECMSK,SRT,SRTE
        ENTRY   PACDMB,POPX,XPOP,PACQE,POPDMS,SETSET,GDAC
        ENTRY   PACGO,CACCF,POPINT,POPST,PUGO,QQEDMS,PACGO1
        ENTRY   FPULST,PUCT,PUEPTR,PUBPTR,PUPAC
        ENTRY   SWAP,UPRL,PTRL,LABEL,CHRL,GDBC
        ENTRY   RRL1,RRL2,RRL3,QUTAB,TIME,TTIME,PUCTR
        ENTRY   RTEX,RREAL,WREAL,CQO,SQO,RSYB,WSYB,DMS
        ENTRY   EPU,FPLST,QSCH,QPUT,IIR,SIR,PGET
        ENTRY   CLINT,CLOCK3,PWFI,TRAPT


*
*       SCHEDULER
*
* EXEC DISMISS
        IF -1
EXDMS   SKN     PQU,2
        BRU     TRAP
        LDX     SS03
        RCH     220B            (CXA,CBX)
        MUL     =3
        LSH     23
        ADD     =Q0
        RCH     440B            (CXB,CAX)
        MIN     0
        BRU     POPDMS
        ENDF
* DISMISS UNTIL INTERRUPT
DMS     LDB     =700005B
        BRU     POPST
*BRS 45 (SIM QUANTUM OFLO)
SQO     LDB     =-1
        STB     TTIME
$SSQO   MIN 0
        BRU     PACQE
*SYSPOP EXIT TO RESTORE CENTRAL REGISTERS
POPX    LDA     SS01
        LDB     SS02
        LDX     SS03
```

```
*SYSPOP EXIT IF CENTRAL REG OK
XPOP  BRR     Ø
*QUANTUM OVERFLOW OCCURED
TRAPT  ZRO
       SKN TIME; BRU* TRAPT
       STA     SSØ1
       STB     SSØ2
       STX     SSØ3
       LDA TRAPT; STA Ø
PACQE  LDB     PACDMB
QQEDMS LDX =QQE; SKN TTIME; LDX =QSQ
*POP DISMISS ENTRY POINT
POPDMS STB     PACLVL
       STX PPREV
       BRM FACP
       LDA     PACPTR
       LDX PPREV
       BRM     QPUT            (PUT PAC ON QUEUE)
       LDB     PACLVL          (PICK UP DISMISS COND)
POPST  LDX     PACPTR
       STB     PTEST,2
POPINT LDA     Ø
       ETR     =5ØØ37777B
       XMA     PL,2
       ETR     =77ØØØØØB
       ADM     PL,2            (SAVE START LOC)
*SAVE CENTRAL REGISTERS
       LDA     SSØ1
       STA     PA,2
       LDA     SSØ2
       STA     PB,2
       LDA     SSØ3
       STA     PX,2
*SET UP PROPER QUANTUM REMAINING
       LDA     TTIME
       SKG     =-1
       LDA     QUTAB
       CLB
       LSH     15
       XMA     PQU,2
       ETR     =6ØØ77777B
       ADM     PQU,2
PACGO  SKN NRCL; MIN RADACT
       LDA TIME; SUB =NSQU-1; SKG =-2Ø; LDA =-2Ø
       CNA; AXC; MIN HISCR,2
       LDX =QSQQ
PACGO2 SKG =9; SKN PNEXT,2; BRU PACGO3
       ADD =1; LDX PNEXT,2; BRU PACGO2
PACGO3 CAX; MIN PACCNT,2
       SKR QQEQF; SKG =Ø; BRM EQQEQ
PACGO1 LDA     =SETTB
       STA     TJOB            (SET ACCOUNT TO SYSTEM)
* START SCHEDULER
       CLA; STA FPFLG
```

```
PLOOP1 LDA REAL; ADD =90
PLOOP2 STA ERFLG; BRM EDRCQ; LDX =NSMEM-NMEM
       LDA ERMT,2; ETR ADMSK; STA ERMT,2; BRX *-3
PLOOP  LDA =-2; STA ACTRP; STA CLOCK3
       ADD =1; STA RUNOUT
       LDA =480; STA ARCVRA; STA CLPCTR
       BRM EXPU; BRM FACP
       BRM PTRY; BRM SPAGE
       LDA =-2; STA FPFLG
       LDA REAL; SKG ERFLG; BRU PLOOP
       LDX =QSQQ; BRM TQGET; BRU PLOOP1
       LDX =QQE; BRM QPUT; BRU PLOOP1
ERFLG  ZRO

EQQEQ  ZRO; ADD =1; LSH 3; STA QQEQF
       LDX =QQEQ; BRM TQGET; BRR EQQEQ
       LDX =QSQ; BRM QPUT; BRR EQQEQ

QQEQF  DATA 4
FILAC  ZRO; STX FILAC2
FILAC1 SKN PNEXT,2; BRR FILAC; STX FILAC3
       LDX PNEXT,2; STX FILAC4
       BRM CACC; BRU FILAC1
       LDA PNEXT,2; LDB FILAC2; LDX FILAC3
       BRM QGET; LDX =QSQ; LDA FILAC4; BRM QPUT
       LDX FILAC3; BRU FILAC1
FILAC2 ZRO
FILAC3 ZRO
FILAC4 ZRO

CACC   ZRO; LDA PIM,2; SKA =BIT3; BRR CACC; STX CACC1
       LDA PTEST,2; LRSH 15; CAX
       LDA CACLST,2; STA T; LDX CACC1
       LDA* PTEST,2; BRU* T
CACCS  LDA REAL; ADD =90; STA ERFLG; MIN CACC
CACCF  LDX CACC1; BRR CACC
CACC1  ZRO

PTRY   ZRO; LDX =QSQQ; STX PACLVL
PTRY1  SKN PNEXT,2; BRU PTRY3
       BRM PPGET; BRU PTRY1
PTRY4  STX PACPTR; BRU PACACT
PTRY3  LDA ACTRP
       SKG =-1; LDA =-1; STA ACTR
       LDA WEFLG; ETR RUNOUT; SKN FPFLG; STA WWEFLG
       LDX =QQEQ; STX PACLVL; SKN PNEXT,2; BRR PTRY
       BRM PPGET; BRR PTRY; BRU PTRY4

PPGET  ZRO; LDA =SWTIM; SKN ACTR; LDA =HSWTIM; STA TJOB
       STX PPREV; LDX PNEXT,2
       SKN RUNOUT; BRR PPGET
       BRM PGET; BRR PPGET
       MIN PPGET; BRR PPGET
```

```
FACP    ZRO; LDX =QTIQ; BRM FILAC
        LDX =QIOQ; BRM FILAC
        BRR FACP

TQGET   ZRO; SKN PNEXT,2; BRR TQGET
        CXB; LDX PNEXT,2; STX PACPTR
        LDA PNEXT,2; CBX; BRM QGET
        LDA PACPTR; MIN TQGET; BRR TQGET

*ACTIVATE PROGRAM
PACACT  LDA     =700001B
        XMA     PTEST,2
        STA     PPTEST          (SAVE PTEST FOR INT SIM)
        MIN ACTCNT
        LDA     PNEXT,2
        LDB     PACLVL
        LDX     PPREV
        BRM     QGET            (GET PROG OFF QUEUE)
PACAC1  LDX     PACPTR
        LDA     PQU,2
        LRSH    15
        ETR     =177B
        STA     TTIME
*SET UP TIME,TTIME, AND ACTR
        LDA     =NSQU-1
        STA     TIME
        STA QTIME
        LDA =12
        STA CLPCTR
*CHECK FOR INTERRUPT AND SET UP START LOC (0)
        LDA PPTEST; LRSH 15; CAX
        LDA CACLST,2
        LRSH    15
        ADD     =ACTLST
        STA     T
        LDX PACPTR
        BRU*    T
PACSRT  LDA     PL,2
        STA     0
PACOVF  LDA     PA,2
        LDB     PB,2
        LDX     PX,2
        BRI     0
*CAUSE A PROG INTERRRUPT
PACINT  LDA     PPTEST
        ETR     =37B
        LDB     PL,2
        SKN     PL,2
        LDB     SBRSRT
        CAX
        LDX     200B,6
        STB*    0,6
        CXA
        ADD     =1; MRG =BIT0
```

```
        STA     0
        LDX     PACPTR
        BRU     PACOVF
*ACTIVATION TEST ROUTINES
CACLST  0       CAC0
        0       CAC1
        0       CAC2
        0       CAC3
        0       CAC4
        1       CACCS
        0       CACCF
        0       CACCF
        0       CACCF
        0       CAC9
        0       CAC10
* SPECIAL ACTIVATION ROUTINES
ACTLST  BRU     PACSRT
        BRU     PACINT


CAC0    CACR    0,(SKG =0)
CAC1    CACR    1,(SKG =0)
CAC2    CACR    0,(SKG =-1)
CAC3    CACR    1,(SKG TTYEMG)
CAC4    BRU*    PTEST,2
CAC9    CACR    0,(SKA X2)
CAC10   CACR    1,(SKG =-1)


* PHANTOM USER
EXPU    ZRO
* P.U. SCHEDULER
PUGO    LDA     PUCTR
        SKG     =0
        BRR EXPU
        LDX     =PUBPTR
PUSCN   STX     PUCPTR
        LDX     0,2
        LDA     2,2
        RSH     12
        STA     PUPAC
        CLA
        LCY     12
        STA     FILE
        LDA     1,2
        STA     PUTST
        LRSH    15
        AXC
        BRU*    PUCLST,2
* CAN'T PROCESS THIS ENTRY
PUNXT   LDX*    PUCPTR
        CXA
        SKE     PUEPTR
```

```
        BRU     PUSCN
        BRR EXPU
* PHANTOM USER ACTIVATION TESTS
PUCLST  DATA    PUDIO,PURBT,PUTRTW,PUBRTW,PUCRTW,PUNXT
        DATA    PUDHU,PUNXT,PUNXT,PUROF
        DATA    PUDSON

PUDSON  LDA FILE; MRG CPTST; STA *+1; SKS* 36200B,2
        BRU *+2; BRU PUACT; LDA REAL; LDX FILE
        SKG TTYTIM,2; BRU PUNXT; BRU PUACT


* TEST DATA-SET TIME OUT
PUDHU   LDX     FILE
        LDA     REAL
        SKG     TTYTIM,2
        BRU     PUNXT
        BRU     PUACT
        IF -1
* TEST IF CARD PUNCH READY
PUCPTW  EXU     CPTW
        BRU     PUNXT
        BRU     PUBRTW
* TEST IF PRINTER READY
PUPRTW  EXU     PRTW
        BRU     PUNXT
        BRU     PUBRTW
        ENDF
* TEST IF TAPE READY
PUTRTW  LDX     FILE
        LDA     FC,2
        RSH     15
        ETR     =77B
        CAX
        EXU     TRTW,2
        BRU     *+2
        BRU     PUNXT
* TEST W-BUFFER
PUBRTW  CLA
        SKE     BLK31
        BRU     PUNXT
        SKS     14000B          TEST W CHANNEL READY
        BRU     PUNXT
        LDX     FILE
        LDA     FA,2
        LRSH    17
        COPY    AX,B
        STX     JOB
        LDA     PMTP,2
        ADD     =20000000B-NCMEM
        STA     PMTJOB
        CLA
        LDX     RL3,2
        MIN ACTRP


-----
```

```
        BRM     SWAP
        BRU     PUNXT
        BRM     LABEL
        BRU     PUACT1
PUACTW  LDX     FILE
        LDA     XN2
        ADM     FD,2
        LDB     FD,2
        STB     DEV
        LDA     FC,2
        ETR     ADMSK
        STA     BUFF
        LDB     PUTST
        BRM     ED
        BRU     PUGO
        BRU     *
        BRU     PUGO
* TEST CARD READER READY
PUCRTW  EXU     CFTW
        BRU     PUACT
        EXU     CRTW
        BRU     PUNXT
        BRU     PUBRTW
* TEST IF DRUM FILE READY
PUDIO   LDX     FILE
        LDA     FD,2
        SKA     =DBB
        BRU     PUNXT
        BRU     PUACT
* TEST IF RUBOUT APPLICABLE
PURBT   LDX     FILE
        SKN     TTYASG,2
        BRU     PURBT2
        LDX     TTYASG,2
        LDA     =PURBTA
        BRM     SCFK
        STX     PUPAC
        BRU     PUACT
PURBT2  LDA FPLST; SKG =0; BRU PUNXT
        SKN FULST; BRU PUACT1; BRM RCVR

PURBTA  ZRO
        DIR
        LDA     PIM,2
        SKA     =BIT2+BIT3
        BRU     PURBT1
        EIR
        BRR     PURBTA
PURBT1  MRG     X2
        EIR
        STA     PIM,2
        BRU     PUNXT
PUROF   LDX FILE; SKN TTYASG,2; BRU PUROF1
        LDX TTYASG,2; STX PUPAC; BRM EFK
```

```
        LDA =PUROFA; BRM SCFK; BRU PUACT
PUROF1  LDA =PUGO+100000B; STA PUTST; BRU PUACT1
PUROFA  ZRO; LDA PIM,2; SKA =BIT2+BIT3; BRU PUNXT; BRR PUROFA
* ACTIVATE P.U. REQUEST ROUTINE
PUACT   LDX     PUPAC
        BRM     PGET
        BRU     PUNXT
PUACT1  LDX     PUCPTR
        DIR
        LDA     FPULST
        XMA*    0,2
        XMA     0,2
        STA     FPULST
        SKE     PUEPTR
        BRU     *+2
        STX     PUEPTR
        LDA     PUTST
        LRSH    15
        SKR     PUCTR
        CAX
        EIR
        BRU*    PUCSET,2
* PHANTOM USER REQUEST PROCESSING PREPARATION
PUCSET  ZRO     PUDIOS
        ZRO*    PUTST
        DATA    PUACTW,PUACTW,PUACTW,PUGO
        ZRO*    PUTST
        DATA    PUACTW,PUACTW
        ZRO* PUTST
        ZRO* PUTST
* CONTINUE DRUM BRS
PUDIOS  LDX PUPAC; LDA PIM,2; ETR =(NOT)BIT3; STA PIM,2; LDX FILE
        LDA     XN2
        ADM     FD,2
        LDA     FC,2
        ETR     ADMSK
        STA     BUFF
        CAX
        BRR     PUTST
* MAKE P. U. ENTRY
EPU     ZRO
        MIN ACTR
        STA EPU1; STB EPU2; LDX PUBPTR
EPU4    CXA; SKE =PUBPTR; BRU EPU3
        LDA EPU1; LDB EPU2
        XMA*    FPULST
        SKE =0; BRU *+2; BRM RCVR (PUQ OVRFLO)
        MIN     PUCTR
        XMA     FPULST
        CAX
        XMA*    PUEPTR
        STX     PUEPTR
        XMA     0,2
        STA     1,2
```

```
        STB     2,2
        EIR
        BRR     EPU
EPU3    LDA 1,2; LDB 2,2; LDX 0,2; SKE EPU1; BRU EPU4
        CBA; SKE EPU2; BRU EPU4; EIR; BRR EPU
EPU1    ZRO
EPU2    ZRO

* GET USER TO CORE
PGET    ZRO
        STX     PUPAC
        SKN     PUPAC
        BRU     PGET1
        LDA     PTAB,2
        LRSH    15
        ETR     =177B
        STA     JOB
        MIN ACTRP
        CAX
        ADD     =DBA
        STA     DBAJOB          (SET UP DBA POINTER)
        LDA     PMTP,2
        ADD     =20000000B-NCMEM
        STA     PMTJOB          (SET UP PMT PTR)
        LDA     TTNO,2
        STA     UTTY            (SET UP USER TELETYPE)
        LDX     PUPAC
        BRM     CHRL
        BRU PGET3
PGET2   LDX     JOB
        EAX     ETTB,2
        STX     TJOB            SET UP TIME CHARGING
PGET1   MIN     PGET
PGET3   LDX PUPAC
        BRR     PGET


* SET UP REAL RELABELING
LABEL   ZRO
        STA     RRL1
        STB     RRL2
        STX     RRL3
        LRR1
        POT     RRL1
        LRR2
        POT     RRL2
        LRR3
        POT     RRL3
        BRR     LABEL
CHRL    ZRO
        LDA     RL1,2
        LDB     RL2,2
        LDX     JOB
```

```
        LDX     RL3,2
        BRM     SWAP
        BRR     CHRL
        BRM     LABEL
        MIN     CHRL
        BRR     CHRL
```

SWAPPER J.T. HOLLAND 4 MAY 1967

*THIS SWAPPER WORKS IN TWO MODES: A SCHEDULER MODE AND A UTILITY
*MODE. THE MODE IS DETERMINED BY THE SETTING OF SFLG, WHICH IS
*NEGATIVE FOR SCHEDULER MODE AND NON-NEGATIVE FOR UTILITY MODE.
*THE SCHEDULER MODE SHOULD ONLY BE USED FOR CALLS FROM PACACT.
*SWAP IS CALLED WITH PSEUDO-RELABELING IN A,B AND X. IT UNPACKS
*THE RELABELING INTO THE ARRAY SRT. IF SFLG INDICATES UTILITY
*MODE, IT COPIES SRT INTO THE WORKING ARRAY WSRT. OTHERWISE, IT
*CONTINUES TRYING TO FIND PAGES FOR THE ARRAY ALREADY IN WSRT. IF
*IT SHOULD SUCCEED IN ASSIGNING PAGES TO EVERY REQUEST IN WSRT,
*IT WILL COPY SRT INTO WSRT AND BEGIN PAGE ASSIGNMENT. IN ANY CASE,
*SWAP WILL NOT SKIP UNLESS ALL THE PAGES REQUESTED ARE IN CORE.
*THUS IN ALL CALLS TO SWAP OTHER THAN IN PACACT, THERE SHOULD BE
*HANG LOOPS IF SUCCESSFUL SWAPPING IS NECESSARY. IF ALL THE PAGES
*REQUESTED ARE IN CORE, SWAP WILL SET UP THE REAL RELABELING,
*(WITH THE READ ONLY BIT ON FOR EACH PAGE IN THE PROGRAM RELABELING)
*AND RETURN IT IN A,B, AND X. THE READ ONLY TRAP ROUTINE INTERROGATES
*TO SEE WHETHER THE PAGE ADDRESSED WAS REALLY READ ONLY. IF NOT,
*IT TURNS OFF THE READ-ONLY BIT, AND TURNS OFF THE BIT IN THE
*PMT ENTRY WHICH INDICATES A VALID RAD COPY. THUS, IF A PAGE IS
*NOT ALTERED, IT NEED NOT BE WRITTEN OUT. FURTHERMORE, ON EVERY CALL
*TO SWAP, IT SCANS THE PAGES IN MEMORY. A WRITE IS INITIATED FOR
*ANY PAGE WHICH HAS NOT BEEN REQUESTED BY SOMEBODY AND DOES NOT
*HAVE A VALID RAD IMAGE.

PMT/SMT STATUS BITS

*BIT 0 INDICATES THAT A VALID COPY OF THIS PAGE EXISTS ON THE RAD

*BIT 1 WHEN THIS BIT IS ON, THE PMT ENTRY REALLY POINTS TO AN
*   ENTRY IN THE SMT. THE BYTE NUMBER IS IN THE ADDRESS FIELD.
*   IF THIS BIT SHOULD GET TURNED ON ACCIDENTALLY, ALL HELL WILL
*   BREAK LOOSE.
*BIT 2 MGET TURNS THIS BIT ON WHEN IT ASSIGNS A FRESH
*   PAGE. IT SIGNIFIES THAT ALTHOUGH THERE IS NO COPY OF THE
*   PAGE IN CORE, NO RAD READ IS REQUIRED. ANY OLD JUNK
*   WILL DO.
*BIT 3 INDICATES THAT THERE IS A VALID COPY OF THIS PAGE IN CORE

*BIT 4 INDICATES THAT THERE IS A READ IN PROGRESS.
*BIT 5 INDICATES THAT THERE IS A WRITE IN PROGRESS.
*BIT 6 INDICATES THAT THIS PAGE IS TO BE MADE READ ONLY.
*BIT 7 IS UNASSIGNED
*BIT 8 SAYS THAT THIS PAGE CAN BE ACCESSED ONLY BY PROGRAMS WITH
*   EXEC STATUS.

```
*BITS 9 THRU 17 CONTAIN THE RAD ADDRESS OF THE PAGE.
*BIT 18 IS THE READ ONLY BIT AND IS ON IF THE PAGE IS READ ONLY.
*BITS 19 TO 23 CONTAIN THE REAL PAGE ADDRESS, IF THE PAGE IS
*       IN CORE.  IF THERE IS NO VALID CORE IMAGE OF THE PAGE
*       THESE BITS SHOULD BE ZERO.

*RMT STATUS BITS*

*IF THE RMT ENTRY FOR A PAGE IS ALL ZERO, THE PAGE IS UNUSED.
*OTHERWISE THE FOLLOWING CONVENTIONS HOLD.

*BIT 0 MUST BE ZERO SINCE THERE IS A LOT OF INDIRECT ADDRESSING
*       THROUGH THE RMT TABLE

*BIT 1 HAD BETTER BE 0, SINCE SWAP DOES A FAIR AMOUNT OF
*       INDIRECT ADDRESSING THROUGH RMT, AND THE INDEX BIT
*       BETTER BE 0.
*BIT 2 MEANS THAT ON SOME CALL TO SWAP, THIS PAGE HAS BEEN ASSIGNED.
*       IN SCHEDULER MODE, THIS PAGE WILL NOT BE AVAILABLE TO SATISFY
*       A REQUEST.  IN UTILITY MODE THIS BIT WILL BE ON ONLY IF
*       IT HAS BEEN ASSIGNED TO SATISFY THE CURRENT REQUEST.

*BITS 3 THRU 8 ARE NOT USED.
*BIT 9 MUST BE ZERO BECAUSE OF INDIRECT ADDRESSING VIA THE RMT TABLE
*BITS 10 THRU 23 CONTAIN THE CORE ADDRESS OF THE PMT OR SMT ENTRY
*       WHICH IS RESPONSIBLE FOR THIS PAGE.

*SWAP*

SWAP    ZRO
        BRM     SWIN            INITIALIZE THE SWAPPER
        BRM     GPAGE           SCAN THE REQUESTED PAGES AND START READS
        BRM     MKRL            MAKE UP RELABELING IF POSSIBLE
        MIN     SWAP            POSSIBLE, SO SKIP
        BRR     SWAP

*SWIN*

SWIN    ZRO
        STA     SWR1            SAVE REGISTERS IN CASE OF NEED
        STB     SWR2
        STX     SWR3
        BRM     UPRL            UNPACK PSEUDO RELABELING
        LDA     =-1
        STA     FLFLG           INITIALIZE SWAP-FAILURE FLAG
        BRM     DCRL            FIND PMT ENTRIES
        BRR     SWIN

*DCRL*

DCRL    ZRO
        LDX     =-10
DCRL1   LDA     SRTE,2          FETCH PSEUDO-RELABELING BYTE
        COPY    XB,AX
```

```
        SKG     =NCMEM-1        PMT OR SMT?
        BRU     DCRL3           SMT
        EAX*    PMTJOB          COMPUTE TRUE ADDRESS
DCRL7   LDA     0,2             FETCH PMT ENTRY
        SKA     =BIT1           DOES IT POINT TO SMT?
        BRU     DCRL4           YES
        COPY    XA,BX
        STA     SRTE,2
DCRL2   BRX     DCRL1
        BRR     DCRL
DCRL4   COPY    AX,E
DCRL3   EAX     SMT,2
        BRU     DCRL7

*CPAGE*

CPAGE   ZRO
        SKN DRCC; BRU *+2; BRU CPAGE6
        BRM CWIP
        BRU CPAGE3
CPAGE6  LDX     =-10
CPAGE1  LDA*    SRTE,2          FETCH PMT ENTRY
        SKA     =BIT3           VALID PAGE IN CORE?
        SKA     =BIT6           YES, CAN WE TOUCH IT?
        BRU     CPAGE4
        BRX     CPAGE1
        LDX     =-10            ALL PAGES IN CORE AND IDLE
CPAGE2  LDA*    SRTE,2
        BRM     MPAGE           MARK THE PAGE
        BRX     CPAGE2
        BRR     CPAGE           RETURN SKIPPING
CPAGE3  MIN     FLFLG; MIN RUNOUT
        BRR     CPAGE
CPAGE4  MIN CPAGE
        BRR     CPAGE

CWIP    ZRO; LDX =-10; LDA =BIT5
        SKA* SRTE,2; BRR CWIP; BRX *-2
        MIN CWIP; BRR CWIP

*GPAGE*

GPAGE   ZRO
        BRM     CPAGE           CHECK FOR PAGES IN CORE
        BRR     GPAGE           YES
GPAGE3  SKN     WEFLG
        BRM     PSCN
        SKN     FLFLG           HAVE WE FINISHED WITH THE LAST REQUEST?
        BRR     GPAGE           NO, CAN'T DO ANY MORE
        LDX     =-10            YES, MOVE THIS REQUEST
        LDA     SRTE,2
        STA     WSRTE,2
        BRX     *-2
        BRM     PSCN            TRY TO GET PAGES FOR THIS GUY
```

```
        SKN     FLFLG
        MIN     WEFLG
        MIN     FLFLG           INDICATE FAILURE
        BRR     GPAGE
*PSCN*

*PSCN SCANS THROUGH THE PAGES REQUESTED.  IF ANY SUCH PAGE IS NOT
*IN CORE, PSCN WILL CALL FPAGE TO FIND AN AVAILABLE PAGE IN MEMORY.
*PSCN WILL THEN ASSIGN THE CORE TO THE PAGE REQUESTED AND INITIATE
*A READ.  IT WILL ALSO TAKE CARE OF THE NECESSARY OPERATIONS TO
*MAKE A PAGE READ-ONLY, IF THAT HAS BEEN REQUESTED.

PSCN    ZRO
        LDX     =-10
PSCN11  LDA =(NOT)BIT5; DIR; ETR* WSRTE,2; EIR; STA* WSRTE,2
        SKA     =BIT3+BIT4+37B
        BRM     MPAGE
PSCN12  BRX     PSCN11
        LDX     =-10
PSCN1   LDA*    WSRTE,2         YES, FETCH PMT ENTRY
        SKA     =BIT6           SHOULD WE MAKE THIS PAGE READ ONLY?
        BRU     PSCN7           YES
        SKA =BIT7; BRU PSCN15
PSCN10  SKA     =BIT3+BIT4+BIT7 VALID PAGE OR READ IN PROGRESS?
        BRU     PSCN2           YES
        SKA     =37B            PAGE ALREADY ASSIGNED?
        BRU     PSCN13          YES
        BRM     FPAGE           NO, TRY TO FIND A PAGE FOR THIS ONE
        BRU     PSCN14          NO PAGE AVAILABLE.  DEFER READ.
        MRG*    WSRTE,2          PUT REAL PAGE NO IN PMT ENTRY
        SKA     =BIT2           PAGE-GRAB?
        BRU     PSCN5           YES
PSCN13  MRG     =BIT4           NO, MARK READ IN PROGRESS
        MIN     SWRCTR
PSCN3   LDB     WSRTE,2
        STA*  WSRTE,2
        BRM     SRADC           PUT READ IN QUEUE
PSCN8   BRM     MPAGE
PSCN2   BRX     PSCN1
        BRR     PSCN
PSCN5   MRG     =BIT3           MARK AS VALID CORE IMAGE
        STA*  WSRTE,2
        BRU     PSCN8
PSCN7   SKA     =BIT5           WRITE IN PROGRESS?
        BRU     PSCN8           YES
        SKA     =BIT0+40B       VALID COPY ON RAD?
        BRU     PSCN9           YES
        MRG     =BIT5           MARK WRITE IN PROGRESS
        MIN     SWWCTR
        BRU     PSCN3
PSCN9   DIR
        LDA*    WSRTE,2
        MRG     =40B            MARK AS READ ONLY
        ETR     =(NOT)BIT6
```

```
        STA*    WSRTE,2
        EIR
        BRU     PSCN10          CONTINUE
PSCN16  MIN  FLFLG; BRU PSCN2
PSCN14  LDA  WIPFLG; SKG DRCC; BRU PSCN16
        LDA  =LDRQ-1; SKG DRCC; BRU PSCN16
        LDA* WSRTE,2; MRG =BIT7; STA* WSRTE,2
        MIN  DRCIP; LDA* DRCIP; SKA =7774000ØB; ADM DRCIP
        LDA  WSRTE,2; STA* DRCIP; MIN DRCC; BRU PSCN2
PSCN15  MIN  FLFLG; SKN WIPFLG; BRU PSCN10
        LDA  =(NOT)BIT7; DIR; ETR* WSRTE,2
        EIR; STA* WSRTE,2; BRU PSCN10

*MPAGE*

MPAGE   ZRO
        ETR     =37B            EXTRACT REAL PAGE NUMBER
        COPY    AX,XB
        LDA  =BIT2; DIR; MRG RMT,2; EIR; STA RMT,2
        LDA  SREAL; SUB =1ØØB; STA SREAL
        STB  MPAGE1; LDB* RMT,2; SKB =4ØB; LSH 1; LRSH 2; LDB MPAGE1
        SKG  RMA,2; LDA RMA,2
        STA     RMA,2
        CBX
        BRR     MPAGE
MPAGE1  ZRO


*FPAGE*

*FPAGE SCANS THROUGH REAL MEMORY LOOKING FOR A PAGE WHICH CAN BE
*RELEASED AND REASSIGNED.  IN SCHEDULER MODE, IT WILL NOT RELEASE
*ANY PAGE FOR WHICH BIT Ø OF RMT IS SET, AND WILL RELEASE
*THE OLDEST SUCH PAGE IT FINDS.  IN UTILITY MODE, IT WILL RELEASE
*AN ASSIGNED PAGE IF NECESSARY, AND THE YOUNGEST ONE FIRST.  IN
*NO EVENT WILL FPAGE RELEASE A PAGE IS THERE IS ANY RAD ACTIVITY
*ASSOCIATED WITH IT.

FPAGE   ZRO
        STX     FPAGE9
        BRM     OPAGE
        BRU     FPAGE1
        BRM     RPAGE
        LDX     FPAGE9
        LDB     WSRTE,2
        XXA
        STB     RMT,2
        XXA
        MIN     FPAGE
        BRR     FPAGE
FPAGE1  BRM  SPAGE
        LDX     FPAGE9
        BRR     FPAGE
FPAGE9  ZRO
```

```
*OPAGE*

OPAGE  ZRO
       LDX     =NSMEM-NMEM
       LDA     XX
       STA     OPAGE4
OPAGE3 SKN     ERMC,2          LOCKED?
       BRU     OPAGE1          YES, IGNORE IT
       LDB     ERMT,2
       SKB     =-1             PERHAPS IT IS ZERO?
       BRU     *+2             NO
       BRU     OPAGE2          THIS IS OUR BOY
       SKB     =BIT2           PAGE ALREADY ASSIGNED?
       BRU     OPAGE1          YES
       LDB*    ERMT,2
       SKB     =BIT0+BIT2+40B  CAN WE THROW CORE IMAGE AWAY?
       SKG     ERMA,2          YES, OLDER?
       BRU     OPAGE1          NO, LEAVE HIM ALONE
       LDA     ERMA,2
       STX     OPAGE4
OPAGE1 BRX     OPAGE3
       SKN     OPAGE4
       BRR     OPAGE
       LDX     OPAGE4
OPAGE2 MIN     OPAGE
       CXA
       ADD     =NMEM
       CAX
       BRR     OPAGE
OPAGE4 ZRO

*RPAGE*

RPAGE  ZRO
       ABC
       SKE     RMT,2
       BRU     *+3
       CBA
       BRR     RPAGE
       LDA*    RMT,2           FETCH PMT ENTRY
       ETR     =50177740B      RELEASE CORE IMAGE
       SKA     =BIT2           PAGE GRAB?
       ETR     =10177740B      YES, THROW AWAY VALIDITY BITS
       STA*    RMT,2
       BAC
       STB     RMT,2
       BRR     RPAGE

*EDRCQ*

EDRCQ  ZRO
EDRCQ2 DIR
```

```
        SKN     DRCC
        BRU     EDRCQ1
        EIR
        BRR     EDRCQ
EDRCQ1  LDA     =1
        ADM     DRCOP
        LDA*    DRCOP
        SKA     =77740000B
        BRU     *-3
        EIR
        SKR     DRCC
        NOP
        CAX
        LDA =(NOT)BIT7; DIR; ETR 0,2; EIR; STA 0,2
        BRU     EDRCQ2
*SPAGE*

*SPAGE SCANS THROUGH REAL MEMORY LOOKING FOR UNREQUESTED PAGES
*WHICH DO NOT HAVE A VALID RAD IMAGE.  IT INITIATES WRITES FOR
*EACH SUCH PAGE WHICH IT FINDS.

SPAGE   ZRO
        LDX     =NSMEM-NMEM
SPAGE2  SKN     ERMC,2          NO, LOCKED?
        BRU     SPAGE1          YES, LEAVE HIM ALONE
        LDB     ERMT,2
        SKB     =-1             PAGE UNASSIGNED?
        SKB     =BIT2           PAGE ATTACHED?
        BRU     SPAGE1          NO NEED TO WRITE
        LDA*    ERMT,2
        SKA     =BIT0+BIT5+BIT4+BIT2+40B   SHOULD WE WRITE HIM OUT?
        BRU     SPAGE1          NO
        MRG     =BIT5           MARK WRITE IN PROGRESS
        STA*    ERMT,2
        BRM     SRADC           GENERATE WRITE COMMAND
        MIN     SWWCTR          COUNT NO OF WRITES
SPAGE1  BRX SPAGE2
        BRR     SPAGE

*SRADC*

SRADC   ZRO
        STA     SRADC1
        STB     SRADC2
        STX     SRADC3
        SKA =BIT5; MIN WIPFLG
        SKA =BIT4; MIN RIPFLG
        BRM     GRC
        LDX     ERCL
        MIN     3,2             SET UP INTERRUPT ROUTINE
        DIR
        BRM IRTC
        BRM RTS
        LDA     SRADC1
```

```
        LDB     SRADC2
        LDX     SRADC3
        BRR     SRADC
SRADC1  ZRO
SRADC2  ZRO
SRADC3  ZRO

*GRC*

GRC     ZRO
        CBX
        LSH     11
        ETR     =174000B        EXTRACT ABSOLUTE CORE ADDRESS
        LDB     0,2             FETCH PMT ENTRY
        SKB     =BIT5           WRITE?
        MRG     =BIT0           YES, MARK IT SO
        COPY    AX,BA
        LSH     5
        ETR     =3774000B       EXTRACT RAD ADDRESS
        COPY    AB,XA
        LDX     =4000B          WORD COUNT
        BRM     RTC
        BRR     GRC

*MKRL*

MKRL,   ZRO
        SKN     FLFLG           NO, FAILURE FLAG ON?
        BRU     MKRL3           YES, SKIP RETURN
MKRL9   SKN FPFLG               FIRST PASS THRU SCHEDULER?
        BRU MKRL3               YES, DON'T SKIP
        LDX =-10; LDA* SRTE,2; SKA =BIT2; BRU MKRL7
MKRL6   BRX *-3
        LDX     =-8
MKRL1   LDA =44177777B
        DIR
        ETR*    SRT+8,2
        EIR
        STA*    SRT+8,2
        SKA     =BIT0           VALID COPY ON RAD?
        MRG     =40B            YES, MAKE IT READ-ONLY
        ETR     =77B
        STA     SRT+8,2
        BRX     MKRL1
        LDX     =-2
MKRL2   LDA =41777778
        DIR
        ETR*    SRTE,2
        EIR
        STA*    SRTE,2
        ETR     =77B
        STA     SRTE,2
        BRX     MKRL2
        LDA =-1; STA WEFLG
```

```
        LDX     =NSMEM-NMEM
MKRL5   LDA ADMSK; DIR; ETR ERMT,2; EIR; STA ERMT,2
        LDA ERMA,2; LRSH 1; STA ERMA,2
        BRX     MKRL5
        LDA XX; XMA SREAL; LRSH 1; STA SREALP
        BRM     PTRL            PUT THE RELABELING BACK TOGETHER
        BRR     MKRL            NO-SKIP RETURN WITH REAL RELABELING
MKRL7   ETR =37B; XMA RRL3; LRR3; POT RRL3; STA MKRL8
        COPY A,XB; LDX =-4000B; STA 0,2; BRX *-1
        LDA MKRL8; STA RRL3; LRR3; POT RRL3
        CBX; BRU MKRL6
MKRL8   ZRO
MKRL3   LDA     SWR1            RESTORE REGISTERS
        LDB     SWR2
        LDX     SWR3
        MIN     MKRL
        BRR     MKRL            SKIP RETURN

*PTRL*

PTRL    ZRO; CLEAR; LDA SRT+8; LSH 6; MRG SRT+9; STA PTRL1
        LDX =-8; CLA; MRG SRT+8,2; LCY 6; BRX *-2
        LCY 18; LDX PTRL1; BRR PTRL
PTRL1   ZRO

*UPRL*

UPRL    ZRO; STB UPRL1; LRSH 18; STA SRT
        CLA; LCY 6; STA SRT+1; CLA; LCY 6; STA SRT+2
        CLA; LCY 6; STA SRT+3; LDA UPRL1; LRSH 18; STA SRT+4
        CLA; LCY 6; STA SRT+5; CLA; LCY 6; STA SRT+6
        CLA; LCY 6; STA SRT+7; CXA; RSH 6; ETR =77B; STA SRT+8
        CLA; LCY 6; STA SRT+9; BRR UPRL
UPRL1   ZRO

*IRS*

*IRS IS THE CLEAN UP ROUTINE CALLED BY THE RAD INTERRUPT
*PROCESSOR.  ON READS, IT CHECKS FOR A SUCCESSFUL READ, AND
*IF SO TURNS ON THE VALID CORE COPY BIT.  IN ANY CASE IT TURNS
*OFF THE READ IN PROGRESS BIT.
*ON WRITES, IT FIRST CHECKS THE WRITE IN PROGRESS BIT TO SEE IF
*IT IS STILL ON.  IF NOT, IT FIGURES THAT THE WRITE TURNED
*OUT TO BE REDUNDANT, AND THE CORE COPY IS ALREADY SUBJECT TO
*FURTHER MODIFICATION.  IF THE WRITE IN PROGRESS BIT IS STILL
*ON, IT TURNS IT OFF AND TURNS ON THE VALID DRUM COPY BIT IF THE
*WRITE WAS SUCCESSFUL.

$IRS    ZRO
        COPY    BX,AB
        LDX     3,2
        XXB
        LDA*    RMT,2           FETCH PMT ENTRY
        SKB     =BIT0           WRITE?
```

```
        BRU     IRS1            YES
        SKA     =BIT4           STILL INTERESTED IN THIS READ?
        BRU     *+2             YES
        BRR     IRS             NO
        SKR RIPFLG; NOP
        ETR     =(NOT)BIT4      TURN OFF READ IN PROGRESS BIT
        SKN     RADTRY          UNRECOVERABLE RAD ERROR
        MRG     =BIT3           NO, MARK CORE COPY AS VALID
        STA*    RMT,2
        BRR     IRS
IRS1    SKR WIPFLG; NOP
        SKA     =BIT5           STILL INTERESTED IN THIS WRITE?
        BRU     *+2
        BRR     IRS             NO
        SKN     RADTRY          UNRECOVERABLE RAD ERROR?
        MRG     =BIT0
        ETR =(NOT)BIT5; STA* RMT,2
        SKN RADTRY; BRU IRS2; BRR IRS
IRS2    SKN DRCC; BRU *+2; BRR IRS
        LDA =1
        ADM DRCOP; LDA* DRCOP; SKA =77740000B; BRU *-3
        SKR DRCC; NOP
        XXA; LDB 0,2; XXA; SKB =BIT7; BRU *+2; BRU IRS2
        CAB; LDA* RMT,2
        ETR =40577740B; STA* RMT,2
        LDA SREALP; STA RMA,2
        MIN RMC,2; LDA* FRQLST; XMA FRQLST
        ADD =1; STA IRCL1
        CBA
        SKN WEFLG; MRG =BIT2
        STA RMT,2
        XXA; ETR =37B; MRG 0,2
        ETR =(NOT)BIT7
        MRG =BIT4; STA 0,2
        CAX; LRSH 1; ETR =37740B; STA* IRCL1
        CXA; LSH 2; ETR =140B; MRG IRS3
        XXA; LSH 11; ETR =34000B
        LDB IRCL1; XXB; STA 1,2; STB 2,2
        LDA =1; STA 3,2
        MIN RIPFLG
        MIN SWRCTR
        SKN ICOUNT; BRU IRS4
        STX IRCL; BRM RRETRY
IRS5    MIN ICOUNT; BRR IRS
IRS4    BRM IRTC; BRU IRS5
IRS3    EOD 17202B,4

*TRAPR*

*TRAPR IS THE READ-ONLY TRAP ROUTINE.  IT CHECKS TO SEE IF THE
*READ ONLY TRAP IS LEGITIMATE.  IF SO, IT BRANCHES OFF TO THE
*DIAGNOSTIC ROUTINE.  OTHERWISE, IT RESETS THE RELABELING
*APPROPRIATELY AND TURNS OFF THE VALID RAD COPY BIT.
```

```
STRAPR  ZRO
        STA     TRAPR1
        STB     TRAPR2
        STX     TRAPR3
        LDA TRAPR
        BRM     CAE             COMPUTE EFFECTIVE ADDRESS
        NOP     0               WE DON'T CARE ABOUT INDIRECT ADDRESSING
        LRSH    11
        ETR     =7              GET BYTE NUMBER
        MUL     =3              COMPUTE SHIFT
        CBX
        LDA     RRL2            GET REAL RELABELING
        LDB     RRL1
        LCY     6,2
        ETR     =37B            EXTRACT PAGE NUMBER
        COPY    AX,XB
        LDA*    RMT,2           GET PMT/SMT ENTRY
        SKA     =40B            REALLY READ ONLY?
        BRU     TRAPR4          YES
        DIR
        LDA*    RMT,2
        ETR     XX              TURN OFF VALID RAD COPY BIT
        STA*    RMT,2
        EIR
        CBX
        LDA     RRL1
        LDB     RRL2
        LCY     0,2
        ETR     XX              TURN OFF READ ONLY BIT
        RCY     0,2
        LDX     RRL3
        BRM     LABEL           RESET THE RELABELING
        LDA     TRAPR1
        LDB     TRAPR2
        LDX     TRAPR3
        BRI     TRAPR           RETURN TO PROGRAM
TRAPR4  LDA     TRAPR
        STA     0               SET THINGS UP FOR THE TRAP ROUTIN
        LDA     TRAPR1
        STA     SS01
        LDA     TRAPR2
        STA     SS02
        LDA     TRAPR3
        STA     SS03
        BRU     MTRAP
TRAPR1  ZRO
TRAPR2  ZRO
TRAPR3  ZRO

*FLAGS, POINTERS AND STORAGE*


SWR1    ZRO
SWR2    ZRO


-----
```

```
SWR3    ZRO

DRCIP   DATA    DRCQ
DRCOP   DATA    DRCQ
DRCC    DATA    -1
DRCQ    BSS     LDRQ
        DATA    -LDRQ
SREAL   DATA 37777777B
SREALP  DATA 37777777B
$WIPFLG DATA -1                 NEGATIVE IF NO WRITES IN PROGRESS
RIPFLG  DATA -1
WWEFLG  DATA -1
RUNOUT  DATA -1
WEFLG   DATA    -1              NEGATIVE IF WSRT IS EMPTY
ACTRP   DATA    -2
FPFLG   ZRO     0               NON-NEGATIVE FIRST PASS THRU SCHEDULER
FLFLG   ZRO     0               IF NON-NEGATIVE, THE SWAP HAS FAILED
SRT     BSS     10              TABLE FOR UNPACKING RELABELING
SRTE    EQU     *
WSRT    RPT     10
        DATA    SMT
        ENDR
WSRTE   EQU     *
RRL1    ZRO
RRL2    ZRO
RRL3    ZRO


* READ OR WRITE 2K (BRS 104,105)
WSYB    LDB     =RTW
        BRU     RSYB+1
RSYB    LDB     =RTC
        STB     RSYBT1
        ETR     =34000B
        SKN     PQU,2
        BRU     TRAP
        CAX
        LDB     0,6
        STB 0,6
        BRM     DTH
        LDB     SS02
        SKB =74000000B; BRU TRAP
        LDX     =4000B
        BRM*    RSYBT1
        BRM DISA; LDX ERCL; ADM 3,2; DIR; BRM IRTC
        BRM RTS; BRU ARWD2
$RSYBI  ZRO; LDX IRCL1; LDA 3,2; LRSH 6; MRG PLMSK
        CAX; SKN RADTRY; MIN PL,2; LDA PIM,2
        ETR =(NOT)BIT3; STA PIM,2; BRR RSYBI
RSYBT1  ZRO

PACLVL  ZRO
PPREV   ZRO
PPTEST  ZRO
```

```
FPLST  ZRO

PUCT   BSS     NPUQ*3
$EPUCT3 EQU    PUCT+NPUQ*3+3
$EPUCT EQU     PUCT+NPUQ*3
$EPUCM3 EQU    PUCT+NPUQ*3-3
FPULST ZRO
PUBPTR ZRO
PUCTR  ZRO
PUCTR1 ZRO
PUCPTR ZRO
PUEPTR ZRO
PUPAC  ZRO
PUTST  ZRO
$PUTIM  ZRO
$ARCVRA ZRO 3600

*
*      CLOCK ROUTINES AND TABLES
*
QUTAB  ZRO     NFQU            NUMBER OF CLOCK CYCLES IN QUANTUM
QTIME  ZRO
TIME   ZRO
TTIME  ZRO
CLOCK3 ZRO
* CLOCK INTRRUPT ROUTINE  AVG. TIME= .047 MS
CLINT  ZRO
       MIN     REAL
       MIN*    TJOB
       SKN     CLINT
       MIN     STIME
       SKN ACTR; MIN HREAL
       SKS 14000B; MIN WBTIM
       CATE; MIN CETIM
       SKR     ARCVRA
       BPT4
       BRM     RCVR
       SKR TTIME; NOP
       SKR     TIME
       BRI     CLINT
       SKN NRCL; BRI CLINT
       SKN     ACTR
       BRU CLOUT; SKR QTIME; BRI CLINT; SKN TTIME
       BRI     CLINT
CLOUT  EOM 22400B; SKR CLPCTR; BRI CLINT
       ARMI SATIW; MIN CLINTI; BRI CLINT
$CLINTA SKR CLINTI; NOP; SKN ATII; BRU *+2; BRU INT31B
       LDA =TRAP; STA ATII; STA CLPCTR; BRU INT31B
$CLINTI DATA -1
$CLPCTR DATA 3600
* FAST CLOCK (POWER OFF INTERRUPT)
PWFI   ZRO
       BRI     PWFI
```

```
* BRS'S FOR TIMING
RTEX    LDX     JOB
        LDA     ETTB,2
        STA     SSØ1
        BRU     POPX
RREAL   LDA     REAL
        LDB DMIN; LDX SSØ3; BRU XPOP
WREAL   MUL     =1727024B
        SKB     X4
        ADD     =1
        ADD     REAL
        STA     SSØ1
        LDX     =QTI
        LDB     WREC1
        MIN     Ø
        BRU     POPDMS
WREC1   4       *+1
        LDA     PA,2
        CACR    1,(SKG REAL)
CQO     LDA     TIME
        SKG     =1
        BRU     SSQO
        BRU     POPX
$CONT   ZRO; CKF; LDA =16*20000B; LDB DCWBIT; LDX =40000B
        BRM GDAC; BRU *-4; BRM GB6; BRU CONTA
$LOAD   BRM GB6; BRU CONTA


*
* SYSTEM RESTART
*
RSTART ZRO
        CKF
        LDA     =DMPBND*20000B
        LDB     DCWBIT
        LDX     =40000B
        BRM     GDAC
RESET1 HLT      Ø
        BRM     GB6
        BRR     RSTART

RCVR1   ZRO
RCVR2   ZRO
RCVR3   ZRO
$RCVR   ZRO
        STA RCVR1; STB RCVR2; STX RCVR3
        ARMI =600000B; CKF
        BRI =*+1
        SKN NDCL; BRU *-1; SKN NRCL; BRU *-1
        BRM     RSTART
        BRU     CONTA  RE INITIALIZE SYSTEM- MUST RE BITMAP

* SYSTEM START
$SETSET BRM     GB6
```

```
        BRU     SETSA

* GET DRUM BLOCK IN A INTO REAL CORE ADDRESS IN B
GDAC    ZRO     0
        ARMI    =600000B
        STB     GDBCL
        LRSH    6
        STA     GDBCL+1
        LDA     GDBCL
        ETR     ADMSK
        RSH     14
        CXA
        ETR     =77777B
        LCY     14
        STA     GDBCL+2
        LSH     19
        LDA     GDBCL
        ETR     =140000B
        CBX
        RSH     14
        CXB
        LSH     5
        MRG     =IOSDE
        STA     GDAC3
GDAC1   RSR
        BRU     *-1
        ALR
        POT     GDBCL+1
        EOD*    10000B
GDAC3   BRU*    *
        POT     GDBCL+2
        SKN     GDBCL
        BRU     *+3
        WRF
        BRU     *+2
        RRF
        RSE
        BRR     GDAC
        RSR
        BRU     *-3
        CETE
        BRR     GDAC
        MIN     GDAC
GDAC2   BRR     GDAC
GDBCL   DATA    4000B,0,0
GDBC    ZRO
        LDX     GDBC
        STX     GDAC
        LDX     =4000B
        BRU     GDAC+1

* GET S1M INTO REAL BLOCK 6
GB6     ZRO     0
        DISW
```

```
        TSN
        LDA     MSMT
        LSH     5
        ETR     =3774000B
        LDB     =30000B
        BRM     GDBC
        HLT     0
        LDA =607B; STA RRL3; LRR3; POT RRL3
        BRR     GB6

*
*       QUEUE ROUTINES
*
QPUT    ZRO
        LDB     2,2
        XXA
        STB     PNEXT,2
        XXA
        LDB     1,2
        STA     1,2
        CBX
        STA     PNEXT,2
        BRR     QPUT
QGET    ZRO
        STA     PNEXT,2
        XXB
        SKG     =0
        BRR     QGET
        STB     PNXTP1,2
        BRR     QGET
QSCH    ZRO; XMA PTEST,2
        EOR     =700000B
        SKA =7700000B; BRU *+2; BRR QSCH
        LDA PNEXT,2; STA QSCH1; STX QSCH4; LDA =QTI
QSCH2   SKG =0; BRU QSCH3; SUB =PNEXT; CAB
QSCH3   CAX; LDA PNEXT,2; SKE QSCH4; BRU QSCH2; LDA QSCH1
        BRM     QGET
        LDX     QSCH4
        BRR     QSCH
QSCH1   ZRO
QSCH4   ZRO
*
*       INTERRUPT LOGIC
*
IIR     ZRO
        SKA     PIM,2
        BRU     *+2
        BRR     IIR
        STA     IIR1
        STX     IIR2
        DIR
        EOR     PIM,2
        EIR
        STA     PIM,2
```

```
        CLEAR
        LDA     IIR1
        NOD     24
        CXA
        SUB     =500000B-2
        CNA
        CAB
        LDX     IIR2
        LDA     PTEST,2
        SKE     =700004B
        BRU     IIR3
        LDA     PIM,2
        SKA     X1
        BRR     IIR                 (NON-TERMINABLE BRS)
        STB     IIR1
        LDA     PPTR,2
        RSH     12
        CAX
        BRM     DFK
        LDB     IIR1
IIR3    CBA
        LDX     IIR2
        BRM     QSCH
        CXA
        MRG     X7
        LDX     =QIO
        BRM     QPUT
        LDX     IIR2
        MIN     IIR
        BRR     IIR
IIR1    ZRO
IIR2    ZRO
* SEARCH FORK STRUCTURE AND INTERRUPT
SIR     ZRO
        STA     SIR1
        LDA     PPTR,2
        MRG     PLMSK
        CAX
        LDA     PPTR,2
        LRSH    12
SIR2    SKA     PRMSK
        BRU     SIR3
SIR4    LDA     PPTR,2
        SKA     PRMSK
        BRU     *+2
        BRR     SIR
        MRG     PLMSK
        CAX
        LDA     SIR1
        BRM     IIR
        BRU     SIR4
SIR5    MIN     SIR
        BRR     SIR
SIR3    MRG     PLMSK
```

```
        CAX
        LDA     SIR1
        BRM     IIR
        BRU     *+2
        BRU     SIR5
        LDA     PQU,2
        BRU     SIR2
SIR1    ZRO
        END
```