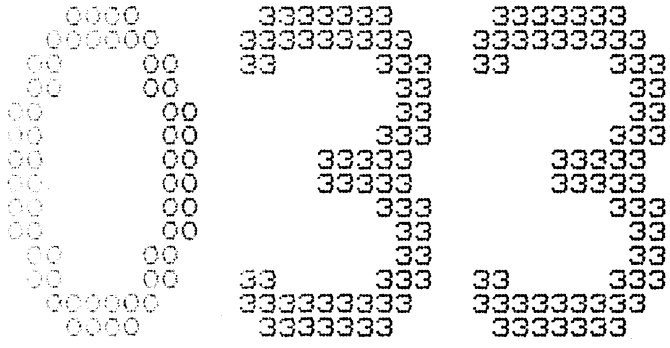
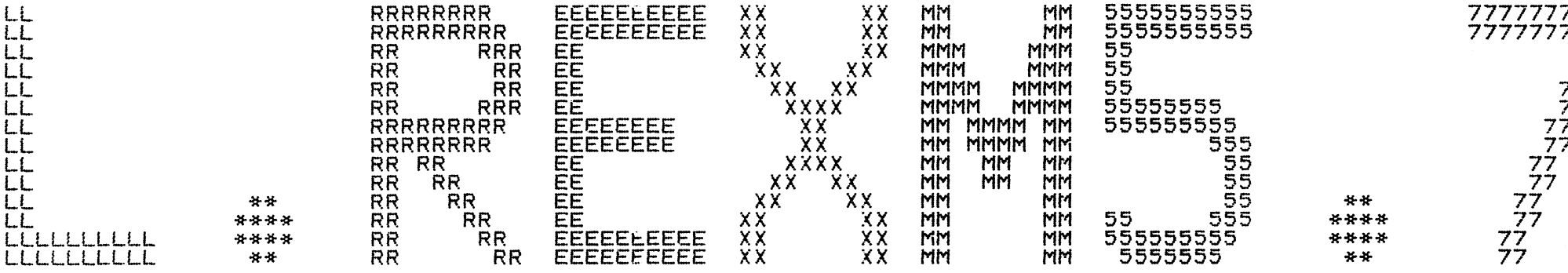
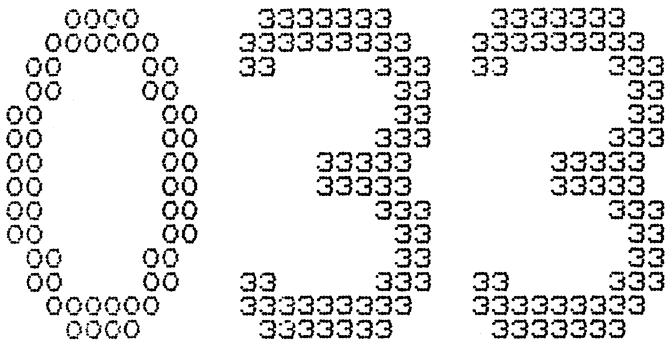
```
LL                    RRRRRRRR    EEEEEEEEEEE  XX       XX  MM       MM  5555555555              77777777
LL                    RRRRRRRRR   EEEEEEFEEEE  XX       XX  MM       MM  5555555555              77777777
LL                    RR      RRR EE           XX       XX  MMMM     MMMM  55                            7
LL                    RR        RR EE            XX     XX   MMMM     MMMM  55                            7
LL                    RR        RR EE              XX   XX    MMMMM   MMMMM  55                           77
LL                    RR      RRR EE                XXXXX     MMMMM   MMMMM  55555555                     77
LL                    RRRRRRRRR   EEEEEEEEE          XX       MM MMMMM MM  555555555                     77
LL                    RRRRRRRR    EEEEEEEEE          XX       MM MMMMM MM           555                   77
LL                    RR RR       EE               XXXXX      MM   MM  MM            55                  77
LL                    RR  RR      EE             XX   XX      MM   MM  MM            55                  77
LL                 **  RR   RR     EE            XX     XX     MM       MM            55        **       77
LL                ****  RR    RR   EE           XX       XX   MM       MM  55      555      ****      77
LLLLLLLLLL        ****  RR     RR  EEEEEEEEEEE  XX       XX   MM       MM  555555555     ****     77
LLLLLLLLLL         **   RR      RR EEEEEEEEEEE  XX       XX   MM       MM   5555555       **      77
```



```
File:  1/L.REXM5.7033          Printed:  MAR  2, 1987  11:48:36
File size:   904  blocks       Copy 1 of 1
```

Printed for RDC 9.1





File: 1/L.REXMS.7033 Printed: MAR 2, 1987 11:48:44
File size: 904 blocks Copy 1 of 1

Printed for RDC 9.1

```
.EOT ;"RxxJCL.REXM5"
.EOT  ;REX "RESD" FOR "IRIS" RESIDENT EXECUTIVE R9.0
.EOT  ;REX "SCHD" FOR "IRIS" RESIDENT EXECUTIVE
.EOT  ;REX "QUEM" FOR "IRIS" RESIDENT EXECUTIVE R9.0
.EOT  ;REX "MEMM" FOR "IRIS" RESIDENT EXECUTIVE
.EOT  ;REX "DPMP" FOR "IRIS" RESIDENT EXECUTIVE
.EOT  ;REX "CALL" FOR "IRIS" RESIDENT EXECUTIVE
.EOT  ;REX "GSUB" FOR "IRIS" RESIDENT EXECUTIVE
.EOT  ;REX "PCHR" FOR "IRIS" RESIDENT EXECUTIVE
.EOT  ;REX "MTTY" FOR "IRIS" RESIDENT EXECUTIVE
.EOT  ;REX "INTS" FOR "IRIS" RESIDENT EXECUTIVE
.EOT  ;REX "PFRS" FOR "IRIS" RESIDENT EXECUTIVE R9.0
.EOT  ;REX "FALT" FOR "IRIS" RESIDENT EXECUTIVE
.END
```

```
ASM 1/A.REXM5.7033!,@1/L.REXM5.7033!,B660,-B661,B662,B663,B664,B665,B666
MAR  2, 1987  11:05:40


;       Batchfile:   R91JCL.REXM5

;
;       ;D=7033  (YDDD)
;       ; NAME = REXM5
;       ; TYPE = 77000
;
;       -R91DEFS
;       * MK3=0
;       * MK5=1
;       * MARK3=0
;       * MARK5=1
;       * MARK8=0
;       R91REXRESDSA
;       R91REXSCHDSA
;       R90REXQUEMSA
;       R90REXMEMMSA
;       R90REXDPMPSB
;       R90REXCALLSC
;       R91REXGSUBSA
;       R90REXPCHRSC
;       R90REXMTTYSA
;       R90REXINTSSA
;       R91REXPFKSSA
;       R91REXFALTSA
;       *.END
;
                .EOT ;"RxxJCL.REXM5"
```

```
;                          << SI = R91REXRESDSA; BO = 1/A.REXM5.7033! >>
; 13 SEP 86, RB.

; "REX" == RESIDENT EXECUTIVE FOR "IRIS" R9.0
; 25 FEB 86, RB.: ADD DFTCALL
;  6 APR 86, RB.: ADD MAPCALL
; 17 APR 86, RB.: MODIFY ESCAPE INTERCEPT LINKAGE


               12        .RDX 10

               13 MONTH = 11
                4 DAY   = 4
             3702 YEAR  = 1986

;                    ALL RIGHTS RESERVED
;      COPYRIGHT (C) 1974, EDUCATIONAL DATA SYSTEMS
;      COPYRIGHT (C) 1979, EDUCATIONAL DATA SYSTEMS
;      COPYRIGHT (C) 1982, POINT 4 DATA CORPORATION
;      COPYRIGHT (C) 1986, POINT 4 DATA CORPORATION
;     THIS DOCUMENT CONTAINS SECRET AND CONFIDENTIAL
;    INFORMATION OF POINT 4 DATA CORPORATION.   IT MAY
;  NOT BE REPRODUCED,  USED, OR DISCLOSED WITHOUT THE
; PRIOR WRITTEN PERMISSION OF POINT 4 DATA CORPORATION.

; REX ASSEMBLY DATE (DAYS AFTER JAN 1 OF BASE YEAR)
            4761 RDATE = YEAR-BASEYEAR*12+MONTH-1*31+DAY-1


; EDITING NOTE: THE DEFINITIONS ON THIS PAGE (MONTH, YEAR, ETC.) ARE NOT
; INCLUDED IN THE PZ FILE, SO THAT THEY MAY BE USED IN OTHER SOURCES
; WHICH GET ASSEMBLED WITH DEFS & PZ.  DO NOT ADD ANYTHING ON THIS PAGE
; THAT IS REQUIRED IN PZ FILE.
```

```
;                       << SI = R91REXRESDSA;  BO = 1/A.REXM5.7033! >>

                  1     .TXTM  1
                  0     .LOC   0
                 10     .RDX   8


      0       151     JFLTO  ;IDLE LOCATION; also Trap 0 at 0
      1     12237     I.IGN  ;initial INTERRUPT VECTOR = Ignore interrupts


      2         2 C2:   2
      3         3 C3:   3
      4         0 PIB:  0       ;PARTITION INFO. BLOCK POINTER


      5         0 RUP:  0       ;REGNANT USER PCB POINTER
      6         0 IOP:  0       ;I/O CONTROL  PCB POINTER (INBYTE, OUTBYTE, STOUTPUT)
      7         0 RTP:  0       ;REGNANT TASK JCB POINTER

     10     30000 .BSA: BPS-2200 ;BLOCK SWAP AREA
     11     30400 .HBA: BPS-1600 ;HEADER BLOCK AREA
     12     31000 .HXA: BPS-1200 ;HEADER EXTENDER AREA
     13     31400 .SSA: BPS-0600 ;SUBROUTINE SWAP AREA

     14         0 .ABA: 0       ;AUXILIARY BUFFER AREA (SET BY 'SIR')

     15      2062 TASKQ:SNODE   ;TASK QUEUE HEAD POINTER


     16         0 BPI:  0       ;USED BY DSP AND DBUG FOR BREAKPOINTS
     17     73000        LDBUG  ;USED BY DSP AND DBUG FOR BREAKPOINTS
     20    177421        -357   ;USED BY ICMEC DISC, POWER FAIL RESTART
```

```
;                    << SI = R91REXRESDSA;  BO = 1/A.REXM5.7033! >>

       21 170000 C170K: 170000 ;TOP DIGIT MASK (4 BITS)
       22  77400 C774C: 77400  ;TOP BYTE MASK (7 BITS)
       23 177400 CM400: -400   ;TOP BYTE MASK (8 BITS)

       24      4 C4:    4
       25      5 C5:    5
       26      6 C6:    6
       27      7 C7:    7
       30     10 C10:   10
       31     11 C11:   11
       32     12 C12:   12
       33     13 C13:   13
       34     14 C14:   14
       35     15 C15:   15
       36     16 C16:   16
       37     17 C17:   17
       40      0 SBA:   0       ;SOURCE BYTE ADDRESS           *** PLACED AT 40 AND 41 SO THEY
       41      0 DBA:   0       ;DEST.  BYTE ADDRESS           *** CORR.  TO MK8 BYTE LOAD INSTR.
       42     20 C20:   20
       43     37 C37:   37      ;MASK FOR FILE TYPE
       44     40 C40:   40      ;SIZE OF EACH PCB
       45      1 .BLK   1       ;RESERVED FOR STK.  BASE PTR.  ON MK8 CPU

       46      0        0       ;THESE TWO WORDS MUST BE
       47  12025        TRAP3   ;AT 46 AND 47 FOR NOVA 3

       50     77 C77:   77
       51    100 C100:  100
       52    177 C177:  177
       53    200 C200:  200     ;NULL
       54    205 C205:  205     ;  ^E
       55    215 C215:  215     ;  CR
       56    240 C240:  240     ;  SP
       57    244 C244:  244     ;  $
       60    260 C260:  260     ;  0
       61    271 C271:  271     ;  9
       62    300 C300:  300     ;  @
       63    334 C334:  334     ;  \
       64    377 C377:  377     ;RUBOUT,  BYTE MASK
       65    400 C400:  400
       66    777 C777:  777
       67   1000 C1000: 1000
       70   1777 C1777: 1777
       71   2000 C2000: 2000
       72   4000 C4000: 4000

       73      0 ESCF:  0       ;ESCAPE FLAG
       74      0 ETSF:  0       ;END OF TIME SLICE FLAG
       75      0 BSACF: 0       ;"BSA CHANGED" FLAG
       76      0 ERRF:  0       ;ERROR FLAG,  USED BY DECIMAL

       77  32200 .BPS:  BPS     ;BEGINNING OF PROCESSOR STORAGE
      100    600 .INFO: INFO    ;SYSTEM INFORMATION TABLE
```

```
;                    << SI = R91REXRESDSA; BO = 1/A.REXM5.7033! >>

                ; SYSTEM COMMAND CALLS

      6101 CALL=     JSR @.  ;CALL A SYSTEM SUBROUTINE
101   5542            CALLS

      6102 FLAGC=    JSR @.  ;CHANGE OR CHECK A FLAG
102   7664            FLAGX

     34103 IOCALL=   LDA 3,. ;CALL STANDARD CHARACTER I/O ROUTINE
      6103 QCHAR=    JSR @.  ;QUEUE A CHARACTER TO BE PROCESSED
103  10052            QCHAX

      6104 QUEUE=    JSR @.  ;PUT A TASK IN THE QUEUE
104   2306            QUE

      6105 DQUEUE=   JSR @.  ;REMOVE REGNANT TASK FROM THE QUEUE
105   2562            DQUE

     34106 DFTCALL=  LDA 3,. ;CALL PREFIX FOR DFT ACCESS ROUTINES
      6106 CHANNEL=  JSR @.  ;PERFORM A CHANNEL OPERATION
106   6061            CHANX

      6107 FREENODE=JSR @.   ;GET OR RELEASE A FREE 32-WORD NODE
107   2665            FREEX

     34110 DMCALL=   LDA 3,. ;CALL PREFIX FOR DMA MAP MANAGER
      6110 DATAPUMP=JSR @.   ;INITIATE A DMA DATA TRANSFER
110   3351            DPUMX



111  12114 .INTR: INTSR
112   6011 .NRET: CALLN
113   5705 .SRET: CALLR
114  12470 .LCM:  TFALT   ;$LCM INIT LINKS ITSELF THROUGH HERE
```

```
;                    << SI = R91REXRESDSA;  BO = 1/A.REXM5.7033! >>
                   ; SYSTEM SUBROUTINE CALLS

        6115 BINDIVIDE  =JSR @.
 115    7232           BDIV
        6116 BINMULTIPLY=JSR @.
 116    7212           BMUL
        6117 BUMPUSER   =JSR @.
 117    1370           BUMP
        6120 DECIMAL    =JSR @.
 120    7261           DECX    ;CHANGED BY $DEC OR $DAU
        6121 FIX        =JSR @.
 121    7261           DECX    ;CHANGED BY $DEC OR $DAU
        6122 FLOAT      =JSR @.
 122    7261           DECX    ;CHANGED BY $DEC OR $DAU
        6123 FINDLUT    =JSR @.
 123    6517           FLUT
        6124 GETBYTE    =JSR @.
 124    6673           ACBY
        6125 INBYTE     =JSR @.
 125    6655           ACIB
        6126 INSTBYTE   =JSR @.
 126    6663           ACSB
        6127 ISA2DIGIT  =JSR @.
 127    7655           IA2D
        6130 ISA2LETTER =JSR @.
 130    7640           IA2L
        6131 LOADDA     =JSR @.
 131    7261           DECX    ;CHANGED BY $DEC OR $DAU
        6132 OUTBYTE    =JSR @.
 132    6714           STOB
        6133 OUTTEXT    =JSR @.
 133    6625           OTXT
        6134 PUTBYTE    =JSR @.
 134    6745           STBY
        6135 READBLOCK  =JSR @.
 135    3366           RBLK
        6136 RELJMPRET  =JSR @.
        6136 RJSR       =JSR @.
 136    7770           XRJSR
        6137 STORDA     =JSR @.
 137    7261           DECX    ;CHANGED BY $DEC OR $DAU
        6140 STINPUT    =JSR @.
 140    7273           STI
        6141 STOUTPUT   =JSR @.
 141    7360           STO
        6142 TRAPFAULT  =JSR @.
 142   12470           TFALT
        6143 WRITBLOCK  =JSR @.
 143    3423           WBLK
        6144 XGETBYTE   =JSR @.
 144    6645           XACBY
        6145 XPUTBYTE   =JSR @.
 145    6641           XSTBY
        6146 SPINPUT    =JSR @.
 146    7275           SSI
        6147 STINTERACTION=JSR @.
 147    1561           SINT    ;START AN INTERACTION
```

```
;                       << SI = R91REXRESDSA;  BO = 1/A.REXM5.7033! >>

                    ;POINTERS USED ONLY WITHIN "REX"

      150    1361 .BRKP: BRKP
      151    2152 JFLTO: JMP     @.FLTO     ;allows either JMP 0 or
      152   12464 .FLTO: FALTO               ;     JMP @0 to be trapped

      153       5         .BLK    160-.      ;OVERLAP CHECK AND PATCH SPACE

                    ;DECIMAL FLOATING-POINT REGISTERS

      160       0 DA:     0          ;DECIMAL ACCUMULATOR
      161       0         0
      162       0         0
      163       0         0
      164       0 DAC:    0          ;CHARACTERISTIC
      165       0 DAS:    0          ;SIGN

      166       0 DB:     0          ;DECIMAL BUFFER REGISTER
      167       0         0
      170       0         0
      171       0         0
      172       0 DBC:    0          ;CHARACTERISTIC
      173       0 DBS:    0          ;SIGN

      174     160 .DA:    DA
      175     163 .DA3:   DA+3
      176     166 .DB:    DB
      177     171 .DB3:   DB+3


              174 C160 =.DA       ;THESE POINTERS USED AS CONSTANTS
              175 C163 =.DA3
              176 C166 =.DB
              177 C171 =.DB3

              100 C600 =.INFO

      200       0 .BLK   INFO-400-.        ;OVERLAP CHECK
```

```
;                    << SI = R91REXRESDSA; BO = 1/A.REXM5.7033! >>

          600          .LOC    INFO       ;SYSTEM INFORMATION TABLE

      600    7033 7033            ;SDAT.     ;REX ASSEMBLY DATE (DAYS AFTER JAN 1 OF BASE YEAR)
      601    2000 2000            ;SPED.     ;AVERAGE CPU SPEED (INSTR/MSEC)
      602       1 1               ;MILU.     ;MAXIMUM # OF INSTALLED LOGICAL UNITS
      603       3 3               ;NDCH.     ;NUMBER OF DATA CHANNELS PER PORT
      604   65740 65740           ;LPCA.     ;LOCATION OF PORT CONTROL AREA
      605       1 1               ;TNAP.     ;TOTAL NUMBER OF ACTIVE PORTS
      606  100000 100000          ;SPCF.     ;SPECIAL CONDITIONS (BIT 15=1 ==> NO DIRTY BUFFERS)
      607   43600 MEPS            ;LEPS.     ;LOCATION OF END OF PROCESSOR STORAGE
      610   77777 77777           ;TOPW.     ;TOP WORD OF CORE TO BE USED
      611       0 0               ;ABUF.     ;AUXILIARY BUFFER SIZE (NUMBER OF WORDS)
      612       0 0               ;UDSB.     ;# OF USER DISCSUBS
      613       4 4               ;NCQN.     ;NUMBER OF CHARACTER QUEUE NODES
      614      10 10              ;NNOD.     ;NUMBER OF FREE NODES
      615      10 10              ;NSIG.     ;NUMBER OF SIGNAL BUFFER NODES
      616     230 DFNSUB          ;SDSB.     ;MAXIMUM SUBROUTINE NUMBER IN DISCSUBS
      617   24003 24003           ;KTSL.     ;TIME SLICE PARAMS, LONG TIME SLICE * 400 + SHORT TIME SLICE
      620 1777// 177777          ;DFLU.     ;DEFAULT LOGICAL UNIT #(FOR CHAIN AND SCOPE  FIND FILE SEARCHES
      621   77377 77377           ;RESERVED
      622      13 SZLT            ;SZLINK    ;SIZE OF PSEUDO-DEVICE LINKAGE TABLE
      623       0 0               ;HCON.     ;HARDWARE CONDITION WORD
      624       0 0               ;HCN1.     ;UPPER BYTE RESERVED FOR # MUX PORTS. LOWER BYTE = CPU REVISION

      625       5 .BLK  SZICON+INFO-.        ;(RESERVED)

      632    3674 BASEY           ;BASY.     ;BASE YEAR FOR SYSTEM TIME
      633     160 DA              ;.TSA.     ;TEMPORARY STORAGE "A" POINTER (6 WORDS)
      634     160 DA              ;.TSB.     ;TEMPORARY STORAGE "B" POINTER (6 WORDS)
      635     160 DA              ;.TSQ.     ;TEMPORARY STORAGE "Q" POINTER (6 WORDS)
      636     160 DA              ;.TSZ.     ;TEMPORARY STORAGE "Z" POINTER (6 WORDS)
      637     160 DA              ;.TSC.     ;TEMPORARY STORAGE "C" POINTER (16 WORDS)
      640       0 0               ;HRS.      ;CPU TIME -- HOURS SINCE JAN 1 OF BASE YEAR
      641       0 0               ;TSC.      ;  PART OF HOUR IN TENTH-SECONDS
      642 1777// -1               ;CPLU.     ;CURRENT PROCESSOR LOGICAL UNIT
      643 177777 -1               ;CPDA.     ;CURRENT PROCESSOR DISC ADDRESS
      644 17777/ -1               ;CPTN.     ;CURRENT PROCESSOR TYPE NUMBER
      645      40 -CHM1*4         ;SDFT.     ;SIZE OF EACH PORT'S DATA FILE TABLE, MIN=2 NEG + 2 POS
      646 17777/ -1               ;DSCO.     ;DISC ADDRESS OF "SCOPE"     \
      647 1777// -1               ;DBYE.     ;DISC ADDRESS OF "BYE"        ) SET
      650 1777// -1               ;DDSP.     ;DISC ADDRESS OF "DSP"        ) UP
      651 1777/7 -1               ;DSUB.     ;DISC ADDRESS OF "DISCSUBS"   ) BY
      652 17777/ -1               ;DMSG.     ;DISC ADDRESS OF "MESSAGES"   ) SIR
      653 177777 -1               ;DCON.     ;DISC ADDRESS OF "CONFIG"    /
      654   27/2 LUSRX            ;.LSR.     ;A(LOADUSER) MAY BE CHANGED BY $LUSR OR $LUSR.LCM
      655      1 1                ;MAXB.     ;MAX # BLOCKS IN PARTITION (SET BY SIR)
```

```
;                        << SI = R91REXRESDSA; BO = 1/A.REXM5.7033! >>

656    2062 SNODE          ;.STN.    ;POINTER TO SCHEDULER TASK NODE
657    5763 .STK           ;..STK    ;POINTER TO "CALL" STACK POINTER
660       1 1              ;FNOD.    ;# AVAILABLE FREE NODES + 1
661     575 INFO-3         ;.LUT.    ;POINTER TO LOGICAL UNIT TABLE (SET BY SIR)
662   40000 M3*MARK3!MC    ;SCON.    ;SYSTEM CONDITION (STATE) WORDS
663       0 0              ;SCN1.    ;
664       0 0              ;SCN2.    ;
665       0 0              ;IRUPT    ;INTERRUPT FLAG:  -1 = NORMAL, >=0 = PROCESSING INTERRUPT
666  177777 -1             ;.BPT.    ;POINTER TO BUFFER POOL TABLE (SET BY SIR)
667    6427 CRST           ;.CTT.    ;POINTER TO "CALL" TRANSLATE TABLE
670    1623 .SIGB          ;.SGB.    ;POINTER TO SIGNAL BUFFER POINTERS
671  177777 -1             ;.TTT.    ;POINTER TO TERMINAL TYPE TABLE (SET BY $TERMS)
672  177777 -1             ;.UPT.    ;POINTER TO USER PARTITION (SET BY SIR)
673       0 0              ;THTC.    ;TEN HERTZ TASK COUNTER
674  177774 177774         ;MASK.    ;INITIAL INTERRUPT MASK (ENABLE DEVICE 10/11)
675       0 0              ;PFRF.    ;POWER FAIL FLAG
676  177777 -1             ;LINKT    ;POINTER TO PSEUDO-DEVICE TABLE
677  177777 -1             ;LCALT    ;LOCATION OF CALL TABLE IN MEMORY  (-1 ==> NONE)
700   12470 TFALT          ;.MAP.    ;$SYSMAP ENTRY POINT
701   12164 .INTERCEPT     ;.INT.    ;INTERRRUPT RETURN INTERCEPT LOCATION
702       0 0              ;APCB.    ;ACTIVE PCB WHILE TIMESHARING IS SUSPENDED
703  177777 177777         ;LSMNX    ;POINTER TO $SMTBL TABLES
704   11401 .ESCLINK       ;ESCL.    ;ESCAPE PROCESSING INTERCEPT LOCATION
705      -1 EXM            ;NIOP.    ;DEFAULT NIOP IS 64K MODE


706      20 .BLK   INFO+INVT.-2-.     ;RESERVED FOR INFO TABLE EXPANSION

726   12707 PATSP          ;BPSP.    ;BEGIN PATCH SPACE (AFTER LAST PATCH TO REX)
727   32200 BPS            ;ENDP.    ;END OF PATCH SPACE (SET BY SIR)

730       0 .BLK   INFO+INVT.-.      ;OVERLAP CHECK
```

```
;                       << SI = R91REXRESDSA; BO = 1/A.REXM5.7033! >>

                    ; INTERRUPT VECTOR TABLE

      730   12360 INTZ   ; 0 ==> INTERRUPT NOT ACKNOWLEDGED
      731   12224 PZIH-INTX*MK3+INIX

                    ; END OF INTERRUPT VECTOR TABLE FOR MARK 3

      732   12224 INTX
      733   12224 INTX
      734   12224 INTX
      735   12224 INTX
      736   12224 INTX
      737   12224 INTX
      740   11743 TTIS   ;10
      741   11753 TTOS
      742   12224 INTX
      743   12224 INTX
      744   12224 INTX
      745   12224 INTX
      746   12224 INTX
      747   12224 INTX
      750   12224 INTX   ;20
      751   12224 INTX
      752   12224 INTX
      753   12224 INTX
      754   12224 INTX
      755   12224 INTX   ;25 MIGHTY MUX
      756   12224 INTX
      757   12224 INTX
      760   12224 INTX   ;30
      761   12224 INTX
      762   12224 INTX
      763   12224 INTX
      764   12224 INTX
      765   12224 INTX
      766   12224 INTX
      767   12224 INTX
      770   12224 INTX   ;40
      771   12224 INTX
      772   12224 INTX
      773   12224 INTX
      774   12224 INTX
      775   12224 INTX
      776   12224 INTX
      777   12224 INTX
```

```
;                          << SI = R91REXRESDSA; BO = 1/A.REXM5.7033! >>
     1000   12224 INTX   ;50
     1001   12224 INTX
     1002   12224 INTX
     1003   12224 INTX
     1004   12224 INTX
     1005   12224 INTX
     1006   12224 INTX
     1007   12224 INTX
     1010   12224 INTX   ;60
     1011   12224 INTX   ;61 LCM
     1012   12224 INTX
     1013   12224 INTX
     1014   12224 INTX
     1015   12224 INTX
     1016   12224 INTX
     1017   12224 INTX
     1020   12224 INTX   ;70
     1021   12224 INTX
     1022   12224 INTX
     1023   12224 INTX
     1024   12224 INTX
     1025   12224 INTX
     1026   12224 INTX
     1027   12360 INTZ   ;77 = CPU !?


; COMMONLY USED DEVICE CODES:

                          ;10 = MASTER TELETYPE INPUT
                          ;11 = MASTER TELETYPE OUTPUT
                          ;12 = PAPER TAPE READER
                          ;13 = PAPER TAPE PUNCH
                          ;14 = REAL TIME CLOCK
                          ;16 = CARD READER
                          ;17 = PARALLEL LINE PRINTER
                          ;20 = DGC 4019, AMPEX MEGASTORE
                          ;24 = BIT-BANGER MULTIPLEXER
                          ;25 = POINT 4 MIGHTY-MUX
                          ;30 = DCC 116446 DISC
                          ;33 = DGC 4046/4234/6051, TELEFILE DISC
                          ;36 = MCT802/902 50MB DISC
                          ;40 = SI 3015/3045 DISC
                          ;42 = DCC ALU MUX INPUT
                          ;43 = DCC ALU MUX OUTPUT
                          ;50 = TELETYPE #2 INPUT
                          ;51 = TELETYPE #2 OUTPUT
                          ;61 = LCM


          1030 .LOC   INFO+INVT.+2-.*MK3+.; RESTORE LOCATION COUNTER IN CASE OF MARK 3
               .EOT   ;REX "RESD" FOR "IRIS" RESIDENT EXECUTIVE R9.0
```

```
;                          << SI = R91REXSCHDSA; BO = 1/A.REXM5.7033! >>
; 2 NOV 86, RB.

;;- --------- ------- - ------------------------------------------------
;;
;;              R E X :   S C H E D U L I N G   F U N C T I O N S
;;
;;     MAJOR COMPONENTS:
;;
;;               1.   START AN INTERACTION
;;               2.   EXIT TO SCOPE
;;               3.   SWAP OUT
;;               4.   DSP BREAKPOINT INTERCEPT
;;               5.   BUMP USER
;;               6.   SCHEDULER
;;                    A.  SELECT JOB
;;                    B.  START AN INTERACTION
;;               7.   TEN HERTZ TASK
;;
;;     UPDATE RECORD:
;;
;;              5-30-80    LLL    REARRANGED FOR READABILITY
;;            11 NOV 81    RMS    RESET ETSF AFTER TRAP #34
;;             2 FEB 82    RB.    new DATAPUMP and QLINK
;;             7 MAR 86    RB.    ADD PROVISIONS FOR DFT'S IN EXTENDED MEMORY
;;             6 APR 86    RB.    IOB'S IN XMEM
;;             5 MAY 86    TWM    ADD DETACHED PORT FEATURES
```

```
;                        << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

;;-------- ---------- ---------- ---------- ----------------------------
;;
;;     1.   START TIME SLICE
;;
;;          ENTRY:   A2 <-- ADDRESS OF PCB TO BECOME REGNANT
;;                   URA.(PCB) <-- EXECUTION ADDRESS
;;                   CPDA.(INFO) <-- CURRENT PROCESSOR DISC ADDRESS
;;                   CPLU.(INFO) <-- CURRENT PROCESSOR LU NUMBER
;;
;;                   IF (DSP BREAKPOINT TO BE SET UP)
;;                       FLW.(PCB) BIT 14 <-- 1    <<20000>>
;;                       DSPS(AFH) <-- -1
;;                       DSPS+1 <-- ADDRESS WHERE BREAKPOINT TO BE SET
;;
;;          EXIT:    RUP <-- ADDRESS OF REGNANT PCB
;;                   RTP <-- ADDRESS OF REGNANT PCB
;;                   ETSF <-- 0
;;
;;                   CONTROL IS PASSED TO THE PROCESSOR VIA URA.(PCB)
;;                       IF URA.(PCB) < 100000, JUMPS TO THAT ADDRESS
;;                       ELSE, JUMPS INDIRECT THROUGH THAT ADDRESS
;;
;;          PURPOSE:  SETS UP PROCESSOR AREA FOR A USER TO EXECUTE,
;;                    AS WELL AS SETTING DSP BREAKPOINTS.
;;---------- ---------- ---------- ---------- ----------------------------

1030    647 .DBYE: INFO+DBYE.
1031  32204 .BPS4: BPS+4
1032      0 THEND: 0
1033      0 A2TEM: 0

1034  34015 STIME: LDA     3,TASKQ  ;PREPARE TO ACTIVATE A PROCESSOR
1035  51410        STA     2,TCBP,3
1036  50005        STA     2,RUP    ;MAKE THE JOB REGNANT
1037  50006        STA     2,IOP
1040  50007        STA     2,RTP
1041  34106        DFTCALL
1042   77/4        DFTREAD          ;READ DFT INTO CORE FROM XMEM,  IF ANY
1043  30005        LDA     2,RUP
1044  35025        LDA     3,DFT.,2
1045  21012        LDA     0,FLW.,2
1046  24044        LDA     1,C40
1047 107415        AND#    0,1,SNR  ;LOG-OFF BIT SET ?
1050    407        JMP     STIMM    ;   NO
1051  26757        LDA     1,@.DBYE ;YES,  MAKE HIS PROCESSOR = BYE
1052  457/1        STA     1,FDA+CHM1,3
1053 126400        SUB     1,1
1054  457/0        STA     1,FLU+CHM1,3
1055  24754        LDA     1,.BPS4
1056  45014        STA     1,URA.,2
1057  21770 STIMM: LDA     0,FLU+CHM1,3
1060  257/1        LDA     1,FDA+CHM1,3
1061  34100        LDA     3,.INFO
1062  31443        LDA     2,CPDA.,3
```

```
;                    << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

    1063 146414          SEQ     2,1        ;IS JOB'S PROCESSOR IN CORE ?
    1064    404          JMP     STIMO      ;   NO
    1065  31442          LDA     2,CPLU.,3  ;   MAYBE
    1066 142415          SNE     2,0
    1067    475          JMP     STIM1      ;   YES
    1070  41442 STIMO:   STA     0,CPLU.,3  ;   NO
    1071  45443          STA     1,CPDA.,3
    1072   6110          DATAPUMP           ;READ PROCESSOR HEADER
    1073      0          GETBLOCK
    1074 100000          @0
    1075  34100          LDA     3,.INFO    ;CHECK FOR CONSISTENT HEADER
    1076  21176          LDA     0,UNIT,2
    1077  25442          LDA     1,CPLU.,3
    1100 106414          SEQ     0,1
    1101   6142 HDRXX:   TRAPFAULT          ;LU'S DON'T MATCH!!!?
    1102 104410          11*K!NOP
    1103  21177          LDA     0,DHDR,2
    1104  25443          LDA     1,CPDA.,3
    1105 106414          SEQ     0,1
    1106   4773          JSR     HDRXX      ;RDA IN HEADER DOESN'T MATCH !!!?
    1107  20061          LDA     0,C377
    1110 143000          ADD     2,0        ;A (END RDA LIST)
    1111  40721          STA     0,THEND
    1112  34053          LDA     3,C200
    1113 157000          ADD     2,3        ;STARTING ABSOLUTE ADDRESS OF RDA LIST
    1114  25175          LDA     1,CORA,2;ADDRESS TO CHECK
    1115  50716          STA     2,A2TEM    ;SAVE AC2
    1116  21400 INPRC:   LDA     0,0,3      ;GET WORD FROM RDA LIST
    1117 101015          SNZ     0,0        ;AN RDA?
    1120    417          JMP     INPR1      ;   NO
    1121  20053          LDA     0,C200     ;YES, SEE IF IT SITS INTO PROG STORAGE
    1122 122415          SNE     1,0        ;PZ BLOCK?
    1123    414          JMP     INPR1      ;   YES, CHECK NEXT BLOCK
    1124  20077          LDA     0,.BPS     ;NO, CONTINUE CHECKING THIS BLOCK
    1125 122032          SGE     1,0        ;BELOW BPS?
    1126   6142          TRAPFAULT          ;   YES
    1127 135010          72*K!NOP           ;NO, NOT IN PROCESSOR STORAGE AREA
    1130  30100          LDA     2,.INFO
    1131  21007          LDA     0,LEPS.,2
    1132  30065          LDA     2,C400
    1133 142400          SUB     2,0        ;A(LAST BLOCK OF PROCESSOR STORAGE)
    1134 122433          SLE     1,0        ;ABOVE PROCESSOR STORAGE?
    1135   6142          TRAPFAULT          ;   YES
    1136 135010          72*K!NOP           ;NO
    1137 175400 INPR1:   INC     3,3
    1140  20065          LDA     0,C400
    1141 107000          ADD     0,1        ;ADDRESS TO CHECK
    1142  20670          LDA     0,THEND
    1143 162432          SGR     3,0        ;CHECKED ALL ADDRESSES?
    1144    752          JMP     INPRC      ;   NO, CONTINUE CHECKING
    1145  30666          LDA     2,A2TEM    ;YES
    1146  34100          LDA     3,.INFO
    1147  21010          LDA     0,TYPE,2;SAVE PROCESSOR TYPE NUMBER
    1150  24043          LDA     1,C37
    1151 107400          AND     0,1
    1152  45444          STA     1,CPTN.,3
```

```
;                       << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

 1153   6110          DATAPUMP            ;NOW READ IN THE PROCESSOR . . .
 1154 100022          DFILREAD+UL         ;AND RELEASE THE HEADER BUFFER
 1155 100000          @0
 1156  32543          LDA     2,@ASCON    ;SYSTEM CONDITION WORD
 1157  20543          LDA     0,VCT
 1160 113415          AND#    0,2,SNR     ;MONITORING ENABLED ?
 1161    403          JMP     STIM1       ;   NO
 1162   6016          JSR     @CTRVECTOR
 1163      1          CTIRDA
 1164  34005 STIM1:   LDA     3,RUP
 1165 102400          SUB     0,0
 1166  25412          LDA     1,FLW.,3
 1167  30526          LDA     2,BRMSK
 1170 133415          AND#    1,2,SNR     ;SET BREAKPOINT ?
 1171    441          JMP     STIM3       ;   NO
 1172  25427          LDA     1,AHA.,3    ;   YES, READ ACTIVE FILE HEADER
 1173   6110          DATAPUMP
 1174      0          GETBLOCK
 1175 100000          @0
 1176  34005          LDA     3,RUP       ;SAVE ACTIVE HEADER LOCATION
 1177  35425          LDA     3,DFT.,3
 1200  51762          STA     2,CBN+CHM2,3;IN DSP CHANNEL
 1201  21050          LDA     0,DSPS+0,2
 1202  25051          LDA     1,DSPS+1,2
 1203 101415          INC#    0,0,SNR     ;FILE SELECTED ?
 1204 125015          SNZ     1,1
 1205    421          JMP     STIM2       ;   NO, CONTINUE
 1206  21760          LDA     0,FLU+CHM2,3;   YES, SEE IF BREAK IN CURRENT PORCESSOR
 1207  25761          LDA     1,FDA+CHM2,3
 1210 101014          SKZ     0,0         ;SELECTED FILE ON LU #0 ?
 1211    415          JMP     STIM2       ;   NO, CONTINUE
 1212  30100          LDA     2,.INFO
 1213  21043          LDA     0,CPDA.,2;CURRENT PROCESSOR HDR RDA
 1214 106414          SEQ     0,1         ;BREAKPOINT CURRENT PROCESSOR ?
 1215    411          JMP     STIM2       ;   NO, CONTINUE
 1216  21760          LDA     0,FLU+CHM2,3;   YES, FORCE PROC CHANGE
 1217 100000          COM     0,0
 1220  41042          STA     0,CPLU.,2
 1221  31762          LDA     2,CBN+CHM2,3
 1222  23052          LDA     0,@DSPS+2,2;GET INSTRUCTION AT BP LOCATION
 1223  40016          STA     0,BPI       ;...AND SAVE IT
 1224  20472          LDA     0,JBRKP     ;SET THE BREAKPOINT
 1225  43052          STA     0,@DSPS+2,2
 1226  31762 STIM2:   LDA     2,CBN+CHM2,3;<< FROM ABOVE
 1227   6110          DATAPUMP            ; RELEASE AFH
 1230      3          UNLATCH
 1231 100000          @0
 1232  30100 STIM3:   LDA     2,.INFO
 1233  31056          LDA     2,.STN.,2
 1234 102000          ADC     0,0
 1235  41032          STA     0,SN.LU,2;CLR SAVED RECORD ACCESS MEMORY
```

```
;                       << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

     1236  34077          LDA     3,.BPS    ;LET PROCESSOR DO ITS SETUP
     1237  30005          LDA     2,RUP
     1240 102400          SUB     0,0
     1241  40040          STA     0,SBA
     1242  40041          STA     0,DBA
     1243  40076          STA     0,ERRF    ;0 --> DECIMAL'S ERROR FLAG
     1244 126400          SUB     1,1
     1245   7400          JSR     @0,3      ;PROCESSOR'S "SWAPI"
     1246 102400          SUB     0,0
     1247  40074          STA     0,ETSF    ;END-OF-TIME-SLICE <-- FALSE
     1250  34100          LDA     3,.INFO
     1251  21417          LDA     0,KTSL.,3;LONG/SHORT TS PARAM'S
     1252  24446          LDA     1,SIODS   ;I/O DONE STATE MASK
     1253  60277          INTDS
     1254  31462          LDA     2,SCON.,3;SYSTEM CONDITION WORD
     1255 147415          AND#    2,1,SNR   ;IS SYSTEM IN I/O DONE STATE ?
     1256 101300          MOVS    0,0       ;   NO, USE LONG TIME SLICE
     1257  24064          LDA     1,C377
     1260 123400          AND     1,0
     1261  42436          STA     0,@SCTSL  ;SET TIME SLICE
     1262  30005          LDA     2,RUP
     1263  21015          LDA     0,ABN.,2
     1264  34474          LDA     3,ABNMSK  ;CLEAR ESC AND CNTL-C
     1265 117400          AND     0,3       ;   BITS IN ABN.
     1266 162400          SUB     3,0
     1267  55015          STA     3,ABN.,2
     1270  60177          INTEN
     1271  40073          STA     0,ESCF
     1272 176400          SUB     3,3       ;ASSUME NORMAL ENTRY, NOT ESC OR CTRL-C
     1273 101202          MOVR    0,0,SZC   ;ESCAPE ?
     1274  34416          LDA     3,.BPS2   ;   YES
     1275 101202          MOVR    0,0,SZC   ;CONTROL C ?
     1276  34415          LDA     3,.BPS3   ;   YES
     1277  21014          LDA     0,URA.,2
     1300  24414          LDA     1,ABPS4
     1301 175014          SKZ     3,3       ;NO ESCAPE
     1302 106415          SNE     0,1       ;   OR INITIAL ENTRY ?
     1303 115000          MOV     0,3       ;   YES, USE URA
     1304 175113          SSN     3,3
     1305   1400          JMP     0,3
     1306 102400          SUB     0,0       ;GOING IN ESCAPE ENTRY
     1307  40073          STA     0,ESCF    ;SO CLEAR FLAG
     1310 177240          ADDOR   3,3       ;PROCESSOR MUST BE IN LOWER CORE ***
     1311   3400          JMP     @0,3

     1312 132202 .BPS2:  @BPS+2
     1313 132203 .BPS3:  @BPS+3
     1314  32204 ABPS4:  BPS+4
     1315  20000 BRMSK:  20000
     1316   2150 JBRKP:  JMP     @.BRKP
     1317   2101 SCTSL:  TCTSL              ;TIME SLICE COUNTER
     1320   4000 SIODS:  ID                 ;I/O DONE STATE MASK
     1321    662 ASCON:  INFO+SCON.
     1322    100 VCT:    CT
```

```
;                       << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

;;---------- - ---- --------------- ----- ---------------------------
;;
;;     2.   EXIT TO "SCOPE"
;;
;;          ENTRY:   INVOKED BY A PROCESSOR ISSUING THE FOLLOWING:
;;
;;                        CALL
;;                        SCOPE
;;
;;          EXIT:    GOES TO START INPUT
;;
;;          PURPOSE:   A PROCESSOR INVOKES THIS ROUTINE WHEN IT HAS
;;                   COMPLETELY FINISHED ITS WORK, AND WILL NOT BE
;;                   SWAPPED IN AGAIN.
;;                    THIS ROUTINE SWAPS OUT THE PROCESSOR FOR THE LAST
;;                   TIME, AND CLEARS ANY CHANNELS WHICH MAY BE LEFT OPEN.
;;
;;---------- ------------------------ -- ---------------------------
     1323  32204          BPS+4
     1324   6101  EXITX:  CALL                ;CLEAR ALL DATA CHANNELS
     1325 100003          ALLCLEAR
     1326 102000          ADC     0,0
     1327   6106          CHANNEL             ;CLEAR PROCESSOR CHANNEL
     1330     27          CLEAR
     1331   6142          TRAPFAULT           ;CHANNEL #-1 IS ILLEGAL !?
     1332 114010          30*K!NOP
     1333  30005          LDA     2,RUP
     1334  34100          LDA     3,.INFO     ;OPEN "SCOPE" AS PROCESSOR
     1335  21446          LDA     0,DSCO.,3
     1336  35025          LDA     3,DFT.,2
     1337  41771          STA     0,FDA+CHM1,3
     1340 126400          SUB     1,1
     1341  44073          STA     1,ESCF      ;MAKE SURE AN "ESCAPE" IS NOT PENDING
     1342  45770          STA     1,FLU+CHM1,3
     1343  45773          STA     1,STS+CHM1,3;CLEAR "LOCKABLE PGM" FLAG
     1344   6102          FLAGCHANGE          ;ENABLE CTRL O- ALLOWS THE USER
     1345 100012          RESET+FLW.          ;...  TO SUPPRESS PRINTING LOGON DATA
     1346     11          11
     1347  11000          ISZ     0,2         ;ENABLE ECHO
     1350   6101          CALL
     1351 100011          WONA                ;EMPTY OUTPUT BUFFER
     1352   6133          OUTTEXT             ;...THEN
     1353 106643          .TXTF   ;<215>#
     1354      0 ;

     1355   6141          STOUTPUT            ;FORCE A NEW LINE & ISSUE PROMPT
;
;     SET UP NEXT SCHEDULE POINT AT PROCESSOR'S INITIAL ENTRY
;
     1356  34745          LDA     3,EXITX-1
     1357   2140          JMP     @STINP&377;BUMP USER AND AWAIT INPUT DONE
```

```
;                        << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

    1360 1777/4 ABNMSK: 177774          ;MASK TO CLEAR ESC AND CNTL-C BITS IN ABN.

;;-----------------------------------------------------------------------
;;
;;     4.   DSP BREAKPOINT INTERCEPT
;;
;;          ENTRY:    A3 <-- RETURN ADDRESS (1 MORE THAN BROKEN
;;                       INSTRUCTION)
;;
;;          EXIT:     AFTER CALLING BREAK IT FALLS INTO BUMP
;;
;;          PURPOSE:    WHEN A DSP BREAKPOINT IS ENCOUNTERED, IT
;;                    PERFORMS A "JSR @BPI+1" TO END UP AT THIS INTERCEPT
;;                    ROUTINE, FROM WHICH "BREAK" IS CALLED.
;;
;;-----------------------------------------------------------------------

    1361  6017/ BRKP:  INTEN              ;INHIBIT SWAPOUT
    1362  54017        STA     3,BPI+1
    1363   6101        CALL
    1364     14        BREAK
    1365  30005        LDA     2,RUP
    1366  55014        STA     3,URA.,2
    1367    42?        JMP     BUMP2       ;SKIP USER'S SWAP-OUT ROUTINE


;;-----------------------------------------------------------------------
;;
;;     5.   BUMP REGNANT USER
;;
;;        ENTRY:
;;                  A3 <-- USER'S RETURN ADDRESS (NEXT SCHEDULE POINT)
;;                       CONTROL IS TRANSFERED HERE FROM A CALL TO BUMPUSER
;;                       OR AFTER A CALL TO STINPUT OR WONA AND OUTPUT
;;                       IS ACTIVE OR A DSP BREAKPOINT IS INTERCEPTED.
;;
;;         EXIT:  ENTERS THE SCHEDULER LOOP TO DETERMINE THE NEXT
;;                USER TO BE MADE REGNANT.
;;
;;        PURPOSE:  TO REMOVE THE REGNANT USER IN AN ORDERLY MANNER.
;;
;;-----------------------------------------------------------------------

    1370  30005 BUMP:  LDA     2,RUP       ;BUMP REGNANT USER
    1371  55014        STA     3,URA.,2
    1372  3407/        LDA     3,.BPS
    1373   7401        JSR     @1,3        ;LET PROCESSOR DO ITS SWAP-OUT
    1374    415        JMP     BUMP2       ;   NON-SKIP ==> RETAIN PARTITION
    1375  20005        LDA     0,RUP
    1376  34100        LDA     3,.INFO
    1377  35454        LDA     3,.LSR.,3
    1400   7765        JSR     @XLCPA,3    ;CHECK FOR USER'S PARTITION
    1401 151015        SNZ     2,2         ;PART'N ASSIGNED ?
    1402    407        JMP     BUMP2       ;   NO
```

```
;                       << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

     1403 102400          SUB     0,0         ;YES
     1404  41002          STA     0,JCP.,2
     1405  31003          LDA     2,AFP.,2
     1406   6110          DATAPUMP            ;RELEASE ACTIVE FILE HEADER
     1407      3          UNLATCH
     1410 100000          @0
     1411  30005 BUMP2:   LDA     2,RUP
     1412  34106          DFTCALL
     1413   77/5          DFTWRITE            ;WRITE DFT OUT TO XMEM,  IF ANY
     1414  34100          LDA     3,.INFO
     1415  21411          LDA     0,TSC.,3    ;TENTH SECOND COUNTER
     1416  34004          LDA     3,PIB       ;PART'N INFO BLOCK
     1417 175014          SKZ     3,3         ;PART'N ASSIGNED ?
     1420  41404          STA     0,TLU.,3    ;   YES, SET TIME LAST USED
     1421  30005          LDA     2,RUP       ;PCB ADDRESS
     1422 102400          SUB     0,0
     1423  6027/          INTDS
     1424  24073          LDA     1,ESCF      ;PICK UP ESCAPE FLAG
     1425  40073          STA     0,ESCF      ;CLEAR ESCAPE FLAG
     1426 125014          SKZ     1,1         ;DID PROCESSOR IGNORE ESC FLAG ?
     1427    414          JMP     BUMP5       ;   YES, LEAVE I/O CONDITIONS AS THEY ARE
     1430  24062          LDA     1,C300      ;NO,  I/O MASK
     1431  35012          LDA     3,FLW.,2    ;FLAG WORD
     1432 167400          AND     3,1         ;ISOLATE INPUT/OUTPUT ACTIVE BITS
     1433  35026          LDA     3,PDC.,2    ;PAUSE COUNTER
     1434 125015          SNZ     1,1         ;ANY I/O ACTIVE...
     1435 175014          SKZ     3,3         ;...OR IS A PAUSE ACTIVE ?
     1436  41021          STA     0,AOI.,2    ;   YES, END INTERACTION
     1437  35015          LDA     3,ABN.,2;CLEAR ESC AND CNTL-C BITS IN ABN.
     1440  24720          LDA     1,ABNMSK
     1441 137400          AND     1,3
     1442  55015          STA     3,ABN.,2
     1443  40004 BUMP5:   STA     0,PIB       ;CLEAR PART'N INFO BLOCK ADDRESS
     1444  40005          STA     0,RUP       ;CLEAR PORT CONTROL BLOCK ADDRESS
     1445  40076          STA     0,ERRF      ;CLEAR ERROR FLAG
     1446  6017/          INTEN
;
;AT THIS TIME THERE IS NO REGNANT USER
;
```

```
;                        << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

;;--- --- ---  ---------- --------- ----------------------------------------
;;
;;  6.   SCHEDULER
;;
;;
;;       ENTRY:     From BUMPUSER
;;                  From INTERRUPT SERVICE
;;
;;                    AC's  None
;;
;;       EXIT:      To 'Deferred Task' via QLINK/DEQUEUE when there are one or more
;;                  tasks on the Defer Task Queue.
;;
;;                    AC's  None
;;
;;                  To START TIME SLICE
;;
;;                    AC2 - PCB Address of next job
;;
;;                  To IDLE
;;
;;                    AC's  None
;;
;;       PURPOSE:   Place any deferred tasks on the task queue.  If there are no
;;                  deferred tasks then select next job to receive a time slice.
;;
;;--- ------- ------- ------------------------ ----------------------------------

 1447  30451 SCHED: LDA     2,DEFRQ   ;CHECK DEFERRED TASK QUEUE << FROM INTST, BMPX
 1450 151014        SKZ     2,2       ;ANY TASKS BEEN DEFERRED ?
 1451    443        JMP     SCH3      ;   YES, DO THEM NOW
 1452   6101 SCHO:  CALL              ;CLEAN UP
 1453 100017        XFIXBUFFERS
 1454  30100        LDA     2,.INFO
 1455   6102        FLAGCHECK         ;INTRAUSER BUFFERING ?
 1456  20006        SKIPO+SPCF.
 1457  20000        TDP
 1460    404        JMP     SCH1      ;   NO
 1461   6110        DATAPUMP          ;YES
 1462     17        ALLFLUSH
 1463 100000        @0
 1464   4452 SCH1:  JSR     SELP      ;SELECT NEXT JOB
 1465 151014        SKZ     2,2       ;IS A JOB SELECTED ?
 1466    413        JMP     SCH5      ;   YES, CHECK TIMESHARING-SUSPENDED FLAG
 1467 126400 IDLE:  SUB     1,1       ;NO, SET IDLE PC = 0:  JMP 0
 1470   6027/       INTDS
 1471  44000        STA     1,0
 1472  22404        LDA     0,@.IDLL  ;SET CARRY LAMP
 1473 101200        MOVR    0,0
 1474   6017/       INTEN
 1475      0        JMP     0         ;IDLE AT LOCATION ZERO
```

```
;                       << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

    1476    2100 .IDLL: IDLEL
    1477    1034 .STIM: STIME   ;START TIME SLICE
    1500      10 MST:   ST

    1501   34100 SCH5:  LDA     3,.INFO
    1502   25502         LDA     1,APCB.,3;A(PCB) THAT SUSPENDED T/S
    1503   35462         LDA     3,SCON.,3
    1504   20774         LDA     0,MST     ;MASK FOR SUSPEND FLAG
    1505  117415         AND#    0,3,SNR   ;TIMESHARING SUSPENDED ?
    1506    2771         JMP     @.STIM    ;   NO, BEGIN NORMAL TIME SLICE
    1507  125015         SNZ     1,1       ;YES, PCB PTR SET ?
    1510    6142         TRAPFAULT         ;   NO
    1511  132414         SEQ     1,2       ;IS THIS THE PCB WHO SUSPENDED T/S ?
    1512     755         JMP     IDLE      ;   NO, DO NOTHING
    1513    2764         JMP     @.STIM    ;   YES, START HIM


                 ;   THIS SECTION PROCESSES THE DEFERRED TASK QUEUE BY FIRST
                 ;   CREATING A VERY HIGH PRIORITY TASK (77776) AND FORCING IT
                 ;   INTO THE TASK QUEUE AT THE TOP.  THIS TASK THEN AWAKENS ALL
                 ;   OF THOSE TASKS WHICH ARE CURRENTLY ON THE DEFERRED QUEUE
                 ;   BY PLACING THEM ON THE TASK QUEUE.


    1514    6406 SCH3:  JSR     @XQLNK

    1515    1524        SCH4
    1516  177777        -1        ;use current TCBP
    1517   77776        QP.AW     ;priority just below FAULT task
    1520       0 DEFRQ: 0         ;(A3 not used in AWAKE) <-- THIS IS DEFER QUEUE HEAD POINTER

    1521     731        JMP     SCH0      ;return to scheduler

    1522    2475 XQLNK: QLINK


;  AWAKE TASK :  SCANS DEFER QUEUE AND AWAKENS ANY TASKS THERE

    1523       0        0                 ;no conflict flags
    1524   30015 SCH4:  LDA     2,TASKQ
    1525  102400         SUB     0,0
    1526   41035         STA     0,AUXL,2 ;BREAK AUXL SO WE CAN DEQUEUE IN PEACE
    1527   30771 SCH4L: LDA     2,DEFRQ   ;CONFLICT RESOLUTION LOOP
    1530  151015         SNZ     2,2       ;ANY CONFLICTING TASKS TO QUEUE ?
    1531    6105         DQUEUE            ;   NO, RESUME NORMAL TASKING
    1532    6101         CALL              ;   YES, WAKE IT UP
    1533  100025         AWAKE
    1534     773         JMP     SCH4L     ;DO UNTIL DEFRQ IS EMPTY
```

```
;                          << SI = R91REXSCHDSA; BO = 1/A.REXM5.7033! >>
;-----------------------------------------------------------------------------
;
; 6.A.   SELECT JOB
;
;     ENTRY:     From SCHEDULER
;
;                AC3 - Return
;
;     EXIT:      To Caller, nonskip
;
;                AC2 - PCB Address     ==> Next Job
;                AC2 - 0               ==> System Idle
;
;     PURPOSE:   Provides an interim scheduler for system operating
;                under a minimum config.
;
;
;-----------------------------------------------------------------------------

  1535      0        0               ;RETURN (IF MIN CONFIG)
  1536  30100 SELP:  LDA    2,.INFO  ;PRESERVE AC3 (RETURNS @"SCH1"+1 FROM $SYS.SCHED)
  1537  310/6        LDA    2,LINKT,2;A(PSEUDO-DEVIVE LINKAGE TABLE)
  1540 150015        COM#   2,2,SNR  ;ANY TABLE ?
  1541    404        JMP    SELPO    ;   NO, OK IF MIN CONFIG
  1542  31002        LDA    2,SCHDN,2;A(OF SCHEDULER) (i.e. $SYS.SCHED)
  1543 151014        SKZ    2,2      ;IS THERE A SCHEDULER ?
  1544   1000        JMP    0,2      ;   YES, GO THERE
  1545  54770 SELPO: STA    3,SELP-1 ;SAVE RETURN
  1546  30100        LDA    2,.INFO
  1547   6102        FLAGCHECK       ;MIN CONFIG ?
  1550  20062        SCON.+SKIPO
  1551  40000        MC
  1552  77277        77277  ;   NO, NEED A SCHEDULER LINKED VIA PSEUDO-DEVICE TABLE ($SYS.SCHED)
  1553  32405        LDA    2,@SLPCA ;   YES, CHECK PORT 0
  1554  21021        LDA    0,AOI.,2 ;AGE OF INTERACTION
  1555 101015        SNZ    0,0      ;READY ?
  1556 152400        SUB    2,2      ;   NO, INDICATE IDLE STATE
  1557   2756        JMP    @SELP-1  ;RETURN

  1560    604 SLPCA: INFO+LPCA.      ;PORT CONTROL AREA
```

```
;                        << SI = R91REXSCHDSA; BO = 1/A.REXM5.7033! >>

;-- --------- -------------------- ---- ------------------------------------
;
;  6.B.   START AN INTERACTION
;
;
;  ENTRY:       From TEN HERTZ, PC
;
;               AC2 - PCB
;               AC3 - Return
;
;
;  EXIT:        To Caller, nonskip
;
;               AC2 - PCB
;
;
;  PURPOSE:     Does the initilization at a port that starts a new interaction.
;
;                   a. RUI <-- 0
;                   b. AOI <-- Initial AGE (from KSCH)
;
;               Places the system in an interactive state.
;
;                   a. SCON(ID) <-- True  (Interactive State Flag)
;                   b. TCTSL    <-- Short TS  (If TCTSL > Short TS)
;
;
;  NOTE:        This routine will not start an interaction until each of the
;               following are true:
;
;                   a. Pause Ctr is zero (PDC)
;                   b. Output is not active (FLW)
;                   c. Input is not active (FLW)
;
;               This routine is reentrant.
;
;               This routine will be modified when the Job Selection subroutine
;               is modified.
;
;------------ -------- ------------- -- ------------------------------------
```

```
;                       << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

     1561  21026 SINT:  LDA     0,PDC.,2 ;PAUSE COUNTER
     1562 101014        SKZ     0,0      ;PAUSE ACTIVE ?
     1563   1400        JMP     0,3      ;   YES, WAIT
     1564  21012        LDA     0,FLW.,2
     1565  24062        LDA     1,C300
     1566 123414        AND#    1,0,SZR  ;I/O ACTIVE ?
     1567   1400        JMP     0,3      ;   YES, WAIT
     1570  21023        LDA     0,PF1.,2 ;GET EFP (EFFECTIVE PRIORITY)
     1571  24066        LDA     1,C777
     1572 123400        AND     1,0      ;EFP
     1573  24002        LDA     1,C2     ;MAX BACKGROUND EFP
     1574 106432        SGR     0,1      ;BACKGROUND ?
     1575    414        JMP     SIN11    ;   YES, LEAVE SYSTEM STATE AS IS
     1576  22421        LDA     0,@STCON ;SET INTERACTIVE STATE FLAG
     1577  24421        LDA     1,STIOD
     1600 123415        AND#    1,0,SNR  ;ALREADY SET ?
     1601 123000        ADD     1,0      ;   NO
     1602  42415        STA     0,@STCON
     1603  22413        LDA     0,@STTSL ;REDUCE TIME SLICE
     1604  24064        LDA     1,C377
     1605 123400        AND     1,0      ;SHORT TS
     1606  26413        LDA     1,@STCTR ;TIME SLICE COUNTER
     1607 106432        SGR     0,1      ;MORE THAN A SHORT TS REMAINING ?
     1610  42411        STA     0,@STCTR ;   YES, REDUCE TO A SHORT TS
     1611 102400 SINT1: SUB     0,0      ;SET INITIAL SERVICE
     1612  41020        STA     0,RUI.,2 ;INITIAL SERVICE
     1613 101400        INC     0,0
     1614  41021        STA     0,ADI.,2 ;INITIAL AGE
     1615   1400        JMP     0,3      ;RETURN


     1616    617 STTSL: INFO+KTSL.       ;LONG-SHORT TIME SLICE
     1617    662 STCON: INFO+SCON.       ;SYSTEM CONDITION WORD
     1620   4000 STIOD: ID               ;INTERACTIVE STATE FLAG BIT
     1621   2101 STCTR: TCTSL            ;TIME SLICE COUNTER
     1622 1777/5 VMXTA: MXTA             ;MUX "SEND" COMMAND TO TERMINATE ALL I/O


     1623      0 .SIGB: 0        ;BEGIN SIGNAL BUFFER  \
     1624      0 .SIGN: 0        ;NEXT ENTRY POSITION   } SET BY SIR
     1625      0 .SIGE: 0        ;END OF SIGNAL BUFFER /
```

```
;                        << SI = R91REXSCHDSA; BO = 1/A.REXM5.7033! >>

;;-------- ------------------------------ ------------------
;;
;;     7.   TEN HERTZ TASK
;;
;;          ENTRY:   QUEUED EACH TENTH-SECOND BY REAL TIME CLOCK DRIVER.
;;           NOTE:   THE RTC DRIVER IS NORMALLY IN THE $MMUX DRIVER.
;;
;;           EXIT:   DQUEUE
;;
;;        PURPOSE:   FORCE EXECUTION OF VARIOUS SYSTEM FUNCTIONS AT
;;                   KNOWN INTERVALS.
;;
;;-------- -- ------------------- ------ ----------------------------------------

    1626       0        0                    ;TASK CONFLICT FLAGS

                   ;  UPDATE SYSTEM CLOCK COUNTERS IN INFO TABLE

    1627   34100 TNHZT: LDA     3,.INFO
    1630  152400         SUB     2,2
    1631   21441         LDA     0,TSC.,3  ;TENTH SECONDS COUNTER
    1632   25473         LDA     1,THTC.,3;RESET TEN HERTZ TASK COUNTER
    1633   51473         STA     2,THTC.,3
    1634   44562         STA     1,TSC     ;REMEMBER # TICKS TO PROCESS
    1635  123000         ADD     1,0
    1636   41441         STA     0,TSC.,3
    1637   24462         LDA     1,TSPH
    1640  122423         SUBZ    1,0,SNC   ;END OF HOUR ?
    1641     462         JMP     THZ3      ;   NO, AWAKEN THE SLEEPERS
    1642   41441         STA     0,TSC.,3;YES, CLEAR TNTH SEC CNTR
    1643   11440         ISZ     HRS.,3    ;BUMP HOURS, OVERFLOW ?
    1644  100010         NOP

;  THIS CODE IS EXECUTED ONCE AN HOUR

    1645   20455         LDA     0,HRDAY   ;HRs PER DAY
    1646   25440         LDA     1,HRS.,3
    1647    6115         BINDIVIDE
    1650  125014         SKZ     1,1       ;CHANGE DAYS ?
    1651     421         JMP     TNHZ1     ;   NO, ONLY CHANGED THE HOUR

;THIS CODE IS EXECUTED ONCE A DAY (AT MIDNIGHT)

    1652   26416         LDA     1,@.HRS.
    1653    6416 DAYCK: JSR     @.LDAY    ;LOCATE TABLE
    1654   21400 DAYC1: LDA     0,0,3
    1655  101415         INC#    0,0,SNR   ;END OF TABLE ?
    1656     414         JMP     TNHZ1     ;   YES, THIS DAY IS LEGITIMATE
    1657  106415         SNE     0,1       ;ILLEGITIMATE DAY ?
    1660     403         JMP     DAYC2     ;   YES
    1661  175400         INC     3,3
    1662     772         JMP     DAYC1     ;CONTINUE CHECKING TABLE
```

```
;                    << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

1663   26405 DAYC2: LDA     1,@.HRS.  ;   YES, ADD 24 HOURS SO MONTH WILL CHANGE
1664   30436        LDA     2,HRDAY   ;HOURS IN ONE DAY
1665  147000        ADD     2,1
1666   46402        STA     1,@.HRS.
1667     764        JMP     DAYCK     ;NOW CHECK AGAIN IN CASE MONTH IS FEB

1670     640 .HRS.: INFO+HRS.
1671    2223 .LDAY: LDAYS

                ;   CULL OUT AGED SIGNALS FROM SIGNAL BUFFER (ENTERED AT END OF AN HOUR)

1672   34731 TNHZ1: LDA     3,.SIGB   ;A(BEG SIGNAL BUFFER)
1673   20731        LDA     0,.SIGN   ;PREPARE TO SCAN SIGNAL BUFFER
1674  162033 THZ1:  SLS     3,0       ;ANY MORE SIGNALS ? <<FRO THZ2, THZ10
1675     426        JMP     THZ3      ;   NO, AWAKEN ANY SLEEPERS
1676   25400        LDA     1,0,3     ;   YES, GET TIME ON QUEUE
1677  125113        SSN     1,1       ;IS THIS ONE OLDER THAN 1 HOUR ?
1700     413        JMP     THZ2      ;   NO, FLAG IT FOR DELETION
1701  171000        MOV     3,2       ;   YES, DELETE IT

1702   25004 THZ10: LDA     1,4,2     ;COPY BACK REST OF SIGNAL BUFFER
1703   45000        STA     1,0,2
1704  151400        INC     2,2
1705  142032        SGE     2,0
1706     774        JMP     THZ10
1707   24024        LDA     1,C4
1710  122400        SUB     1,0       ;MOVE BACK BUFFER POINTER
1711   40713        STA     0,.SIGN
1712     762        JMP     THZ1

1713  127240 THZ2:  ADDOR   1,1       ;FLAG TO DELETE SIGNAL NEXT HOUR
1714   45400        STA     1,0,3
1715   30024        LDA     2,C4
1716  157000        ADD     2,3
1717     755        JMP     THZ1

1720    2660 .SLPQ: SLEPQ             ;*** REF *** (QUEM)
          12        .RDX    10
1721  106240 TSPH:  60*60*10          ;TENTH SECONDS PER HOUR
1722      30 HRDAY: 24                ;HOURS IN ONE DAY
          10        .RDX    8

                ;   AWAKEN SLEEPING TASKS

1723   6027/ THZ3:  INTDS             ;cannot allow interrupt till sleeper is on task queue
1724   32774        LDA     2,@.SLPQ  ;CHECK SLEEP QUEUE
1725  151015        SNZ     2,2       ;ANY NODES ASLEEP NOW ?
1726     411        JMP     THZ40     ;   NO
1727   21012        LDA     0,PAUZ,2  ;YES
1730   24466        LDA     1,TSC
1731  122422        SUBZ    1,0,SZC   ;calc.rel.awake time:  positive ?
1732     404        JMP     THZ39     ;   yes, not ready to awaken
1733    6402        JSR     @.AWAK    ;no, then wake it up now (does INTEN)
1734     767        JMP     THZ3      ;CHECK FOR ANOTHER

1735    2470 .AWAK: AWAKX              ;bypass CALL mechanism to avoid premature INTEN
```

```
;                    << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

1736   41012 THZ39: STA     0,PAUZ,2 ;UPDATE REMAINING SLEEP TIME
1737   60177 THZ40: INTEN
1740   30100        LDA     2,.INFO  ;PREPARE TO SCAN ALL PCB'S
1741   21005        LDA     0,TNAP.,2
1742   40533        STA     0,THZC   ;COUNT <-- TOTAL # ACTIVE PORTS
1743   31004        LDA     2,LPCA.,2
1744   21021 THZ4:  LDA     0,AOI.,2 ;SCAN ALL PORTS
1745   25026        LDA     1,PDC.,2
1746  101015        SNZ     0,0      ;INTERACTION STARTED,
1747  125014        SKZ     1,1      ;  OR IN A PAUSE ?
1750     406        JMP     THZ4A    ;    YES
1751   20047 THZ6:  LDA     0,C40    ;NO, STEP TO NEXT PCB
1752  113000        ADD     0,2
1753   14522        DSZ     THZC     ;LAST PORT ?
1754     770        JMP     THZ4     ;  NO
1755     545        JMP     THZ7     ;  YES


1756   60277 THZ4A: INTDS            ;Disable interrupts for PDC. and FLW.
1757   21021        LDA     0,AOI.,2 ;AGE OF INTERACTION
1760   25026        LDA     1,PDC.,2 ;PAUSE CTR
1761   34435        LDA     3,TSC    ;# TICKS TO PROCESS
1762  101014        SKZ     0,0      ;INTERACTION STARTED ...
1763  125014        SKZ     1,1      ;... AND PAUSE COUNTER = 0 ?
1764     404        JMP     THZ5     ;   NO
1765  163022        ADDZ    3,0,SZC  ;AGE THE INTERACTION -- OVERFLOW ?
1766  102000        ADC     0,0      ;  YES, LIMIT TO 177777
1767   41021        STA     0,AOI.,2
1770  125015 THZ5: SNZ     1,1      ;IS JOB PAUSED ?
1771     465        JMP     THZ6A    ;  NO
1772  166423        SUBZ    3,1,SNC  ;COUNT DOWN THE PAUSE -- UNDERFLOW ?
1773  126400        SUB     1,1      ;  YES, SET TO ZERO
1774   45026        STA     1,PDC.,2
1775  125014        SKZ     1,1      ;PAUSE NOW DONE ?
1776     460        JMP     THZ6A    ;  NO
1777   50477        STA     2,THPCB  ;YES, END OF A PAUSE
2000   35012        LDA     3,FLW.,2
2001   60177        INTEN            ;Re-enable interrupts
2002   24051        LDA     1,C100   ;END OF A PAUSE
2003  137414        AND#    1,3,SZR  ;INPUT ACTIVE ?
2004     417        JMP     THZ5E    ;  YES, TIME OUT
2005   20005        LDA     0,RUP
2006  112415        SNE     0,2      ;NO, IS USER REGNANT ?
2007     410        JMP     THZ5A    ;  YES, END HIS TIME SLICE & SET PDC = .1
2010    6147        STINTERACTION    ;NO, START INTERACTION
2011    6102        FLAGCHANGE       ;CLEAR "ACTIVATE FROM SIGNAL" FLAG
2012  100012        RESET+FLW.
2013    4000        4000
2014   30462        LDA     2,THPCB
2015     734        JMP     THZ6

2016       0 TSC:   0                ;SAVE # TICKS BEING PROCESSED THIS INSTANCE
```

THZ5C

ADC
8-31-87

; << SI = R91REXSCHDSA; BO = 1/A.REXM5.7033! >>

```
2017  10074 THZ5A: ISZ     ETSF        ;YES, END HIS TIME SLICE
2020  11026 THZ5B: ISZ     PDC.,2      ;SET THE PAUSE DELAY = .1 SEC
2021    730        JMP     THZ6

2022 177772 VMXSS: MXSS                ;MUX START SINGLE CHAR. INPUT (NON-DMA)

2023   6102 THZ5E: FLAGCHANGE          ;INPUT TIMED OUT
2024 100012        RESET!FLW.
2025    100        100                 ;CLEAR INPUT-ACTIVE BIT
2026  30450        LDA     2,THPCB
2027  20713        LDA     0,VMXSS
2030   7032        JSR     @SND.,2     ;SEND A STOP DMA INPUT COMMAND TO HARDWARE
2031   6102        FLAGCHANGE          ;Set "Input-Timed-out"
2032 140015        SET!ABN.
2033    200        PORTTOUT
2034  30442        LDA     2,THPCB     ;Recover PCB address
2035  21015        LDA     0,ABN.,2
2036  24422        LDA     1,VTHPD
2037 107414        AND#    0,1,SZR     ;Is port detached?
2040    414        JMP     THZ5F       ;   Yes - don't signal the timeout
2041  34103        IOCALL              ;No - get input length
2042   7746        IBCOUNT             ;GET # BYTES IN INPUT BUFFER
2043 111000        MOV     0,2         ;BECOMES 2ND PARAMETER
2044 126400        SUB     1,1         ;INDICATE TIME OUT
2045  20431        LDA     0,THPCB     ;SEND SIGNAL TO SELF
2046   6104        QUEUE
2047      0        SIGNAL              ;SIGNAL TASK
2050 177776        -2                  ;-2 IMPLIES SPECIAL SIGNAL
2051  30000        QP.SI               ;PRIORITY
2052      0        0                   ;A3 (NOT USED)
2053  30423        LDA     2,THPCB
2054  20405 THZ5F: LDA     0,VQIDN
2055   6103        QCHARACTER          ;QUEUE "INPUT DONE" TASK
2056  60177 THZ6A: INTEN
2057    672        JMP     THZ6        ;PROCESS NEXT PORT

2060     40 VTHPD: PORTDTCH            ;PORT DETACHED
2061    403 VQIDN: QCIDN               ;INPUT-DONE CODE FOR QCHARACTER
```

THZ5C: INTEN

THZ5C

```
LDA  0,C200
AND# 0,3,SZR   ;OUTPUT ACTIVE?
JMP  THZ5B     ;  YES, keep pause
SUB  1,3       ;  NO, clear INPUT
STA  3,FLW.,2
```

RDC
8-31-87

```
;                       << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>


; SCHEDULER TASK NODE

    2062       0 SNODE: 0        ;   0 ;A2
    2063       0        0        ;   1 ;A1
    2064       0        0        ;   2 ;A0
    2065       0        0        ;   3 ;A3
    2066    1447        SCHED    ;   4 ;PC
    2067       0        0        ;   5 ;CPRI
    2070       0        0        ;   6 ;SSBA
    2071       0        0        ;   7 ;SDBA
    2072       0        0        ;  10 ;TCBP
    2073    1447        SCHED    ;  11 ;TASK
    2074       0        0        ;  12 ;SN.Z;RECORD # OF LAST DATA FILE ACCESS
    2075       0 THZC:  0        ;  13 ;PCB COUNTER                                \
    2076       0 THPCB: 0        ;  14 ;PCB POINTER                                 \
    2077       5 IDLEC: 5        ;  15 ;COUNTER TO FLASH CARRY LIGHT                 } TENHZ
    2100       0 IDLEL: 0        ;  16 ;STATE OF CARRY LIGHT FOR IDLE(BIT 0)       /
    2101       0 TCTSL: 0        ;  17 ;TIME SLICE COUNTER                        /
    2102       0        0        ;  20 ;SN.0;
    2103       0        0        ;  21 ;SN.1;
    2104       0        0        ;  22 ;SN.2;
    2105       0        0        ;  23 ;SN.3;
    2106       0        0        ;  24 ;SN.4;
    2107       0        0        ;  25 ;SN.5;
    2110       0        0        ;  26 ;SN.6;
    2111       0        0        ;  27 ;SN.7;
    2112       0        0        ;  30 ;SN.8;
    2113       0        0        ;  31 ;SN.9;
    2114       0        0        ;  32 ;SN.A;LU    \ OF LAST DATA FILE ACCESS
    2115       0        0        ;  33 ;SN.B;HRDA /
    2116       0        0        ;  34 ;NSTS
    2117       0        0        ;  35 ;AUXL
    2120      15        TASKQ    ;  36 ;PLNK
    2121       0        0        ;  37 ;LINK (END OF QUEUE)
```

```
;                       << SI = R91REXSCHDSA;  BO = 1/A.REXM5.7033! >>

     2122  30005 THZ7:  LDA     2,RUP
     2123 151015         SNZ     2,2         ;IS THERE A REGNANT USER ?
     2124    435         JMP     THZ8        ;   NO
     2125  34074         LDA     3,ETSF      ;YES
     2126 175015         SNZ     3,3         ;IS REGNANT USER OVERTIME ?
     2127    414         JMP     THZ7A       ;   NO
     2130  24067         LDA     1,C1000     ;YES
     2131 166432         SGR     3,1         ;MORE THAN 25.6 SECONDS ?
     2132    411         JMP     THZ7A       ;   NO
     2133   4402         JSR     .+2         ;YES,  ABORT PROCESSOR !
     2134 116010         34*K!NOP
     2135  54730         STA     3,SNODE+A3;A3 = A(34*K!NOP)
     2136  20730         LDA     0,SNODE+PC
     2137  40725         STA     0,SNODE+A0;A0 = OLD RETURN ADDRESS
     2140  20142         LDA     0,TRAPF&377
     2141  40725         STA     0,SNODE+PC;RETURN ADDRESS = TFALT
     2142 176520         SUBZL   3,3         ;SET ETSF = 1 TO PREVENT DOUBLE TRAP
     2143  21020 THZ7A:  LDA     0,RUI.,2
     2144  24652         LDA     1,TSC
     2145 123022         ADDZ    1,0,SZC     ;CHARGE INTERACTION TSC TENTH-SECONDS -- OVERFLOW ?
     2146 102000         ADC     0,0         ;   YES, LIMIT TO 177777
     2147  41020         STA     0,RUI.,2
     2150  31017         LDA     2,PCX.,2
     2151  21007         LDA     0,RUA.,2
     2152 123022         ADDZ    1,0,SZC     ;CHARGE JOB TSC TENTH-SECONDS -- OVERFLOW ?
     2153  11006         ISZ     RUAOV,2     ;   YES,  INTO SECOND WORD OF 2-WORD PRECISION
     2154  41007         STA     0,RUA.,2
     2155  31000         LDA     2,PCB.,2
     2156 167000         ADD     3,1
     2157 175014         SKZ     3,3         ;IS REGNANT USER OVERTIME ?
     2160  44074         STA     1,ETSF      ;   YES,  ACCUMULATE BY HOW MUCH
     2161  14716 THZ8:   DSZ     IDLEC       ;END OF A HALF-SECOND ?
     2162    405         JMP     THZ8A       ;   NO
     2163  24025         LDA     1,C5        ;YES
     2164  44713         STA     1,IDLEC
     2165  10713         ISZ     IDLEL       ;TOGGLE IDLE CARRY FLAG
     2166 100010         NOP
     2167 151015 THZ8A:  SNZ     2,2         ;IS THERE A REGNANT USER ?
     2170    413         JMP     THZ9        ;   NO
     2171  20710         LDA     0,TCTSL     ;YES
     2172  24624         LDA     1,TSC
     2173 122423         SUBZ    1,0,SNC     ;COUNT DOWN THE TIME SLICE COUNTER -- UNDERFLOW ?
     2174 102400         SUB     0,0         ;   YES,  LIMIT TO 0
     2175  40704         STA     0,TCTSL
     2176 101014         SKZ     0,0         ;DID TIME SLICE EXPIRE ?
     2177    404         JMP     THZ9        ;   NO
     2200  10074         ISZ     ETSF        ;YES,  SET END OF TIME SLICE FLAG
     2201   6105         DQUEUE              ;EXIT
     2202  63077         HALT                ;IMPOSSIBLE !
```

```
;                       << SI = R91REXSCHDSA; BO = 1/A.REXM5.7033! >>

; FLUSH ONE DIRTY POOL BUFFER EVERY TENTH-SECOND

   2203  30015 THZ9:  LDA     2,TASKQ
   2204  31037        LDA     2,LINK,2 ;A(NEXT TASK NODE)
   2205  25004        LDA     1,PC,2   ;GET THE EXECUTE ADDR.
   2206 125014        SKZ     1,1      ;RETURNING TO IDLE ?
   2207   6105        DQUEUE           ;   NO, JUST DO NEXT TASK
   2210   6110        DATAPUMP         ;yes, flush a pool buffer
   2211     15        LRUFLUSH         ;flushes the least recently used dirty buffer
   2212 100000        @0
   2213   6105        DQUEUE


; DEFERRED TASK TO END USER'S PAUSE

; CALLING SEQUENCE:
;       (A2) = PCB
;   QUEUE
;   ENDPAUSE
;   {TCB IGNORED}
;   QP.PA
;   {A3 IGNORED}
;   <-- RETURN WITH A2 PRESERVED

   2214 100000        100000           ;TASK CONFLICT FLAG
   2215  60277 EPAUS: INTDS
   2216  21026        LDA     0,PDC.,2 ;USER'S CURRENT PAUSE DELAY
   2217 126520        SUBZL   1,1      ;FORM A 1
   2220 106433        SLE     0,1      ;IS REMAINING PAUSE MORE THAN 1 ?
   2221  45026        STA     1,PDC.,2 ;   YES, THEN REDUCE IT TO 1
   2222   6105        DQUEUE           ;FUNCTION ACCOMPLISHED, DEQUEUE
```

```
;                    << SI = R91REXSCHDSA; BO = 1/A.REXM5.7033! >>

    2223   5400 LDAYS: JSR    0,3       ;LOCATE TABLE ADDRESS

;TABLE OF FIRST HOUR IN DAYS THAT ARE NON-EXIST (e.g. FEB 30)

              12        .RDX   10

    2224   2640             1*31+29*24  ;FEB 30 BASE YEAR
    2225   2670             1*31+30*24  ;FEB 31 BASE YEAR
    2226   5610             3*31+30*24  ;APR 31 BASE YEAR
    2227  10530             5*31+30*24  ;JUN 31 BASE YEAR
    2230  15020             8*31+30*24  ;SEP 31 BASE YEAR
    2231  17740            10*31+30*24  ;NOV 31 BASE YEAR

    2232  24150        12+1*31+28*24    ;FEB 29 BASE YEAR + 1
    2233  24200        12+1*31+29*24    ;FEB 30 BASE YEAR + 1
    2234  24230        12+1*31+30*24    ;FEB 31 BASE YEAR + 1
    2235  27150        12+3*31+30*24    ;APR 31 BASE YEAR + 1
    2236  32070        12+5*31+30*24    ;JUN 31 BASE YEAR + 1
    2237  36360        12+8*31+30*24    ;SEP 31 BASE YEAR + 1
    2240  41300        12+10*31+30*24   ;NOV 31 BASE YEAR + 1

    2241  45510        24+1*31+28*24    ;FEB 29 BASE YEAR + 2
    2242  45540        24+1*31+29*24    ;FEB 30 BASE YEAR + 2
    2243  45570        24+1*31+30*24    ;FEB 31 BASE YEAR + 2
    2244  50510        24+3*31+30*24    ;APR 31 BASE YEAR + 2
    2245  53430        24+5*31+30*24    ;JUN 31 BASE YEAR + 2
    2246  57720        24+8*31+30*24    ;SEP 31 BASE YEAR + 2
    2247  62640        24+10*31+30*24   ;NOV 31 BASE YEAR + 2

    2250  67050        36+1*31+28*24    ;FEB 29 BASE YEAR + 3
    2251  67100        36+1*31+29*24    ;FEB 30 BASE YEAR + 3
    2252  67130        36+1*31+30*24    ;FEB 31 BASE YEAR + 3
    2253  72050        36+3*31+30*24    ;APR 31 BASE YEAR + 3
    2254  74770        36+5*31+30*24    ;JUN 31 BASE YEAR + 3
    2255 101260        36+8*31+30*24    ;SEP 31 BASE YEAR + 3
    2256 104200        36+10*31+30*24   ;NOV 31 BASE YEAR + 3

    2257 110440        48+1*31+29*24    ;FEB 30 BASE YEAR + 4
    2260 110470        48+1*31+30*24    ;FEB 31 BASE YEAR + 4
    2261 113410        48+3*31+30*24    ;APR 31 BASE YEAR + 4
    2262 116330        48+5*31+30*24    ;JUN 31 BASE YEAR + 4
    2263 122620        48+8*31+30*24    ;SEP 31 BASE YEAR + 4
    2264 125540        48+10*31+30*24   ;NOV 31 BASE YEAR + 4

    2265 131750        60+1*31+28*24    ;FEB 29 BASE YEAR + 5
    2266 132000        60+1*31+29*24    ;FEB 30 BASE YEAR + 5
    2267 132030        60+1*31+30*24    ;FEB 31 BASE YEAR + 5
    2270 134750        60+3*31+30*24    ;APR 31 BASE YEAR + 5
    2271 137670        60+5*31+30*24    ;JUN 31 BASE YEAR + 5
    2272 144160        60+8*31+30*24    ;SEP 31 BASE YEAR + 5
    2273 147100        60+10*31+30*24   ;NOV 31 BASE YEAR + 5
    2274 177777        -1               ;-1 TERMINATOR
              10        .RDX   8

                .EOT   ;REX "SCHD" FOR "IRIS" RESIDENT EXECUTIVE
```

```
;                          << SI = R90REXQUEMSA; BO = 1/A.REXM5.7033! >>
; 16 JUN 86, RB.

;;---- ---- ----- ---------------------------- ------ ----------------------------
;;
;;            R E X :   Q U E U I N G   F U N C T I O N S
;;
;;      MAJOR COMPONENTS:
;;
;;            1.   CREATE A NODE ON THE TASK QUEUE
;;            2.   EXTRACT A NODE FROM A QUEUE
;;            3.   AWAKE, IE. EXTRACT A NODE & PLACE ON TASK QUEUE
;;            4.   CREATE A NODE ON TASK QUEUE & LINK TO CALLER
;;            5.   DEQUEUE REGNANT TASK
;;            6.   PUT NODE ON SLEEP QUEUE
;;
;;      DESCRIPTION OF SYSTEM QUEUES:
;;
;;            TASK QUEUE - QUEUE OF TASKS TO BE EXECUTED IN ORDER OF
;;                 PRIORITY.
;;
;;            SLEEP QUEUE - QUEUE OF TASKS OR NODES WHICH ARE WAITING
;;                 FOR AN EXTERNAL EVENT, OR A TIME TO EXPIRE.
;;
;;            DEFER QUEUE - TASKS WITH CONFLICT FLAGS WAITING FOR
;;                 END OF REGNANT USER'S TIME SLICE
;;
;;     UPDATE RECORD:
;;
;;            6-3-80   (LLL)  REARRANGED FOR READABILITY
;;           10-09-80 (GAD) UPDATE,DOCUMENT & CORRECT
;;            1-19-81 (GAD) CORRECTED ERRORS
;;                          ASSIGNED TRAP #'S
;;            4- 1-81 (GAD) INCORPORATED PATCHES
;;            7-20-81 (GAD) INCORPORATED PATCHES
;;           11 NOV81 (RMS) IMPROVED QUEING TRAPS
;;            1-29-82 (rb.) added qlink
;;            6-05-86 (JPM) Increased reserved words in task table
;;
;;      EXTERNAL REFERENCES:
;;          GTNOD (MEMM)
;;          FRNOD (MEMM)
;;          INTSF (INTS)
;;          INTEX (INTS)
;;          TNHZT \
;;          SSIGT /  TASKS
;;
;;      GLOBAL REFERENCES:
;;          SUBROUTINES:
;;              TRAPFAULT
;;          DATA:
;;              TASKQ
;;              TSC.(INFO)
;;---------- ---------- ---------- ----------------------------------------------
```

```
;                        << SI = R9OREXQUEMSA;  BO = 1/A.REXM5.7033! >>
;;-------------------------------------------------------------------------
;;     1.    QUEUE A TASK                                <<RE-ENTRANT>>
;;
;;           ENTRY:  A0, A1, A2 <-- PARAMETERS FOR TASK
;;                   A3 <-- A (PARAMETERS FOLLOWING CALL)
;;
;;                   CALLING SEQUENCE:
;;                        QUEUE
;;                        <TASK EXEC. ADDR ! INDEX INTO TASK TABLE>
;;                        <TCB PTR> OR -1 (IF REGNANT TASK'S TCB IS TO BE USED
;;                        <PRIORITY>
;;                        <CONTENTS OF A3 FOR TASK>
;;
;;           NOTE:   A TASK TO BE QUEUED MUST HAVE ITS CODE IN CORE, & MUST
;;                   HAVE A FLAGWORD PRECEDING THE ENTRY POINT.   THE FLAGS
;;                   ARE DEFINED AS FOLLOWS:
;;                   BIT 15 <-- POSSIBLE TASK CONFLICT:  put on Defer Queue
;;
;;                   THIS ROUTINE MAY BE CALLED FROM INTERRUPT HANDLERS
;;
;;           EXIT:   RETURNS AT POINT OF CALL + 4
;;                   A0, A1 ARE UNCHANGED
;;                   A2 <-- A(QUEUED NODE)
;;                   NSTS(NODE) <-- 0 " ON SYSTEM TASK QUEUE "
;;                   AUXL(node) <-- 0
;;                   LINK(node) <-- next lower priority task node
;;                   PLNK(node) <-- next higher priority task node
;;
;;           PURPOSE:   THIS ROUTINE GETS A FREENODE, CHECKS THE QUEUEING
;;                   PRIORITY, SEARCHES THE TASK QUEUE IN ORDER TO FIND THE
;;                   PROPER POSITION FOR THE NEW NODE, AND PLACES IT ON THE
;;                   QUEUE.
;;-------------------------------------------------------------------------

;  Algorithm:
;       INTDS
;       Get a new node
;       Save accumulators in new node
;       Get parameters from QUEUE parameter list
;       INTEN
;       Configure new node per parameters
;       Check that given PRI <= Regnant task's PRI, unless IRUPT <> -1
;       INTDS
;       If conflict flag is set, put new node on Defer queue
;       Else insert new node in Task Queue at proper place
;       Restore accumulators
;       INTEN
;       Return
```

```
;                    << SI = R90REXQUEMSA;  BO = 1/A.REXM5.7033! >>

2275    2277 TASKT: .+2      ;TASK NUMBER TABLE
2276       3        ETSKS-1-.       ;NUMBER OF FIXED TASKS
2277   11717        SSIGT   ;  0 = SEND SIGNAL TASK
2300    1627        TNHZT   ;  1 = TEN HERTZ TASK
2301    2215        EPAUS   ;  2 = END PAUSE TASK
2302       0 ETSKS: 0       ;Reserved
2303       0        0       ;Reserved
2304       0        0       ;Reserved
2305       0 QURTN: 0

2306   60277 QUE:   INTDS             ;GET A NODE
2307   54776        STA     3,QURTN
2310    6525        JSR     @.GTND    ;(saves A2 in node's A2 cell)
2311   41002        STA     0,A0,2    ;SAVE A0
2312   45001        STA     1,A1,2    ;and A1
2313   34772        LDA     3,QURTN
2314   20024        LDA     0,C4
2315  163000        ADD     3,0       ;adjust return address past params.
2316   41014        STA     0,QRTN,2  ;SAVE RETURN ADDRESS
2317   25403        LDA     1,3,3     ;SAVE A3 (FROM PARAMETER LIST)
2320   45003        STA     1,A3,2
2321   21401        LDA     0,1,3     ;SET UP TCB POINTER
2322  101415        INC#    0,0,SNR   ;REGNANT TASK'S TCB TO BE USED ?
2323   20007        LDA     0,RTP     ;   yes
2324   41010        STA     0,TCBP,2
2325   25402        LDA     1,2,3     ;SPECIFIED PRIORITY
2326   35400        LDA     3,0,3     ;GET TASK IDENTIFIER
2327   60177        INTEN
2330   20746        LDA     0,TASKT+1
2331  162033        SLS     3,0       ;TASK NUMBER GIVEN ?
2332     404        JMP     QUE0      ;   NO, MUST BE A POINTER
2333   20742        LDA     0,TASKT   ;   YES, GET POINTER FROM TABLE
2334  117000        ADD     0,3
2335   35400        LDA     3,0,3     ;TASK POINTER
2336   55011 QUE0:  STA     3,TASK,2
2337   55004        STA     3,PC,2    ;STARTING ADDRESS
2340  124537        SGZ     1,1       ;LEGAL PRIORITY ?
2341    4420        JSR     QUEF      ;   NO
2342  127000        ADD     1,1       ;SHIFT PRI LEFT (PRESERVE CARRY)
2343  121200        MOVR    1,0       ;COMBINE CARRY AND PRIORITY
2344   41005        STA     0,CPRI,2
2345   21777        LDA     0,-1,3    ;PICK UP TASK FLAGS
2346  101112        SSP     0,0       ;ANY CONFLICTS ?
2347     443        JMP     QUE5      ;   YES, MUST PUT ON DEFRQ
2350   34015 QUE0A: LDA     3,TASKQ   ;   NO, FIND POSITION IN QUEUE
2351   22413        LDA     0,@.INTF
2352   60277        INTDS
2353  101414        INC#    0,0,SZR   ;are we in an interrupt handler ?
2354     412        JMP     QUE2      ;   yes, higher PRI is OK
2355   21405        LDA     0,CPRI,3  ;no
2356  101120        MOVZL   0,0
2357  122432        SGR     1,0       ;is given PRI <= regnant PRI ?
2360     406        JMP     QUE2      ;   yes, OK to queue it
2361   21014 QUEF:  LDA     0,QRTN,2  ;ILLEGAL PRIORITY GIVEN !?
2362    6142 QUEF2: TRAPFAULT
2363  117010        36*K!NOP
```

```
;                       << SI = R9OREXQUEMSA;  BO = 1/A.REXM5.7033! >>

     2364    665 .INTF: INFO+IRUPT       ;INTERRUPT FLAG

     2365  35437 QUE1:  LDA     3,LINK,3 ;POINTER TO NEXT LOWER PRI NODE
     2366 175015 QUE2:  SNZ     3,3      ;A(TASK QUEUE NODE) = 0 ?
     2367   4772        JSR     QUEF     ;   YES, QUEING FAULT !?!?
     2370  21405        LDA     0,CPRI,3 ;NEXT NODE'S PRIORITY
     2371 101120        MOVZL   0,0      ;SHIFT OUT CARRY BIT
     2372 106033        SLS     0,1      ;PAST HIGHER PRIORITY TASKS ?
     2373    772        JMP     QUE1     ;   no, CONTINUE SCANNING QUEUE
     2374  55037        STA     3,LINK,2 ;LINK NEW NODE AFTER THIS ONE
     2375  21436        LDA     0,PLNK,3
     2376  41036        STA     0,PLNK,2 ;LINK BACK TO HIGHER PRI TASKS
     2377  53436        STA     2,@PLNK,3;enter new node as previous' next
     2400  24557        LDA     1,CLINK
     2401 147000        ADD     2,1
     2402  45436        STA     1,PLNK,3 ;LINK NEXT NODE BACK TO THIS ONE
     2403 102400 QUE4:  SUB     0,0
     2404  41034        STA     0,NSTS,2 ;0 ==> ON SYSTEM TASK QUEUE
     2405  21002        LDA     0,A0,2   ;RESTORE ACCUMULATORS A0 AND A1
     2406  25001        LDA     1,A1,2
     2407  60177        INTEN
     2410   3014        JMP     @QRTN,2  ;RETURN TO CALLER


;   IN ANY CASE OF TASK CONFLICT, THE TASK WILL NOT BE ENQUED
;   ON THE TASK QUEUE.  INSTEAD, IT WILL BE QUEUED ON THE TOP
;   OF THE DEFER QUEUE AND WILL BE TRANSFERED TO THE TASK QUEUE
;   ONLY WHEN THE SCHEDULER IS NEXT INVOKED.

     2411   1520        DEFRQ
     2412  60277 QUE5:  INTDS
     2413  36776        LDA     3,@QUE5-1;PLACE CONFLICTING TASKS ON DEFER QUEUE
     2414 126400        SUB     1,1      ;GEN A ZERO
     2415  45037        STA     1,LINK,2 ;IN FOWARD LINK FOR NEW NODE
     2416 175014        SKZ     3,3      ;IS QUEUE EMPTY?
     2417    404        JMP     QUE6     ;NOPE, CHASE DOWN LINKS
     2420  52771        STA     2,@QUE5-1;YEP, POINT TO NEW FIRST NOTE
     2421  20770        LDA     0,QUE5-1 ;PREVIOUS LINK FOR FIRST NODE
     2422    411        JMP     QUE7     ;JOIN COMMON CODE
     2423 165000 QUE6:  MOV     3,1      ;SAVE A(THIS NODE ON CHAIN)
     2424  35437        LDA     3,LINK,3 ;POINTER TO NEXT NODE
     2425 175014        SKZ     3,3      ;THIS THE END OF LINKS?
     2426    775        JMP     QUE6     ;NOPE, KEEP SEARCHING LINKS
     2427 135000        MOV     1,3      ;RESTOR A (LAST NODE ON CHAIN)
     2430  51437        STA     2,LINK,3 ;LINK IN A NEW NODE TO END OF CHAIN
     2431  20526        LDA     0,CLINK  ;OFFSET INTO NOTE FOR LINK CELL
     2432 163000        ADD     3,0      ;POINT TO END OF PREVIOUS LAST NOTE
     2433  41036 QUE7:  STA     0,PLNK,2 ;SET POINTER TO PREVIOUS LINK
     2434    747        JMP     QUE4     ;COMPLETE NODE AND EXIT

     2435   2734 .GTND: GTNOD   ;*** REF *** (MEMM)
```

```
;                       << SI = R9OREXQUEMSA;  BO = 1/A.REXM5.7033! >>
;;-------- ------------------------ ----- -- ------------------------------
;;
;;     2.   EXTRACT A NODE FROM A QUEUE                    <<RE-ENTRANT>>
;;
;;               CALLING SEQUENCE:
;;                       CALL
;;                       XQUEUE (@26)
;;
;;               ENTRY:  A2 <-- A(NODE)
;;
;;               EXIT:   A2 <-- A(NODE)
;;                       NSTS(NODE) <-- -1
;;
;;                       LINK(PREVIOUS-NODE) <-- A(NEXT-NODE)
;;                       PLNK(NEXT-NODE) <-- A(link(PREVIOUS-NODE))
;;
;;                       Leaves interrupts disabled so that caller can
;;                       put the node somewhere without fear of interrupt
;;
;;             PURPOSE:  EXTRACT A NODE FROM ANY QUEUE IT'S IN
;;                       IF SLEEP QUEUE, RECOMPUTE NEXT NODE'S PAUZ
;;-------------------------------------- ------------------------------

 2436 161000 XQUEX: MOV     3,0       ;EXTRACT NODE FROM A QUEUE
 2437  35034        LDA     3,NSTS,2
 2440 175112        SSP     3,3
 2441   6142        TRAPFAULT         ;NODE NOT ON ANY QUEUE !?
 2442 120010        40*K!NOP
 2443  60277        INTDS
 2444  35037        LDA     3,LINK,2
 2445  57036        STA     3,@PLNK,2;LINK PREV TO NEXT
 2446  25036        LDA     1,PLNK,2
 2447 175015        SNZ     3,3       ;END OF QUEUE ?
 2450    414        JMP     XQUE1     ;   YES
 2451  45436        STA     1,PLNK,3  ;   NO, LINK NEXT TO PREV
 2452  25034        LDA     1,NSTS,2
 2453 125014        SKZ     1,1       ;EXTRACTING FROM TASK QUEUE ...
 2454 125235        MOVZR#  1,1,SNR   ;... OR OTHER THAN SLEEP QUEUE ?
 2455    407        JMP     XQUE1     ;   YES
 2456  41034        STA     0,NSTS,2  ;NO:  SLEEP QUEUE.   SAVE A0
 2457  21012        LDA     0,PAUZ,2
 2460  25412        LDA     1,PAUZ,3
 2461 107000        ADD     0,1       ;ADD OUR PAUZ TO NEXT GUY'S
 2462  45412        STA     1,PAUZ,3
 2463  21034        LDA     0,NSTS,2  ;RECOVER A0
 2464 126000 XQUE1: ADC     1,1
 2465  45034        STA     1,NSTS,2  ;-1 ==> NOT ON ANY QUEUE
 2466 115000        MOV     0,3
 2467   1400        JMP     0,3       ;RETURN
```

```
;                        << SI = R90REXQUEMSA;  BO = 1/A.REXM5.7033! >>
;;-----------  ------------------------------------------  ------------------------
;;
;;     3.   AWAKE A NODE                                          <<RE-ENTRANT>>
;;
;;               CALLING SEQUENCE:
;;                       CALL
;;                       AWAKE (@25)
;;
;;               ENTRY:  A2 <-- A(NODE)
;;
;;               EXIT:   A2 <-- A(NODE)
;;                       NSTS(NODE) <-- 0
;;                       LINK(PREVIOUS-NODE) <-- A(NEXT-NODE)
;;                       PLNK(NEXT-NODE) <-- A(link(PREVIOUS-NODE))
;;                       GOES TO "QUEOA"
;;                       Interrupts are kept off until node is on Task queue
;;
;;               PURPOSE:
;;                       EXTRACT NODE FROM ANY QUEUE IT'S IN AND
;;                       PLACE IT ON THE SYSTEM TASK QUEUE
;;
;;               NOTE:   DOES **NOT** CHECK CONFLICT FLAG;  THEREFORE
;;                       CONFLICT-PRONE TASKS MUST NOT BE PUT ON SLEEP QUEUE
;;-- ---------  --------  ------------------------  ---  -------  ------------------------

2470   55014 AWAKX: STA     3,QRTN,2 ;awaken node at A2
2471    4745        JSR     XQUEX    ;extract the node from its queue (DISABLES INTERRUPTS)
2472   25005        LDA     1,CPRI,2
2473  125120        MOVZL   1,1      ;PRI*2 as used by QUE2
2474     654        JMP     QUEOA
```

```
;                         << SI = R9OREXQUEMSA; BO = 1/A.REXM5.7033! >>
;;------------------------------------------------------------------------
;;
;;   4.               Q L I N K
;;
;;   QUEUES A HIGHER PRIORITY TASK THAN THE CALLER, AND SETS ITS
;;   AUXL CELL TO POINT TO CALLER'S TASK CONTROL NODE.
;;   ALLOWS HIGH PRIORITY SUBROUTINES TO BE CALLED IN A STANDARD WAY.
;;   THE CALLED ROUTINE CAN ACCESS THE CALLER'S TCN FOR PARAMETER PASSING,
;;   AND HAS ITS OWN TCN FOR LOCAL VARIABLE STORAGE.
;;
;;   CALLING SEQUENCE:
;;           INTDS      <-- THIS IS REQUIRED ONLY FOR RE-ENTRANCY
;;           STA 3,.+5 (SHOULD USE A LABEL)
;;           JSR QLINK
;;           TASK         \
;;           TCB OR -1     } SAME DEFINITIONS
;;           PRIORITY      } AS FOR QUEUE
;;           A3           /
;;           (RETURN)
;;
;;   PRESERVES CPU STATUS (ALL 4 ACCUMULATORS, CARRY, PC FOR RETURN,
;;   SBA AND DBA) IN CALLER'S TCN. THEN CREATES THE NEW TCN, LINKS IT INTO
;;   THE TASK QUEUE AHEAD OF THE CALLER, SETS ITS TASK, TCB, AND PRIORITY
;;   APPROPRIATELY, SETS ITS AUXL TO POINT TO THE CALLER'S TCN, THEN FINALLY
;;   RESTORES ACCUMULATORS AND FALLS INTO THE NEW TASK.
;;
;;   THE CALLED TASK MAY ACCESS THE CALLER'S TCN TO RECEIVE AND RETURN
;;   PARAMETERS AS PER THE SPECIFICATION OF THE PARTICULAR CALL. WHEN IT
;;   IS DONE, IT MERELY DQUEUE'S ITSELF AND CONTROL WILL IN DUE COURSE
;;   RETURN TO THE CALLER.
;;
;;   INTERRUPTS ARE DISABLED FOR 102 MEMORY CYCLES (42 MICROSECONDS ON A
;;   POINT 4 CPU)
;;------------------------------------------------------------------------


    2475   60277 QLINK: INTDS
    2476   52015        STA    2,@TASKQ ;SAVE A2 IN CALLER'S TCN
    2477   30015        LDA    2,TASKQ
    2500   41002        STA    0,A0,2   ;ALSO A0
    2501   45001        STA    1,A1,2   ;AND A1
    2502   21403        LDA    0,3,3
    2503   41003        STA    0,A3,2   ;AND A3
    2504   20024        LDA    0,C4     ;CALCULATE PC FOR RETURN
    2505  117000        ADD    0,3
    2506   55004        STA    3,PC,2   ;SAVE PC FOR WHEN NEW TASK DQUEUE'S
    2507   20040        LDA    0,SBA
    2510   41006        STA    0,SSBA,2 ;ALSO PROTECT SBA
    2511   20041        LDA    0,DBA
    2512   41007        STA    0,SDBA,2 ;AND DBA
```

```
;                                << SI = R90REXQUEMSA; BO = 1/A.REXM5.7033! >>
    2513   6722          JSR     @.GTND      ;GET A NEW NODE
    2514  34015          LDA     3,TASKQ
    2515  50015          STA     2,TASKQ     ;LINK NEW NODE IN FRONT OF CALLER
    2516  55037          STA     3,LINK,2
    2517  21436          LDA     0,PLNK,3
    2520  41036          STA     0,PLNK,2
    2521  20436          LDA     0,CLINK
    2522 143000          ADD     2,0
    2523  41436          STA     0,PLNK,3
    2524 102460          SUBC    0,0         ;PRESERVE CARRY !
    2525  41034          STA     0,NSTS,2    ;MARK IT AS ON TASK QUEUE
    2526  55035          STA     3,AUXL,2    ;SET ITS AUXL TO CALLERS TCN
    2527  25405          LDA     1,CPRI,3    ;REMEMBER CALLER'S PRIORITY
    2530 125100          MOVL    1,1
    2531 125220          MOVZR   1,1         ;CLEAR MSB WITHOUT CHANGING CARRY
    2532  35404          LDA     3,PC,3      ;RECOVER CALLER'S RETURN ADDRESS
    2533  21774          LDA     0,-4,3
    2534  41011          STA     0,TASK,2    ;COPY THE TASK ADDRESS INTO NEW NODE
    2535  40421          STA     0,QLPC      ;SAVE IT TO FALL INTO
    2536  21775          LDA     0,-3,3      ;COPY TCB FROM CALL TABLE
    2537 101415          INC#    0,0,SNR     ;IS IT -1 ?
    2540  20007          LDA     0,RTP       ;   YES, USE REGNANT TASK'S POINTER
    2541  41010          STA     0,TCBP,2    ;ENTER INTO THE NEW NODE
    2542  21776          LDA     0,-2,3      ;COPY PRIORITY FROM CALL TABLE
    2543  35035          LDA     3,AUXL,2    ;RECOVER CALLER'S TCN
    2544 106032          SGE     0,1         ;IS GIVEN PRIORITY >= CALLER'S ?
    2545    413          JMP     QLNKF       ;   NO, ILLEGAL PRIORITY !?
    2546 103200          ADDR    0,0         ;FEED IN CARRY --> MSB
    2547  41005          STA     0,CPRI,2
    2550  21402          LDA     0,A0,3      ;RESTORE ACCUMULATORS
    2551  25401          LDA     1,A1,3
    2552  31400          LDA     2,A2,3
    2553  35403          LDA     3,A3,3
    2554  6017?          INTEN
    2555  2401           JMP     @.+1        ;FALL INTO THE NEWLY QUEUE'D TASK
    2556     0 QLPC:     0

    2557    37 CLINK:    LINK

    2560 21403 QLNKF:    LDA     0,A3,3      ;PICK UP CALLER'S RETURN ADDRESS
    2561  4601           JSR     QUEF2       ;GO TO QUEUE'S TRAPFAULT
```

```
;                       << SI = R9OREXQUEMSA; BO = 1/A.REXM5.7033! >>
;;----------------------------------------------------------------------
;;
;;      5.   DEQUEUE REGNANT TASK                  <<RE-ENTRANT>>
;;
;;           ENTRY:  TASKQ <-- A(CURRENT TASK NODE)
;;                   LINK(NODE) <-- A(NEXT TASK NODE)
;;                   PLNK(NODE) <-- A(PREVIOUS TASK NODE)
;;
;;           EXIT:   GOES TO "INTEX" IN INTERRUPT SERVICE MODULE SO IT
;;                   CAN DROP THRU TO NEXT TASK
;;
;;           NOTE:   DQUE ACTUALLY PERFORMS A "FREENODE" WHICH IS A
;;                   MEMORY MANAGEMENT OPERATION.  IT DOESN'T CALL FREENODE
;;                   BECAUSE WE NEED TO KEEP INTERRUPTS DISABLED.  IT ALSO
;;                   PERFORMS AN "XQUEUE" IN THE SAME MANNER, WITHOUT
;;                   USING THE "CALL" MECHANISM.
;;
;;           PURPOSE:  TO PROCEED TO THE NEXT HIGHEST PRIORITY TASK IN AN
;;                   ORDERLY FASHION WHILE RETURNING THE PREVIOUSLY
;;                   REGNANT TASK NODE TO THE FREE NODE CHAIN.
;;----------------------------------------------------------------------

    2562  60277 DQUE:  INTDS                ;REMOVE REGNANT TASK FROM QUEUE
    2563  30015        LDA     2,TASKQ
    2564  25037        LDA     1,LINK,2
    2565 125015        SNZ     1,1          ;SCHEDULER NODE ?
    2566    404        JMP     DQUET        ;   YES, OUCH !?
    2567   4647        JSR     XQUEX        ;EXTRACT NODE FROM TASKQ
    2570   6405        JSR     @.FRND       ;RETURN NODE TO FREE CHAIN
    2571   2405        JMP     @.INTX       ;FALL INTO NEXT TASK (GOTO INTEX)

    2572 161000 DQUET:MOV     3,0          ;   TRYING TO DQUE THE SCHEDULER NODE
    2573   6142        TRAPFAULT
    2574 117410        37*K!NOP

    2575   2707 .FRND:FRNOD                ;*** REF *** (MEMM)
    2576  12165 .INTX:INTEX                ;*** REF *** (INTS)
```

```
;                        << SI = R90REXQUEMSA;  BO = 1/A.REXM5.7033! >>
;;-------------------------------------------------------------------------------
;;     6.   SLEEP                                  <<RE-ENTRANT>>
;;
;;
;;          ENTRY:   A2 <-- A(NODE)
;;                   A0 <-- PAUSE VALUE (DELTA IN TENTHS OF SECONDS)
;;                          ANY VALUE IS LEGAL; 0 WILL WAKE UP AT NEXT TENHZ TICK
;;
;;          EXIT:    A2 <-- A(NODE)
;;                   NSTS(NODE) <-- 1 "SLEEPING"
;;                   IF (NODE WAS REGNANT) THEN FALLS INTO NEXT TASK BY
;;                   GOING TO INTEX.
;;
;;          PURPOSE:   EXTRACT NODE FROM WHATEVER QUEUE IT IS ON, AND
;;                   PLACE IT ON THE SLEEP QUEUE, IN ASCENDING ORDER BY WAKE
;;                   UP TIME.  EACH NODE'S PAUZ CELL HOLDS AN INCREMENTAL PAUSE
;;                   VALUE SO THAT TENHZ TASK NEED ONLY UPDATE THE FIRST NODE
;;
;;          NOTE:    A PROCESSOR MAY NOT PUT ITSELF TO SLEEP !!
;;
;;          NOTE 2:  DO NOT PUT CONFLICT-TYPE TASKS ON SLEEP QUEUE !!
;;                   (AWAKE DOESN'T CHECK CONFLICT FLAGS).
;;-------------------------------------------------------------------------------

 2577   55014 SLEPX: STA     3,QRTN,2 ;PUT NODE AT (A2) ON SLEEP QUEUE
 2600   24461        LDA     1,STSCH
 2601  132415        SNE     1,2      ;TRYING TO PUT SCHEDULER TO SLEEP ?
 2602    6142        TRAPFAULT        ;   YES, ILLEGAL !!
 2603  120410        41*K!NOP
 2604   34453        LDA     3,SLPQL  ;FIND POSITION IN SLEEP QUEUE
 2605   60277        INTDS
 2606   55013 SLEP1: STA     3,QTMP,2 ;PRESERVE THE NODE POINTER
 2607   35437        LDA     3,LINK,3 ;POINTER TO NEXT NODE
 2610  175015        SNZ     3,3      ;END OF QUEUE ?
 2611     407        JMP     SLEP2    ;   YES
 2612   25412        LDA     1,PAUZ,3 ;PICK UP EXISTING PAUSE ON SLEEP QUEUE NODE
 2613  122422        SUBZ    1,0,SZC  ;CALCULATE REMAINING PAUSE -- UNDERFLOW ?
 2614     772        JMP     SLEP1    ;   NO, CHECK NEXT SLEEP NODE

                     ; *** NO CODE HERE ***
```

```
;                       << SI = R9OREXQUEMSA;  BO = 1/A.REXM5.7033! >>

; FOUND PLACE FOR NEW NODE -- LINK IT AHEAD OF (A3), BEHIND (QTMP)

   2615 123000          ADD     1,0         ;YES, RESTORE OUR LAST PAUSE INCREMENT
   2616 106400          SUB     0,1         ;ADJUST NEXT NODE'S PAUSE INCREMENT
   2617  45412          STA     1,PAUZ,3
   2620  41012 SLEP2:   STA     0,PAUZ,2
   2621  21034          LDA     0,NSTS,2    ;PLACE FOUND FOR THIS NODE
   2622 101112          SSP     0,0         ;IS IT IN A QUEUE ?
   2623    406          JMP     SLEP3       ;   NO
   2624  21036          LDA     0,PLNK,2    ;   YES, EXTRACT IT
   2625  35037          LDA     3,LINK,2
   2626 175014          SKZ     3,3         ;IS THERE A NEXT NODE ?
   2627  41436          STA     0,PLNK,3    ;   YES, LINK NEXT TO PREV
   2630  57036          STA     3,@PLNK,2   ;LINK PREV TO NEXT
   2631  35013 SLEP3:   LDA     3,QTMP,2    ;LINK NODE INTO SLEEP QUEUE
   2632  25437          LDA     1,LINK,3    ;A(NEXT NODE)
   2633  45037          STA     1,LINK,2    ;INTO LINK OF CURRENT NODE
   2634  24723          LDA     1,CLINK     ;OFFSET OF LINKWORD IN NODE
   2635 167000          ADD     3,1         ;A(LINK OF PREV NODE)
   2636  45036          STA     1,PLNK,2
   2637  51437          STA     2,LINK,3
   2640  35037          LDA     3,LINK,2
   2641  24716          LDA     1,CLINK     ;OFFSET OF LINKWORD IN NODE
   2642 147000          ADD     2,1
   2643 175014          SKZ     3,3         ;THIS NODE AT END OF QUEUE ?
   2644  45436          STA     1,PLNK,3    ;   NO
   2645 126520          SUBZL   1,1
   2646  45034          STA     1,NSTS,2    ;1 ==> ON SLEEP QUEUE
   2647  24015          LDA     1,TASKQ
   2650 146415          SNE     2,1         ;PUTTING SELF TO SLEEP?
   2651    403          JMP     SLEP4
   2652  60177          INTEN               ;   NO...
   2653   3014          JMP     @QRTN,2     ;   ...RETURN TO CALLER

   2654  35014 SLEP4:   LDA     3,QRTN,2    ;YES, SET PC
   2655  55004          STA     3,PC,2
   2656   2720          JMP     @.INTX      ;FALL INTO NEXT TASK

   2657   2621 SLPQL:   SLEPQ-LINK
   2660      0 SLEPQ:   0           ;SLEEP QUEUE HEAD POINTER
   2661   2062 STSCH:   SNODE


                .EOT  ;REX "QUEM" FOR "IRIS" RESIDENT EXECUTIVE R9.0
```

```
;                          << SI = R90REXMEMMSA; BO = 1/A.REXM5.7033! >>
; 5 JUL 86, RB.
;;--- ---- -- ------- ---------- -- ------------ ---------------------
;;
;;
;;      REX:   MEMORY / PARTITION MANAGEMENT
;;
;;
;;
;;      MAJOR COMPONENTS:
;;
;;           1.   FREENODE
;;           2.   GETNODE
;;           3.   LOADUSER
;;
;;
;;
;;      DEFINITIONS:
;;
;;           NODE          - An area of memory, 32 words, used for passing control
;;                           information and used for temporary storage.  Node
;;                           format is defined in DEFS.
;;
;;           PASSIVE FILE  - A file that is loaded into a processor's partition
;;                           (work space) for the purpose of being operated on
;;                           or executed.
;;
;;           ACTIVE FILE   - An area on disc that is used to store a processor's
;;                           partition between time slices.
;;
;;
;;
;;      UPDATE RECORD:
;;
;;           6-03-80 (GAD)  R8.0 GETNODE, FREENODE
;;           4-05-83 (DLF)  R8.2 Modular Fixed Partition LOADUSER
;;
;;
;;---------- - ---------------------- ----------------------------------
```

```
;                        << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>

;;-------------  ------ -- ------------- ------------------------------------
;;
;;    F R E E N O D E :                              << RE-ENTRANT>>
;;
;;    PURPOSE:   FREENODE SERVES THE DUAL PURPOSE OF ACQUIRING AND
;;               FREEING NODES FROM THE FREE-NODE-CHAIN.   IT WILL
;;               ALSO RELEASE ANY AUXILIARY NODES LINKED VIA THE 'AUXL' CELL.
;;
;;    TO ACQUIRE A FREE NODE:
;;
;;         ENTRY:   A2 <-- 0
;;
;;         EXIT:    A1 <-- UNCHANGED FROM ENTRY
;;                  A2 <-- A (GOTTEN NODE)
;;                  AUXL(NODE) <-- 0
;;                  NSTS(NODE) <-- -1 << NOT ON ANY CHAIN OR QUEUE >>
;;                  LINK(NODE) <-- 0
;;                  PLNK(NODE) <-- 0
;;
;;     TO FREE A NODE:
;;
;;         ENTRY:   A2 <-- A (NODE TO BE FREED)
;;                  AUXL(NODE) <-- 0 OR A (AUXILIARY NODE)
;;                  NSTS(NODE) <-- -1 <<NOT ON ANY CHAIN OR QUEUE >>
;;
;;         EXIT:    A0 <-- CONTENTS OF A0 CELL IN PRIMARY NODE BEING FREED
;;                  A1 <-- SAME AS IT WAS UPON ENTRY
;;                  NSTS(NODE) <-- -2 <<ON FREE CHAIN>>
;;                  LINK(NODE) <-- LINK TO NEXT NODE ON FREE CHAIN
;;
;;-------------- - ---- --- ------------------ ----------------------------

2662       0 FRERT: 0                    ;RETURN ADDRESS
2663     660 NODES: INFO+FNOD.           ;NODE COUNTING CELL
2664       0 FREE:  0                    ;FREE NODE CHAIN POINTER (SET BY SIR)

2665   6347/ FREEX: SKPBN   CPU          ;INTERRUPTS ON ?
2666     412         JMP     FREX        ;   NO
2667   6027/         INTDS               ;YES, DISABLE INTERRUPTS
2670   547/2         STA     3,FRERT     ;SAVE RETURN
2671  151015         SNZ     2,2         ;GETNODE REQUESTED?
2672     403         JMP     FREEG
2673    4414         JSR     FRNOD       ;   NO, FREE THIS NODE
2674     402         SKIP
2675    4437 FREEG:  JSR     GTNOD       ;   YES GET A FREE NODE
2676   6017/         INTEN               ;REENABLE INTERRUPTS
2677    2763         JMP     @FRERT      ;RETURN

2700   54762 FREX:   STA     3,FRERT     ;SAVE RETURN
2701  151015         SNZ     2,2         ;GETNODE REQUESTED?
2702     403         JMP     FREXG
2703    4404         JSR     FRNOD       ;   NO, FREE THIS NODE
2704    2756         JMP     @FRERT
2705    4427 FREXG:  JSR     GTNOD       ;   YES GET A FREE NODE
2706    2754         JMP     @FRERT      ;RETURN
```

```
;                        << SI = R90REXMEMM3A;  B0 = 1/A.REXM5.7033! >>

;;--- -------- ------------------- --------- ---------- --------------------
;;
;;    1.   RETURN A NODE TO THE FREE CHAIN
;;
;;         ENTRY:    INTERRUPTS MUST BE DISABLED
;;                   A2 <-- A(NODE TO BE FREED)
;;                   AUXL(NODE) <-- 0 OR A(AUXILIARY NODE TO BE FREED)
;;                   NSTS(NODE) <-- -1 "NOT ON ANY CHAIN"
;;
;;          EXIT:    A0 <-- CONTENTS OF A0 CELL IN PRIMARY NODE FREED
;;                   A1 <-- SAME AS UPON ENTRY
;;                   NSTS(NODE) <-- -2 "ON FREE CHAIN"
;;                   LINK(NODE) <-- LINK TO NEXT NODE ON FREE CHAIN
;;                   AUXL(NODE) <-- 0
;;
;;       PURPOSE:   THIS ROUTINE PERFORMS ACTUAL CHAINING OF A NODE
;;                  BACK ONTO THE FREE CHAIN.  THIS SHOULD BE THE ONLY
;;                  ROUTINE WHICH PERFORMS SUCH A FUNCTION.  IF THE AUXL CELL
;;                  IS NON-ZERO, FRNOD WILL PLACE THAT NODE ON THE FREECHAIN
;;                  AS WELL, ETC.
;;
;;       NOTE:  IF WE RUN TEMPORARILY OUT OF FREE NODES, FREE = 0.  THEN
;;                  WHEN A NODE IS RETURNED, FRNOD WILL CLOBBER LOC.  0.
;;                  IF A TRAP AT 0 OCCURRED BEFORE NEXT INTERRUPT IT WOULD'T WORK
;;--- -------- ------------------- --------- ---------- --------------------


2707   52755 FRNOD: STA     2,@FREE   ;save A2 in first free node
2710   21034 FRNO2: LDA     0,NSTS,2
2711   12752        ISZ     @NODES    ;COUNT UP ONE MORE FREE NODE AVAILABLE
2712  101414        INC#    0,0,SZR   ;is given node's status = -1 = LOOSE ?
2713    6142        TRAPFAULT         ;   no, illegal !!
2714  121010        42*K!NOP
2715   15034        DSZ     NSTS,2    ;yes, make it -2 = FREE
2716   21035        LDA     0,AUXL,2
2717  101015        SNZ     0,0       ;end of chain ?
2720     406        JMP     FRNO3     ;   yes
2721   41037        STA     0,LINK,2  ;no, link to next
2722  102400        SUB     0,0
2723   41035        STA     0,AUXL,2  ;clear AUXL
2724   31037        LDA     2,LINK,2  ;process next node
2725     763        JMP     FRNO2

2726   20736 FRNO3: LDA     0,FREE    ;end of chain of nodes being freed
2727   41037        STA     0,LINK,2  ;link ahead of existing free nodes
2730   32734        LDA     2,@FREE   ;recover given A2
2731   50733        STA     2,FREE    ;make it the head of FREE chain
2732   21002        LDA     0,A0,2    ;restore A0
2733    1400        JMP     0,3       ;return
```

```
;                       << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>

;;-------- - ------- -------- ------------------------ ----- -----------------
;;
;;    2.   GET A FREE NODE FROM CHAIN
;;
;;     NOTE:   THIS ROUTINE IS NOT RE-ENTRANT, BUT INTERRUPTS MUST BE
;;           DISABLED FOR THE DURATION, SO IT MAY BE CALLED FROM AN INTERRUPT
;;           HANDLER.
;;
;;        ENTRY:   A3 <-- RETURN ADDRESS
;;                 INTERRUPTS MUST BE DISABLED
;;
;;        EXIT:    A2 <-- A(NODE)
;;                 NSTS(NODE) <-- -1   "NOT ON ANY CHAIN OR QUEUE"
;;                 AUXL(NODE) <-- 0 (THIS IS DONE BY FRNOD)
;;                 A2(NODE) <-- CONTENTS OF A2 AT ENTRY
;;
;;                 A0,A1 <-- SAME AS ENTRY
;;                 INTERRUPTS ARE KEPT DISABLED
;;
;;        PURPOSE:   TO EXTRACT A NODE FROM THE FREE CHAIN
;;
;;        NOTE:   TIME IN THIS ROUTINE IS 20 MACHINE CYCLES.  INTERRUPTS
;;                ARE DISABLED FOR THAT PERIOD.
;;-----------  ---- ------------------ ----- -------- ----------------------

2734   52730 GTNOD: STA     2,@FREE   ;protect (A2) in node's A2 cell
2735   30727        LDA     2,FREE    ;A(TOP FREE NODE)
2736   16725        DSZ     @NODES    ;ONE LESS FREE NODE AVAILABLE
2737  151015        SNZ     2,2       ;ANY FREE NODES LEFT ?
2740     414        JMP     NFREE     ;   NO, NONE
2741   55003        STA     3,A3,2    ;PROTECT (A3)
2742   41002        STA     0,A0,2    ;SAVE A0
2743   36720        LDA     3,@NODES  ;NUMBER FREE NODES NOT IN USE
2744   20412        LDA     0,MINOD   ;LOWEST NUMBER OF UNUSED FREE NODES SINCE LAST IPL
2745  116033        SLS     0,3       ;HAVE LESS UNUSED NODES NOW THAN EVER BEFORE ?
2746   54410        STA     3,MINOD   ;   YES, SAVE NEW LOW NODE USAGE
2747   21002        LDA     0,A0,2    ;RESTORE A0
2750   35037        LDA     3,LINK,2  ;A(NEXT NODE)
2751   54713        STA     3,FREE    ;CUT AWAY THE NODE
2752   11034        ISZ     NSTS,2    ;set status = loose
2753    3003        JMP     @A3,2     ;RETURN

2754   73377 NFREE: 73377   ;"NO FREE NODES" HALT
2755     77/        JMP     .-1       ;CATCH FOR NO FREE NODES.

2756 1777// MINOD: -1        ;NUMBER OF FREE NODES AVAILABLE THAT HAVE NOT BEEN USED SINCE LAST IPL
```

<< SI = R9OREXMEMMSA; BO = 1/A.REXM5.7033! >>

3. LOADUSER

System call to load processor storage when a job begins or resumes.

It has two functions:

1. Partition Management

   a. assign partition (workspace)
   b. load partition

2. Page Zero Management

   a. restore processor page zero
   b. restore DA cells

```
;                    << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>
;
;          LOADUSER CALLING SEQUENCE
;
;               CALL
;               LOADUSER
;               < Types Don't Match > Rtn
;               < Types Match > Rtn
;
;               AC0    AC1
;
;                x      -1        ALLOCATE FREE PARTITION
;
;                                 1. Assign Part'n
;                                 2. Set AF Type = Proc Type
;                                 3. Types Match Rtn
;
;                x       0        LOAD ACTIVE FILE, CHECK TYPE
;
;                                 1. Assign Part'n
;
;                                 If AF Type = Proc Type
;                                 Then
;                                 2. Load Active File (if not in memory)
;                                 3. Load Processor Page Zero (if any)
;                                 4. Load DA Cells
;                                 5. Types Match Rtn
;                                 Else
;                                 2. Types Don't Match Rtn
;
;
;                x       1        LOAD ACTIVE FILE, DON'T CHECK TYPE
;
;                                 1. Assign Part'n
;
;                                 If AF Type <> 37 (Scratched)
;                                 Then
;                                 2. Load Active File (if not in memory)
;                                 3. Load Processor Page Zero (if any)
;                                 4. Load DA Cells
;                                 5. Types Match Rtn
;                                 Else
;                                 2. Types Don't Match Rtn
;
;               LU     RDA       LOAD PASSIVE FILE (AC1 > 1)
;
;                                 1. Assign Part'n
;
;                                 If PF Type    1<type<20
;                                 Then
;                                 2. Load Passive File
;                                 3. Set AF Type = Proc Type
;                                 4. Types Match Rtn
;                                 Else
;                                 2. Types Don't Match Rtn
```

```
;                        << SI = R90REXMEMMSA;  BO = 1/A.REXM5.7033! >>
;
;           LOADUSER EXIT CONDITIONS
;
;                For both 'Types Match' and 'Types Don't Match' returns
;
;                AC0 - Active File Type
;                AC1 - Processor File Type
;                AC2 - Partition Address
;                AC3 - Partition Size (Words)
;
;                PIB - Partition Information Block Address
;                SBA - Partition Address
;                DBA - Partition Address
;
;
;                PART'N [ AFP.  ] - Active File Header Buffer Address
;                PART'N [ JCP.  ] - Job Control Block Address (PCB)
;                AFH [ CORA ]     - Partition Address
;                PCB [ DFT [ CHM1 [ CBN ]]] - Partition Information Block Address
;
;                If another job's partition is written out to its Active File
;                then for that job:
;
;                PCB [ DFT [ CHM1 [ CBN ]]] - 0
;
;
;
;
;           ABNORMAL TERMINATIONS
;
;
;                TRAP 43 - "File larger than partition"
;                TRAP 44 - "Unable to assign partition"
;                TRAP 47 - "Partition Linkage Error, PIB [JCP.] <> RUP"
;                TRAP 54 - "Partition Linkage Error, PCB [AHA.] <> PIB [AFP.[DHDR]]"
```

```
;                        << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>
;
;            PARTITION TABLE
;
;
;            The PARTITION INFORMATION TABLE is external to the partition area and
;            resides in lower memory.  It contains a single four word HEADER plus
;            a number of six word PARTITION INFORMATION BLOCKs, one for
;            each partition.  (See DEFS).
;
;
;
;                                                                        NOTES:
;
;             Header    /    NPT.  = 0      Number of Partitions                    (1)
;                       (    MP0.  = 1      PSIZE
;                       (    MP1.  = 2      MTYPE
;                        \   MP2.  = 3      NO. PARTITIONS REQUESTED IN CONFIG
;
;
;
;                         /  PAD.  = 0      Partition Address                       (2)
;                       (    SZP.  = 1      Partition Size (words)
;             For       (    JCP.  = 2      Job Control Block Pointer  (a PCB)      (3)
;             Each      (    AFP.  = 3      Active File Header Buffer Address
;             Part'n    (    TLU.  = 4      Time Last Used (TSC.)
;                        \   PBN.  = 5      Physical Block Number                   (4)
;
;
;
;
;       (1) INFO [.UPT.] Points Here
;       (2) PIB Points Here
;       (3) Zero ==> Unused Part'n
;       (4) Used On Mapped Systems
```

```
;                       << SI = R90REXMEMMSA; BO = 1/A.REXM5.7033! >>
;           PARTITION SET UP PARAMETERS
;
;
;           The General Information Table in CONFIG contains the parameters
;           that SIR uses to allocate the partition area and create the
;           partition table.
;
;                   LOC     CONTENTS        COMMENT
;
;                   400     PSIZE           User partition size.
;                                           Range 400 to 100000 words.
;
;                   401     NPART           Number of user partitions.
;                                           Range 1 to 377
;
;                   402     MTYPE           Memory Type
;
;                                           0 - Standard
;                                           1 - Mapped
;                                           2 - Banked (not implimented)
;
;
;
;---------- ------- ---------------- --- ------------------------------

; SUBROUTINE ADDRESS TABLE (FIXED NEG. OFFSETS FROM LUSRX):

   2757   3223 .LCPA:LCPA    ;CHECK FOR CALLER'S PARTITION
   2760   3300 .LWAC:LWAC    ;WRITE ACTIVE FILE
   2761   3321 .LSET:LSET    ;SET UP PARTITION
   2762   3043 .LACT:LACT    ;LOAD ACTIVE FILE
   2763   3127 .LPAS:LPAS    ;LOAD PASSIVE FILE
   2764   3161 .LAFP:LAFP    ;ALLOCATE FREE PARTITION
   2765   3166 .LWUP:LWUP    ;WRAP UP
   2766   3241 .LSPT:LSPT    ;SELECT PARTITION
   2767   3034 .LSMM:LNOP    ;SELECT MEMORY
   2770   3041 .LTOTAL: LUTOTAL;ACTIVITY COUNTERS
   2771   3125 .LDREAD: LUDREAD
                ; ***** NO CODE HERE *****
```

```
;                       << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>

                    ; ***** NO CODE HERE *****

; EXECUTIVE

   2772   54443 LUSRX: STA     3,LCALR   ;SAVE RETURN
   2773   40443        STA     0,LCAL0   ;SAVE CALL PARAM 0
   2774   44443        STA     1,LCAL1   ;SAVE CALL PARAM 1
   2775   10445        ISZ     LUTOTAL+1;INCREMENT ACTIVITY CNTR
   2776     402        SKIP
   2777   10442        ISZ     LUTOTAL
   3000   20005        LDA     0,RUP
   3001    6756        JSR     @.LCPA    ;CHECK FOR CALLER'S PART'N
   3002   50436        STA     2,LRFLG   ;SET MEMORY RESIDENT FLAG
   3003  151014        SKZ     2,2       ;PART'N ASSIGNED ?
   3004     410        JMP     LUSR1     ;   YES, SELECT MEMORY
   3005    6761        JSR     @.LSPT    ;   NO, SELECT PART'N
   3006    6761        JSR     @.LSMM    ;SELECT MEMORY
   3007   21002        LDA     0,JCP.,2  ;JOB CONTROL BLOCK
   3010  101014        SKZ     0,0       ;PART'N IN USE ?
   3011    6747        JSR     @.LWAC    ;   YES, WRITE ACTIVE FILE
   3012    6747        JSR     @.LSET    ;SET UP PART'N
   3013     402        SKIP              ;
   3014    6753 LUSR1: JSR     @.LSMM    ;SELECT MEMORY
   3015   50004        STA     2,PIB     ;SET PART'N INFORMATION BLOCK ADDRESS
   3016   20420        LDA     0,LCAL0   ;CALL PARAM 0
   3017   24420        LDA     1,LCAL1   ;CALL PARAM 1
   3020   30420        LDA     2,LRFLG   ;MEMORY RESIDENT FLAG
   3021   34741        LDA     3,.LACT   ;A(LACT)
   3022  125235        MOVZR#  1,1,SNR   ;LOAD ACTIVE FILE ?
   3023     405        JMP     LUSR2     ;   YES
   3024   34737        LDA     3,.LPAS   ;A(LPAS)
   3025  125414        INC#    1,1,SZR   ;LOAD PASSIVE FILE ?
   3026     402        JMP     LUSR2     ;   YES
   3027   34735        LDA     3,.LAFP   ;ALLOCATE FREE PART'N !
   3030    5400 LUSR2: JSR     0,3       ;EXECUTE CALL REQUEST
   3031   10404        ISZ     LCALR     ;TYPES MATCH RTN (SKIPPED IF NO MATCH)
   3032    6733        JSR     @.LWUP    ;WRAP UP
   3033    2402        JMP     @LCALR    ;RETURN

   3034    1400 LNOP:  JMP     0,3       ;NOP RETURN

   3035       0 LCALR: 0       ;CALL RETURN
   3036       0 LCAL0: 0       ;CALL PARAM 0 (AC0)
   3037       0 LCAL1: 0       ;CALL PARAM 1 (AC1)
   3040       0 LRFLG: 0       ;MEMORY RESIDENT FLAG (0 => LOAD AF)
   3041       0 LUTOTAL: 0     ;ACTIVITY COUNTER
   3042       0        0
```

```
;                        << SI = R9OREXMEMMSA; BO = 1/A.REXM5.7033! >>
; LOAD ACTIVE FILE
;
;
;
; ENTRY:    From EXECUTIVE
;
;           Part'n assigned
;
;               PZ  [PIB]      - Part'n Information Block address
;               PIB [AFP.]     - Active File Header buffer address
;               PIB [JCP.]     - Job Control Block (PCB) address
;               AFH [CORA]     - Partition address
;               PCB [DFT [CHM1 [CBN]]] - Part'n Information Block address
;
;               AC0 - x
;               AC1 - Type check parameter  ( 0 => check types)
;               AC2 - Memory Resident flag  ( 0 => load Active File)
;               AC3 - Return
;
;
; EXIT:     To caller, nonskip => good completion
;
;               AC's - none
;
;           To caller, skip - error detected
;
;               AC's - none
;
;
;
; PURPOSE:
;
;           1. Load Active File (if requested, i.e. AC2 = 0)
;           2. Restore DA cells from Active File header DA save
;              area (DASA thru DASA+5)
;           3. Restore processor page zero from Active File header PZ save
;              area (FMAP), DASA+6 contains page zero load address (0 => no saved PZ)
;                         , DASA+7 contains page zero size (words)
;
;
;
; ERROR RETURNS:
;
;           1. Active File is type 37 (scratched)
;           2. Type check requested and Active File type <> Processor type
;
;
;
; ABNORMAL TERMINATIONS:
;
;           TRAP 43 - FILE LARGER THAN PART'N
;
;               AC0 - File Size
;               AC1 - Part'n Size
```

```
;                       << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>

3043  54537 LACT:  STA     3,LSUBR   ;SAVE RETURN
3044  50537         STA     2,LSUBX   ;SAVE MEMORY RESIDENT FLAG
3045  34004         LDA     3,PIB     ;PART'N INFORMATION BLOCK ADDRESS
3046  31403         LDA     2,AFP.,3  ;ACTIVE FILE HEADER BUFFER
3047  21010         LDA     0,TYPE 2  ;TYPE WORD
3050  34043         LDA     3,C37     ;
3051 163400         AND     3,0       ;ACTIVE FILE TYPE
3052 162415         SNE     3,0       ;SCRATCHED FILE (TYPE 37) ?
3053    445         JMP     LACT4     ;   YES, ERROR
3054 125014         SKZ     1,1       ;CHECK TYPES ?
3055    404         JMP     LACT1     ;   NO, SKIP TYPE CHECK
3056  26526         LDA     1,@LCPTN  ;PROCESSOR TYPE
3057 106414         SEQ     0,1       ;ACTIVE FILE TYPE = PROCESSOR TYPE ?
3060    440         JMP     LACT4     ;   NO, ERROR
3061  20522 LACT1:  LDA     0,LSUBX   ;MEMORY RESIDENT FLAG
3062 101014         SKZ     0,0       ;MEMORY RESIDENT ?
3063    412         JMP     LACT2     ;   YES, RESTORE SAVE AREAS
3064   4523         JSR     LCFS      ;DOES ACTIVE FILE FIT INTO PART'N ?
3065   6142         TRAPFAULT         ;   NO, FILE LARGER THAN PART'N !
3066 121410         43*K!NOP
3067   6110         DATAPUMP          ;READ ACTIVE FILE
3070     22         DFILREAD          ;BYPASS BUFFER POOL
3071 100000         @0
3072  10434         ISZ     LUDREAD+1
3073    402         SKIP
3074  10431         ISZ     LUDREAD
3075  20426 LACT2:  LDA     0,LDASA   ;RESTORE DA
3076 143000         ADD     2,0       ;BEG SAVED DA
3077  24025         LDA     1,C5      ;DA SIZE -1
3100 107000         ADD     0,1       ;END SAVED DA
3101  30174         LDA     2,.DA     ;DA
3102   6101         CALL              ;MOVE SAVED DA INTO DA
3103 100015         MOVEWORDS
3104  30004         LDA     2,PIB     ;RESTORE PROCESSOR PAGE ZERO
3105  35003         LDA     3,AFP.,2  ;ACTIVE FILE HEADER BUFFER
3106  31446         LDA     2,DASA+6,3;PROCESSOR SAVE AREA DESTINATION ADDRESS
3107 151015         SNZ     2,2       ;ANY SAVE AREA ?
3110    407         JMP     LACT3     ;   NO, RETURN
3111  20413         LDA     0,LFMAP   ;DISPLACEMENT TO SAVED PROCESSOR AREA
3112 163000         ADD     3,0       ;BEG SAVED PROCESSOR AREA
3113  25447         LDA     1,DASA+7,3;SAVED AREA SIZE
3114 107000         ADD     0,1       ;END SAVED PROCESSOR AREA
3115   6101         CALL              ;MOVE SAVED PROCESSOR AREA INTO PROCESSOR
3116 100015         MOVEWORDS
3117   2463 LACT3:  JMP     @LSUBR    ;GOOD COMPLETION

3120  10462 LACT4:  ISZ     LSUBR     ;ERROR DETECTED
3121 100010         NOP               ;
3122    7/5         JMP     LACT3     ;RETURN
3123     40 LDASA:  DASA    ;DA SAVE AREA
3124     70 LFMAP:  FMAP    ;PROCESSOR SAVE AREA
3125      0 LUDREAD:0       ;
3126      0          0
```

```
;                        << SI = R9OREXMEMMSA; BO = 1/A.REXM5.7033! >>
; LOAD PASSIVE FILE
;
;
; ENTRY:    From EXECUTIVE
;
;           Part'n assigned
;
;               PZ  [PIB]      - Partition Information Block address
;               PIB [AFP.]     - Active File Header buffer address
;               PIB [JCP.]     - Job Control Block (PCB) address
;               AFH [CORA]     - Partition address
;               PCB [DFT [CHM1 [CBN]]] - Partition Information Block address
;
;               AC0 - LU
;               AC1 - RDA
;               AC2 - x
;               AC3 - Return
;
;
; EXIT:     To caller, nonskip - good completion
;
;               AC's - None
;
;           To caller, skip - error detected
;
;               AC's - None
;
;
; PURPOSE:
;
;           1. Load Passive file
;
;
; ERROR RETURNS:
;
;           1. Illegal Passive file type  (the legal range is  1<type<20)
;
;
; ABNORMAL TERMINATIONS:
;
;           TRAP 43 - FILE LARGER THAN PART'N
;
;               AC0 - File Size
;               AC1 - Part'n Size
```

```
;                       << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033!  >>

     3127  54453 LPAS:  STA     3,LSUBR   ;SAVE RETURN
     3130 152400         SUB     2,2
     3131   6110         DATAPUMP          ;GET PASSIVE FILE HEADER
     3132      0         GETBLOCK
     3133 100000         @0
     3134  21010         LDA     0,TYPE,2  ;TYPE WORD
     3135  24043         LDA     1,C37
     3136 123400         AND     1,0       ;TYPE
     3137  24042         LDA     1,C20
     3140 101234         MOVZR#  0,0,SZR   ;TYPE > 1 ..
     3141 106033         SLS     0,1       ;..AND < 20 ?
     3142    412         JMP     LPAS2     ;  NO, ERROR
     3143   4444         JSR     LCFS      ;DOES PASSIVE FILE FIT ?
     3144   6142         TRAPFAULT         ;  NO, FILE LARGER THAN PART'N !
     3145 121410         43*K!NOP
     3146  36004         LDA     3,@PIB    ;PART'N ADDRESS
     3147  55175         STA     3,CORA,2  ;SET MEMORY ADDRESS
     3150   6110         DATAPUMP          ;LOAD PASSIVE FILE
     3151 120012         FILREAD+UL+CF     ;USE POOL, UNLATCH HEADER
     3152 100000         @0
     3153   2427 LPAS1: JMP     @LSUBR    ;RETURN

     3154   6110 LPAS2: DATAPUMP          ;RELEASE HEADER
     3155      3         UNLATCH
     3156 100000         @0
     3157  10423         ISZ     LSUBR     ;ERROR DETECTED
     3160    773         JMP     LPAS1     ;RETURN
```

```
;                       << SI = R9OREXMEMMSA; BO = 1/A.REXM5.7033! >>
; ALLOCATE FREE PARTITION
;
;
;
; ENTRY:    From EXECUTIVE
;
;           Partition assigned
;
;               PZ  [PIB]      - Partition Information Block address
;               PIB [AFP.]     - Active File Header buffer address
;               PIB [JCP.]     - Job Control Block (PCB) address
;               AFH [CORA]     - Partition address
;               PCB [DFT [CHM1 [CBN]]] - Partition Information Block address
;
;               AC3 - Return
;
;
;
; EXIT:     To caller, nonskip
;
;               AC's - None
;
;
;
; PURPOSE:  Set Active File type equal to Processor type


   3161  30004 LAFP: LDA     2,PIB     ;PART'N INFORMATION BLOCK ADDRESS
   3162  31003       LDA     2,AFP.,2  ;ACTIVE FILE HEADER BUFFER
   3163  22421       LDA     0,@LCPTN  ;CURRENT PROCESSOR TYPE NUMBER
   3164  41010       STA     0,TYPE,2  ;SET ACTIVE FILE TYPE
   3165   1400       JMP     0,3       ;RETURN
```

```
;                       << SI = R90REXMEMMSA; BO = 1/A.REXM5.7033! >>

;  WRAP UP
;
;
;  ENTRY:       From EXECUTIVE
;
;               Part'n assigned
;
;                    PZ  [PIB]      - Partition Information Block address
;                    PIB [AFP.]     - Active File Header buffer address
;                    PIB [JCP.]     - Job Control Block (PCB) address
;                    AFH [CORA]     - Partition address
;                    PCB [DFT [CHM1 [CBN]]] - Partition Information Block address
;
;                    AC3 - Return
;
;
;  EXIT:        To caller, nonskip
;
;                    AC0 - Active File or Passive File Type
;                    AC1 - Processor Type
;                    AC2 - Partition Address
;                    AC3 - Partition size (words)
;
;                    SBA - Partition Address
;                    DBA - Partition Address
;
;

;PURPOSE:       Set Reg's for exit


   3166   54414 LWUP:  STA     3,LSUBR   ;SAVE RETURN
   3167   30004        LDA     2,PIB     ;PART'N INFORMATION BLOCK ADDRESS
   3170   35003        LDA     3,AFP.,2  ;ACTIVE FILE HEADER
   3171   21410        LDA     0,TYPE,3  ;TYPE WORD
   3172   24043        LDA     1,C37     ;
   3173  123400        AND     1,0       ;ACTIVE FILE TYPE
   3174   26410        LDA     1,@LCPTN  ;CURRENT PROCESSOR TYPE
   3175   35001        LDA     3,SZP.,2  ;PART'N SIZE
   3176   32004        LDA     2,@PIB    ;PART'N ADDRESS
   3177   50040        STA     2,SBA     ;SOURCE BYTE ADDRESS
   3200   50041        STA     2,DBA     ;DESTINATION BYTE ADDRESS
   3201    2401        JMP     @LSUBR    ;RETURN

;  GLOBAL

   3202       0 LSUBR: 0                 ;SUBROUTINE RETURN
   3203       0 LSUBX: 0                 ;SUBROUTINE STORAGE

   3204     641 LCPTN: INFO+CPTN.        ;CURRENT PROCESSOR TYPE NUMBER
   3205     672 LUPT.: INFO+.UPT.        ;PARTITION TABLE POINTER
```

```
;                       << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>

; CHECK FILE SIZE
;
;
;
; ENTRY:      From LOAD ACTIVE FILE
;                  LOAD PASSIVE FILE
;
;                  AC2 - Header buffer address
;                  AC3 - Return
;
;
;             To caller, nonskip - file larger than partition
;
;                  AC0 - File size
;                  AC1 - Partition size
;                  AC2 - Header Buffer address
;
;
;             To caller, skip - file fits into partition
;
;                  AC2 - Header Buffer address
;
;
;
; PURPOSE:    Verify that file being loaded fits into partition


  3206      0        0                   ;RETURN

  3207  54777 LCFS:  STA     3,.-1       ;SAVE RETURN
  3210  21173         LDA     0,CSIZ,2    ;FILE SIZE
  3211  24064         LDA     1,C377      ;ROUND UP MODULO 400
  3212 123000         ADD     1,0         ;
  3213 124000         COM     1,1         ;
  3214 123400         AND     1,0         ;FILE SIZE
  3215  34004         LDA     3,PIB       ;PART'N INFORMATION BLOCK ADDRESS
  3216  25401         LDA     1,SZP.,3    ;PART'N SIZE
  3217 106432         SGR     0,1         ;DOES IT FIT ?
  3220  10766         ISZ     LCFS-1      ;   YES, GOOD RETURN
  3221 100010         NOP                 ;   NO, ERROR RETURN
  3222   2764         JMP     @LCFS-1     ;RETURN
```

```
;                        << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>
;  CHECK FOR CALLER'S PARTITION
;
;
;  ENTRY:     From EXECUTIVE
;
;                 AC0 - CALLER'S PCB
;
;
;  EXIT:      To caller, nonskip
;
;                 AC2 - 0 ==> Partition not assigned
;                 AC2 - P ==> Partition Information Block address
;
;
;  PURPOSE:   Determine if caller has a partition


   3223   32762 LCPA:  LDA     2,@LUPT.
   3224   25000         LDA     1,NPT.,2 ;NUMBER OF PARTITIONS TO CHECK
   3225   44756         STA     1,LSUBX
   3226   24450         LDA     1,LPIHS  ;SIZE OF PIT HEADER
   3227  133000         ADD     1,2      ;CALC.  FIRST PIB ADDRESS
         3230 LCPALOOP:
   3230   25002         LDA     1,JCP.,2
   3231  106415         SNE     0,1      ;DOES THIS PARTITION BELONG TO CALLER ?
   3232    1400         JMP     0,3      ;   YES, RETURN WITH A2 --> PIB
   3233   24441         LDA     1,LPIBS  ;NO, GET PIB SIZE
   3234  133000         ADD     1,2      ;CALC.  NEXT PIB ADDRESS
   3235   14746         DSZ     LSUBX    ;ANY MORE PIB'S LEFT ?
   3236     772         JMP     LCPALOOP ;   YES, REPEAT LOOP
   3237  152400         SUB     2,2      ;NO, USER DOES NOT HAVE A PARTITION
   3240    1400         JMP     0,3      ;RETURN WITH (A2) = 0
```

```
;                        << SI = R9OREXMEMMSA; BO = 1/A.REXM5.7033! >>
; SELECT PARTITION
;
;
; ENTRY:      From EXECUTIVE
;
;                  AC3 - Return
;
;
; EXIT:       To caller, nonskip
;
;                  AC2 - Partition Information Block address
;
; PURPOSE:  Select partition for use by call using the following prece:
;
;              1. Unused partition
;              2. Partition (Job) with highest CTC value

; ABNORMAL TERMINATIONS :
;
;      TRAP 44 -  "LOADU unable to select partition" !
;
;              AC0 - x
;              AC1 - x
;              AC2 - x
```

```
;                    << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>

3241   54741 LSPT:  STA     3,LSUBR   ;SAVE RETURN
3242  102400        SUB     0,0       ;INIT TO ZERO
3243   40430        STA     0,LPART   ;SELECTED PARTITION'S PIB ADDRESS
3244   40430        STA     0,LRELP   ;ASSOCIATED JOB'S RELATIVE PRIORITY
3245   32740        LDA     2,@LUPT.  ;PART'N TABLE ADDRESS
3246   21000        LDA     0,NPT.,2  ;NUMBER OF PART'Ns
3247   40426        STA     0,LPCTR   ;SET A COUNTER
3250   20426        LDA     0,LPIHS   ;SIZE OF PART'N TABLE HEADER
3251  113000 LSPT1: ADD     0,2       ;PIB ADDRESS
3252   35002        LDA     3,JCP.,2  ;PCB ADDRESS
3253  175015        SNZ     3,3       ;PART'N IN USE ?
3254     413        JMP     LSPT3     ;   NO, SELECT THIS PART'N
3255   21422        LDA     0,CTC.,3  ;CURRENT JOB'S RELATIVE PRIORITY
3256   24416        LDA     1,LRELP   ;PREVIOUS JOB'S RELATIVE PRIORITY
3257  106032        SGE     0,1       ;CURRENT JOB? >= PREVIOUS JOB
3260     403        JMP     LSPT2     ;   NO, NEXT PART'N
3261   40413        STA     0,LRELP   ;   YES, SELECT PART'N
3262   50411        STA     2,LPART   ;
3263   20414 LSPT2: LDA     0,LPIBS   ;PIB SIZE
3264   14411        DSZ     LPCTR     ;SCANNED ALL PART'Ns?
3265     764        JMP     LSPT1     ;   NO, NEXT PART'N
3266   30405        LDA     2,LPART   ;SELECTED PARTITION'S PIB ADDRESS
3267  151015 LSPT3: SNZ     2,2       ;PART'N SELECTED ?
3270    6142        TRAPFAULT         ;   NO,"LOADUSER UNABLE TO SELECT PARTITION" !
3271  122010        44*K!NOP          ;
3272    2710        JMP     @LSUBR    ;RETURN

3273       0 LPART: 0       ;SELECTED PARTITION'S PIB ADDRESS
3274       0 LRELP: 0       ;ASSOCIATED JOB'S RELATIVE PRIORITY
3275       0 LPCTR: 0       ;PART'N COUNTER

3276       4 LPIHS: PIHS    ;TABLE HEADER SIZE
3277       6 LPIBS: PIBS    ;PIB SIZE
```

```
;                        << SI = R9OREXMEMMSA;  BO = 1/A.REXM5.7033! >>

; WRITE ACTIVE FILE
;
; ENTRY:       From EXECUTIVE
;
;                  AC2 - Partition Information Block address
;                  AC3 - Return
;
; EXIT:        To caller, nonskip
;
;                  AC2 - Partition Information Block address
;
; PURPOSE:     1. Write a partition to its Active File
;              2. Release that partition's Active File Header
;              3. Set partition to an unused state    ( PIB [JCP.] = 0 => unused )
;
;                  PIB (JCP (DFT (CHM1 (CBN))))   <--   0
;                  PIB (JCP)  <--  0
;
; ABNORMAL TERMINATIONS:
;
;              TRAP 54 - Partition linkage error, PCB [AHA.] <> PIB[AFP.[DHDR]]
;
;                  AC0 - Active File Header RDA, HEADER [ DHDR ]
;                  AC1 - Active File Header RDA, PCB [ AHA.]
;                  AC2 - Active File Header Buffer


   3300  54702 LWAC:  STA     3,LSUBR   ;SAVE RETURN
   3301  50702         STA     2,LSUBX   ;SAVE PART'N INFORMATION BLOCK ADDRESS
   3302  35002         LDA     3,JCP.,2  ;JOB CONTROL BLOCK (PCB) ADDRESS
   3303  31003         LDA     2,AFP.,2  ;ACTIVE FILE HEADER
   3304  21177         LDA     0,DHDR,2  ;HEADER RDA
   3305  25427         LDA     1,AHA.,3  ;HEADER RDA
   3306 106414         SEQ     0,1       ;AGREE?
   3307   6142         TRAPFAULT         ;   NO, PART'N LINKAGE ERROR !
   3310 126010         54*K!NOP
   3311   6110         DATAPUMP          ;WRITE PART'N TO ACTIVE FILE
   3312 100023         DFILWRITE+UL      ;BYPASS POOL, UNLATCH AND WRITE HEADER
   3313 100000         @0
   3314  30667         LDA     2,LSUBX   ;PART'N INFORMATION BLOCK ADDRESS
   3315 102400         SUB     0,0
   3316  41002         STA     0,JCP.,2
   3317   2663         JMP     @LSUBR    ;RETURN

   3320 1777/0 VCHM1:CHM1
```

```
;                       << SI = R90REXMEMMSA;  BO = 1/A.REXM5.7033! >>
;  SET UP PARTITION
;
;
;  ENTRY:       From EXECUTIVE
;
;                   AC2 - Partition Information Block address
;                   AC3 - Return
;
;
;  EXIT:        To caller, nonskip
;
;                   AC2 - Partition Information Block address
;
;
;  PURPOSE:     Set up new partition
;
;               1. Read and latch Active File Header
;
;               2. Set up Partition Information Block
;                  a. PIB (AFP.) <-- Active File Header Buffer address
;                  b. PIB (JCP.) <-- Port Control Block address (RUP)
;
;               3. Set up Active File Header
;                  a. HDR (CORA)     <-- Partition Address
;
;               4. Set up Port Control Block
;                  a. PCB ( DFT ( CHM1 ( CBN )))   <-- Partition Information Block address



 3321   54661 LSET:  STA     3,LSUBR   ;SAVE RETURN
 3322   50661        STA     2,LSUBX   ;SAVE PART'N INFORMATION BLOCK ADDRESS
 3323  102400        SUB     0,0       ;LU 0
 3324   30005        LDA     2,RUP
 3325   25027        LDA     1,AHA.,2  ;ACTIVE FILE HEADER RDA
 3326    6110        DATAPUMP          ;GET ACTIVE FILE HEADER
 3327       0        GETBLOCK
 3330  100000        @0
 3331   34652        LDA     3,LSUBX   ;PART'N INFORMATION BLOCK ADDRESS
 3332   51403        STA     2,AFP.,3  ;ACTIVE FILE HEADER BUFFER ADDRESS
 3333   21400        LDA     0,PAD.,3  ;PART'N ADDRESS
 3334   41175        STA     0,CORA,2  ;SET IT
 3335   30005        LDA     2,RUP
 3336   51402        STA     2,JCP.,3  ;JOB CONTROL BLOCK
 3337  171000        MOV     3,2       ;PART'N INFORMATION BLOCK ADDRESS
 3340    2642        JMP     @LSUBR    ;RETURN


                .EOT  ;REX "MEMM" FOR "IRIS" RESIDENT EXECUTIVE
```

```
;                          << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>
;  13 SEP 86, RB.

;;----------- ------------------------------------------------------------
;;
;;                              D A T A P U M P
;;
;;       CALLING SEQUENCE:
;;             A0        A1        A2        USED BY:
;;           - ----    ------    ------    ----------------------
;;             LU        RDA      A(BUF)    PUTBLOCK, (Q)READ, (Q)WRITE
;;             LU        RDA         X      GETBLOCK
;;              X          X      A(BUF)    UNLATCH, SETDIRTY, PUBLIC
;;              X          X      A(BUF)    FILEREAD, FILEWRITE, BUFFLUSH, FILEFLUSH
;;             LU          X         X      LUFLUSH
;;              X          X         X      GETPRIVATE, LRUFLUSH, ALLFLUSH
;;           # OF     RDA LIST
;;           BLKS      OFFSET       X       FILEREAD!PF,FILEWRITE!PF,FILEFLUSH!PF
;;
;;           DATAPUMP (EQUIVALENT TO JSR DPUMX)
;;           COMMAND/MODE
;;           NON-STANDARD RETURN (RELATIVE)
;;           STANDARD RETURN
;;
;;       STANDARD RETURN:
;;           A0 = UNCHANGED
;;           A1 = UNCHANGED
;;           A2 = UNCHANGED EXC. A(BUF) IN GETBLOCK & GETPRIVATE
;;           A3 = A(PTE)  <--  GRATUITOUS; CALLER SHOULD NOT USE (CALLD USES IT)
;;
;;       NON-STANDARD RETURN:
;;           A0 = ERROR CODE
;;
;;       THEORY:
;;           THE DATAPUMP CALL IMMEDIATELY QUEUES UP THE DATAPUMP TASK AT
;;           HIGH PRIORITY.  ALL DATAPUMP ACTION IS PERFORMED BY THAT TASK,
;;           THUS PREVENTING ANY RE-ENTRANCY PROBLEMS SINCE OTHER DATAPUMP
;;           CALLS WOULD BE QUEUED UP BEHIND THE REGNANT ONE.
;;-------- ------------------------- ------ ------------------------------

;   8 APR 86, RB.: PROVISIONS FOR DMA MAPPING
;   1 MAR 86, RB.: ALLOW OUT-OF-BUFFERS ERROR RETURN
;   5 MAY 86, TWM: ADD CHECK TO INHIBIT EXTENDED MEMORY BUFFERING
```

```
;                       << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>

; FIXED NEGATIVE OFFSETS FROM DATAPUMP ENTRY POINT:

    3341    3362        LDMAMAP             ;-10;DATA CHANNEL MAPPING REQUEST
    3342    3363        LDMAXMAP            ; -7;DATA CHANNEL MAP TO EXTENDED MEMORY

; ACTIVITY COUNTERS -- ALL ARE 2-WORD COUNTERS

    3343    4467        DPSTOTAL            ; -6;TOTAL # BUFFER POOL TABLE SEARCHES
    3344    4471        DPSNFND             ; -5;# TIMES BUFFER POOL TABLE SEARCH WAS UNSUCCESSFUL
    3345    5272        DPRREQ              ; -4;# DATAPUMP READ REQUESTS (INCL. LCM/LXM/MAP)
    3346    5274        DPWREQ              ; -3;# DATAPUMP WRITE REQUESTS (INCL. LCM/LXM/MAP)
    3347    5276        DPDREAD             ; -2;# ACTUAL DISK READS
    3350    5300        DPDWRITE            ; -1;# ACTUAL DISK WRITES

; MAIN DATAPUMP ENTRY POINT:

    3351  6027/ DPUMX: INTDS               ;for (future) re-entrancy protection of DPQA3
    3352  54405         STA     3, DPQA3
    3353   6406         JSR     @.QLINK
    3354   3536         DPTASK             ;DATAPUMP TASK
    3355 177777         -1                 ;USE REGNANT TASK'S TCB
    3356  62000         QP.DP              ;TASK QUEUE PRIORITY FOR DATAPUMP
    3357      0 DPQA3:  0                  ;VALUE OF A3 TO PASS TO TASK
                        ;
; DATAPUMP TASK OPERATES NOW.
; WHEN DONE IT DQUEUE'S DIRECTLY BACK TO THE CALLER OR TO TRAPFAULT
                        ;
    3360  6307/         HALT               ;SHOULD ABSOLUTELY NEVER GET HERE !!


    3361   2475 .QLINK:       QLINK


; DEFAULT ROUTINES IN LIEU OF DMA MAPPING

          3362 LDMAMAP:
    3362   1401         JMP     1,3

          3363 LDMAXMAP:
    3363 161000         MOV     3,0        ;POSITION RETURN ADDRESS FOR TRAP
    3364   6142         TRAPFAULT
    3365 131010         62*K+NOP
```

```
;                       << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>
;-------- ----- ---------- ---------- --------------------
;  ALTERNATE CALLING SEQUENCE:
;
;       SAME DEFINITION OF ACCUMULATORS
;       READBLOCK  or  WRITBLOCK
;       (RETURN)
;
;       ALL TRAPS AND ERRORS ARE HANDLED BY DATAPUMP
;---------- ---------- ---------- ---------- ----------

     3366  54423 RBLK:  STA     3,RBQA3
     3367  34010         LDA     3,.BSA
     3370 156414         SEQ     2,3         ;READING INTO BSA ?
     3371    414         JMP     RBLK2       ;  NO
     3372  34425         LDA     3,P.BSA
     3373 175415         INC#    3,3,SNR     ;IS BSA IN USE ?
     3374    411         JMP     RBLK2       ;  NO, JUST QUEUE THE READBLOCK TASK
     3375  35404         LDA     3,P.DA,3    ;YES
     3376 136414         SEQ     1,3         ;IS DESIRED DISC ADDRESS SAME AS (BSA) ?
     3377    405         JMP     RBLK1       ;  NO
     3400  34417         LDA     3,P.BSA     ;  MAYBE
     3401  35403         LDA     3,P.LU,3
     3402 116415         SNE     0,3
     3403   2406         JMP     @RBQA3      ;  YES, JUST RETURN (MAY NEED A3=BPTE) *****
     3404   4425 RBLK1: JSR     CBSAX       ;NO, CLEAR BSA IF CHANGED
     3405   6754 RBLK2: JSR     @.QLINK
     3406   3513         RBTASK             ;READBLOCK TASK
     3407 177777         -1                 ;USE REGNANT TASK'S TCB
     3410  62000         QP.DP              ;TASK QUEUE PRIORITY FOR DATAPUMP
     3411      0 RBQA3: 0                   ;VALUE OF A3 TO PASS TO TASK

;  READBLOCK TASK OPERATES NOW.
;  WHEN DONE IT DQUEUE'S DIRECTLY BACK TO THE CALLER OR TO TRAPFAULT


;  SPECIAL FIXED BUFFER P.  CELLS MUST REMAIN JUST AHEAD OF WBLK

     3412  30000 P...:  BPS-2200           ;\
     3413  30400         BPS-1600           ;  } USED BY TRANSLATOR IN DATAPUMP'S
     3414  31000         BPS-1200           ;  } COPY-TO/FROM-POOL ROUTINE
     3415  31400         BPS-0600           ;/
     3416      0         0         ;TERMINATOR
     3417 17777/ P.BSA: -1         ;OR BUFFER POOL TABLE ENTRY FOR (BSA)
     3420 17777/ P.HBA: -1         ;OR BUFFER POOL TABLE ENTRY FOR (HBA)
     3421 1777/7 P.HXA: -1         ;OR BUFFER POOL TABLE ENTRY FOR (HXA)
     3422 1777// P.SSA: -1         ;OR BUFFER POOL TABLE ENTRY FOR (SSA)

     3423  54405 WBLK:  STA     3,WBQA3
     3424   6735         JSR     @.QLINK
     3425   3515         WBTASK             ;WRITBLOCK TASK
     3426 1777/7         -1                 ;USE REGNANT TASK'S TCB
     3427  62000         QP.DP              ;TASK QUEUE PRIORITY FOR DATAPUMP
     3430      0 WBQA3: 0                   ;VALUE OF A3 TO PASS TO TASK

;  WRITBLOCK TASK OPERATES NOW.
;  WHEN DONE IT DQUEUE'S DIRECTLY BACK TO THE CALLER OR TO TRAPFAULT
```

```
;                       << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

; CLEAR BSA: CHECK BSA CHANGED FLAG, IF IT IS SET WRITE BSA OUT TO
; ITS POOL IMAGE (OR TO DISC IF NO DIRTY BUFFERS ARE ALLOWED)

; CALLING SEQUENCE:   CALL
;                     CBSA        OR:   JSR CBSAX
;
; EXIT: A0 AND A1 ARE PRESERVED, A2 = .BSA

   3431  54426 CBSAX: STA     3,CLBA3   ;SAVE RETURN ADDRESS
   3432  34075        LDA     3,BSACF
   3433  30010        LDA     2,.BSA
   3434 175015        SNZ     3,3       ;IS BSA-CHANGED FLAG SET ?
   3435   2422        JMP     @CLBA3    ;   NO, RETURN
   3436  40417        STA     0,CLBA0   ;YES, SAVE A0 AND A1
   3437  44417        STA     1,CLBA1
   3440  34757        LDA     3,P.BSA
   3441 161405        INC     3,0,SNR   ;IS P.BSA A VALID POOL POINTER ?
   3442    407        JMP     CLBS2     ;   NO, ERROR !
   3443  21403        LDA     0,P.LU,3  ;YES, PICK UP BSA'S LU AND DA
   3444  25404        LDA     1,P.DA,3
   3445   6143        WRITBLOCK         ;WRITE BSA OUT TO DISC OR POOL
   3446  20407        LDA     0,CLBA0   ;RESTORE A0 AND A1
   3447  24407        LDA     1,CLBA1
   3450   2407        JMP     @CLBA3    ;RETURN

; ERROR:  BSACF SET BUT BSA NOT ASSOCIATED WITH A DISC BLOCK BY A READ
; OR A WRITE

   3451  40075 CLBS2: STA     0,BSACF   ;FIRST CLEAR BSACF
   3452  20405        LDA     0,CLBA3
   3453   6142        TRAPFAULT
   3454 124010        50*K!NOP


   3455      0 CLBA0: 0
   3456      0 CLBA1: 0
   3457      0 CLBA3: 0
```

```
;                        << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

;;-------------------------- -------- ----------------------------
;;
;;                          DATAPUMP TASK
;;
;;---------- ------------------------------ ---------------------
;;
;;   ON ENTRY, ACCUMULATORS ARE AS THEY WERE WHEN DATAPUMP WAS CALLED.
;;   TASK CONTROL NODE CONTAINS AN AUXL TO CALLER'S NODE WHICH ALSO
;;   CONTAINS CALLER'S ACCUMULATORS AS THEY WERE.
;;
;;   DATAPUMP IS DRIVEN BY A TABLE OF ROUTINES CONDITIONALLY EXECUTED
;;   UNDER THE CONTROL OF DPEXECUTIVE ON THE BASIS OF CERTAIN
;;   CONTROL BITS STORED IN A CONTROL WORD IN THE TASK CONTROL NODE.
;;   IT SAVES MOST ALL ITS VARIABLES IN THE TCN (THIS DOESN'T MAKE IT RE-ENTRANT)
;;
;;                       TABLE OF CONTENTS:
;;   DEFINITION OF TCN CELLS USED BY DATAPUMP (THIS PAGE)
;;   DEFINITION OF DATAPUMP CONTROL WORD D.CW (NEXT 2 PAGES)
;;   DESCRIPTION OF ALL ROUTINES AVAILABLE TO DPEXECUTIVE (1 P.)
;;   THE ACTUAL DATAPUMP TASK CODE, INCLUDING THE DRIVING TABLE (7 P.)
;;   CODE OF DPEXECUTIVE (1 P.)
;;   EXPLANATION OF BUFFER POOL TABLE STRUCTURE (2 P.)
;;   CODE OF THE AVAILABLE ROUTINES (BULK OF THE LISTING)


; TASK CONTROL NODE CELLS USED BY DATAPUMP

              12 D.TY= 12      ;TYPE OF CALL:  0 = READ/WRITBLOCK, 2 = DATAPUMP
              13 D.IP= 13      ;INITIAL OFFSET IN HEADER FOR FILE OPS (200 UNLESS PARTIAL XFER)
              14 D.T1= 14      ;TEMP.  STORE USED IN GETNEXTPE, DISK TRANSFER
              15 D.T2= 15      ;    "          "          "
              16 D.LU= 16      ;LOGICAL UNIT NUMBER
              17 D.DA= 17      ;REAL DISK ADDRESS
              20 D.CA= 20      ;ABSOLUTE CORE ADDRESS
              21 D.PE= 21      ;BUFFER POOL TABLE ENTRY
              22 D.CW= 22      ;DATAPUMP CONTROL WORD
              23 D.XP= 23      ;POINTER TO EXECUTE TABLE
              24 D.HP= 24      ;POINTER INTO HEADER BLOCK
              25 D.WR= 25      ;DISK WRITE FLAG
              26 D.BC= 26      ;NUMBER OF BLOCKS THIS TRANSFER
              27 D.TB= 27      ;TOTAL # OF BLOCKS TO TRANSFER IN A FILE READ OR WRITE
              30 D.TS= 30      ;TEMP.  STORE USED IN SEVERAL ROUTINES
              31 D.LF= 31      ;POINTER TO LUFIX TABLE
              32 D.LV= 32      ;POINTER TO LUVAR TABLE
              33 D.TO= 33      ;TIME-OUT COUNTER USED IN DISK TRANSFER
```

```
;                       << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>

; MEANINGS OF CONTROL BITS IN DPEXECUTIVE CONTROL WORD D.CW

                                ;BIT GROUP 1        GROUP 2
                                ;--- ---------     ---------
              ;UL= 100000       ;15     UNLATCH (SUFFIX)
              ;CB= 40000        ;14     CLEAR BLOCK (SUFFIX)
              ;CF= 20000        ;13     CLEAR FILE (SUFFIX)
              ;PF= 10000        ;12     PARTIAL FILE (SUFFIX)
      4000 G2=    4000          ;11   --- 0 ---       --- 1 ---
      2000 GP=    2000          ;10  GETPRIVATE      FILEFLUSH
      1000 GB=    1000          ; 9_ GETBLOCK        DFILREAD
       400 PB=    400           ; 8  PUTBLOCK        DFILWRITE
       200 RD=    200           ; 7  READ            FILEREAD
       100 WR=    100           ; 6_ WRITE           FILEWRITE
        40 QR=    40            ; 5  QREAD           SETDIRTY
        20 QW=    20            ; 4  QWRITE          UNLATCH
        10 DR=    10            ; 3_ D.READ          BUFFLUSH
         4 DW=    4             ; 2  D.WRITE         LRUFLUSH
         2 PU=    2             ; 1  PUBLIC          LUFLUSH
         1 SV=    1             ; 0_ SAVDISCSUB      ALLFLUSH

      6000 FF=    G2+2000;FILEFLUSH \
      5000 LR=    G2+1000;DFILREAD   \
      4400 LW=    G2+400 ;DFILWRITE   }
      4200 FR=    G2+200 ;FILEREAD    }
      4100 FW=    G2+100 ;FILEWRITE   }
      4040 SD=    G2+40  ;SETDIRTY    } MNEMONIC
      4020 UN=    G2+20  ;UNLATCH     } SYMBOLS
      4010 BF=    G2+10  ;BUFFLUSH    } FOR GROUP 2
      4004 LF=    G2+4   ;LRUFLUSH    } COMMANDS
      4002 UF=    G2+2   ;LUFLUSH    /
      4001 AF=    G2+1   ;ALLFLUSH  /

         0 ..=    0      ;USED FOR COLUMN ALIGNMENT


    17377/ MSK=   UL+CB+CF+PF+G2-1;MASK FOR ALL LEGAL CODES EXCEPT GROUP ID.
```

```
;                       << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

; TABLE FROM WHICH D.CW IS PICKED UP FOR GIVEN COMMAND (MODE) WORD
; SPECIFIES THE BASIC COMMAND PLUS LEGAL SUFFIXES

          3460 DPFIRST:       ; MODES MUST BE IN NUMERICAL ORDER:
  3460    1000         GB       ; GETBLOCK (0)
  3461    2000         GP       ; GETPRIVATE (1)
  3462      40         QR       ; QREAD (2)
  3463    4020         UN       ; UNLATCH (3)
  3464      20         QW       ; QWRITE (4)
  3465       2         PU       ; PUBLIC (5)
  3466  100400         PB+UL    ; PUTBLOCK (6)
  3467  104040         SD+UL    ; SETDIRTY (7)
  3470     200         RD       ; READ (10)
  3471     100         WR       ; WRITE (11)
  3472  134200         FR+UL+CF+PF       ; FILEREAD (12)
  3473  134100         FW+UL+CF+PF       ; FILEWRITE (13)
  3474  104010         BF+UL    ; BUFFLUSH (14)
  3475    4004         LF       ; LRUFLUSH (15)
  3476   44002         UF+CB    ; LUFLUSH (16)
  3477   44001         AF+CB    ; ALLFLUSH (17) <-- LAST CALL IN DEFS
  3500      10         DR       ; D.READ (20)   } SPECIAL FOR
  3501       4         DW       ; D.WRITE (21) } DSP AND FAULT ONLY
  3502  115000         LR+UL+PF          ; DFILREAD (22)   } SPECIAL FOR
  3503  114400         LW+UL+PF          ; DFILWRITE (23) } LOADUSER ONLY
  3504       1         SV       ; SAVDISCSUB (24) <-- USED BY CALLD ONLY
  3505   36000         FF+CF+PF          ; FILEFLUSH (25) <-- USED WITHIN FILE READ, WRITE IN DATAPUMP
          3505 DPLAST=.-1

  3506       0 MINMODEWORD: GETBLOCK ; <-- CHANGE IF THIS IS NOT 1ST MODE
  3507      25 NO.MODES:    DPLAST-DPFIRST
            24 SAVDISCSUB=  DFILWRITE+1
  3510    3460 DPTABLE:     DPFIRST
  3511  170000 MSKSUFFIX:   UL+CB+CF+PF
  3512    4022 .DPEXECUTIVE:DPEXECUTIVE



; SUMMARY OF PROCEDURES AVAILABLE TO DPEXECUTIVE (SEE NEXT PAGE)


; FOUR CELLS IN DATAPUMP'S TASK CONTROL NODE GOVERN MOST OF THE
; DATAPUMP/BUFFER POOL ROUTINES' OPERATION.   THESE ARE:
;       D.LU = LOGICAL UNIT         D.DA = REAL DISK ADDRESS
;       D.CA = CORE ADDRESS         D.PE = POOL TABLE ENTRY ADDRESS
```

```
;                        << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

;    ROUTINES FOR EXECUTION FLOW CONTROL
;10. GOTOXXIFFOUND:  CONTINUES PROCESSING AT XX IF D.PE > 0
;11. GOTOYYIFNOTDONE:  CONTINUES PROCESSING AT YY IF D.TB > 0
;12. GOTOZZIFNOTDONE:  CONTINUES PROCESSING AT ZZ IF D.PE > 0
;13. SKIFNOTDIRTY:  SKIP NEXT ROUTINE IF D.PE'S DIRTY FLAG IS 0
;14. IFDONE.EXIT:  TERMINATE DPEXECUTIVE PROCESSING IF D.PE = 0

;    ROUTINES TO COPY DATA FROM/TO CALLER'S NODE (VIA AUXL), ETC.
;20. CCLUDA:  COPIES A0, A1 INTO D.LU, D.DA
;21. CPELUDA:  COPIES LU/DA FROM POOL ENTRY (D.PE) INTO D.LU, D.DA
;22. PECOMPUTE:  COMPUTES D.PE FROM A2 OR A1 IF IN POOL, ELSE 0
;23. A2TOCALLER:  COPIES D.CA INTO CALLER'S A2
;24. CKBSACF:  IF BSACF<>0 AND DISK BLOCK IS IN BSA, COPY BSA-->POOL & MKDIRTY

;    ROUTINES TO CHECK FOR LEGALITY
;30. SKIFLATCHEDPUBLIC:  SKIP NEXT ROUTINE IF LATCHED PUBLIC
;31. OKIFPRIVATE:  GOTO OK IF PRIVATE
;32. OKIFFIXEDBUFF:  IF D.CA --> FIXED BUFFER, GOTO OK
;33. OKIFNOTINPOOL:  GOTO OK IF D.CA IS OTHER LEGAL CORE SPACE

;    ROUTINES TO SEARCH THE POOL TABLE TO OBTAIN A POOL ENTRY POINTER D.PE
;40. GETLRU:  GETS LEAST RECENTLY USED NON-LATCHED PUBLIC BUFFER; TRAP IF OUT
;41. GETNEXTPE:  GETS LEAST RECENTLY USED BUFFER FOR FLUSHING AND/OR CLEARING
;42. DASEARCH:  GETS POOL ENTRY WITH MATCHING LU/DA, ELSE 0

;    ROUTINES TO PUT POOL ENTRY INTO A DIFFERENT PLACE IN THE CHAIN
;50. MKVIRGIN:  MARKS ENTRY VIRGIN AND PUTS IT AT LRU END
;51. PUTMRU:  PUTS ENTRY AT MRU (MOST RECENTLY USED) END

;    ROUTINES TO MARK THE TABLE ENTRY POINTED AT BY D.PE
;60. MKPRIVATE:  MARKS THE BUFFER PRIVATE
;61. MKLUDA:  COPIES LU AND DA FROM D.LU, D.DA
;62. MKDIRTY:  SETS THE DIRTY FLAG IF DIRTY BUFFERS OK, ELSE DISKWRITES
;63. MKNOTDIRTY:  CLEARS THE DIRTY FLAG IF SET
;64. MKLATCHED:  INCREMENTS LATCH COUNT, TRAPS IF OVERFLOW
;65. MKUNLATCH:  DECREMENTS LATCH COUNT, TRAPS IF UNDERFLOW
;66. MKINFIXED:  IF CA=FIXED BUFF, SETS IN_FIXED AND APPROPRIATE P.xx CELL(S)

;    ROUTINES TO COPY A BLOCK OF 256 WORDS
;70. COPYTOPOOL:  COPIES FROM (D.CA) TO ADR(D.PE) OR FIXED BUFFER
;71. COPYFROMPOOL:  COPIES FROM ADR(D.PE) OR FIXED BUFFER TO (D.CA)
;    NOTE:  COPY ROUTINES HAVE SPECIAL PROVISIONS FOR FIXED BUFFERS

;    ROUTINES TO PREPARE FOR A FILE READ OR WRITE
;80. CKHEADER:  TRAPS IF BLOCK AT ADR(D.PE) IS NOT A VALID FILE HEADER
;81. GATHERCONFIG:  DFILR/W:  GATHERS A SET OF CONTIGUOUS BLOCKS FROM FILE HEADER
;82. RWAFILE:  ROUTINE FOR FILE READ, WRITE, CALLS FILEFLUSH, DFILE READ,WRITE

;    ROUTINES TO DO A DISK TRANSFER (MULTI-BLOCK IF T.BC > 0)
;90. DISKWRITE:  INVOKES THE SOV TO WRITE FROM D.CA TO D.LU/DA
;91. DISKREAD:  INVOKES THE SOV TO READ BLOCK D.LU/DA TO D.CA
```

```
;                       << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

;  AND NOW HERE, FINALLY, BEGINS THE ACTUAL DATAPUMP CODE !
;  FIRST, THE TWO ALTERNATE CALLS:
;       (NOTE: These TASKs do not need "CONFLICT FLAGS"
;        because they are called via QLINK, not QUEUE)

          3513 RBTASK:                    ;ENTRY FROM READBLOCK
  3513    4422        JSR     RWTASK
  3514      10        READ

          3515 WBTASK:                    ;ENTRY FROM WRITBLOCK
  3515   34011        LDA     3,.HBA
  3516  156414        SEQ     2,3         ;WRITING FROM HBA ?
  3517     407        JMP     WBTCONTINUE;   NO
  3520   31576        LDA     2,UNIT,3    ;YES,
  3521  112414        SEQ     0,2         ;DOES GIVEN AO MATCH LU IN HBA ?
  3522    4406        JSR     HBAERROR    ;   NO, HBA ERROR
  3523   31577        LDA     2,DHDR,3    ;YES
  3524  132414        SEQ     1,2         ;DOES GIVEN A1 MATCH RDA IN HBA ?
  3525    4403        JSR     HBAERROR    ;   NO, ERROR
          3526 WBTCONTINUE:
  3526    4407        JSR     RWTASK
  3527      11        WRITE

          3530 HBAERROR:                  ;ERROR IN HEADER, WRITING HBA
  3530    6402        JSR     @XDPERROR
  3531      11        11                  ;GENERATE TRAP 11

          3532 XDPERROR:
  3532    3756        DPERRETURN

          3533 DPCERROR:                  ;DATAPUMP CALL ERROR
  3533    67//        JSR     @XDPERROR
  3534      26        26                  ;GIVE TRAP 26

          3535 RWTASK:
  3535  102401        SUB     0,0,SKP     ;ENTRY FROM READBLOCK OR WRITBLOCK
          3536 DPTASK:                    ;***** DATAPUMP TASK MAIN ENTRY *****
  3536   20002        LDA     0,C2
  3537   30015        LDA     2,TASKQ
  3540   41012        STA     0,D.TY,2    ;SET TYPE-OF-CALL WORD
  3541   21400        LDA     0,0,3       ;MODE WORD
  3542   34747        LDA     3,MSKSUFFIX
  3543  117400        AND     0,3         ;EXTRACT SUFFIX CODE
  3544  162400        SUB     3,0         ;REMOVE IT TO GET BASIC MODE
  3545   24741        LDA     1,MINMODEWORD
  3546   30741        LDA     2,NO.MODES
  3547  122422        SUBZ    1,0,SZC     ;IS IT IN LEGAL RANGE ?
  3550  112433        SLE     0,2
  3551    4762        JSR     DPCERROR    ;   NO
  3552   30736        LDA     2,DPTABLE;YES
  3553  113000        ADD     0,2
  3554   21000        LDA     0,0,2       ;PICK UP DP CONTROL WORD
```

```
;                       << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

    3555  24734         LDA     1,MSKSUFFIX
    3556 107400         AND     0,1         ;A1 = MAX. LEGAL SUFFIX
    3557 122400         SUB     1,0         ;A0 = BASIC CONTROL WORD
    3560 167400         AND     3,1         ;LEGAL PART OF ACTUAL SUFFIX
    3561 136405         SUB     1,3,SNR     ;IS ACTUAL = LEGAL ?
    3562 123005         ADD     1,0,SNR     ;   YES, ADD IT TO BASIC CW: NON-ZERO ?
    3563   4750         JSR     DPCERROR    ;     NO, ERROR IN DATAPUMP CALL OR TABLE
    3564  30015         LDA     2,TASKQ
    3565  41022         STA     0,D.CW,2    ;STORE THE COMPLETED DP CONTROL WORD
    3566 102520         SUBZL   0,0
    3567  41026         STA     0,D.BC,2    ;INITIALIZE BLOCK COUNT TO 1
    3570  55021         STA     3,D.PE,2    ;AND BPTE POINTER TO 0 FOR SAFETY
    3571   6721         JSR     @.DPEXECUTIVE

                    ; ***** NO CODE HERE *****


; OVERVIEW OF TABLE-DRIVEN PROCESSING FOR GROUP 1 COMMANDS

;                                    |GP|GB|PB|RD|WR|QR|QW|DR|DW|PU|SV*|
; Legal core is:                     |--|--|--|--|--|--|--|--|--|--|--|
;     Latched public buffer          |  |  |PB|  |  |  |  |  |  |  |  |
;     Private buffer                 |  |  |  |RD|WR|QR|QW|DR|DW|PU|  |
;     Fixed buffer                   |  |  |  |RD|WR|  |  |DR|DW|  |SV|
;     Other legal core               |  |  |  |RD|WR|QR|QW|DR|DW|  |  |
; Search pool for LU/DA match        |  |GB|PB|RD|WR|QR|GW|  |  |  |SV|
; IF match found copy CA             |  |GB|PB|RD|WR|QR|QW|  |  |  |SV|
; ELSE get LRU non-latched pub.      |GP|GB|PB|RD|WR|  |  |  |  |  |  |
;     IF not dirty SKIP              |GP|GB|PB|RD|WR|  |  |  |  |  |  |
;     DISKWRITE                      |GP|GB|PB|RD|WR|  |QW|  |DW|  |  |
;     Mark pool entry virgin         |GP|GB|PB|RD|WR|  |  |  |  |PU|  |
;     DISKREAD Caller's block        |  |GB|  |RD|  |QR|  |DR|  |  |  |
;     EXIT                           |  |  |  |  |  |QR|QW|  |  |PU|  |
; COPY user's block into pool        |  |  |  |  |WR|  |QW|  |  |  |SV|
; COPY pool block to user            |  |  |  |RD|  |QR|  |  |  |  |  |
; Mark pool entry dirty              |  |  |PB|  |WR|  |QW|  |  |  |  |
; Put entry at MRU end               |GP|GB|PB|RD|WR|  |  |  |  |  |SV|
; Mark pool entry private            |GP|  |  |  |  |  |  |  |  |  |  |
; Increment latch count              |  |GB|  |  |  |  |  |  |  |  |SV|
; Decrement latch count              |  |  |  |  |  |  |  |  |  |  |  |<-- if UL bit
; Mark pool entry in-fixed if so     |  |  |  |RD|WR|  |  |  |  |  |  |
;                                                ^  ^     ^  ^
;                                                |  |     |  |
;                                    via pool ---+--+     +--+--- not via pool

; * SV (= SAVDISCSUB) is for special use by CALLDiscsub only


; AN "@" IN THE FOLLOWING TABLE (WHICH SETS MSB = 1) INDICATES THAT
; THE ROUTINE REQUIRES A VALID BPTE (BUFFER POOL TABLE ENTRY) FOR
; PROPER OPERATION

                    ; ***** NO CODE HERE *****
```

```
;                        << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>
                                                    ; SOURCE OF NEW DATA INTO TCN:
                                                    ; D.LU/DA  D.CA      D.PE
                                                    ; --------  --------  --------
; FIRST, SPLIT OFF GROUP 2 COMMANDS

    3572    4000 G2!..!..!..!..!..!..!..!..!
    3573    3666         GROUP2

; CHECK FOR LEGALITY OF GIVEN CORE ADDRESS OR PE POINTER

    3574     777 ..!..!PB!RD!WR!QR!QW!DR!DW!PU!SV
    3575    4141         PECOMPUTE                  ;           CALLER'S CALLER'S

    3576     315 ..!..!..!RD!WR!..!..!DR!DW!..!SV
    3577    4244         OKIFFIXEDBUFFER

    3600     374 ..!..!..!RD!WR!QR!QW!DR!DW!..!
    3601    4256         OKIFNOTINPOOL

    3602     376 ..!..!..!RD!WR!QR!QW!DR!DW!PU!
    3603    4231         OKIFPRIVATE

    3604     400 ..!..!PB!..!..!..!..!..!..!..!
    3605    4221         SKIFLATCHEDPUBLIC

    3606     777 ..!..!PB!RD!WR!QR!QW!DR!DW!PU!SV
    3607    3754         DPER10

; LEGALITY CHECKED, NOW SEARCH POOL FOR MATCHING LU,DA

            3610 OK:
    3610    1775 ..!GB!PB!RD!WR!QR!QW!DR!DW!..!SV
    3611    4125         CCLUDA                     ; CALLER'S

    3612    1761 ..!GB!PB!RD!WR!QR!QW!..!..!..!SV
    3613    4427         DASEARCH                   ;           [PE]      MATCH OR 0

    3614    1761 ..!GB!PB!RD!WR!QR!QW!..!..!..!SV
    3615    4041         GOTOXXIFFOUND

; IF NOT FOUND, DO:

    3616    3700 GP!GB!PB!RD!WR!..!..!..!..!..!
    3617    4325         GETLRU                     ; LRU       LRU       LRU

    3620    3700 GP!GB!PB!RD!WR!..!..!..!..!..!
    3621  104052         @SKIFNOTDIRTY

    3622    3724 GP!GB!PB!RD!WR!..!QW!..!DW!..!
    3623    5140         DISKWRITE

    3624    3702 GP!GB!PB!RD!WR!..!..!..!..!PU!
    3625  104473         @MKVIRGIN
```

```
;                       << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

                                                    ; SOURCE OF NEW DATA INTO TCN:
                                                    ; D.LU/DA  D.CA      D.PE
                                                    ; --------  --------  --------
     3626    1700 . .!GB!PB!RD!WR!. .!. .!. .!. .!. .!
     3627    4125         CCLUDA                     ; CALLER'S

     3630    1250 . .!GB!. .!RD!. .!QR!. .!DR!. .!. .!
     3631    5156         DISKREAD

     3632      76 . .!. .!. .!. .!. .!QR!QW!DR!DW!PU!
     3633    3741         DPRETURN          ; EXIT PROCESSING

     3634    1700 . .!GB!PB!RD!WR!. .!. .!. .!. .!. .!
     3635  104535         @MKLUDA

; END "IF NOT FOUND" CASE

       3636 XX:
     3636    3000 GP!GB!. .!. .!. .!. .!. .!. .!. .!. .!
     3637  104171         @A2TOCALLER

     3640    1240 . .!GB!. .!RD!. .!QR!. .!. .!. .!. .!
     3641  104175         @CKBSACF

     3642     521 . .!. .!PB!. .!WR!. .!QW!. .!. .!. .!SV
     3643  104632         @COPYTOPOOL                ;           CALLER'S

     3644     240 . .!. .!. .!RD!. .!QR!. .!. .!. .!. .!
     3645  104633         @COPYFROMPOOL              ;           CALLER'S

     3646     300 . .!. .!. .!RD!WR!. .!. .!. .!. .!. .!
     3647  104601           @MKINFIXED

     3650     520 . .!. .!PB!. .!WR!. .!QW!. .!. .!. .!
     3651  104543         @MKDIRTY          ;may do disc write if no dirty buffers allowed   . '.  may trap

     3652    3701 GP!GB!PB!RD!WR!. .!. .!. .!. .!. .!SV
     3653  104502         @PUTMRU

     3654    2000 GP!. .!. .!. .!. .!. .!. .!. .!. .!. .!
     3655  104524         @MKPRIVATE

     3656    1001 . .!GB!. .!. .!. .!. .!. .!. .!. .!. .!SV
     3657  104566         @MKLATCHED

     3660  100000 . .!. .!UL!. .!. .!. .!. .!. .!. .!. .!
     3661    4141         PECOMPUTE                  ;           CALLER'S CALLER'S

     3662  100000 . .!. .!UL!. .!. .!. .!. .!. .!. .!. .!
     3663  104567         @MKUNLATCH

     3664  177777         -1                ;UNCONDITIONALLY EXIT
     3665    3741         DPRETURN
```

```
;                          << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>
; REMAINDER OF TABLE DRIVES GROUP 2 COMMANDS

            3666 GROUP2:
    3666    4534         JSR     DPEXECUTIVE

    3667    3770 FF!LR!LW!FR!FW!SD!UN!BF!..!..!..&MSK
    3670    4141         PECOMPUTE

    3671     300 ..!..!..!FR!FW!..!..!..!..!..!..&MSK
    3672    5054         RWAFILE

    3673    3470 FF!LR!LW!..!..!SD!UN!BF!..!..!..&MSK
    3674    4221         SKIFLATCHEDPUBLIC

    3675    3470 FF!LR!LW!..!..!SD!UN!BF!..!..!..&MSK
    3676    3754         DPER10

    3677    3400 FF!LR!LW!..!..!..!..!..!..!..!..&MSK
    3700  104676         @CKHEADER

    3701    3470 FF!LR!LW!..!..!SD!UN!BF!..!..!..&MSK
    3702  104133         @CPELUDA

    3703       2 ..!..!..!..!..!..!..!..!..!UF!..&MSK
    3704    4125         CCLUDA

            3705 ZZ:
    3705    2007 FF!..!..!..!..!..!..!..!LF!UF!AF&MSK
    3706    4337         GETNEXTPE

    3707    2007 FF!..!..!..!..!..!..!..!LF!UF!AF&MSK
    3710    4052         SKIFNOTDIRTY

    3711    2417 FF!..!LW!..!..!..!..!BF!LF!UF!AF&MSK
    3712    5140         DISKWRITE

    3713    2417 FF!..!LW!..!..!..!..!BF!LF!UF!AF&MSK
    3714  104562         @MKNOTDIRTY

    3715   60000 ..!..!..!..!..!..!..!..!..!CB!CF&MSK
    3716  104473         @MKVIRGIN

    3717    2003 FF!..!..!..!..!..!..!..!..!UF!AF&MSK
    3720    4050         GOTOZZ
```

```
;                       << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>

          3721 YY:
  3721    1400 . . !LR!LW!. . !. . !. . !. . !. . !. . !. . &MSK
  3722    5015         GATHERCONTIG

  3723    1000 . . !LR!. . !. . !. . !. . !. . !. . !. . !. . &MSK
  3724    5156         DISKREAD

  3725     400 . . !. . !LW!. . !. . !. . !. . !. . !. . !. . &MSK
  3726    5140         DISKWRITE

  3727    1400 . . !LR!LW!. . !. . !. . !. . !. . !. . !. . &MSK
  3730    4045         GOTOYYIFNOTDONE

  3731      40 . . !. . !. . !. . !. . !SD!. . !. . !. . !. . &MSK
  3732  104543         @MKDIRTY

  3733    1760 . . !LR!LW!FR!FW!SD!UN!. . !. . !. . !. . &MSK
  3734  104502         @PUTMRU

  3735  100020 . . !UL!UL!UL!UL!UL!UN!UL!. . !. . !. . &MSK
  3736  104567         @MKUNLATCH

  3737  177777         -1                      ;UNCONDITIONAL EXIT
  3740    3741         DPRETURN


          3741 DPRETURN:                       ;PREPARE FOR NORMAL DATAPUMP RETURN
  3741   30015         LDA     2,TASKQ
  3742   35035         LDA     3,AUXL,2 ;POINTER TO CALLER'S TCN
  3743   25403         LDA     1,A3,3   ;PREPARE CALLER'S RESUME ADDRESS
  3744   21012         LDA     0,D.TY,2
  3745  107000         ADD     0,1      ;ADJUST DEPENDING ON CALL TYPE
          3746 DPSPECIALRETURN:
  3746   45404         STA     1,PC,3   ;PUT RETURN ADDRESS IN HIS PC CELL
  3747   21021         LDA     0,D.PE,2
  3750   41403         STA     0,A3,3   ;COPY PE --> A3
  3751  102400         SUB     0,0
  3752   41035         STA     0,AUXL,2 ;UNHOOK CALLER'S NODE
  3753    6105         DQUEUE
```

```
;                    << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>
; ERROR RETURNS OUT OF DATAPUMP ROUTINES:

          3754 DPER10:
  3754   20030        LDA     0,C10      ;ILLEGAL CORE ADDRESS = TRAP 10
  3755     402        JMP     DPERO

          3756 DPERRETURN:
  3756   21400        LDA     0,0,3      ;PICK UP POTENTIAL TRAP NUMBER
  3757   30015 DPERO: LDA     2,TASKQ
  3760   55033        STA     3,D.TO,2   ;SAVE ERROR LOC.  IN NODE FOR FAULT ANALYSIS
  3761   35012        LDA     3,D.TY,2
  3762  175015        SNZ     3,3        ;WAS IT A DATAPUMP CALL ?
  3763     411        JMP     DPTRAP     ;   NO, THEN THERE IS NO "SPECIAL RETURN"
  3764   35035        LDA     3,AUXL,2   ;YES, GET CALLER'S NODE
  3765   35403        LDA     3,A3,3     ;CALLER'S RETURN ADDRESS
  3766   25401        LDA     1,1,3      ;DATAPUMP ERROR CONTROL WORD
  3767  125112        SSP     1,1        ;SPECIAL RETURN REQUESTED ?
  3770  127112        ADDL#   1,1,SZC    ;  (i.e.  A1>0 OR 140000<=A1<=177777)
  3771     416        JMP     DPERSPECIAL;   YES
  3772   34052        LDA     3,C177     ;NO
  3773  167705        ANDS    3,1,SNR    ;DID CALLER WANT HIS OWN ERROR NO.?
          3774 DPTRAP:
  3774  105300        MOVS    0,1        ;   NO, USE DATAPUMP'S
  3775   20420        LDA     0,NOPKEYWORD
  3776  123000        ADD     1,0        ;COMPUTE TRAP NUMBER KEYWORD
  3777   40407        STA     0,DPFKW
  4000   30015        LDA     2,TASKQ    ;NOW PREPARE ACCUMULATORS FOR TRAPFALT
  4001   35035        LDA     3,AUXL,2
  4002   21403        LDA     0,A3,3     ;A0 = CALLER'S RETURN ADDRESS
  4003   25017        LDA     1,D.DA,2   ;A1 = RDA
  4004   31020        LDA     2,D.CA,2   ;A2 = CORE ADDRESS
  4005    6142        TRAPFAULT
  4006       0 DPFKW: 0                  ;TRAP NUMBER EMBEDDED IN NOP


          4007 DPERSPECIAL:              ;CALCULATE SPECIAL RETURN ADDRESS
  4007  167000        ADD     3,1
  4010  125400        INC     1,1
  4011   30015        LDA     2,TASKQ
  4012   35035        LDA     3,AUXL,2
  4013   41402        STA     0,A0,3
  4014     732        JMP     DPSPECIALRETURN


  4015  100010 NOPKEYWORD:   NOP
```

```
;                       << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

;  DATAPUMP EXECUTIVE
;  CALLS ALL DATAPUMP/BUFFER POOL SERVICE ROUTINES
;
;  CALLING SEQUENCE: (WITH A2 --> TASK CONTROL NODE)
;           JSR DPEXECUTIVE
;           CONDITION MASK #1
;           ROUTINE ADDRESS #1
;           CONDITION MASK #2
;           ROUTINE ADDRESS #2
;
;           CONDITION MASK #N
;           ROUTINE ADDRESS #N
;           0
;           (RETURN)
;
;           THE "CONDITION MASK" IS AND'ED WITH D.CW IN THE TCN AND
;           IF RESULT = 0 NEXT ROUTINE IS SKIPPED
;
;  ON ENTRY TO ANY OF THESE ROUTINES,
;       A2 = TASK CONTROL NODE POINTER
;       A3 = BUFFER POOL TABLE ENTRY POINTER,  IF ANY,  ELSE 0
;
;  NORMAL RETURN FROM ALL ROUTINES IS TO DPEXRETURN,  ABNORMAL RETURN
;       MAY BE TO TRAPFAULT OR TO DPERRETURN WITH A0 = ERROR CODE



          4016 DPEXRETURN:                ;NORMAL RETURN FROM ROUTINE JUST EXECUTED
    4016  30015        LDA     2,TASKQ
    4017  35023        LDA     3,D.XP,2  ;POINTS TO ROUTINE JUST COMPLETED
    4020  25022        LDA     1,D.CW,2
          4021 DPEXLOOP:
    4021 175401        INC     3,3,SKP   ;ADVANCE A3 TO NEXT CONDITION WORD
          4022 DPEXECUTIVE:               ;INITIAL ENTRY TO DPEXECUTIVE
    4022  25022        LDA     1,D.CW,2  ;PICK UP DATAPUMP CONTROL WORD
    4023  21400        LDA     0,0,3     ;AND CONDITION WORD IN DRIVING TABLE
    4024 175400        INC     3,3       ;ADVANCE A3 TO NEXT ROUTINE POINTER
    4025 107415        AND#    0,1,SNR   ;CONDITION MET ?
    4026    773        JMP     DPEXLOOP  ;   NO,  TRY NEXT TABLE ENTRY
    4027  55023        STA     3,D.XP,2  ;YES,  SAVE TABLE ENTRY POINTER
;  Good place for a DBUG breakpoint to see which procedures are being done
    4030  35400        LDA     3,0,3     ;PICK UP ROUTINE POINTER
    4031 175100        MOVL    3,3       ;MOVE "@" BIT INTO CARRY
    4032 102500        SUBL    0,0       ;IF "@" (BPTE REQUIRED),  A0 <-- 0,  ELSE A0 <-- 1
    4033 175220        MOVZR   3,3       ;GENERATE ROUTINE ADDRESS WITHOUT "@" BIT
    4034  55030        STA     3,D.TS,2  ;SAVE POINTER TO ROUTINE TO BE EXECUTED
    4035  35021        LDA     3,D.PE,2  ;BUFFER POOL TABLE ENTRY POINTER
    4036 117014        ADD#    0,3,SZR   ;IS BPTE REQUIRED BUT NON-EXISTENT ?
    4037   3030        JMP     @D.TS,2   ;   NO,  JUMP TO THE ROUTINE
    4040   6142        TRAPFAULT         ;YES,  CODING ERROR INSIDE DATAPUMP
```

```
;                        << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>
;;
;;          B U F F E R   P O O L   T A B L E   S T R U C T U R E
;;
;;                 BPMRU:
;;               +----+----+----+----+----+
;;  .BPMRU:      | 0  | NX |PRI.| -1 | -1 | <-- DUMMY MRU ENTRY
;;               + ---+-^--+----+----+----+
;;                      |
;;                   +--+
;;                   |
;;               + -v-+----+----+----+----+----+
;;               | PR | NX | FL | LU | DA | CA | <-- MOST RECENTLY USED ENTRY
;;               + - -+-^--+----+----+----+----+
;;                      |
;;                   +--+
;;                   |
;;               + -v-+----+----+----+----+----+
;;               | PR | NX | FL | LU | DA | CA | <-- OTHER BUFFER ENTRIES
;;               +---- +-^--+----+----+----+----+
;;                      |
;;                   +--+
;;                   |
;;               + -v-+----+----+----+----+----+
;;               | PR | NX | FL | LU | DA | CA | <-- LEAST RECENTLY USED ENTRY
;;               +---- +-^--+----+----+----+--^-+
;;                      |                     |
;;                   +--+       IN REAL CORE CA IS IN SORTED SEQUENTIAL ORDER
;;                   |
;;               + -v-+----+----+----+----+
;;  .BPLRU:      | PR |  0 |PRI.| -1 | -1 | <-- DUMMY LRU ENTRY
;;               + ---+----+----+----+----+
;;               BPLRU
;;
;;   PLEASE NOTE THAT THE FORWARD AND BACKWARD LINKS POINT AT EACH OTHER


; THE BUFFER POOL TABLE IS A DOUBLY LINKED LIST KEPT IN LRU (LEAST
; RECENTLY USED) ORDER. EVERY TIME A POOL BUFFER IS ACCESSED, ITS ENTRY
; IS PUT AT THE MRU END OF THE CHAIN; WHEN A BLOCK IS NEEDED FOR A
; NEW ASSIGNMENT THE ONE AT THE LRU END IS TAKEN. IF THE LRU BUFFER IS
; DIRTY IT IS WRITTEN TO DISK BEFORE BEING REASSIGNED, BUT DIRTYNESS
; DOES NOT KEEP A BUFFER FROM BEING USED IF IT IS THE LRU.

; THE BUFFER POOL TABLE IS A CONTIGUOUS AREA IN CORE AND THE PHYSICAL
; POSITION OF EACH ENTRY PERMANENTLY IDENTIFIES THE CORRESPONDING
; BUFFER'S ADDRESS. THE BUFFER CORE ADDRESSES ARE ENTERED BY SIR IN
; SORTED ORDER, AND IN AT MOST 4 CONTIGUOUS SETS, TO ALLOW EASY
; COMPUTATION OF PE NUMBER FROM BUFFER ADDRESS (WITHOUT REQUIRING A
; POOL SEARCH). IT FOLLOWS THAT THE BUFFER'S CORE ADDRESS DOES NOT
; NEED TO BE STORED IN THE BPT ENTRY; IT IS KEPT THERE MAINLY FOR
; CONVENIENCE IN DEBUGGING. IT IS THE LAST ITEM IN THE ENTRY SO THAT
; IT CAN BE DELETED AT SOME FUTURE TIME.
```

```
;                       << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>

; DEFINITIONS OF BUFFER POOL TABLE ENTRY (BPTE) CELLS:

              0 P.PR= 0       ;LINK TO PREVIOUS POOL TABLE ENTRY'S P.NX
              1 P.NX= 1       ;LINK TO NEXT POOL TABLE ENTRY'S P.PR
              2 P.FL= 2       ;FLAGS AND LATCH COUNT (SEE BELOW)
              3 P.LU= 3       ;LU OF DISK BLOCK IN BUFFER IF PUBLIC
                              ;OWNER'S TCB POINTER IF PRIVATE, ELSE -1
              4 P.DA= 4       ;RDA OF BLOCK IN BUFFER IF PUBLIC (ELSE -1)
              5 P.CA= 5       ;BUFFER'S CORE ADDRESS (PUT IN BY SIR)


                              ;BIT        MEANING OF BIT IN P.FL WORD:

         100000 PRIVAT=100000 ;15_       Buffer is private
                              ;14        Buffer is busy (not implemented)
                              ;13        Buffer is locked (not implemented)
          10000 INFIXED=10000 ;12_       Block is also IN a FIXED buffer
           4000 VIRGIN=  4000 ;11        Bit is always set (NOT A FLAG)
                              ;10
           1000 DIRTY=   1000 ; 9_       Buffer is dirty
            400 LCGUARD=  400 ; 8        GUARD bit for the latch counter
                              ; 7  \
                              ; 6_  \
                              ; 5    \
                              ; 4     } Latch counter (Counts users
                              ; 3_    } who are currently using buffer)
                              ; 2    /  must = 0 if not public
                              ; 1   /
                              ; 0  /

; A VIRGIN BUFFER HAS P.FL = 4000, P.LU = P.DA = -1
```

```
;                          << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>

;;----------  ----------------------------------------
;; ROUTINES FOR PROGRAM EXECUTION FLOW CONTROL
;;----------  ----------------------------------------


;  ====
;((10)) GO TO XX IF POOL TABLE ENTRY WAS FOUND (IE.  IF D.PE > 0)
;  ====

                4041 GOTOXXIFFOUND:
    4041   20452          LDA     0,.XX
                4042 GOTOTEST:
    4042  175014          SKZ     3,3         ;IS D.PE (OR OTHER) = 0 ?
                4043 GOTOUNCONDITIONAL:
    4043   41023          STA     0,D.XP,2 ;   NO, SET "GOTO" DESTINATION
    4044     752          JMP     DPEXRETURN


;  ====
;((11)) GO TO YY IF TOTAL BLOCK COUNT REMAINING TO BE TRANSFERRED
;  ====   IS NON-ZERO (FOR FILE READ/WRITE)

                4045 GOTOYYIFNOTDONE:
    4045   20447          LDA     0,.YY
    4046   35027          LDA     3,D.TB,2
    4047     773          JMP     GOTOTEST


;  ====
;((12)) GO TO ZZ
;  ====

                4050 GOTOZZ:
    4050   20445          LDA     0,.ZZ
    4051     772          JMP     GOTOUNCONDITIONAL
```

```
;                       << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>

;  ====
; ((13)) SKIP NEXT ROUTINE IF POOL ENTRY IS NOT DIRTY
;  ====

          4052 SKIFNOTDIRTY:
     4052  21402          LDA     0,P.FL,3
     4053  24443          LDA     1,K.DIRTY
     4054 107414          AND#    0,1,SZR   ;IS POOL BUFFER DIRTY ?
     4055    741          JMP     DPEXRETURN;  YES, NORMAL RETURN
     4056  22441          LDA     0,@LSCON
     4057  24442          LDA     1,XMEM
     4060 107415          AND#    0,1,SNR   ;LCM OR MAP ACTIVE ?
     4061    427          JMP     SKIPROUTINE;  NO
     4062  21017          LDA     0,D.DA,2
     4063 101415          INC#    0,0,SNR   ;IS THIS BLOCK IN USE ?
     4064    424          JMP     SKIPROUTINE;  NO
     4065  21016          LDA     0,D.LU,2 ;  YES, FIND LUVAR FOR ITS LU
     4066   6123          FINDLUT
     4067   6434          JSR     @ILUER    ;ILLEGAL OR INACTIVE LU
     4070  30015          LDA     2,TASKQ   ;RESTORE NODE(ADDRESS
     4071  21403          LDA     0,LVLUC,3;LOAD LU CHARACTERISTICS
     4072  24432          LDA     1,KNXMB
     4073 123414          AND#    1,0,SZR   ;EXTENDED MEMORY CACHING DISABLED ?
     4074    414          JMP     SKIPROUTINE;  YES, NO CACHE
     4075  22422          LDA     0,@LSCON ;  NO, DETERMINE MAP OR LCM CACHE
     4076  34114          LDA     3,.LCM
     4077  24421          LDA     1,XLC
     4100 107415          AND#    0,1,SNR
     4101  36421          LDA     3,@LMAP.
     4102 102000          ADC     0,0
     4103  41025          STA     0,D.WR,2 ;SET "PUT" CMD
     4104  21017          LDA     0,D.DA,2
     4105   5400          JSR     0,3       ;PUT TO LCM OR MAP
     4106 102520          SUBZL   0,0
     4107  41026          STA     0,D.BC,2 ;RESTORE BLK CNT TO 1
          4110 SKIPROUTINE:
     4110  11023          ISZ     D.XP,2    ;NO, SKIP NEXT PROCEDURE
     4111  11023          ISZ     D.XP,2
     4112    704          JMP     DPEXRETURN


     4113   3635 .XX:   XX-1
     4114   3720 .YY:   YY-1
     4115   3704 .ZZ:   ZZ-1
     4116   1000 K.DIRTY:       DIRTY
     4117    662 LSCON: INFO+SCON.
     4120   2000 XLC:   LC
     4121   2040 XMEM:  LC+MM
     4122    700 LMAP.: INFO+.MAP.
     4123   5127 ILUER: LUERROR
     4124  20000 KNXMB: LVLNXMB
```

```
;                        << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>

;;-------- -------- ----------------------------------
;; ROUTINES TO COPY DATA FROM CALLER'S NODE (VIA AUXL)
;;-------- ---------- ------------------------------

;  ====
;((20)) COPY CALLER'S LU/RDA INTO DATAPUMP NODE
;  ====

          4125 CCLUDA:
    4125  35035        LDA     3,AUXL,2 ;GET LINK TO CALLER'S NODE
    4126  21402        LDA     0,A0,3   ;COPY HIS A0
    4127  41016        STA     0,D.LU,2 ;   INTO OUR D.LU
    4130  21401        LDA     0,A1,3   ;AND HIS A1
    4131  41017        STA     0,D.DA,2 ;   INTO OUR D.DA
    4132    664        JMP     DPEXRETURN


;  ====
;((21)) COPY LU & RDA FROM POOL TABLE ENTRY (D.PE) INTO D.LU, D.DA
;  ====

          4133 CPELUDA:
    4133  21403        LDA     0,P.LU,3
    4134  41016        STA     0,D.LU,2
    4135  21404        LDA     0,P.DA,3
    4136  41017        STA     0,D.DA,2
    4137    657        JMP     DPEXRETURN



    4140    666 I.BPT: INFO+.BPT.       ;POINTER TO BUFFER POOL TABLE
```

```
;                        << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

;  ====
; ((22)) COMPUTE POOL TABLE ENTRY FROM CALLER'S A2,  IF IN POOL
;  ====  ELSE SET D.PE = 0
;        COPY ITS PE AND CA INTO TCN BUT NOT ITS LU OR DA

;             D.CA <-- A2
;             IF A2 IS IN POOL THEN
;                  D.PE <-- PE COMPUTED FROM A2
;             ELSE D.PE <-- 0

;  THEORY:  BUFFERS IN BUFFERLIST ARE IN CONTIGUOUS GROUPS (AT MOST 4)
;  AND CORRESPOND 1-TO-1 WITH POOL TABLE ENTRIES IN REAL CORE (IE.
;  IGNORING THE WAY THEY MAY BE CHAINED TOGETHER):  THE FIRST BUFFER
;  CORRESPONDS TO THE FIRST POOL ENTRY,  THE SECOND TO THE SECOND,  ETC.


        4141 PECOMPUTE:
4141   23035         LDA     0,@AUXL,2;CALLER'S A2
4142   41020         STA     0,D.CA,2 ;SAVE CALLER'S A2 IN TCN
4143   34512         LDA     3,BUFFERLIST
4144   55030         STA     3,D.TS,2 ;SCAN BUFFERLIST TO SEE IF A2 IS IN IT
4145  102400         SUB     0,0      ;A0 WILL BE BUFFER NUMBER
        4146 PELOOP:
4146   37030         LDA     3,@D.TS,2;PICK UP BASE OF A GROUP OF BUFFERS
4147  175015         SNZ     3,3      ;END OF BUFFERLIST ?
4150     417         JMP     PEDONE   ;   YES, SET PE=0 (A2 NOT IN POOL)
4151   25020         LDA     1,D.CA,2 ;NO
4152  166700         SUBS    3,1      ;CONVERT CORE OFFSET TO BUFFER NUMBER
4153   11030         ISZ     D.TS,2
4154   37030         LDA     3,@D.TS,2;PICK UP # OF BUFFERS IN THIS GROUP
4155   11030         ISZ     D.TS,2
4156  136033         SLS     1,3      ;IS A2 IN THIS GROUP OF BUFFERS ?
4157  163001         ADD     3,0,SKP  ;   NO, ACCUMULATE #BUFFERS & CONT.
4160  123001         ADD     1,0,SKP  ;   YES,  CALCULATE BUFFER NO.
4161     765         JMP     PELOOP
4162   36756         LDA     3,@I.BPT ;CALCULATE POOL ENTRY ADDRESS:
4163  103000         ADD     0,0      ;D.PE = A(BUFFER POOL TABLE) +
4164  117000         ADD     0,3      ;         6 * POOL ENTRY NO.
4165  117000         ADD     0,3
4166  117000         ADD     0,3      ;NOW A3 = POOL TABLE ENTRY ADDRESS
        4167 PEDONE:
4167   55021         STA     3,D.PE,2
4170     626         JMP     DPEXRETURN
```

```
;                       << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>

;  ====
;((23)) COPY POOL BUFFER'S CORE ADDRESS INTO CALLER'S A2
;  ====

            4171 A2TOCALLER:
    4171   21405         LDA     0,P.CA,3 ;CORE ADDRESS IN BPTE
    4172   35035         LDA     3,AUXL,2 ;CALLER'S TASK CONTROL NODE
    4173   41400         STA     0,A2,3   ;SET HIS A2 = CORE ADDRESS
    4174     622         JMP     DPEXRETURN


;  ====
;((24)) CHECK BSA-CHANGED FLAG (ONLY USED IN A READ):
;  ====  IF DESIRED DISK BLOCK IS IN BSA AND BSA-CHANGED FLAG IS SET
;        THEN COPY BSA INTO ITS POOL BUFFER, CLEAR BSACF AND MARK BUFFER DIRTY

            4175 CKBSACF:
    4175   22423         LDA     0,@XP.BSA;PICK UP (P.BSA)
    4176   24075         LDA     1,BSACF
    4177  162415         SNE     3,0       ;IS PE = P.BSA (IE. DISK BLOCK IN BSA) ...
    4200  125015         SNZ     1,1       ;  ... AND BSA-CHANGED FLAG SET ?
    4201     615         JMP     DPEXRETURN;  NO, THEN NOTHING SPECIAL
    4202   30010         LDA     2,.BSA    ;YES, THEN ...
    4203   35405         LDA     3,P.CA,3
    4204   24023         LDA     1,CM400
    4205   21000 CKBS2:  LDA     0,0,2     ;COPY BSA INTO POOL BUFFER
    4206   41400         STA     0,0,3
    4207  151400         INC     2,2
    4210  175400         INC     3,3
    4211  125404         INC     1,1,SZR   ;DONE ?
    4212     773         JMP     CKBS2     ;  NO
    4213   44075         STA     1,BSACF   ;YES, CLEAR BSA-CHANGED FLAG
    4214   30015         LDA     2,TASKQ   ;RESTORE A2 AND A3
    4215   35021         LDA     3,D.PE,2
    4216    2401         JMP     @.+1      ;AND GO TO "MKDIRTY"
    4217    4543         MKDIRTY


    4220    3417 XP.BSA:        P.BSA
```

```
;                        << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

;;----------  ----------------------  ----------
;;  ROUTINES TO CHECK IF GIVEN CORE ADDRESS IS LEGAL
;;----------  -------  ----------  ------------


;  ====
;((30)) SKIP NEXT ROUTINE IF POOL ENTRY IS A LATCHED PUBLIC BUFFER
;  ====  NON-SKIP RETURN IF NOT IN POOL;  TRAP 17 IF IN POOL BUT NOT LATCHED

          4221 SKIFLATCHEDPUBLIC:
  4221 175015          SNZ     3,3         ;IN POOL ?
  4222   2466          JMP     @DPEX1RETURN;   NO, DON'T SKIP
  4223  21402          LDA     0,P.FL,3
  4224  24064          LDA     1,C377
  4225 107414          AND#    0,1,SZR   ;YES, LATCH COUNT > 0 ?
  4226    662          JMP     SKIPROUTINE;   YES, SKIP NEXT ROUTINE
  4227   6554          JSR     @ZDPERROR
  4230     17          17                  ;NO, GIVE A TRAP 17


;  ====
;((31)) GOTO OK IF IT'S A PRIVATE BUFFER -- ELSE RETURN
;  ====

          4231 OKIFPRIVATE:
  4231  21402          LDA     0,P.FL,3
  4232  24462          LDA     1,C.PRIVATE
  4233 175014          SKZ     3,3         ;IN POOL . .
  4234 107405          AND     0,1,SNR   ; . . .  AND PRIVATE ?
  4235   2453          JMP     @DPEX1RETURN;   NO, JUST RETURN
  4236  21403          LDA     0,P.LU,3 ;TCB POINTER IF PRIVATE
  4237  24007          LDA     1,RTP     ;REGNANT TASK POINTER
  4240 106414          SEQ     0,1         ;BELONGS TO REGNANT TASK ?
  4241   2447          JMP     @DPEX1RETURN;   NO, JUST RETURN
          4242 GOTO.OK:
  4242  20411          LDA     0,..OK
  4243    600          JMP     GOTOUNCONDITIONAL


;  ====
;((32)) GOTO OK IF IT IS A FIXED BUFFER -- ELSE RETURN
;  ====

          4244 OKIFFIXEDBUFFER:
  4244  21020          LDA     0,D.CA,2
  4245  24010          LDA     1,..BSA
  4246 122700          SUBS    1,0
  4247  24024          LDA     1,C4
  4250 106033          SLS     0,1         ;IS GIVEN ADDRESS A FIXED BUFFER ?
  4251   2437          JMP     @DPEX1RETURN;   NO
  4252    770          JMP     GOTO.OK   ;   YES

  4253   3607 ..OK:             OK-1
  4254 11037/ C.UNAVAIL:        PRIVATE+INFIXED+LCGUARD-1
  4255   547/ BUFFERLIST:       BUF1
```

```
;                       << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

;  ====
;((33)) GOTO OK IF IT IS OTHER LEGAL CORE SPACE -- ELSE RETURN
;  ====  LEGAL CORE SPACE = PROCESSOR PAGE ZERO (INFO - 400)
;        AND ALL SPACE ABOVE REX EXCEPT POOL BUFFERS

          4256 OKIFNOTINPOOL:
     4256  21020          LDA     0,D.CA,2
     4257  24064          LDA     1,C377    ;AVOID 177400+400 =0
     4260 123000          ADD     1,0
     4261  34100          LDA     3,.INFO
     4262 116015          ADC#    0,3,SNR   ;PROCESSOR'S PAGE ZERO ?
     4263    757          JMP     GOTO.OK   ;   YES, OK
     4264  24423          LDA     1,PTSPLUS377
     4265 106032          SGE     0,1       ;BELOW BEGINNING OF PATCH SPACE ?
     4266   2422          JMP     @DPEX1RETURN;   YES, NOT OK
     4267  34766          LDA     3,BUFFERLIST;NO, CHECK IF IT'S A POOL BUFFER
     4270  55030          STA     3,D.TS,2
          4271 CKLOOP:
     4271  37030          LDA     3,@D.TS,2;FIRST BUFFER IN A GROUP
     4272 175015          SNZ     3,3       ;END OF BUFFER LIST ?
     4273    747          JMP     GOTO.OK   ;   YES, THEN OK
     4274  11030          ISZ     D.TS,2
     4275  27030          LDA     1,@D.TS,2;NO.  BUFFERS IN THIS GROUP
     4276  11030          ISZ     D.TS,2
     4277 125700          INCS    1,1       ;CALCULATE AREA IN WORDS
     4300 167000          ADD     3,1       ;END OF AREA +1
     4301 124400          NEG     1,1       ;END OF AREA
     4302 124000          COM     1,1
     4303 116033          SLS     0,3       ;IS GIVEN ADDRESS BELOW THIS GROUP,
     4304 106033          SLS     0,1       ;   OR ABOVE IT ?
     4305    764          JMP     CKLOOP    ;   YES, SO KEEP CHECKING
     4306   2402          JMP     @DPEX1RETURN;NO, THEN NOT OK


     4307  13306 PTSPLUS377:  PATSP+377
     4310   4016 DPEX1RETURN: DPEXRETURN



;  DUMMY NODES FOR MRU AND LRU ENDS OF BUFFER POOL TABLE ENTRY CHAIN:

     4311   4312 .BPMRU:       .+1
     4312      0               0         ;PR = 0 TO SIGNAL MRU END OF CHAIN
     4313      0 BPMRU:        0         ;NX ***** SET UP BY SIR *****
     4314 100000 C.PRIVATE:    PRIVATE   ;FL
     4315 177777               -1        ;LU
     4316 177777               -1        ;DA

     4317   4320 .BPLRU:       .+1
     4320      0 BPLRU:        0         ;PR ***** SET UP BY SIR *****
     4321      0               0         ;NX = 0 TO SIGNAL LRU END OF CHAIN
     4322 100000               PRIVATE   ;FL
     4323 177777               -1        ;LU
     4324 177777               -1        ;DA
```

```
;                        << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>

;;-------- ------------------------------------------
;; ROUTINES TO SEARCH POOL TABLE TO OBTAIN A POOL ENTRY POINTER D.PE
;;-------- ------- ----------------------------

;  ====
;((40)) GET LEAST RECENTLY USED "AVAILABLE" BUFFER -- I.E., PUBLIC,
;  ====  NOT LATCHED, AND NOT IN A FIXED BUFFER
;        WHEN FOUND, COPY ITS LU, DA, CA, AND PE INTO TASK CONTROL NODE

          4325 GETLRU:
    4325  34773          LDA     3,BPLRU   ;PICK UP PNTR TO LRU ENTRY'S NX CELL
    4326  24726          LDA     1,C.UNAVAIL
          4327 GTLRULOOP:
    4327  21401          LDA     0,P.FL-1,3
    4330 107415          AND#    0,1,SNR   ;IS BUFFER PRI, LATCHED, OR IN_FIXED ?
    4331     440         JMP     GTCOPY    ;   NO, THEN USE IT
    4332  357??          LDA     3,P.PR-1,3;YES, TRY NEXT
    4333 175014          SKZ     3,3       ;HIT MRU END OF CHAIN ?
    4334     773         JMP     GTLRULOOP;   NO
    4335    6446         JSR     @ZDPERROR;YES, POOL IS OUT OF AVAILABLE BUFFERS!
    4336      71         71                ;TRAP 71 UNLESS USER WANTS TO HANDLE HIMSELF


;  ====
;((41)) GET NEXT BUFFER POOL TABLE ENTRY TO PROCESS, BASED ON COMMAND
;  ====  BEING EXECUTED:

;         IF LRUFLUSH,      CHECK THAT P.FL = VIRGIN+DIRTY
;         IF LUFLUSH,       CHECK THAT P.LU = (AO) & P.FL = VIRGIN+DIRTY
;         IF LUFLUSH!CB,    CHECK THAT P.LU = (AO) & P.DA <> -1
;         IF ALLFLUSH,      CHECK THAT P.FL = VIRGIN+DIRTY
;         IF ALLFLUSH!CB,   CHECK THAT P.DA <> -1
;         IF FILEFLUSH,     CHECK THAT P.LU = (AO) & P.DA IN HDR.
;         IF FILEFLUSH!CF,  CHECK THAT P.LU = (AO) & P.DA IN HDR.  & VIRGIN+DIRTY

          4337 GETNEXTPE:
    4337  34761          LDA     3,BPLRU   ;A3 --> LRU ENTRY'S NX CELL
          4340 GETNLOOP:
    4340  21022          LDA     0,D.CW,2
    4341  24522          LDA     1,VFFUF
    4342 107415          AND#    0,1,SNR   ;DOING LUFLUSH OR FILEFLUSH ?
    4343     406         JMP     GETN2     ;   NO
    4344  21402          LDA     0,P.LU-1,3;YES
    4345  25016          LDA     1,D.LU,2
    4346 106414          SEQ     0,1       ;DOES PE BELONG TO LU WE'RE LOOKING FOR ?
    4347     415         JMP     TRYNEXT   ;   NO
    4350  21022          LDA     0,D.CW,2 ;YES, SO FAR SO GOOD
    4351  24514 GETN2:LDA     1,VCL
    4352 107414          AND#    0,1,SZR   ;ARE WE DOING A CLEAR ?
    4353     432         JMP     CHKFF     ;   YES
    4354  21401          LDA     0,P.FL-1,3;NO, THEN SELECT ONLY DIRTY BUFFERS
    4355  24427          LDA     1,C.FLUSHABLE
    4356 106415          SNE     0,1       ;IS THIS ONE DIRTY & NON-LATCHED PUBLIC ?
    4357     426         JMP     CHKFF     ;   YES, FOUND ONE -- GO CHECK IF FILEFLUSH
    4360     404         JMP     TRYNEXT   ;NO, IGNORE IT
```

```
;                    << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

4361  21403 GETN3: LDA     0,P.DA-1,3;DOING A CLEAR
4362 101414        INC#    0,0,SZR  ;IS THE BUFFER IN USE AND NOT PRIVATE ?
4363    406        JMP     GTCOPY   ;   YES, COPY IT
      4364 TRYNEXT:
4364  35777        LDA     3,P.PR-1,3;PICK UP NEXT YOUNGER PE
4365 175014        SKZ     3,3      ;HIT MRU END OF CHAIN ?
4366    752        JMP     GETNLOOP ;   NO, CONTINUE LOOKING
4367   2401        JMP     @.+1     ;YES, DATAPUMP OPERATION IS DONE
4370   3741        DPRETURN

      4371 GTCOPY:
4371 174400        NEG     3,3      ;NOW CORRECT POINTER TO
4372 174000        COM     3,3      ;POINT TO WORD 0 OF ENTRY
4373  21403        LDA     0,P.LU,3 ;COPY LU,
4374  41016        STA     0,D.LU,2
4375  21404        LDA     0,P.DA,3 ;      RDA, AND
4376  41017        STA     0,D.DA,2
      4377 COPYCA:
4377  21405        LDA     0,P.CA,3 ;      CORE ADDRESS
4400  41020        STA     0,D.CA,2 ;INTO TCN
      4401 SETPE.RETURN:
4401  55021        STA     3,D.PE,2 ;SAVE PE (MAY=0 FROM GETDIRTY)
      4402 DPEX2RETURN:
4402   2706        JMP     @DPEX1RETURN

4403   3756 ZDPERROR:     DPERRETURN
4404   5000 C.FLUSHABLE:  VIRGIN+DIRTY

4405  20457 CHKFF: LDA     0,VFF
4406  25022        LDA     1,D.CW,2
4407 107415        AND#    0,1,SNR  ;FILEFLUSH ?
4410    751        JMP     GETN3    ;   NO
4411  21024        LDA     0,D.HP,2;YES, COPY HDR. PTR., AND TRANS. COUNT TO TEMP CELLS
4412  41014        STA     0,D.T1,2
4413  21027        LDA     0,D.TB,2
4414  41015        STA     0,D.T2,2
      4415 FFSCAN:
4415  23014        LDA     0,@D.T1,2
4416  11014        ISZ     D.T1,2   ;PREPARE NEXT RDA, NEVER SKIPS
4417 101015        SNZ     0,0      ;A GOOD RDA ?
4420    775        JMP     FFSCAN   ;   NO
4421  25403        LDA     1,P.DA-1,3;YES, CHECK AGAINST D.PA
4422 106415        SNE     0,1      ;RDA IN FILE ?
4423    736        JMP     GETN3    ;   YES
4424  15015        DSZ     D.T2,2   ;NO, DONE WITH (TB) BLOCKS ?
4425    770        JMP     FFSCAN   ;   NO, GET NEXT
4426    736        JMP     TRYNEXT  ;   YES, GO TO NEXT POOL ENTRY
```

```
;                       << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>

;  ====
;((42)) SEARCH FOR LU/DA GIVEN IN REGNANT TASK CONTROL NODE (TCN)
;  ====  IF FOUND COPY ITS PE AND CA INTO TCN
;        ELSE SET PE = 0 AND DO NOT CHANGE CA IN TCN

          4427 DASEARCH:
  4427   21016         LDA     0,D.LU,2
  4430    6123         FINDLUT
  4431    6435         JSR     @.LUERROR;ILLEGAL OR INACTIVE LU
  4432   10436         ISZ     DPSTOTAL+1;INCREMENT ACTIVITY COUNTER
  4433     404         JMP     DAS2
  4434   10433         ISZ     DPSTOTAL
  4435     402         JMP     DAS2
  4436   14431         DSZ     DPSTOTAL
  4437   30015 DAS2:   LDA     2,TASKQ
  4440   34651         LDA     3,.BPMRU ;PICK UP PNTR TO MRU DUMMY ENTRY
          4441 DAS2LOOP:
  4441   25017         LDA     1,D.DA,2
          4442 DASLOOP:
  4442   35401         LDA     3,P.NX,3 ;PICK UP POINTER TO NEXT ENTRY
  4443  175015         SNZ     3,3      ;END OF CHAIN ?
  4444     411         JMP     DASNF    ;   YES, NOT FOUND
  4445   21404         LDA     0,P.DA,3 ;NO, PICK UP ENTRY'S RDA
  4446  106414         SEQ     0,1      ;DOES IT MATCH ?
  4447     773         JMP     DASLOOP  ;   NO, KEEP SEARCHING
  4450   21403         LDA     0,P.LU,3 ;YES, TRY LU
  4451   25016         LDA     1,D.LU,2
  4452  106414         SEQ     0,1      ;DOES LU MATCH TOO ?
  4453     766         JMP     DAS2LOOP ;   NO, BACK TO SEARCH
  4454     723         JMP     COPYCA   ;YES, COPY ITS CA

  4455   10415 DASNF:  ISZ     DPSNFND+1;INCREMENT NOT-FOUND COUNTER
  4456     723         JMP     SETPE.RETURN;SET PE = 0 (A3 STILL = 0) AND RETURN
  4457   10412         ISZ     DPSNFND  ;OVERFLOW, INCREMENT 2ND WORD TOO
  4460     721         JMP     SETPE.RETURN
  4461   14410         DSZ     DPSNFND  ;LIMIT 2-WORD NUMBER TO 2^32-1
  4462     717         JMP     SETPE.RETURN

  4463    2002 VFFUF:         FF!UF&MSK;LUFLUSH
  4464    2000 VFF:           FF&MSK   ;FILEFLUSH
  4465   60000 VCL:           CB!CF    ;"CLEAR" SUFFIX
  4466    5127 .LUERROR:      LUERROR
  4467       0 DPSTOTAL:      0        ;ACTIVITY COUNTERS
  4470       0                0
  4471       0 DPSNFND:       0
  4472       0                0
```

```
;                        << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------
;; ROUTINES TO PUT POOL ENTRY INTO A DIFFERENT PLACE IN THE CHAIN
;;-------- ------- ----------------------------

;  ====
;((50)) MARK BUFFER VIRGIN AND PUT AT LRU END OF CHAIN
;  ====

          4473 MKVIRGIN:
    4473  20427        LDA     0,C.VIRGIN
    4474  41402        STA     0,P.FL,3
    4475 102000        ADC     0,0
    4476  41403        STA     0,P.LU,3
    4477  41404        STA     0,P.DA,3
    4500  30617        LDA     2,.BPLRU ;PREPARE TO PUT AT LRU END
    4501    402        JMP     BPTRELINK


;  ====
;((51)) PUT BUFFER POOL TABLE ENTRY AT MRU END OF CHAIN
;  ====

          4502 PUTMRU:
    4502 152400        SUB     2,2      ;FLAGS MRU END
          4503 BPTRELINK:
    4503  60277        INTDS
    4504  21400        LDA     0,P.PR,3 ;FIRST UNLINK ENTRY (A3)
    4505  43401        STA     0,@P.NX,3
    4506  21401        LDA     0,P.NX,3
    4507  43400        STA     0,@P.PR,3
    4510 151015        SNZ     2,2      ;INSERT AT MRU END ?
    4511  30602        LDA     2,BPMRU  ;   YES (CANNOT LDA EARLIER BEC. BPMRU CAN CHANGE)
    4512  51401        STA     2,P.NX,3 ;NOW LINK IT IN FRONT OF ENTRY (A2)
    4513  21000        LDA     0,P.PR,2
    4514  41400        STA     0,P.PR,3
    4515  57000        STA     3,@P.PR,2
    4516 161400        INC     3,0      ;COMPUTE ADDRESS OF P.NX (A3)
    4517  41000        STA     0,P.PR,2 ;COMPLETE THE LINKAGE
    4520  60177        INTEN
    4521    661        JMP     DPEX2RETURN


    4522   4000 C.VIRGIN:    VIRGIN
    4523  10000 C.INFIXED:   INFIXED
```

```
;                          << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

;;----------  ----------------------------------
;;  ROUTINES TO MARK THE TABLE ENTRY POINTED AT BY D.PE
;;----------  ----------  ----------  ----------


;  ====
;((60)) MARK BUFFER POOL TABLE ENTRY AS PRIVATE
;  ====

                4524 MKPRIVATE:
     4524   20007          LDA     0,RTP
     4525   41403          STA     0,P.LU,3
     4526   26406          LDA     1,@.C.PRI
                4527 MARKIT:
     4527   21402          LDA     0,P.FL,3
     4530  107415          AND#    0,1,SNR
     4531  123000          ADD     1,0
                4532 MKEXIT:
     4532   41402          STA     0,P.FL,3
                4533 DPEX3RETURN:
     4533     647          JMP     DPEX2RETURN

     4534    4314 .C.PRI:         C.PRIVATE


;  ====
;((61)) MARK LU/RDA IN BUFFER POOL TABLE ENTRY FROM LU/RDA IN NODE
;  ====  ALSO MARK BUFFER PUBLIC WITH LATCH COUNT = 0

                4535 MKLUDA:
     4535   21016          LDA     0,D.LU,2
     4536   41403          STA     0,P.LU,3
     4537   21017          LDA     0,D.DA,2
     4540   41404          STA     0,P.DA,3
     4541   20761          LDA     0,C.VIRGIN
     4542     7/0          JMP     MKEXIT
```

```
;                       << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

;  ====
;((62)) MARK BUFFER POOL TABLE ENTRY AS DIRTY EXCEPT
;  ====  IF NO-DIRTY-BUFFERS BIT (MSB) IN INFO TABLE SPECIAL CONDITION
;        FLAGWORD IS SET, THEN DISKWRITE

            4543 MKDIRTY:
      4543  21017          LDA     0,D.DA,2
      4544 101015          SNZ     0,0         ;RDA = 0 ?
      4545   2432          JMP     @.DAERROR;   YES, CANNOT MARK BLOCK 0 DIRTY
      4546  22461          LDA     0,@I.SPCF
      4547  24431          LDA     1,C.DIRTY
      4550 101113          SSN     0,0         ;ARE DIRTY BUFFERS ALLOWED ?
      4551    756          JMP     MARKIT      ;   YES, MARK IT DIRTY
      4552  21403          LDA     0,P.LU,3
      4553  41016          STA     0,D.LU,2 ;COPY PE'S LU, DA, CA --> TCN
      4554  21404          LDA     0,P.DA,3
      4555  41017          STA     0,D.DA,2
      4556  21405          LDA     0,P.CA,3
      4557  41020          STA     0,D.CA,2
      4560   2401          JMP     @.+1
      4561   5140          DISKWRITE


;  ====
;((63)) MARK BUFFER POOL TABLE ENTRY AS NOT DIRTY:
;  ====

            4562 MKNOTDIRTY:
      4562  24446          LDA     1,X.DIRTY
      4563  21402          LDA     0,P.FL,3
      4564 123400          AND     1,0
      4565    745          JMP     MKEXIT


;  =======
;((64,65)) BUFFER POOL LATCH AND UNLATCH ROUTINES
;  =======

            4566 MKLATCHED:
      4566 126521          SUBZL   1,1,SKP   ;ADD 1 TO LATCH COUNT
            4567 MKUNLATCH:
      4567 126000          ADC     1,1       ;ADD -1 TO LATCH COUNT
      4570  21402          LDA     0,P.FL,3 ;PICK UP WD CONTAINING LATCH COUNT
      4571 123000          ADD     1,0       ;MODIFY IT
      4572  24433          LDA     1,C.LCGUARDBIT
      4573 107415          AND#    0,1,SNR   ;DID IT PRODUCE OVER/UNDERFLOW ?
      4574    736          JMP     MKEXIT    ;   NO, STORE THE MODIFIED WORD
      4575   6606          JSR     @ZDPERROR
      4576     17          17                ;YES!! LATCHING ERROR


      4577   5134 .DAERROR:     DAERROR
      4600   1000 C.DIRTY:      DIRTY
```

```
;                        << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>

;  ====
;((66)) MARK BUFFER POOL TABLE ENTRY AS HAVING A COPY IN A FIXED BUFFER
;  ====  IF THAT IS THE CASE
;        ALSO, IF IT'S BSA, CLEAR BSA-CHANGED FLAG
;        IF THE FIXED BUFFER'S PRIOR P. CELL IS NOT -1, CLEAR IN-FIXED
;        BIT OF THE POOL TABLE ENTRY IT POINTS TO

            4601 MKINFIXEDBUFF:
    4601   31035         LDA     2,AUXL,2 ;PICK UP CALLER'S CORE ADDRESS
    4602   21000         LDA     0,A2,2
    4603   24010         LDA     1,.BSA
    4604  122705         SUBS    1,0,SNR  ;IS IT BSA ?
    4605   40075         STA     0,BSACF  ;   YES, CLEAR BSACF
    4606   24024         LDA     1,C4
    4607  106033         SLS     0,1      ;IS IT A FIXED BUFFER ?
    4610     723         JMP     DPEX3RETURN;  NO, RETURN
    4611   30415         LDA     2,.P.BSA ;YES, GET POINTER TO P.BSA
    4612  113000         ADD     0,2      ;CALCULATE OUR P. CELL
    4613   21000         LDA     0,0,2    ;PICK UP PRIOR P. CELL
    4614   55000         STA     3,0,2    ;SET P. CELL TO NEW BPTE
    4615  111405         INC     0,2,SNR  ;IS PRIOR P. CELL = -1 ?
    4616     405         JMP     MKIN2    ;   YES
    4617   21001         LDA     0,P.FL-1,2;NO, THEN CLEAR ITS IN-FIXED BIT
    4620   24411         LDA     1,X.INFIXED
    4621  123400         AND     1,0
    4622   41001         STA     0,P.FL-1,2
    4623   24700 MKIN2:  LDA     1,C.INFIXED;MARK NEW BPTE AS IN-FIXED
    4624     703         JMP     MARKIT


    4625     400 C.LCGUARDBIT:LCGUARDBIT
    4626    3417 .P.BSA:      P.BSA
    4627     606 I.SPCF:      INFO+SPCF.;SPCL.COND. FLAGWORD IN INFO TABLE
    4630  176777 X.DIRTY:     -1-DIRTY
    4631  167777 X.INFIXED:   -1-INFIXED
```

```
;                       << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

;;----------  ----------------  ---------------
;;  ROUTINES TO COPY A DATA BLOCK
;;----------  ----------------------------------

;  THESE ROUTINES ARE USED IF CALLER'S DESIRED DISK BLOCK IS FOUND
;  IN A POOL BUFFER.
;
;         IF WRITE, COPY TO POOL BUFFER FROM CALLER'S (A2)
;         IF READ, COPY FROM POOL BUFFER TO CALLER'S (A2)

;  SPECIAL CONSIDERATION FOR FIXED BUFFERS:
;  IF POOL ENTRY HAS IN-FIXED BIT SET (IE. THE BLOCK IS ALSO IN A FIXED BUFFER),
;  THEN FIND THE FIXED BUFFER WHOSE P. CELL EQUALS OUR PE, SET THAT
;  P. CELL = -1, AND CLEAR THE POOL ENTRY'S IN-FIXED BIT.  (NOTE:
;  IF CALLER'S A2 IS THE FIXED BUFFER, THIS LINKAGE WILL BE RE-ESTABLISHED
;  LATER, IN "MKINFIXED").

;  IF DOING A READ, AND CALLER'S A2 = FIXED BUFFER WHICH ALREADY CONTAINS
;  THE DESIRED BLOCK, BY-PASS THE COPY.


;  ====
;((70)) COPY TO POOL FROM USER'S BUFFER
;  ====

            4632 COPYTOPOOL:
    4632 101041         MOVO    0,0,SKP   ;C = 1 MEANS COPY TO POOL (WRITE)
                        ;
;  ====
;((71)) COPY FROM POOL TO USER'S BUFFER
;  ====
                        ;
            4633 COPYFROMPOOL:
    4633 101020         MOVZ    0,0       ;C = 0 MEANS COPY FROM POOL (READ)
    4634  21402         LDA     0,P.FL,3
    4635  24666         LDA     1,C.INFIXED
    4636 107405         AND     0,1,SNR   ;IS BLOCK ALSO IN A FIXED BUFFER ?
    4637    423         JMP     COPY2     ;  NO, COPY AS IS
                        ;
                        ; ***** NO CODE HERE *****
```

```
;                       << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>

     4640 122460          SUBC    1,0         ;PRESERVING CARRY, . . .
     4641  41402          STA     0,P.FL,3    ;CLEAR IN-FIXED BIT IN POOL TABLE ENTRY
     4642  30764          LDA     2,.P.BSA    ;PICK UP POINTER TO P. CELLS
          4643 TRANSLOOP:
     4643  21373          LDA     0,P..-P.BSA,2;AO = A(A FIXED BUFFER)
     4644 101015          SNZ     0,0         ;END OF TABLE ?
     4645   6142          TRAPFAULT           ;   YES !? WITHOUT FINDING MATCH !?
     4646  25000          LDA     1,0,2       ;PICK UP P. CELL
     4647 151400          INC     2,2
     4650 136414          SEQ     1,3         ;MATCH FOUND ?
     4651    772          JMP     TRANSLOOP;   NO
     4652 126000          ADC     1,1         ;YES
     4653  45377          STA     1,-1,2      ;SET FIXED BUFFER'S P. CELL TO -1
     4654  30015          LDA     2,TASKQ
     4655 101012          MOV#    0,0,SZC     ;ARE WE DOING A READ ?
     4656    404          JMP     COPY2       ;   NO, THEN DO THE COPY
     4657  27035          LDA     1,@AUXL,2;YES, AND . . .
     4660 106415          SNE     0,1         ;. . . IS CALLER'S A2 = A(FIXED BUFFER) ?
     4661    652          JMP     DPEX3RETURN;YES, THEN SKIP THE COPYING
     4662  25405 COPY2:   LDA     1,P.CA,3
     4663  33035          LDA     2,@AUXL,2;A2 = CALLER'S A2
     4664 132415          SNE     1,2         ;COPY IN PLACE ?
     4665    646          JMP     DPEX3RETURN;   YES, DON'T BOTHER
     4666 141003          MOV     2,0,SNC     ;COPY IN WHICH DIRECTION ?
     4667 121001          MOV     1,0,SKP     ;   FROM POOL
     4670 131000          MOV     1,2         ;   TO POOL
     4671  24064          LDA     1,C377
     4672 107000          ADD     0,1
     4673   6101          CALL
     4674 100015          MOVEWORDS
     4675    636          JMP     DPEX3RETURN
```

```
;                        << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

;;-------- ----------------------------------------
;; ROUTINES TO PREPARE FOR A FILE READ OR WRITE
;;-------- ----------------------------------------


;  ====
;((80)) CHECK THE BUFFER POINTED AT BY OUR D.PE TO MAKE SURE IT IS A VALID
;  ====  FILE HEADER BLOCK
;        IF IT IS, INITIALIZE D.T8 TO TOTAL NUMBER OF BLOCKS TO BE TRANSFERRED
;        (NOT INCLUDING HEADER), AND D.HP TO POINT TO WORD 200 OF THE BLOCK

          4676 CKHEADER:
  4676   31405         LDA     2,P.CA,3 ;A2 --> HEADER BLOCK
  4677   21177         LDA     0,DHDR,2
  4700   25404         LDA     1,P.DA,3
  4701  106414         SEQ     0,1      ;CONTAINS ITS OWN RDA ?
  4702    4505         JSR     HDRERROR ;  NO, NOT A LEGAL HEADER
  4703   21176         LDA     0,UNIT,2
  4704   25403         LDA     1,P.LU,3
  4705  106414         SEQ     0,1      ;CONTAINS ITS OWN LU ?
  4706    4501         JSR     HDRERROR ;  NO, NOT LEGAL HEADER
  4707   21010         LDA     0,TYPE,2
  4710   24043         LDA     1,C37
  4711  123400         AND     1,0      ;EXTRACT TYPE CODE
  4712   24042         LDA     1,C20
  4713  106033         SLS     0,1      ;LEGAL RANGE FOR FILEREAD/FILEWRITE ?
  4714    4473         JSR     HDRERROR ;  NO
  4715   21012         LDA     0,STAT,2
  4716  101212         SKE     0,0      ;EXTENDED FILE ?
  4717    4470         JSR     HDRERROR ;  YES, NOT LEGAL
  4720   24473         LDA     1,VPF
  4721   34015         LDA     3,TASKQ
  4722   21422         LDA     0,D.CW,3
  4723  107415         AND#    0,1,SNR  ;PARTIAL FILE ?
  4724     417         JMP     NOTPF    ;  NO
  4725   35435         LDA     3,AUXL,3 ;  YES, GET A(CALLERS TCN)
  4726   21402         LDA     0,A0,3   ;# OF BLOCKS
  4727   25401         LDA     1,A1,3   ;RDA LIST OFFSET
  4730   34053         LDA     3,C200
  4731  101015         SNZ     0,0      ;LEGAL # OF BLOCKS ?
  4732    4457         JSR     PFERROR  ;  NO
  4733  123000         ADD     1,0
  4734  125133         MOVZL#  1,1,SNC  ;NEG. OFFSET ?
  4735  162032         SGE     3,0      ;  NO, OUT OF RANGE
  4736    4453         JSR     PFERROR  ;  YES
  4737  122400         SUB     1,0      ;RESTORED A0 TO # OF BLOCKS
  4740  167000         ADD     3,1
  4741   34015         LDA     3,TASKQ
  4742     421         JMP     PFINT
  4743   21173 NOTPF:  LDA     0,CSIZ,2
  4744  101015         SNZ     0,0      ;CURRENT SIZE GIVEN ?
  4745     405         JMP     USE.NBLK ;  NO, USE NBLK
  4746   24064         LDA     1,C377   ;YES, CONVERT TO # BLOCKS
  4747  123300         ADDS    1,0      ;(ROUNDING UP)
  4750  123400         AND     1,0
  4751     404         JMP     D.T8.COMPUTED
```

```
;                    << SI = R9OREXDPMPSB;  BO = 1/A.REXM5.7033! >>

          4752 USE.NBLK:
4752   21011          LDA     0,NBLK,2
4753  100400          NEG     0,0         ;# BLOCKS = NBLK - 1
4754  100000          COM     0,0
          4755 D.TB.COMPUTED:
4755   34015          LDA     3,TASKQ
4756   41427          STA     0,D.TB,3 ;INITIALIZE TOT.BLKS TO BE TRANSF.
4757   24053          LDA     1,C200
4760  106033          SLS     0,1         ;MORE THAN 127 BLOCKS ?
4761    4426          JSR     HDRERROR ;   YES, TOO MUCH
4762  121000          MOV     1,0
4763   45413 PFINT:   STA     1,D.IP,3 ;STORE INITIAL HEADER POINTER (200 UNLESS PF)
4764  133000          ADD     1,2         ;A2 --> FIRST RDA IN HEADER
4765   51424          STA     2,D.HP,3 ;INITIALIZE D.HP
4766  114400          NEG     0,3         ;A3 = -200 = LOOP COUNTER
4767  126400          SUB     1,1         ;A1 WILL ACCUMULATE # RDA'S
          4770 CNTBLOCKS:                 ;COUNT # RDA'S IN HEADER
4770   21000          LDA     0,0,2
4771  101014          SKZ     0,0         ;FOUND ONE ?
4772  125400          INC     1,1         ;   YES, COUNT IT
4773  151400          INC     2,2
4774  175404          INC     3,3,SZR  ;CHECKED 128 LOCATIONS ?
4775     773          JMP     CNTBLOCKS;   NOT YET
4776   30015          LDA     2,TASKQ  ;YES
4777   21022          LDA     0,D.CW,2
5000   34413          LDA     3,VPF
5001  117414          AND#    0,3,SZR  ;PARTIAL FILE ?
5002   45027          STA     1,D.TB,2 ;   YES
5003   21027          LDA     0,D.TB,2
5004  106432          SGR     0,1         ;BLKS TO TRANSF.> # RDA'S IN HEADER ?
5005    2513          JMP     @.DPREI  ;   NO, HEADER IS OK
5006    4401          JSR     HDRERROR ;YES, NOT A GOOD FILE HEADER


          5007 HDRERROR:
5007    6405          JSR     @.DP1ERROR
5010      12          12                  ;ERROR IN FILE HEADER

          5011 PFERROR:
5011    6403          JSR     @.DP1ERROR
5012      27          27                  ;ERROR IN PARTIAL FILE OPERATION

5013   10000 VPF:     PF

5014    3756 .DP1ERR:          DPERRETURN
```

```
;                        << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

;  ====
;((81)) DFILREAD/WRITE: GATHER A GROUP OF CONTIGUOUS BLOCKS FROM FILE HEADER
;  ====        FOR A MULTI-BLOCK DISK TRANSFER, BYPASSING THE POOL


;  ON ENTRY, A2 --> TASK CONTROL NODE, WHICH CONTAINS:

;  THE FOLLOWING VALUES REMAIN FIXED:
;       D.PE --> POOL TABLE ENTRY OF HEADER BLOCK
;       D.CW = DATAPUMP CONTROL WORD
;       D.LU = LU OF FILE

;  THE FOLLOWING VALUES ARE UPDATED BY GATHER.CONTIG:
;       D.HP = HEADER BLOCK POINTER --> RDA WHERE NEXT CONTIG. AREA
;              MAY START (INITIALLY = WORD 200 OF HEADER BLOCK)
;       D.TB = TOTAL NO. BLOCKS REMAINING TO BE TRANSFERRED

;  THE FOLLOWING VALUES ARE GENERATED BY GATHER.CONTIG:
;       D.BC = BLOCK COUNT FOR THIS TRANSFER
;       D.DA = RDA OF FIRST BLOCK IN CONTIGUOUS GROUP
;       D.CA = FILE CORE ADDRESS WHERE CONTIGUOUS AREA STARTS


          5015 GATHER.CONTIG:
  5015  21027        LDA     0,D.TB,2
  5016 101015        SNZ     0,0        ;ANY BLOCKS TO BE TRANSFERRED ?
  5017   2501        JMP     @.DPRET    ;   NO, JUST RETURN
          5020 GCSCAN:                  ;SCAN HEADER FOR FIRST NON-ZERO RDA
  5020  23024        LDA     0,@D.HP,2  ;PICK UP AN RDA ENTRY
  5021 101014        SKZ     0,0        ;BLANK ?
  5022    403        JMP     GCSTART    ;   NO, START GATHERING BLOCKS
  5023  11024        ISZ     D.HP,2     ;YES, SCAN ON
  5024    7/4        JMP     GCSCAN
```

```
;                    << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

          5025 GCSTART:
5025   41017          STA     0,D.DA,2 ;SET FIRST BLOCK'S RDA
5026   21024          LDA     0,D.HP,2 ;PICK UP ITS HEADER POINTER
5027   35405          LDA     3,P.CA,3 ;CORE ADDRESS OF HEADER BLOCK
5030  162400          SUB     3,0      ;CALCULATE RELATIVE POSITION IN HEADER
5031   25013          LDA     1,D.IP,2
5032  122700          SUBS    1,0      ;CONVERT TO #WORDS OFFSET
5033   25575          LDA     1,CORA,3
5034  123022          ADDZ    1,0,SZC  ;ADD TO FILE'S CORA -- WRAP AROUND ?
5035    4752          JSR     HDRERROR ;   YES !!
5036   41020          STA     0,D.CA,2 ;NO, SAVE AS STARTING CORE ADDRESS
5037   25017          LDA     1,D.DA,2 ;RECOVER FIRST BLOCK'S RDA
5040   35022          LDA     3,D.CW,2 ;DATAPUMP CONTROL WORD
5041  102400          SUB     0,0
5042   41026          STA     0,D.BC,2 ;INITIALIZE BLOCK COUNT FOR DFILREAD/WRITE
          5043 GCAPPEND:                ;NOW SEE HOW MANY CONTIGUOUS BLOCKS THERE ARE
5043   11026          ISZ     D.BC,2   ;COUNT THE BLOCK
5044   15027          DSZ     D.TB,2   ;END OF TOTAL BLOCKS TO TRANSFER ?
5045  125401          INC     1,1,SKP  ;   NO, PREPARE FOR NEXT RDA
5046    2452          JMP     @.DPRET  ;   YES, RETURN
5047   11024          ISZ     D.HP,2   ;INCREMENT HEADER POINTER
5050   23024          LDA     0,@D.HP,2;PICK UD NEXT RDA
5051  106415          SNE     0,1      ;IS IT CONTIGUOUS ?
5052     7/1          JMP     GCAPPEND ;   YES, APPEND IT
5053    2445          JMP     @.DPRET  ;NO, RETURN
```

```
;                    << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>

;  ====
;((82))  FILEREAD/FILEWRITE: READ, WRITE A FILE IS MERELY FILEFLUSH, FOLLOWED
;  ====       BY DFILREAD, OR DFILWRITE DEPENDING ON WHEHTER A READ OR WRITE

            5054 RWAFILE:
     5054   35022          LDA     3,D.CW,2
     5055   20444          LDA     0,KCFPF
     5056  117400          AND     0,3
     5057   24444          LDA     1,FFCW
     5060  137000          ADD     1,3
     5061   54412          STA     3,FFCW1   ; PASS ON THE CLEAR BIT TO THE FILE FLUSH
     5062   20444          LDA     0,KNCNF   ;NOW CLEAR THE CF (CLEAR FILE) BIT IN THE CONTROL WORD
     5063   35022          LDA     3,D.CW,2
     5064  117400          AND     0,3
     5065   55022          STA     3,D.CW,2
     5066   35035          LDA     3,AUXL,2
     5067   21402          LDA     0,A0,3
     5070   25401          LDA     1,A1,3
     5071   31020          LDA     2,D.CA,2 ;RESTORE A2 FROM ORIGINAL CALL
     5072    6110          DATAPUMP
     5073      25 FFCW1:   FILEFLUSH         ; MODIFIED FOR PASSING ALONG CLEAR FLAG
     5074  100000          @0
     5075   34015          LDA     3,TASKQ   ; SE A3 TO A(TCN)
     5076   21422          LDA     0,D.CW,3
     5077   24423          LDA     1,VFW
     5100   34424          LDA     3,DFRCW
     5101   24421          LDA     1,VFW
     5102  107414          AND#    0,1,SZR   ;FILE WRITE ?
     5103   34422          LDA     3,DFWCW   ;   YES
     5104   24707          LDA     1,VPF
     5105  107400          AND     0,1
     5106  137000          ADD     1,3       ;PASS ALONG PF SUFFIX
     5107   54406          STA     3,DFCW1
     5110   34015          LDA     3,TASKQ
     5111   35435          LDA     3,AUXL,3
     5112   21402          LDA     0,A0,3
     5113   25401          LDA     1,A1,3
     5114    6110          DATAPUMP
     5115      22 DFCW1:   DFILREAD          ;MODIFIED FOR READ OR WRITE AND PASS ALONG PF FLAG
     5116  100000          @0
     5117    2401          JMP     @.DPRET

     5120    4016 .DPRET:        DPEXRETURN
     5121   30000 KCFPF:CF+PF
     5122     100 VFW:   FW&MSK
     5123      25 FFCW:  FILEFLUSH
     5124      22 DFRCW:DFILREAD
     5125      23 DFWCW:DFILWRITE
     5126  157777 KNCNF:-1-CF
```

```
;                       << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

;;-----------  --------------------  --------------------
;;  DISK TRANSFER -- THIS IS THE ONLY ROUTINE THAT ACTUALLY CALLS
;;  THE DISK DRIVER
;;-----------  --------------------  --------------------


;  ON ENTRY,
;        A2 = A(TCN)

;  THE TASK CONTROL NODE (TCN) MUST CONTAIN:
;      D.LU = LU
;      D.DA = RDA
;      D.CA = CORE ADDRESS
;      D.BC = BLOCK COUNT

;  DISKTRANSFER USES THE FOLLOWING WORDS IN THE TCN:
;      D.LF = LUFIX POINTER
;      D.LV = LUVAR POINTER
;      D.WR = DISK WRITE FLAG
;      D.T1 = # BLOCKS TO BE XFERRED BY DISK (VS. LCM/LXM/MAP)
;      D.TS = TEMP.  STORE BLOCK COUNT ACCEPTED
;      D.TO = TIME-OUT COUNTER

;  NO OTHER WORDS IN TCN ARE TOUCHED

;  NORMAL RETURN IS TO DPEXRETURN WITH NODE ENTRIES AS SHOWN ABOVE
;  ABNORMAL RETURN IS TO DPERRETURN WITH A0 = ERROR CODE (TRAP NO.)

;      ERROR CODE          MEANING
;          1           WRITE PROTECTED
;          2           NO SUCH DISK
;          3           UNCORRECTABLE DISK ERROR
;          4           TIME-OUT
;          5           ILLEGAL DISK ADDRESS
;          6           DISK BUSY AT START OF TRANSFER
;          7           LU INACTIVE


          5127 LUERROR:
5127   20027         LDA     0,C7
          5130 DKERR:
5130  176400         SUB     3,3
5131   54551         STA     3,INDRVRFLG;RESET "IN DISK DRIVER" STATE
5132    2401         JMP     @.+1
5133    3757         DPERO

          5134 DAERROR:
5134   20025         LDA     0,C5
5135     773         JMP     DKERR

          5136 BUSYERROR:
5136   20026         LDA     0,C6
5137     771         JMP     DKERR
```

```
;                        << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

; PREPARE FOR A DISK TRANSFER

; ====
;((90)) WRITE THE BLOCK AT D.CA TO DISK AT D.LU, D.DA
; ====
          5140 DISKWRITE:
    5140  21017         LDA     0,D.DA,2
    5141 101015         SNZ     0,0         ;TRYING TO WRITE OVER BZUD ?
    5142    472         JSR     DAERROR     ;   YES, ILLEGAL DISK ADDRESS
    5143  36524 DWOK;   LDA     3,@I.SCO
    5144  20524         LDA     0,VRO
    5145 117414         AND#    0,3,SZR     ;READ ONLY MODE ?
    5146   2752         JMP     @.DPRET     ;   YES, IGNORE
    5147 102520         SUBZL   0,0         ;SET DISK WRITE FLAG = 1
    5150  10525         ISZ     DPWREQ+1
    5151    414         JMP     DSKRW
    5152  10522         ISZ     DPWREQ
    5153    412         JMP     DSKRW
    5154  14520         DSZ     DPWREQ
    5155    410         JMP     DSKRW

; ====
;((91)) READ FROM DISK THE BLOCK AT D.LU, D.DA TO CORE AT D.CA
; ====
          5156 DISKREAD:
    5156 102400         SUB     0,0         ;SET DISK WRITE FLAG = 0
    5157  10514         ISZ     DPRREQ+1
    5160    405         JMP     DSKRW
    5161  10511         ISZ     DPRREQ
    5162    403         JMP     DSKRW
    5163  14507         DSZ     DPRREQ
    5164    401         JMP     DSKRW
    5165  41025 DSKRW:  STA     0,D.WR,2
    5166  21016         LDA     0,D.LU,2
    5167   6123         FINDLUT
    5170   4737         JSR     LUERROR     ;LU INACTIVE OR ILLEGAL -- TRAP 7
    5171 141000         MOV     2,0
    5172  30015         LDA     2,TASKQ
    5173  41031         STA     0,D.LF,2    ;SAVE LUFIX & LUVAR PNTRS IN TCN
    5174  55032         STA     3,D.LV,2
    5175  21017         LDA     0,D.DA,2
    5176  25026         LDA     1,D.BC,2
    5177 125015         SNZ     1,1         ;# BLOCKS TO TRANSFER = 0 ?
    5200   2720         JMP     @.DPRET     ;   YES, JUST RETURN
```

LDA 0,D.CW,2
LDA 1,C.DWR
SNE 0,1 ;DOING D.WRIT
JMP DWOK ; YES, DON'T C
BLK ZERO

RB.
2-29-88

C.DWR: DW ; D. write

```
;                       << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

     5201   21026 CKLCM: LDA     0,D.BC,2
     5202   35032        LDA     3,D.LV,2
     5203   25403        LDA     1,LVLUC,3;Get LU characteristics word
     5204   34460        LDA     3,VNXMB
     5205  137414        AND#    1,3,SZR   ;Extended memory buffering allowed?
     5206     416        JMP     NOXMEM    ;  No - don't try it.
     5207   26460        LDA     1,@I.SCO  ;  Yes
     5210   34456        LDA     3,VLC
     5211  137414        AND#    1,3,SZR   ;LCM ACTIVE ?
     5212     407        JMP     DOLCM     ;  YES
     5213   34452        LDA     3,VMM
     5214  137415        AND#    1,3,SNR   ;MAP ACTIVE ?
     5215     407        JMP     NOXMEM    ;  NO
     5216   34100        LDA     3,.INFO   ;YES
     5217    7500        JSR     @.MAP.,3
     5220     402        SKIP
     5221    6114 DOLCM: JSR     @.LCM     ;USE $LCM OR $LXM
     5222  101015        SNZ     0,0       ;HAS LCM DONE ALL ?
     5223     561        JMP     JRSTBC    ;  YES, RESTORE D.BC IN TCN TO DEFAULT = 1
          5224 NOXMEM:
     5224   41014        STA     0,D.T1,2  ;NO
     5225   25017        LDA     1,D.DA,2  ;FIRST DISK ADDRESS TO TRANSFER
     5226  123000        ADD     1,0       ;COMPUTE LAST DISK ADDRESS TO TRANSFER + 1
     5227   35032        LDA     3,D.LV,2  ;LUVAR
     5230   25410        LDA     1,FUDA,3  ;FIRST ADDRESS BEOND END OF LOGICAL UNIT
     5231  106433        SLE     0,1       ;WITHIN LU ?
     5232     702        JMP     DAERROR   ;  NO, ILLEGAL DISK ADDRESS
     5233   21025        LDA     0,D.WR,2
     5234  101212        SKE     0,0       ;WRITE REQUEST ?
     5235     407        JMP     D2WRITE   ;  YES
     5236   10441        ISZ     DPDREAD+1;NO, INCREMENT DISKREAD COUNT
     5237     412        JMP     D2RW
     5240   10436        ISZ     DPDREAD   ;INCRMENT OVERFLOW COUNTER
     5241     410        JMP     D2RW
     5242   14434        DSZ     DPDREAD   ;PEG AT 177777
     5243     406        JMP     D2RW

          5244 D2WRITE:
     5244   10435        ISZ     DPDWRITE+1
     5245     404        JMP     D2RW
     5246   10432        ISZ     DPDWRITE
     5247     402        JMP     D2RW
     5250   14430        DSZ     DPDWRITE
     5251   21016 D2RW:  LDA     0,D.LU,2
     5252  101014        SKZ     0,0       ;USING SYSTEM DISK ?
     5253     431        JMP     SRETRYCOUNT;  NO, START TRANSFER
          5254 WTSYSTEMDISKREADY:          ;YES, MAKE SURE IT'S READY
     5254   35031        LDA     3,D.LF,2
     5255   31032        LDA     2,D.LV,2
     5256    7774        JSR     @SLUR,3   ;IS DRIVE READY ?
     5257  102401        SUB     0,0,SKP   ;  NO
     5260     424        JMP     SRETRYCOUNT;  YES, START TRANSFER
     5261   40074        STA     0,EISF    ;PREVENT TRAP 34
     5262   30015        LDA     2,TASKQ
     5263     771        JMP     WTSYSTEMDISKREADY;KEEP WAITING
```

```
;                       << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>
    5264   20000 VNXMB: LVLNXMB          ;LVLUC No-extended memory buffering bit
    5265      40 VMM:   MM
    5266    2000 VLC:   LC                ;LCM ACTIVE MASK
    5267     662 I.SCO: INFO+SCON.
    5270     200 VRO:   RO
    5271     673 I.THT: INFO+THTC.

;DISK READ/WRITE ACTIVITY COUNTERS

    5272       0 DPRREQ:       0
    5273       0               0
    5274       0 DPWREQ:       0
    5275       0               0
    5276       0 DPDREAD:      0
    5277       0               0
    5300       0 DPDWRITE:     0
    5301       0               0

    5302       0 INDRVRFLG:    0          ;ZERO=NOT IN DISK DRIVER
                                          ;NON-ZERO=A(TASK NODE OF TASK DOING DISK TRANSFER)
    5303       0 DMAPCW:       0          ;DMA MAP CONTROL WORD -- INITIALLY MUST = 0


; PERFORM THE ACTUAL DISK TRANSFER BY CALLING THE DISK DRIVER

          5304 SRETRYCOUNT:
    5304  20042        LDA     0,C20
    5305  40442        STA     0,RWBC+1 ;SET RETRY COUNT = 16.
    5306  30015        LDA     2,TASKQ
          5307 DSKRETRY:                 ;RE-ENTER HERE FOR RETRY
    5307  35031        LDA     3,D.LF,2 ;POINTER TO LUFIX
    5310  50772        STA     2,INDRVRFLG;SET "IN DISK DRIVER" STATE
    5311   7775        JSR     @SKNB,3  ;IS CONTROLLER ALREADY BUSY ?
    5312   4624        JSR     BUSYERROR;   YES, ERROR #6
    5313  22756        LDA     0,@I.THT ;10 HZ TICK COUNTER
    5314  24042        LDA     1,C20    ;  1.6 SECONDS
    5315 107000        ADD     0,1
    5316  30015        LDA     2,TASKQ
    5317  45033        STA     1,D.TO,2
    5320  21014        LDA     0,D.T1,2 ;# BLOCKS TO BE XFERRED BY DISK
    5321  24762        LDA     1,DMAPCW ;GET DATA CHANNEL MAP CONTROL WORD
          5322 MAPRETRY:
    5322  40424        STA     0,RWBC   ;SET BLOCK COUNT FOR DISK DRIVER
    5323  30015        LDA     2,TASKQ
    5324  35032        LDA     3,D.LV,2 ;LOAD ADDRESS OF LUVAR
    5325  31020        LDA     2,D.CA,2 ;CORE ADDRESS FOR TRANSFER
    5326  35402        LDA     3,DFLG,3 ;LOAD DRIVE FLAG
    5327 177120        ADDZL   3,3
    5330 177132        ADDZL#  3,3,SZC  ;DMA DEVICE ?
    5331    407        JMP     DONTMAP  ;   NO, DON'T USE DMA MAP
    5332  60277        INTDS
    5333  34110        DMCALL
    5334   7770        DMAMAP           ;REQUEST DATA CHANNEL MAPPING
    5335    765        JMP     MAPRETRY ;   NOT ENOUGH CONTIGUOUS BLOCKS AVAILABLE:
                                        ;   (AO) = MAX. AVAIL; RETRY WITH THAT NUMBER
    5336  44745        STA     1,DMAPCW ;REMEMBER DMA MAP CONTROL WORD FOR NEXT REQUEST
    5337  60177        INTEN
```

```
;                       << SI = R90REXDPMPSB;  BO = 1/A.REXM5.7033! >>

          5340 DONTMAP:
5340   34015         LDA     3,TASKQ
5341   21425         LDA     0,D.WR,3
5342  101200         MOVR    0,0             ;SET C = WRITE FLAG
5343   21432         LDA     0,D.LV,3 ;LUVAR POINTER
5344   25417         LDA     1,D.DA,3 ;RDA
5345    7431         JSR     @D.LF,3  ;USE THE DISK DRIVER <--------<<<<
5346       1 RWBC:   1                       ;BLOCK COUNT
5347       0         0                       ;RETRY COUNT (FOR HEAD OR TIMING OFFSET)
5350   6017/         INTEN
5351   30015         LDA     2,TASKQ
5352   45030         STA     1,D.TS,2 ;SAVE BLOCK COUNT ACCEPTED
          5353 WTNOTBUSY:
5353   30015         LDA     2,TASKQ
5354   35031         LDA     3,D.LF,2 ;LUFIX ADDRESS FORM NODE
5355   22714         LDA     0,@I.THT ;TEN HZ OVERFLOW TICK CONTER
5356   25033         LDA     1,D.TO,2 ;TIME OUT LIMIT
5357  106033         SLS     0,1             ;THIS DISK ACCES >= 1.6 SECONDS ?
5360     506         JMP     TIMEOUTERR;   YES
5361    7775         JSR     @SKNB,3  ;DISK STILL BUSY ?
5362     7/1         JMP     WTNOTBUSY;   YES, WAIT TILL IT ISN'T
5363   35031         LDA     3,D.LF,2 ;NO
5364   31032         LDA     2,D.LV,2
5365    77/6         JSR     @REDS,3  ;READ STATUS
5366  152400         SUB     2,2
5367   50713         STA     2,INDRVRFLG
5370   30015         LDA     2,TASKQ
5371   35031         LDA     3,D.LF,2
5372   35765         LDA     3,EMSK,3
5373  117414         AND#    0,3,SZR  ;ANY ERRORS ?
5374     426         JMP     DSKERROR ;   YES
5375   35014         LDA     3,D.T1,2 ;NO, GET REQUESTED BLOCK COUNT
5376   25030         LDA     1,D.TS,2 ;REIRIEVE BLOCK COUNT ACCEPTED
5377  125014         SKZ     1,1             ;DIDN'T ACCEPT ANY, OR
5400  136423         SUBZ    1,3,SNC  ;MORE THAN WE ASKED FOR ?
5401    6142         TRAPFAULT               ;   YES, SOV ERROR
5402   21026         LDA     0,D.BC,2 ;UPDATE BLK. CNT.
5403  122405         SUB     1,0,SNR  ;ANY LEFT TO DO ?
          5404 JRSTBC:
5404     467         JMP     RESTORBC ;   NO, RESTORE D.BC TO ITS DEFAULT: 1
5405   41026         STA     0,D.BC,2 ;YES, UPDATE BLOCK COUNT IN TCN
5406   21017         LDA     0,D.DA,2 ;ADVANCE THE RDA
5407  123000         ADD     1,0
5410   41017         STA     0,D.DA,2
5411  125300         MOVS    1,1
5412   21020         LDA     0,D.CA,2 ;AND THE CORE ADDRESS
5413  123000         ADD     1,0
5414   41020         STA     0,D.CA,2
5415  175015         SNZ     3,3             ;DID DISK DO PARTIAL ONLY ?
5416    2403         JMP     @.CKLCM  ;   NO, RECHECK LCM
5417   55014         STA     3,D.T1,2 ;YES, SET REM. BLK. CNT.
5420     66?         JMP     SRETRYCOUNT;RE-INITIATE TRANSFER

5421    5201 .CKLCM:          CKLCM
```

```
;                    << SI = R90REXDPMPSB; BO = 1/A.REXM5.7033! >>

;  ------------------------------------------------
;  LUFIX                              LUVAR      RETURN
;   EMSK                               ERRC      ERROR
;  OFFSET   MEANING                   OFFSET     CODE
;  ------   ---------------           ------     ------
;     1     WRITE PROTECTED                        1
;     2     NO SUCH DISK                           2
;     3     D.CHANNEL LATE               2         3
;     4     ADDRESS CK.ERROR             1         3
;     5     ILLEGAL DISK ADDR.                     5
;  (NONE)   DATA CHECK ERROR             0         3
;           TIME-OUT                     3         4
;  ------------------------------------------------

; DISK ERROR HANDLER: ENTERED WITH A2 --> TCN, A0 = STATUS WORD

          5422 DSKERROR:                 ;DISK ERROR HANDLER
  5422 126400         SUB     1,1
          5423 DSKELOOP:                 ;CHECK ALL LUFIX EMSK'S FOR ERROR ID.
  5423 125400         INC     1,1
  5424  34025         LDA     3,C5
  5425 136433         SLE     1,3        ;TRIED ALL EMSK'S ?
  5426    423         JMP     DATAERROR;   YES, DEFAULT = DATA ERROR
  5427  35031         LDA     3,D.LF,2   ;LUFIX POINTER
  5430 137000         ADD     1,3        ;ADD EMSK OFFSET
  5431  35765         LDA     3,EMSK,3
  5432 117415         AND#    0,3,SNR    ;MATCHES OUR ERROR ?
  5433    770         JMP     DSKELOOP   ;   NO, KEEP LOOKING
  5434 121000         MOV     1,0        ;LUFIX EMSK OFFSET
  5435  34003         LDA     3,C3
  5436 162654         SUBOR#  3,0,SZR    ;IS EMSK OFFSET = 3 OR 4 ?
  5437   6426         JSR     @.DKERR    ;   NO, HARD ERROR (A0=ERROR CODE)
  5440  24002         LDA     1,C2       ;ASSUME DATA CHANNEL LATE ERROR
  5441 116415         SNE     0,3        ;IS EMSK OFFSET = 3 ?
  5442    407         JMP     RETRIABLERROR;  YES, DCH LATE
  5443  35031         LDA     3,D.LF,2   ;NO, MUST BE #4 = ADDRESS ERROR
  5444  31032         LDA     2,D.LV,2
  5445 126000         ADC     1,1
  5446   7777         JSR     @SEEK,3    ;RECALIBRATE
  5447 126521         SUBZL   1,1,SKP    ;NOW SET ERROR OFFSET = 1
          5450 DATAERROR:
  5450 126400         SUB     1,1
          5451 RETRIABLERROR:            ;ENTER WITH A1 = ERROR LOG OFFSET
  5451  30015         LDA     2,TASKQ
  5452  35032         LDA     3,D.LV,2
  5453 137000         ADD     1,3        ;MAX. A1 = 3 !
  5454  11411         ISZ     ERRC,3     ;LOG ERROR COUNT
  5455    402         SKIP
  5456  15411         DSZ     ERRC,3     ;?! 64K ERRORS !?
  5457  14670         DSZ     RWBC+1     ;DECREMENT RETRY COUNT - 0 ?
  5460    627         JMP     DSKRETRY   ;   NO, TRY ONE MORE TIME
  5461  20003         LDA     0,C3       ;YES, ASSUME ERROR CODE 3
  5462 122415         SNE     1,0        ;ERROR LOG OFFSET = 3 (TIMOUT) ?
  5463 101400         INC     0,0        ;   YES, RETURN ERROR CODE 4
  5464   2401         JMP     @.DKERR    ;RECORD DISK ERROR
```

```
;                       << SI = R9OREXDPMPSB; BO = 1/A.REXM5.7033! >>

5465    5130 .DKERR:        DKERR

          5466 TIMEOUTERROR:
5466   31032        LDA     2,D.LV,2 ;LUVAR ADDRESS FROM NODE
5467  126000        ADC     1,1
5470    777/        JSR     @SEEK,3  ;RECALIBRATE
5471   24003        LDA     1,C3
5472     757        JMP     RETRIABLERROR


          5473 RESTORBC:              ;RESTORE BLOCK COUNT TO 1 AFTER USING $LCM
5473  102520        SUBZL   0,0
5474   41026        STA     0,D.BC,2
5475    2401        JMP     @.+1
5476    4016        DPEXRETURN



; LIST OF THE 4 (MAX.) CONTIGUOUS SETS OF POOL BUFFERS

                    ;THE FOLLOWING 8 VALUES ARE PUT HERE BY SIR:
                    ;
5477       0 BUF1:  0       ;FIRST BUFFER IN FIRST CONTIGUOUS GROUP
5500       0        0       ;NUMBER OF BUFFERS IN FIRST GROUP
5501       0        0       ;FIRST BUFFER IN 2ND CONTIGUOUS GROUP
5502       0        0       ;NUMBER OF BUFFERS IN 2ND GROUP
5503       0        0       ;FIRST BUFFER IN 3RD CONTIGUOUS GROUP
5504       0        0       ;NUMBER OF BUFFERS IN 3RD GROUP
5505       0        0       ;FIRST BUFFER IN 4TH CONTIGUOUS GROUP
5506       0        0       ;NUMBER OF BUFFERS IN 4TH GROUP
5507       0        0       ;DEFAULT TERMINATOR

            .EOT   ;REX "DPMP" FOR "IRIS" RESIDENT EXECUTIVE
```

```
;                          << SI = R90REXCALLSC; BO = 1/A.REXM5.7033! >>
; 29 OCT 86, RB.

;;- --------- ------------------ -------- ------------------------------
;;        R E X :   S Y S T E M   -   D I S C     C A L L
;;
;;    MAJOR COMPONENTS:
;;         1.   CALL A SYSTEM SUBROUTINE
;;         2.   CALL A DISC SUBROUTINE
;;         3.   CHANNEL OPERATIONS
;;
;;    UPDATE RECORD:
;;         6-12-80   (LLL)   COMMENTED FOR READABILITY
;;         8-13-81   (GAD)   CORRECTED CALL RETURN ROUTINE
;;         8-20-81   (GAD)   ADDED SYSTEM SUBR. RSFBX TO CALL TABLE.
;;                           UPDATED CALL RETURN LOGIC
;;         1-13-82   (rb.)   modified for new DATAPUMP
;;         5-21-83   (RMS)   added READ(WRIT)MAINTENANCE support to CHANNEL
;;       23 JAN 86    RB.    DISCSUBS IN XMEM (MK 12 OR MAP)
;;        2 MAR 86    RB.    DFT'S IN XMEM
;;       27 MAR 86   (PLR)   Add RENAME and WRITN to Attributes Key Table
;;
;;--- --- --- ------------------- ----- ------------------ ---------
```

```
;                        << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

;;-------------   ------------------- --------- ---------------------------------
;;
;;     1.   CALL A SYSTEM SUBROUTINE                      <<RE ENTRANT>>
;;
;;
;;
;;     PURPOSE:   DETERMINES IF A SYSTEM OR DISC SUBROUTINE IS BEING
;;                CALLED, AND BRANCHES TO "CALLD" IF IT IS A DISCSUB.
;;                OTHERWISE IT JUMPS TO THE APPROPRIATE SYSTEM SUBROUTINE.
;;
;;
;;
;;     ENTRY:     A0,A1,A2 <-- PARAMETERS FOR THE CALLED SUBROUTINE
;;                A3 <-- RETURN ADDR
;;                (A3)+1 <-- SUBROUTINE KEYWORD
;;
;;     EXIT:      A0,A1,A2 <-- SAME AS ENTRY
;;                IF (DISCSUB) THEN GOES TO "CALLD"
;;                     A3 <-- UNDEFINED
;;                     REGS+3 <-- RETURN ADDR
;;                     INTERRUPTS ARE DISABLED
;;                ELSE (SYSTEM SUBROUTINE) GOES TO SUBROUTINE
;;                     A3 <-- RETURN ADDRESS
;;                     ON A FINAL RETURN, SYSTEM SUBROUTINE RETURNS DIRECTLY
;;                        TO THE CALLER.
;;
;;
;;     CALLING SEQUENCE:          CALL
;;                                <SUBROUTINE KEYWORD>
;;
;;
;;       <SUBROUTINE KEYWORD>:  <INDEX > 0>                = DISCSUB
;;                              <msb = 1> + <INDEX > 0> = SYSTEM SUBROUTINE
;;
;;-------------  ---------------------- ---------------------------------

    5510      32        .BLK    32        ;RESEVED FOR 9.1 NEGATIVE OFFSET FROM CALL (101)
```

```
;                       << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

5542  60277 CALLS: INTDS              ;CALL A SYSTEM SUBROUTINE
5543  54424        STA     3,CATMP    ;SAVE KEYWORD POINTER
5544  35400        LDA     3,0,3      ;PICK UP KEYWORD
5545 175113        SSN     3,3        ;IS IT A "REX" SUBROUTINE ?
5546    430        JMP     CALLD      ;   NO
5547  40421        STA     0,CATMP+1  ;   YES
5550  22416        LDA     0,@SSTAB   ;PICK UP MAX # OF SYSTEM SUBROUTINES
5551 162033        SLS     3,0        ;DOES THE SUBROUTINE EXIST ?
5552   4564        JSR     CALLF      ;    NO, NO SUCH SUBROUTINE !!
5553  20413        LDA     0,SSTAB
5554 117120        ADDZL   0,3
5555 175220        MOVZR   3,3
5556  21401        LDA     0,1,3      ;SUBROUTINE ADDRESS
5557  40406        STA     0,SA.
5560  20410        LDA     0,CATMP+1
5561  34406        LDA     3,CATMP
5562 175400        INC     3,3        ;bump return address over subroutine id. word
5563  60177        INTEN
5564   2401        JMP     @SA.


5565      0 SA.:   0
5566   6427 SSTAB: CRST
5567      0 CATMP: 0                  ;TEMP CELLS
5570      0        0
5571   2000 UBIT:  U
5572    612 .UDSB: INFO+UDSB.
5573    616 .SDSB: INFO+SDSB.
5574    662 XSCON: INFO+SCON.
5575    100 MCT:   CT
```

```
;                    << SI = R90REXCALLSC; BO = 1/A.REXM5.7033! >>

;;---------- - ---------- ------ ---------- ----------------------------
;;
;;     2.   CALL A DISC SUBROUTINE                              <<NON REENTRANT>>
;;
;;     NOTE:   THIS ROUTINE IS NOT RE-ENTRANT, BUT THERE IS NO LOCK, OR
;;             QUEUEING MECHANISM TO PREVENT MULTIPLE CALLS.   THUS, THE
;;             FOLLOWING RESTRICTIONS EXIST:
;;             1.   DONT CALL THIS ROUTINE FROM INTERRUPT HANDLER
;;             2.   DONT CALL THIS ROUTINE FROM A NON-USER TASK
;;             3.   IF USERS GET DESCHEDULED ON DISC ACCESSES, THIS ROUTINE
;;                  WOULD HAVE TO BE RE-WRITTEN & SINGLE THREADED, OR MADE
;;                  SERIALLY RE-USABLE.
;;
;;     PURPOSE:   THIS ROUTINE PERFORMS NESTING FUNCTIONS FOR DISCSUBS, AS
;;            WELL AS LOADING & SAVING THEM, AND TRANSFERRING CONTROL TO THEM.
;;            It gets disc address from DAT and starting address from SAT.
;;                 There are two cases:
;;            DA > 0 means disc resident
;;            DA < 0 means core resident
;;
;;     ENTRY:     (FROM CALLS)
;;            INTERRUPTS ARE DISABLED
;;            REGS+3 <-- RETURN ADDRESS
;;            A0,A1,A2 <-- PARAMETERS FOR DISCSUB
;;            (A3)+1 <-- SUBROUTINE INDEX
;;
;;     EXIT:  CALLS SUBROUTINE.   WHEN SUBROUTINE IS FINISHED, IT RETURNS
;;            TO "CALLN" (NON-SKIP) OR "CALLR" (SKIP).
;;
;;     Note on extended discsubs: the first block of extended discsubs
;;            is not included in the nesting/unnesting mechanism.
;;            Therefore an extended discsub may not call an extended
;;            discsub (either directly or indirectly) from its first
;;            block, nor access its own first block after calling an
;;            extended discsub (or otherwise using HXA) from its
;;            second block.
;;---------- - ---------- ------ ---------- ----------------------------
```

```
;                        << SI = R9OREXCALLSC; BO = 1/A.REXM5.7033! >>

; STRUCTURE OF THE STACK: (based on NL = 6)

; STKP when empty     -->  0   (not part of stack)
; STKP in DS level 1  -->  1 : caller's return address
;                     2  -->  2 : return address to level 1
;                     3  -->  3 : return address to level 2
;                     4  -->  4 : return address to level 3
;                     5  -->  5 : return address to level 4
;                     6  -->  6 : return address to level 5
;                             7 = -1 (signifies original caller = core res.)
;                            10 : RDA of DS level 1 (complemented if core res.)
;                            11 : RDA of DS level 2 (complemented if core res.)
;                            12 : RDA of DS level 3 (complemented if core res.)
;                            13 : RDA of DS level 4 (complemented if core res.)
;                            14 : RDA of DS level 5 (complemented if core res.)
;                            15 : RDA of DS level 6 (complemented if core res.)
;                            16 : DISCSUB KEYWORD FOR LEVEL 1
;                            17 : DISCSUB KEYWORD FOR LEVEL 2
;                            20 : DISCSUB KEYWORD FOR LEVEL 3
;                            21 : DISCSUB KEYWORD FOR LEVEL 4
;                            22 : DISCSUB KEYWORD FOR LEVEL 5
;                            23 : DISCSUB KEYWORD FOR LEVEL 6
```

```
;                        << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

;  DATAPUMP calls used in CALLD:
;
;  READBLOCK:   reads discsub via pool into SSA
;               marks pool entry IN-FIXED (does not latch it)
;               sets P.SSA = pool entry pointer
;               clears IN-FIXED bit of previous occupant, if any
;
;  SAVDISCSUB:  (used for nesting discsubs)
;               copies SSA into its pool image
;               clears pool entry's IN-FIXED bit
;               latches pool entry (does not mark it dirty)
;
;  on CALL, we SAVDISCSUB if necessary,
;               then READBLOCK the next one
;
;  on RETURN,  we READBLOCK the previous one
;               then unlatch it


;  CALLD algorithm (based on NL = 6)
;
;  CALLD:  IF current nesting level > 6 THEN TRAPFAULT
;          IF current nesting level = 0 THEN SDA <-- 0
;          calculate DA (disc address), SA (starting address) from DAT, SAT
;          increment STKP
;          save return address at STKP + 0
;          IF DA > 0 (ie. disc resident) THEN
;               IF SDA <> 0 (ie. SSA occupied) THEN
;                    SAVDISCSUB
;               IF calling extended discsub THEN
;                    READBLOCK (DA) --> HXA
;                    INCREMENT DA
;               SDA <-- DA (disc address of discsub to be in SSA)
;               READBLOCK (DA) --> SSA
;          SAVE DA at STKP + 7
;          JMP @SA (ie. use the discsub)

;  CALLR:  increment the return address at (STKP + 0), for skip-return
;          IF (STKP + 7) > 0 (ie. returning FROM disc-resident) THEN SDA <-- 0
;          SA <-- (STKP + 0); prepare return address
;          decrement STKP (pop stack)
;          IF (STKP + 7) > 0 (ie. returning TO disc-resident) THEN
;               IF (SDA) = 0 THEN
;                    SDA <-- (STKP + 7)
;                    READBLOCK (SDA) --> SSA
;                    UNLATCH (SDA); remains "IN-FIXED", though
;          JMP @SA (ie. return to previous level)

;  FOR DISCSUBS IN EXTENDED MEMORY, THE SAME APPLIES EXCEPT FOR
;  NON-CORE-RESIDENT DISCSUBS USE THE 4 STANDARD ACCESS CALLS:
;       INIDSUB   TO INITIALIZE, WHEN STACK IS EMPTY
;       GTDSUB    TO GET A DISCSUB
;       SVDSUB    TO SAVE (NEXT) A DISCSUB
;       RVDSUB    TO RETRIEVE A SAVED DISCSUB
```

```
;                    << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

5576   4546 CALLD: JSR     DUMPR      ;CALL A DISCSUB
5577  30770        LDA     2,CATMP
5600  60177        INTEN
5601  21000        LDA     0,0,2      ;SUBROUTINE KEYWORD
5602  34562 CALD1: LDA     3,STKP     ;<< ENTRY FROM "CHANNEL"
5603  24560        LDA     1,.STK
5604 136405        SUB     1,3,SNR    ;NESTING LEVEL
5605  54563        STA     3,SDA      ;   INITIALIZE SDA
5606  24526        LDA     1,C.NL     ;DIFFERENCE MUST BE < C.NL
5607 166033        SLS     3,1
5610   6142        TRAPFAULT          ;CALLS NESTED TOO DEEP !?
5611 106010        14*K!NOP
5612 175015        SNZ     3,3        ;STACK EMPTY ?
5613   6544        JSR     @.INIDSUB; YES, INITIALIZE XMEM DSUB FUNCTION, IF ANY
5614  40555        STA     0,CDSBN    ;HOLD FULL KEYWORD
5615  36756        LDA     3,@.SDSB   ;# SYSTEM DISCSUBS
5616  24753        LDA     1,UBIT     ;USER DISCSUB ID. BIT
5617 107415        AND#    0,1,SNR    ;CALLING A USER DISCSUB ?
5620    404        JMP     CALD2      ;   NO
5621 163000        ADD     3,0        ;YES, CALC. DSUB ENTRY NUMBER, INTO DAT & SAT
5622  26750        LDA     1,@.UDSB   ;# USER DISCSUBS
5623 137000        ADD     1,3        ;TOTAL # DISCSUBS
5624  24066 CALD2: LDA     1,C777
5625 123400        AND     1,0        ;SUBROUTINE NUMBER
5626 116033        SLS     0,3
5627   4507        JSR     CALLF      ;ILLEGAL SUBROUTINE NUMBER !?
5630  40542        STA     0,DEN      ;SAVE DSUB TABLE ENTRY NO. FOR XMEM CODE
5631  34535        LDA     3,.SAT
5632 117000        ADD     0,3
5633  25400        LDA     1,0,3      ;STARTING ADDR.(IN SSA, HXA or core)
5634  44533        STA     1,SA
5635  36737        LDA     3,@XSCON
5636  24737        LDA     1,MCT
5637 167415        AND#    3,1,SNR
5640    404        JMP     CALD3
5641  24526        LDA     1,SA
5642   6016        JSR     @CTRVECTOR
5643      2        CTECALL
5644  34521 CALD3: LDA     3,.DAT
5645 117000        ADD     0,3
5646  25400        LDA     1,0,3      ;DISC ADDRESS (ON UNIT ZERO)
5647 125415        INC#    1,1,SNR
5650   4466        JSR     CALLF      ;NO SUCH SUBROUTINE !?
5651  10513        ISZ     STKP       ;BUMP STACK POINTER TO NEXT FRAME
5652  34512        LDA     3,STKP
5653 151400        INC     2,2        ;BUMP PAST SUBROUTINE ID
5654  51400        STA     2,0,3      ;SAVE RETURN ADDRESS IN STACK
5655 125112        SSP     1,1        ;CALLING CORE-RESIDENT ?
5656    420        JMP     CALLJ      ;   YES, DON'T CHANGE SSA
5657  20511        LDA     0,SDA      ;NO
5660  44510        STA     1,SDA      ;SAVE NEW DA TO BE IN SSA
5661  30013        LDA     2,.SSA
```

```
;                       << SI = R90REXCALLSC; BO = 1/A.REXM5.7033! >>

     5662 105004          MOV     0,1,SZR  ;is SSA occupied ?
     5663   6476          JSR     @.SVDSUB ;  YES, SAVE IT
     5664  24504          LDA     1,SDA
     5665  34504          LDA     3,CDSBN
     5666 177113          ADDL#   3,3,SNC  ;calling extended discsub ?
     5667    405          JMP     CALD7    ;  no
     5670  30012          LDA     2,.HXA   ;yes, read first block into HXA
     5671   6467          JSR     @.GTDSUB
     5672  10476          ISZ     SDA      ;ADDRESS OF SECOND BLOCK
     5673  24475          LDA     1,SDA
     5674  30013 CALD7:   LDA     2,.SSA
     5675   6463          JSR     @.GTDSUB ;read discsub into SSA
     5676  34466 CALLJ:   LDA     3,STKP
     5677  45407          STA     1,NL+1,3 ;save DA on stack for return
     5700  24471          LDA     1,CDSBN  ;GET CURRENT DSB NUMBER
     5701  45415          STA     1,NL*2+1,3;SAVE DSB NUMBER ON STACK (FOR TRAPFAULT)
     5702   4474          JSR     LOADR    ;RESTORE REGS
     5703   6464          JSR     @SA      ;GIVE CONTROL TO SUBROUTINE
                          ;
     5704    505          JMP     CALLN    ;RETURN HERE VIA (A3) or .NRET,
                          ;
     5705  12457 CALLR:   ISZ     @STKP    ;OR HERE VIA (A3)+1 OR .SRET FOR SKIP-RETURN
     5706  54447          STA     3,REGS+3
     5707   4435          JSR     DUMPR
     5710  14454 CALR1:   DSZ     STKP     ;"POP" THE CALL STACK << FROM NRET
     5711  34453          LDA     3,STKP
     5712  25415          LDA     1,NL*2+1,3
     5713  44456          STA     1,CDSBN
     5714  25410          LDA     1,NL+2,3
     5715 102400          SUB     0,0
     5716 125113          SSN     1,1      ;RETURNING **FROM** DISC-RESIDENT ?
     5717  40451          STA     0,SDA    ;  YES, MARK SSA NOT OCCUPIED
     5720  21401          LDA     0,1,3    ;RETURN ADDRESS
     5721  40446          STA     0,SA
     5722  25407          LDA     1,NL+1,3 ;CALLER'S ORIGINAL DISC ADDRESS
     5723  20445          LDA     0,SDA
     5724 125113          SSN     1,1      ;RETURNING **TO** DISC-RESIDENT . . .
     5725 101014          SKZ     0,0      ;  . . . AND SSA NO LONGER OCCUPIED ?
     5726    404          JMP     CALLL    ;  NO, JUST RESTORE REG'S & RETURN
     5727  44441          STA     1,SDA    ;YES, MARK SSA AS CONTAINING SAVED DISCSUB
     5730  30013          LDA     2,.SSA
     5731   6431          JSR     @.RVDSUB ;RETRIEVE SAVED DISCSUB
     5732   4444 CALLL:   JSR     LOADR    ;RESTORE REGISTERS, AND
     5733   2434          JMP     @SA      ;RETURN CONTROL TO CALLER
```

```
;                       << SI = R90REXCALLSC; BO = 1/A.REXM5.7033! >>

     5734       6 C.NL:  NL      ;discsub nesting limit

     5735       0        0
     5736   54777 CALLF: STA     3,.-1
     5737   20432        LDA     0,CDSBN   ;KEYWORD
     5740   24774        LDA     1,C.NL    ;NEST LIMIT
     5741   30626        LDA     2,CATMP   ;CALLER'S RETURN ADDRESS-1
     5742    6142        TRAPFAULT
     5743  106410        15*K!NOP


     5744   40406 DUMPR: STA     0,REGS    ;DUMP REGISTERS
     5745  101100        MOVL    0,0
     5746   40410        STA     0,REGS+4
     5747   44404        STA     1,REGS+1
     5750   50404        STA     2,REGS+2
     5751    1400        JMP     0,3


     5752       5 REGS:  .BLK 5  ;REGISTER STORAGE FOR "CALL"
                                 ;A0
                                 ;A1
                                 ;A2
                                 ;A3
                                 ;CARRY
```

```
;                       << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

; FIXED OFFSETS FROM CALL STACK POINTER (SEE R83DEFS)
; THE 4 NEGATIVE OFFSET VECTORS GET OVERLAID BY $LXM IF DSUBS ARE IN XMEM

 5757    6031 .INIDSUB:     INIDSUBS ;INITIALIZE XMEM DSUB MECHANISM
 5760    6032 .GTDSUB:      GTDSUB   ;GET A DISCSUB
 5761    6033 .SVDSUB:      SVDSUB   ;SAVE DISCSUB
 5762    6041 .RVDSUB:      RVDSUB   ;RETRIEVE SAVED DISCSUB
 5763       0 .STK:         0        ;POINTER TO CALL STACK <<< ..STK IN INFO POINTS HERE
 5764       0 STKP:         0        ;STACK NESTING POINTER
 5765       0 .DAT:         0        ;DISC ADDRESS TABLE
 5766       0 .SAT:         0        ;STARTING ADDRESS TABLE
 5767       0 SA:           0        ;STARTING ADDRESS OR RETURN ADDRESS
 5770       0 SDA:          0        ;RDA OF THE DISCSUB BLOCK IN SSA IF STILL NEEDED
 5771       0 CDSBN:        0        ;CURRENT DISCSUB NUMBER (KEYWORD)
 5772       0 DEN:          0        ;DISCSUB ENTRY NUMBER (IN DAT, SAT, SZT)
 5773       0               0        ;DISCSUB SIZE TABLE (ONLY USED BY M12 XMEM)
 5774       0               0        ;SIR PUTS ADDRESS OF MAP BUFFER AREA HERE, IF ANY


 5775       0         0
 5776   54777 LOADR:STA     3,.-1    ;LOAD REGISTERS
 5777   20753         LDA     0,REGS+0
 6000   24753         LDA     1,REGS+1
 6001   30753         LDA     2,REGS+2
 6002   34754         LDA     3,REGS+4 ;CARRY
 6003  175200         MOVR    3,3
 6004   34751         LDA     3,REGS+3
 6005    2770         JMP     @LOADR-1


 6006    6112         JSR     @.NRET
 6007  100010         100010
 6010   77400         077400

 6011   54744 CALLN:STA     3,REGS+3 ;NON-SKIP RETURN VIA .NRET
 6012    4732         JSR     DUMPR
 6013   34742         LDA     3,REGS+3
 6014   21777         LDA     0,-1,3
 6015   24771         LDA     1,CALLN-3
 6016  106414         SEQ     0,1      ;JSR @.NRET ?
 6017     671         JMP     CALR1    ;  NO
 6020    2470         LDA     1,CALLN-1;  YES
 6021   21400         LDA     0,0,3    ;POSSIBLE X*K+NOP
 6022  130000         COM     1,2
 6023  107700         ANDS    0,1      ;EXTRACT X FIELD
 6024  113400         AND     0,2      ;EXTRACT NOP FIELD
 6025   34762         LDA     3,CALLN-2
 6026  156415         SNE     2,3      ;IS IT X*K+NOP ?
 6027   44726         STA     1,REGS+3 ;  YES, X --> A3
 6030     660         JMP     CALR1
```

```
;                        << SI = R9OREXCALLSC; BO = 1/A.REXM5.7033! >>

; DEFAULT DISCSUB ACCESS ROUTINES FOR NON-EXTENDED MEMORY OPERATION

; NOTE: IF DISCSUB ARE IN XMEM, DAT CONTAINS THE DISCSUB BLOCK ADDRESS
; IN XMEM, AND SAT CONTAINS THE STARTING ADDRESS IN XMEM DSUB BUFFER AREA

; INITIALIZE XMEM DISCSUB MECHANISM
; MUST PRESERVE AO AND A2

          6031 INIDSUB:
    6031  1400         JMP     0,3


; GET DISCSUB
; A1 = RDA, OR XMEM BLOCK #
; A2 = SSA OR HXA (XMEM ROUTINE MUST INTERPRET AND ADJUST AS REQUIRED)

          6032 GTDSUB:
    6032  2135         JMP     @READBLOCK&377


; SAVE (NEST) THE DISCSUB IN SSA
; A1 = RDA (XMEM BLOCK #) OF DISCSUB IN SSA (OR IN CORRESPONDING XMEM BUFFER)
; A2 = SSA

          6033 SVDSUB:
    6033  54742        STA     3,LOADR-1
    6034 102400        SUB     0,0
    6035   6110        DATAPUMP
    6036     24        SAVDISCSUB
    6037 100000        @0
    6040   2735        JMP     @LOADR-1


; RETRIEVE (UNNEST) LAST SAVED DISCSUB
; AO = 0
; A1 = RDA (XMEM BLOCK #) OF SAVED DISCSUB
; A2 = SSA

          6041 RVDSUB:
    6041  54734        STA     3,LOADR-1
    6042   6135        READBLOCK           ;GET THE DISCSUB BACK FROM POOL
    6043  31405        LDA     2,P.CA,3    ;GET POINTER TO CORRESPONDING POOL BUFFER
    6044   6110        DATAPUMP
    6045      3        UNLATCH             ;UNLATCH THE BLOCK JUST UNNESTED
    6046 100000        @0
    6047   2726        JMP     @LOADR-1
```

```
;                        << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

;;--- ------- ---------------- -- -- --- -- ------------------------------
;;
;;    3.   CHANNEL OPERATIONS
;;
;;    PURPOSE:   PERFORMS SOME CHECKING ON PARAMETERS & AUTHORIZATION, BEFORE
;;               BRANCHING TO "CALLD" TO PERFORM THE SPECIFIED "CALL".
;;               NOTE THAT THIS ENTRY IS FOR USER JOBS ONLY.
;;
;;    CALLING SEQUENCE:      CHANNEL
;;                           <SUBR KEYWORD> (BUILD, BILDD, OPEN, OPENUPDATE,
;;                             OPENREF, OPENLOCK, CLOSE, CLEAR, GETRR, GETRW,
;;                             LOCK, WRITITEM, READITEM, READCONTIG, WRITCONTIG)
;;
;;    ENTRY:   A0,A1,A2 <-- PARAMETERS FOR DISC SUBROUTINE
;;             A3 <-- RETURN ADDRESS
;;
;;    EXIT:    IF (ERROR) RETURNS AT (A3)+1 with a3 = error code
;;             ELSE GOES TO CALLD, EVENTUALLY RETURNS AT (A3)+2
;;--- ---------------- -------- -------- --- ---- ------------------------

    6050     603 N.DCH:INFO+NDCH.

; FIXED NEGATIVE OFFSETS FROM CHANNEL ENTRY POINT (SEE R83DEFS)

    6051       0 LDFTA:        0          ;-10;CORE ADDR. OF SHARED REGNANT DFT, IF DFT'S IN XMEM
    6052       0 LDFTX:        0          ; -7;DELTA: XADR - CORE ADDR., FOR REGNANT USER'S DFT
    6053    6327 .DFEFROMPC:   RDFEFROMPC
    6054    6357 .DFETOPC:     RDFETOPC
    6055    6326 .DFTREAD:     RDFTREAD
    6056    6326 .DFTWRITE:    RDFTWRITE
    6057    6312 .DFEREAD:     RDFEREAD
    6060    6326 .DFEWRITE:    RDFEWRITE

    6061   54674 CHANX:STA     3,REGS+3 ;CHANNEL OPERATION
    6062    4662         JSR     DUMPR
    6063  101200         MOVR    0,0      ;CORRECT FOR CARRY SAVE IN DUMPR
    6064   26671         LDA     1,@REGS+3;GET SUBR KEYWORD
    6065   30066         LDA     2,C777
    6066  147400         AND     2,1      ;EXTRACT DSB# FROM KEYWORD
    6067   30420         LDA     2,CHAN1-1;GET ATTRIBUTES KEY
    6070  133000         ADD     1,2
    6071   25360         LDA     1,-BUILD!177000,2
    6072  101113         SSN     0,0      ;NEGATIVE CHANNEL NUMBER ?
    6073     415         JMP     CHAN1    ;  NO
    6074  111404         INC     0,2,SZR  ;CHANNEL -1 ?
    6075     404         JMP     CHANZ    ;  NO
    6076  125112         SSP     1,1      ;  YES, LEGAL ?
    6077     404         JMP     CHANY    ;    NO
    6100     413         JMP     CHAN2    ;    YES

    6101  151415 CHANZ:INC#    2,2,SNR  ;CHANNEL -2 ?
    6102  127112         ADDL#   1,1,SZC  ;  YES, LEGAL ?
    6103  176401 CHANY:SUB     3,3,SKP  ;    NO, ERROR #0
    6104     407         JMP     CHAN2    ;    YES
    6105   10650 CHANO:ISZ     REGS+3   ;NON-SKIP RETURN
    6106    2647         JMP     @REGS+3
```

```
;                       << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

 6107   6270           ATKEY
 6110  32740 CHAN1:    LDA     2,@N.DCH ;CHANNEL NUMBER >= 0
 6111 112033           SLS     0,2      ;LEGAL CHANNEL NUMBER ?
 6112    771           JMP     CHAN0-2  ;   NO, ERROR #0
 6113  30005 CHAN2:    LDA     2,RUP    ;   YES
 6114  35025           LDA     3,DFT.,2
 6115 101120           MOVZL   0,0      ;MULTIPLY CHANNEL NUMBER BY 8
 6116 101120           MOVZL   0,0      ;(CHANGE IF CHM1 IS CHANGED) ***
 6117 101120           MOVZL   0,0
 6120 117000           ADD     0,3      ;CHANNEL ADDRESS
 6121  54631           STA     3,REGS
 6122  31401           LDA     2,FDA,3
 6123 127125           ADDZL   1,1,SNR  ;CALLING "CLEAR" . . .
 6124 151014           SKZ     2,2      ;AND CHANNEL NOT OPEN ?
 6125    403           JMP     CHN2A    ;   NO
 6126  34627           LDA     3,REGS+3 ;   YES, NO-OP SKIP RETURN
 6127   1402           JMP     2,3

 6130 125103 CHN2A:    MOVL    1,1,SNC  ;ERROR IF CHANNEL OPEN ?
 6131    406           JMP     CHAN4    ;   NO
 6132 151015           SNZ     2,2      ;YES, IS IT OPEN ?
 6133    510           JMP     CHANE    ;   NO, CALL THE ROUTINE
 6134 176521           SUBZL   3,3,SKP  ;   YES, ERROR #1
 6135  34002 CHAN3:    LDA     3,C2
 6136    747           JMP     CHAN0

 6137 151015 CHAN4:    SNZ     2,2      ;CHANNEL MUST BE OPEN -- IS IT ?
 6140    775           JMP     CHAN3    ;   NO, ERROR #2
 6141 125103           MOVL    1,1,SNC  ;   YES, I/O ROUTINE ?
 6142    505           JMP     CHAN6    ;     NO
 6143  31400           LDA     2,FLU,3  ;     YES
 6144 151113           SSN     2,2      ;DEVICE ?
 6145    440           JMP     CHAN7    ;   NO
 6146  31402           LDA     2,CBN,3  ;   YES, DRIVER ADDRESS
 6147 125103           MOVL    1,1,SNC  ;READ ?
 6150    411           JMP     CHANW    ;   NO, WRITE
 6151  31377           LDA     2,-1,2   ;   YES
 6152 151415 CHANJ:    INC#    2,2,SNR  ;ANY READ/WRITE ROUTINE ?
 6153    415           JMP     CHA8A    ;   NO, ILLEGAL OPERATION
 6154  50513           STA     2,CHANA  ;   YES, PRESERVE ROUTINE ADDRESS
 6155   6403           JSR     @CHANW-1 ;RESTORE ALL REGISTERS
 6156 175400           INC     3,3      ;BUMP RETURN ADDRESS
 6157   2510           JMP     @CHANA   ;BRANCH TO ROUTINE (DEVICE)

 6160   5776           LOADR
 6161  31376 CHANW:    LDA     2,-2,2   ;WRITE TO A DEVICE ;GET R/W ROUTINE ADDR)
 6162  25403           LDA     1,STS,3
 6163 125113           SSN     1,1      ;WRITE PROTECTED ?
 6164    766           JMP     CHANJ    ;   NO, BRANCH TO ROUTINE
 6165  34402 CHAN8:    LDA     3,.+2    ;   YES, ERROR #22
 6166    717           JMP     CHAN0
 6167     22           22
 6170  34402 CHA8A:    LDA     3,.+2    ;ILLEGAL DEVICE OPERATION
 6171    714           JMP     CHAN0
 6172     41           41
```

```
;                       << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

 6173  10000 CH10K: 10000
 6174   4000 CH4K:  4000    ;'MAINTENANCE' MODE BIT
 6175  20000 CH20K: 20000
 6176  34000 CH34K: 34000
 6177  20000 VINCLUDED:    N          ;'included' DISCSUB bit
 6200     36 CHRDC: READC
 6201     36 CHRDT: READC   ;(FUTURE READT)
 6202  40142 CHRDP: READP
 6203    115 CHRDM: READMAINTENANCE
 6204      0 CHCNP: 0       ;POLYFILE FLAG (-1=NOT POLYFILE)

 6205  21407 CHAN7: LDA     0,CLP,3   ;Get flags from DFT word
 6206 101223        MOVZR   0,0,SNC   ;Open in Polyfile Mode (make sure AC1 <> -1) ?
 6207 102000        ADC     0,0       ;  No - set flag to -1
 6210  40774        STA     0,CHCNP
 6211 125100        MOVL    1,1       ;Move Read/Write command flag to Carry bit.
 6212  32434        LDA     2,@CHAN6-1;ADDRESS OF COMMAND WORD
 6213  21000        LDA     0,0,2     ;PICK UP COMMAND WORD
 6214  35403        LDA     3,STS,3   ;CHANNEL STATUS
 6215  24756        LDA     1,CH1OK
 6216 167414        AND#    3,1,SZR   ;NOT FORMATTED?
 6217  20762        LDA     0,CHRDT   ;  YES, PICK UP CORRECT COMMAND WORD
 6220  24755        LDA     1,CH2OK
 6221 167414        AND#    3,1,SZR   ;CONTIGUOUS?
 6222  20756        LDA     0,CHRDC   ;  YES, PICK UP CORRECT COMMAND WORD
 6223  24751        LDA     1,CH4K
 6224 167414        AND#    3,1,SZR   ;'maintenance' mode ?
 6225  20756        LDA     0,CHRDM   ;  yes
 6226  24751        LDA     1,VINCLUDED;prepare to 'include' if WRITE
 6227  10755        ISZ     CHCNP     ;POLYFILE?
 6230  20752        LDA     0,CHRDP   ;  YES, PICK UP CORRECT COMMAND WORD
 6231 125402        INC     1,1,SZC   ;FORM INCREMENT TO WRITE COMMAND;  READ?
 6232   2410        JMP     @CHANP+1  ;  YES
 6233 123000        ADD     1,0       ;NO, FORM WRITE COMMAND
 6234 177112        ADDL#   3,3,SZC   ;WRITE PROTECTED?
 6235    730        JMP     CHAN8     ;  YES, ERROR 22
 6236  24740        LDA     1,CH34K   ;  NO
 6237 137415        AND#    1,3,SNR   ;IS IT CONTIGUOUS OR NOT FORMATTED ?
 6240    403        JMP     CHANE     ;  NO, IT'S FORMATTED
 6241   2401 CHANP: JMP     @CHANP+1
 6242   5602        CALD1

 6243  32403 CHANE: LDA     2,@CHAN6-1
 6244  21000        LDA     0,0,2     ;RECOVER DISCSUB KEYWORD
 6245    774        JMP     CHANP
```

```
;                       << SI = R90REXCALLSC; BO = 1/A.REXM5.7033! >>

 6246    5755          REGS+3
 6247   31400 CHAN6:   LDA     2,FLU,3
 6250  127133          ADDZL#  1,1,SNC   ;CLOSE OR CLEAR?
 6251  151113          SSN     2,2       ;DEVICE ?
 6252     771          JMP     CHANE     ;   NO
 6253   31402          LDA     2,CBN,3   ;   YES
 6254   31375          LDA     2,-3,2    ;GET ADDRESS OF WRAP-UP ROUTINE
 6255   25400          LDA     1,FLU,3
 6256  124000          COM     1,1       ;COMPLEMENT THE LUN
 6257   45400          STA     1,FLU,3
 6260   50407          STA     2,CHANA   ;SAVE ADDRESS OF WRAP-UP ROUTINE
 6261    6677          JSR     @CHANW-1  ;COLLECT THE REGISTER
 6262   34405          LDA     3,CHANA   ;AND RETURN
 6263  175414          INC#    3,3,SZR   ;ANY WRAP UP ROUTINE?
 6264    5400          JSR     0,3       ;BRANCH TO WRAP-UP ROUTINE
 6265     756          JMP     CHANE     ;   NO,   JUST EXIT
 6266     755          JMP     CHANE     ;EXIT TO ROUTINE


 6267       0 CHANA:   0

      100000 M1I    =100000;CHANNEL -1 IS ILLEGAL
       40000 M2I    = 40000;CHANNEL -2 IS ILLEGAL
       20000 ECO    = 20000;ERROR IF CHANNEL OPEN
       10000 IOR    = 10000;I/O ROUTINE
       14000 RDR    = 14000;I/O ROUTINE AND READ ROUTINE
        2000 PRC    =  2000;PROCESS CHANNEL ROUTINE, DON'T CLOSE OR CLEAR

        6270 ATKEY =.       ;ATTRIBUTES KEYS FOR "CHANNEL"

 6270  160000 M1I+M2I+ECO   ;BUILD (20)
 6271  160000 M1I+M2I+ECO   ;BILDD (21)
 6272  160000 M1I+M2I+ECO   ;  *** TEST ***  M2I+ECO ;OPEN  (22)
 6273  160000 M1I+M2I+ECO   ;OPENU (23)
 6274  160000 M1I+M2I+ECO   ;OPENL (24)
 6275  160000 M1I+M2I+ECO   ;OPENR (25)
 6276  140000 M1I+M2I       ;CLOSE (26)
 6277       0       0       ;CLEAR (27)
 6300  154000 M1I+M2I+RDR   ;GETRR (30)
 6301  150000 M1I+M2I+IOR   ;GETRW (31)
 6302    2000 PRC           ;LOCK  (32)
 6303  154000 M1I+M2I+RDR   ;READI (33)
 6304  150000 M1I+M2I+IOR   ;WRITI (34)
 6305  150000 M1I+M2I+IOR   ;WRITN (35)
 6306  140000 M1I+M2I       ;READC (36)
 6307  140000 M1I+M2I       ;WRITC (37)
 6310  142000 M1I+M2I+PRC   ;RENAM (40)
 6311  160000 M1I+M2I+ECO   ;OPEN MAINT (41)
```

```
;                    << SI = R90REXCALLSC;  BO = 1/A.REXM5.7033! >>

; DEFAULT DFT ACCESS ROUTINES

          6312 RDFEREAD:                ;"READ" DFT ENTRY WHOSE ADR. IS IN A0; CORE ADR. --> A2,
  6312   30005        LDA     2,RUP     ;  ALSO SET A1 = 0 IFF REGNANT USER'S
  6313   25025        LDA     1,DFT.,2
  6314   30440        LDA     2,DFT.N   ;SIZE OF NEGATIVE CHANNELS
  6315  146400        SUB     2,1       ;MIN. DFTE BELONGING TO REGNANT USER
  6316   32433        LDA     2,@.SDFT  ;SIZE OF ENTIRE DFT
  6317  133000        ADD     1,2       ;END OF REGNANT USER'S DFT
  6320  106033        SLS     0,1       ;DOES GIVEN DFTE BELONG TO R.U.?
  6321  112033        SLS     0,2
  6322     402        SKIP              ;  NO, LEAVE A1 NON-0
  6323  126400        SUB     1,1       ;  YES, SET A1 = 0 (ELSE LEAVE NON-ZERO)
  6324   30431        LDA     2,CC3
  6325  113400        AND     0,2       ;MASK & POSITION CORE ADDRESS
          6326 RDFEWRITE:               ;"WRITE" DFT ENTRY, DEFAULT = NO-OP
          6326 RDFTREAD:                ;READ IN ENTIRE DFT, DEFAULT = NO-OP
          6326 RDFTWRITE:               ;WRITE OUT ENTIRE DFT, DEFAULT = NO-OP
  6326    1400        JMP     0,3

          6327 RDFEFROMPC:              ;CALC. DFTE XADR FROM PORT (A0), CHANNEL (A1)
  6327   32423        LDA     2,@.LPCA  ;(VALID FOR CORE OR XMEM)
  6330  103120        ADDZL   0,0       ;MULTIPLY PORT # BY 40
  6331  103120        ADDZL   0,0
  6332  101120        MOVZL   0,0
  6333  113000        ADD     0,2       ;ADD TO PCA BASE ADDRESS
  6334   20005        LDA     0,RUP
  6335  112414        SEQ     0,2       ;IS GIVEN PORT = REGNANT USER ?
  6336  102401        SUB     0,0,SKP   ;  NO, THE DFT POINTER = XADR
  6337   22414        LDA     0,@.XDFX  ;  YES, PICK UP XADR OFFSET
  6340   31025        LDA     2,DFT.,2
  6341  143000        ADD     2,0       ;CALC. DFT'S XADR --> A0
  6342  125120        MOVZL   1,1       ;MULTIPLY CHANNEL # (A1) BY 10
  6343  125120        MOVZL   1,1       ;(CAN'T USE ADDZL BECAUSE CHANNEL # CAN BE NEGATIVE)
  6344  125120        MOVZL   1,1
  6345  123000        ADD     1,0       ;CALC. XADR OF DFT ENTRY --> A0
  6346    1400        JMP     0,3       ;RETURN

  6347       0 DFRTN:          0
  6350       0 DF.A0:          0
  6351     645 .SDFT:          INFO+SDFT.
  6352     604 .LPCA:          INFO+LPCA.
  6353    6052 .XDFX:          LDFTX
  6354      20 DFT.N:          -CHM2
  6355  177774 CC3:            -1-3
  6356       0 DFCOUNT:        0
```

```
;                    << SI = R90REXCALLSC; BO = 1/A.REXM5.7033! >>

          6357 RDFETOPC:                ;CONVERT DFTE XADR TO PORT, CHANNEL
  6357   54770         STA     3,DFRTN  ;(VALID FOR BOTH CORE AND XMEM)
  6360   34775         LDA     3,CC3
  6361  163400         AND     3,0      ;MASK OFF 2 LSB (FLAGS)
  6362   40766         STA     0,DF.A0  ;SAVE DFTE XADR
  6363   30100         LDA     2,.INFO
  6364   24770         LDA     1,DFT.N  ;SIZE OF NEGATIVE PORTION OF DFT
  6365  107000         ADD     0,1      ;CALC. MAX. DFT VALUE OUR DFTE MIGHT BELONG TO
  6366   35005         LDA     3,TNAP.,2;INITIALIZE FOR A PORT SCAN
  6367   54767         STA     3,DFCOUNT
  6370   31004         LDA     2,LPCA.,2
          6371 DFLOOP:                  ;LOOP THROUGH ALL PORTS & TEST IF DFTE BELONGS TO IT
  6371   34005         LDA     3,RUP
  6372  156414         SEQ     2,3      ;THIS PORT = REGNANT USER ?
  6373     406         JMP     DFL2     ;   NO
  6374   36757         LDA     3,@.XDFX ;YES, PICK UP OFFSET TO XADR
  6375   31025         LDA     2,DFT.,2
  6376  157000         ADD     2,3      ;CALC. DFT XADR
  6377   30005         LDA     2,RUP    ;RESTORE PCB POINTER
  6400     402         SKIP
  6401   35025 DFL2:   LDA     3,DFT.,2 ;PICK UP NON-REGNANT USER'S DFT XADR
  6402   22747         LDA     0,@.SDFT ;TOTAL SIZE OF ENTIRE DFT
  6403  163023         ADDZ    3,0,SNC  ;BEGINNING OF NEXT DFT -- WRAP-AROUND ?
  6404  106433         SLE     0,1      ;   NO, DOES THIS DFT EMBRACE GIVEN DFTE ?
  6405  166433         SLE     3,1
  6406     414         JMP     DFNEXT   ;   NO, TEST NEXT PORT
  6407   22743         LDA     0,@.LPCA ;YES, CALC. PORT # FROM PCB --> A0
  6410  112540         SUBOL   0,2
  6411  141120         MOVZL   2,0
  6412  103300         ADDS    0,0
  6413   24735         LDA     1,DF.A0  ;RETRIEVE GIVEN DFTE (MASKED)
  6414  166640         SUBOR   3,1      ;CALC. CHANNEL # --> A1
  6415  135100         MOVL    1,3      ;RESTORE CARRY
  6416  125200         MOVR    1,1      ;RIGHT-SHIFT WITH SIGN EXTENSION
  6417  135100         MOVL    1,3
  6420  125200         MOVR    1,1
  6421    2726         JMP     @DFRTN

          6422 DFNEXT:                  ;STEP TO NEXT PORT
  6422   34044         LDA     3,C40
  6423  173000         ADD     3,2      ;CALC. NEXT PCB
  6424   14732         DSZ     DFCOUNT  ;ANY MORE PORTS LEFT ?
  6425     744         JMP     DFLOOP   ;   YES, STAY IN LOOP
  6426    6142         TRAPFAULT        ;NO, SOFTWARE BUG !!
```

```
;                       << SI = R9OREXCALLSC;  BO = 1/A.REXM5.7033! >>

; CORE-RESIDENT SUBROUTINE TABLE

   6427 100036 CRST:  @N.SUB  ;NUMBER OF SUBROUTINES
   6430   1324        EXITX   ;   0   =   CALL   SCOPE
   6431   5736        CALLF   ;   1   (RESERVED)
   6432   7461        CHCHX   ;   2   =   CALL   CHKCHANNEL
   6433   7432        ALCLR   ;   3   =   CALL   ALLCLEAR
   6434   5736        CALLF   ;   4   (WAS FOFC)
   6435   5736        CALLF   ;   5   (WAS FOFI)
   6436   2772        LUSRX   ;   6   =   CALL   LOADUSER
   6437   5736        CALLF   ;   7   (RESERVED)
   6440   7447        UNLKX   ;  10   =   CALL   UNLOCK
   6441   7412        WONAX   ;  11   =   CALL   WONA
   6442   7503        CHKPB+2;  12   =   CALL   CHKRPROTECTION
   6443   7502        CHKPB+1;  13   =   CALL   CHKWPROTECTION
   6444   7501        CHKPB   ;  14   =   CALL   CHKCPROTECTION
   6445   7004        MOVEX   ;  15   =   CALL   MOVEWORDS
   6446   7126        MOBYX   ;  16   =   CALL   MOVBYTES
   6447   7730        RLSFB   ;  17   =   CALL   XFIXBUFFERS
   6450   6571        CRLAX   ;  20   =   CALL   CRLA
   6451   6553        CLRAX   ;  21   =   CALL   CLRA
   6452   6467        CRPNX   ;  22   =   CALL   CPPPN
   6453   6504        CPNRX   ;  23   =   CALL   CPNPP
   6454   2577        SLEPX   ;  24   =   CALL   SLEEP
   6455   2470        AWAKX   ;  25   =   CALL   AWAKE
   6456   2436        XQUEX   ;  26   =   CALL   XQUEUE
   6457   7756        RSFBX   ;  27   =   CALL   SFIXBUFFER
   6460   3431        CBSAX   ;  30   =   CALL   CBSA
   6461   7577        VLFNX   ;  31   =   CALL   VOLFIND
   6462   7546        VLKPX   ;  32   =   CALL   VOLLOOKUP
   6463   5736        CALLF   ;  33
   6464   5736        CALLF   ;  34
   6465   5736        CALLF   ;  35

             36 N.SUB=.-CRST-1


                .EOT   ;REX "CALL" FOR "IRIS" RESIDENT EXECUTIVE
```

```
;                       << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>
; 13 SEP 86, RB.

;;---------------------------------------------------------------------------
;;
;;          R E X :   G L O B A L   S U B R O U T I N E S
;;
;;    NOTE:  ONLY THE PERMANENTLY RESIDENT SUBROUTINES ARE INCLUDED HERE.
;;
;;      1.  MAPPING FUNCTIONS, POINTER SET UP
;;          1.1 CONVERT PCB TO PORT
;;          1.2 CONVERT PORT TO PCB
;;          1.3 FIND LU TABLE ENTRY
;;          1.4 CONVERT LOGICAL TO REAL DISC ADDRESS
;;          1.5 CONVERT REAL TO LOGICAL DISC ADDRESS
;;
;;      2.  DATA ACCESS - MOVE
;;          2.1 TEXT OUTPUT
;;          2.2 EXTENDED STORE BYTE
;;          2.3 EXTENDED LOAD BYTE
;;          2.4 ACCESS NEXT NON-BLANK INPUT BYTE
;;          2.5 ACCESS NEXT STRING INPUT BYTE
;;          2.6 GET A BYTE
;;          2.7 OUTPUT BYTE TO IO BUFFER
;;          2.8 STORE A BYTE
;;          2.9 MOVE WORDS
;;          2.10 MOVE BYTES
;;
;;      3.  (deleted)
;;
;;      4.  ARITHMETIC
;;          4.1 BINARY MULTIPLY
;;          4.2 BINARY DIVIDE
;;          4.3 DUMMY DECIMAL
;;
;;      5.  INTERACTIVE IO
;;          5.1 START INPUT
;;          5.2 START OUTPUT
;;          5.3 WAIT FOR OUTPUT NOT ACTIVE
;;
;;      6.  FILE - CHANNEL OPERATIONS
;;          6.1 CLEAR ALL DATA CHANNELS
;;          6.2 UNLOCK RECORD IN CHANNEL
;;          6.3 CHECK IF CHANNEL IS IN USE
;;          6.4 CHECK PROTECTION ON FILE
;;          6.5 DELETED (WAS FOFI)
;;          6.6 DELETED (WAS FOFC)
;;          6.7 POLYFILE VOLUME LOOKUP
;;          6.8 POLYFILE FIND NEXT VOLUME
;;
;;      7.  MISCELLANEOUS
;;          7.1 IS A2 A LETTER?
;;          7.2 IS A2 A DIGIT?
;;          7.3 CHANGE OR CHECK FLAG
;;          7.4 RELEASE FIXED BUFFERS
;;          7.5 RELEASE SPECIFIED FIXED BUFFER
;;          7.6 RELATIVE JSR
```

```
;                       << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>
;;
;;    UPDATE RECORD:
;;      6-11-80   (LLL) REARRANGED FOR FUNCTIONAL MODULARITY &
;;                      DOCUMENTED ENTRY & EXIT CONDITIONS.
;;      1-10-81   (GAD) ADDED POLYFILE VOLUME ROUTINES
;;                      UPDATED CHARGE FOR DISC
;;                      ADDED RELATIVE JSR
;;      8-20-81   (GAD) ADDED RELEASE SPECIFIED FIXED LOGIC
;;      2- 5-82   (rb.) new datapump & task control node defs
;;     2 MAR 86    RB.  DELETE FOFI, FOR NEW DFT'S-IN-XMEM SCHEME
;;     9 APR 86    RB.  MOD. STOUTPUT TO ALLOW FOR IOB'S IN XMEM
;;     7 MAY 86    TWM  RECODED MOVEWORDS FOR GREATER SPEED
;;--- -------  -------- ------------------- ------------------------


;;- -------- ---------- --------- ---- -------------------------
;;     1.1.   CONVERT PCB TO PORT                        <<RE-ENTRANT>>
;;
;;          PURPOSE:   MAP A PCB ADDRESS INTO A PORT NUMBER.
;;
;;          CALLING SEQUENCE:   CALL
;;                              CPPPN
;;
;;          ENTRY:   A0 <-- A(PCB)   (MAY HAVE MSB SET)
;;
;;          EXIT:    GOOD (SKIP)
;;                      A0 <-- PORT #
;;                      A1 <-- PORT #
;;                      A2 PRESERVED
;;                   ERROR (NON SKIP)
;;- -------  ---------------------------------------------------

     6466     604 XLPCA: INFO+LPCA.

     6467  2677/ CRPNX: LDA     1,@XLPCA ;CONVERT PCB POINTER TO PORT #
     6470 122540        SUBOL   1,0      ;2 * OFFSET FROM LPCA (TO IGNORE MSB)
     6471  24044        LDA     1,C40
     6472 101222        MOVZR   0,0,SZC  ;NOT AN EXACT MULTIPLE OF 40 ?
     6473   1400        JMP     0,3      ;   NO, INVALID
     6474 125224        MOVZR   1,1,SZR  ;SHIFTED 6 PLACES YET ?
     6475    775        JMP     .-3      ;   NO
     6476  26405        LDA     1,@XTNAP ;YES
     6477 106033        SLS     0,1      ;PORT # >= TNAP ?
     6500   1400        JMP     0,3      ;   YES, TOO BIG
     6501 105000        MOV     0,1      ;NO, COPY PORT # TO A1
     6502   1401        JMP     1,3      ;GOOD RETURN
```

```
;                        << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-----------------------------------------------------------------------
;;      1.2  CONVERT PORT TO PCB ADDRESS                <<RE-ENTRANT>>
;;
;;           PURPOSE:  MAP A PORT # INTO A PCB ADDRESS.
;;
;;           CALLING SEQUENCE:    CALL
;;                                CPNPP
;;
;;           ENTRY:    A0 <-- PORT #
;;
;;           EXIT:     GOOD (SKIP)
;;                        A0 <-- A (PCB)
;;                        A1 <-- 0
;;                        A2 PRESERVED
;;                     ERROR (NON-SKIP)
;;                        A0 <-- UNCHANGED
;;                        A1 <-- # OF ACTIVE PORTS
;;-----------------------------------------------------------------------

6503     605 XTNAP: INFO+TNAP.

6504   26777 CPNRX: LDA     1,@XTNAP ;CONVERT PORT # TO PCB POINTER
6505  106033       SLS     0,1
6506    1400       JMP     0,3      ;NOT A VALID PORT NUMBER
6507  103120       ADDZL   0,0      ;MULTIPLY PORT # BY 40
6510  103120       ADDZL   0,0
6511  101120       MOVZL   0,0
6512   26754       LDA     1,@XLPCA
6513  123000       ADD     1,0      ;ADD LPCA TO IT
6514    1401       JMP     1,3      ;GOOD RETURN
```

```
;                       << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------
;;     1.3.   FIND LOGICAL UNIT TABLE ENTRY          <<NON-REENTRANT>>
;;
;;           PURPOSE:    MAPS A LOGICAL UNIT # INTO THE ADDRESS OF ITS
;;                       ASSOCIATED LU TABLE ENTRY
;;           CALLING SEQUENCE:   FINDLUT
;;
;;           ENTRY:   A0 <-- LU#
;;           EXIT:    IF (FOUND) THEN SKIP
;;                       A0 <-- LU#
;;                       A1 <-- A(LUT ENTRY)
;;                       A2 <-- A(LUFIX TABLE)
;;                       A3 <-- A(LUVAR TABLE)
;;                    ELSE (NOT FOUND) NON-SKIP
;;                       A1 <-- A(LU ENTRY) OR -1 LU NOT IN TABLE
;;                       A3 <-- STATUS = 14 = "LU NOT ACTIVE "
;;                                     = 30 = "LU TEMPORARILY INACTIVE "
;;-------------------------------------------------------------------

    6515     661          INFO+.LUT.
    6516     602          INFO+MILU.
    6517   26777  FLUT:   LDA     1,@.-1     ;FIND LOGICAL UNIT TABLE ENTRY
    6520   44431          STA     1,FLCNT    ;  (A0) = LOGICAL UNIT NUMBER
    6521   54431          STA     3,FLRTN
    6522  152400          SUB     2,2        ;PRESET A2
    6523   36772          LDA     3,@FLUT-2
    6524   25402  FLUO:   LDA     1,.LLU.,3  ;LU#
    6525  106555          SUBOL#  0,1,SNR    ;THIS LOGICAL UNIT ?
    6526     407          JMP     FLU1       ;   YES
    6527   24003          LDA     1,C3       ;   NO
    6530  137000          ADD     1,3
    6531   14420          DSZ     FLCNT      ;END OF TABLE ?
    6532     772          JMP     FLUO       ;   NO
    6533  176000          ADC     3,3
    6534     410          JMP     FLU2       ;   YES, NOT THERE

    6535   31400  FLU1:   LDA     2,.LFX.,3  ;LOGICAL UNIT FOUND
    6536  150513          NEGL#   2,2,SNC    ;IS IT ACTIVE ?
    6537     405          JMP     FLU2       ;   NO
    6540  165020          MOVZ    3,1        ;   YES
    6541   35401          LDA     3,.LVR.,3
    6542   10410          ISZ     FLRTN
    6543    2407          JMP     @FLRTN

    6544  165000  FLU2:   MOV     3,1        ;PRESERVE A(LU ENTRY)
    6545   34034          LDA     3,C14      ;ASSUME LOGICAL UNIT NOT ACTIVE
    6546  151014          SKZ     2,2        ;LU TEMPORARILY INACTIVE ?
    6547  175120          MOVZL   3,3        ;   YES, A3 = 30 NOW
    6550    2402          JMP     @FLRTN

    6551       0  FLCNT: 0
    6552       0  FLRTN: 0
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-----------------------------------------------------------------------
;;      1.4.   CONVERT LOGICAL TO REAL DISC ADDRESS     <<NON RE-ENTRANT>>
;;
;;           PURPOSE:   BUILD AN RDA FROM LOGICAL COMPONENTS
;;           CALLING SEQUENCE:   CALL
;;                               CLRA
;;           ENTRY:     A0 <-- A(LUVAR)
;;                      A2 <-- A(PARAMETER LIST)
;;                      <PARM LIST> :== CYLINDER
;;                                      TRACK
;;                                      SECTOR
;;           EXIT:      A1 <-- RDA
;;                      A2 <-- UNCHANGED
;;                      A3 <-- LOGICAL SECTOR
;;-----------------------------------------------------------------------

    6553   54451 CLRAX:STA     3,CRLRA   ;CONVERT LOGICAL TO REAL ADDRESS
    6554    4432       JSR     GETCF     ;GET CONVERSION FACTORS
    6555  105000       MOV     0,1       ;NSCT
    6556   22445       LDA     0,@CRLPL  ;LOGICAL CYLINDER
    6557    6116       BINMULTIPLY       ;A0 = CYL# * NTRK
    6560   30443       LDA     2,CRLPL
    6561   35001       LDA     3,1,2     ;LOGICAL TRACK
    6562  163000       ADD     3,0
    6563  131000       MOV     1,2       ;NSCT
    6564    6116       BINMULTIPLY       ;A0 = ((CYL# * NTRK) + TRK#) * NSCT
    6565   30436       LDA     2,CRLPL
    6566   25002       LDA     1,2,2     ;LOGICAL SECTOR
    6567  107000       ADD     0,1       ;REAL DISC ADDRESS
    6570    2434       JMP     @CRLRA
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-- ------ --- ----------------------------------------------------
;;      1.5.  CONVERT REAL DISC ADDR TO LOGICAL       <<NON-RE-ENTRANT>>
;;
;;            PURPOSE:    CONSTRUCT LOGICAL DISC ADDRESS FROM A REAL
;;                        ADDRESS.
;;            CALLING SEQUENCE:   CALL
;;                                CRLA
;;            ENTRY:   A0 <-- A(LUVAR)
;;                     A1 <-- RDA
;;                     A2 <-- A(PARAMETER LIST TO BE RETURNED, 3 WORDS RESERVED)
;;            EXIT:    A1 <-- LOGICAL TRACK
;;                     A2 <-- UNCHANGED
;;                     A3 <-- LOGICAL CYLINDER
;;                     <PARM LIST> ::= LOGICAL CYLINDER
;;                                     "     TRACK
;;                                     "     SECTOR
;;---- ------- ------- ------------------- ------- ----------------------

6571   54433 CRLAX: STA     3,CRLRA   ;CONVERT REAL TO LOGICAL ADDRESS
6572    4414        JSR     GETCF
6573   50427        STA     2,CRLNT   ;NTRK
6574    6115        BINDIVIDE         ;A3 = RDA / NSCT, A1 = REM.
6575   30426        LDA     2,CRLPL
6576   45002        STA     1,2,2     ;REMAINDER = LOGICAL SECTOR
6577  165000        MOV     3,1
6600   20422        LDA     0,CRLNT   ;NTRK
6601    6115        BINDIVIDE         ;A3 = ABOVE QUOTIENT / NTRK
6602   30421        LDA     2,CRLPL
6603   55000        STA     3,0,2     ;QUOTIENT = LOGICAL CYLINDER
6604   45001        STA     1,1,2     ;REMAINDER = LOGICAL TRACK
6605    2417        JMP     @CRLRA


6606   50415 GETCF: STA     2,CRLPL   ;GET CONVERSION FACTORS
6607  111000        MOV     0,2
6610   31007        LDA     2,NTRS,2
6611   20050        LDA     0,C77
6612  143400        AND     2,0       ;NUMBER OF SECTORS
6613  151220        MOVZR   2,2
6614  151220        MOVZR   2,2
6615  151220        MOVZR   2,2
6616  151220        MOVZR   2,2
6617  151220        MOVZR   2,2
6620  151220        MOVZR   2,2       ;NUMBER OF TRACKS
6621    1400        JMP     0,3


6622       0 CRLNT: 0         ;TEMP. STORE NTRK
6623       0 CRLPL: 0         ;PARAMETER LIST POINTER
6624       0 CRLRA: 0         ;RETURN ADDRESS
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-------- ------------------------------------------------------------
;;     2.1.   TEXT OUTPUT                                  << NOT RE-ENTRANT >>
;;
;;          PURPOSE:   PLACE TEXT STRING IN REGNANT USERS OUTPUT BUFFER
;;
;;          CALLING SEQUENCE:    OUTTEXT
;;                               .TXTF "TEXT"
;;           ENTRY:    "TEXT" IS A STRING TERMINATED BY A NULL BYTE
;;            EXIT:
;;                    COPIES THE NULL BYTE INTO OUTPUT BUFFER
;;                     OBP (PCB) <-- BAC(LAST NON-NULL BYTE)
;;                     RETURNS TO FIRST INSTRUCTION FOLLOWING NULL BYTE
;;
;;
;;----------- ------------------------------ -- ----------------------

     6625 175120 OTXT:  MOVZL   3,3       ;OUTPUT TEXT TO I/O BUFFER
     6626  54533         STA     3,OTEMP
     6627  24532 OTXT2: LDA     1,OTEMP
     6630  10531         ISZ     OTEMP
     6631   4412         JSR     ACBY      ;GET BYTE
     6632 141005         MOV     2,0,SNR   ;TERMINATOR ? (MUST RETURN (A0) = 0)
     6633    403         JMP     OTXT4     ;   YES
     6634   4460         JSR     STOB      ;NO, STORE IN IOB
     6635    772         JMP     OTXT2

     6636  34523 OTXT4: LDA     3,OTEMP
     6637 175620         INCZR   3,3       ;COMPUTE RETURN ADDRESS
     6640   1400         JMP     0,3
```

```
;                        << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033!  >>

;;-------- ---------------------------- ----- ----------------------------
;;      2.2.   EXTENDED STORE BYTE                          <<RE-ENTRANT>>
;;
;;          PURPOSE:  STORE A BYTE IN 0-64K ADDRESS SPACE
;;          CALLING SEQUENCE:    XPUTBYTE
;;
;;          ENTRY:      DBA <-- WORD BASE ADDRESS OF DESTINATION
;;                      A0  <-- BYTE
;;                      A1  <-- DESTINATION BYTE OFFSET FROM WORD BASE
;;
;;          EXIT:       A0 <-- BYTE
;;                      A1 <-- RESULTING WORD
;;                      A2 <-- WORD ADDRESS OF DESTINATION
;;                      CY <-- 0 = LEFT BYTE, 1 = RIGHT BYTE
;;-------- ---------- ---------- ---- ----------------------------

    6641   30041 XSTBY: LDA     2,DBA     ;EXTENDED CORE STORE BYTE
    6642  125220        MOVZR   1,1
    6643  133000        ADD     1,2
    6644     502        JMP     LSTBY     ; GOTO CODE IN STOB


;;-------- ---------- ---------- ---- ----------------------------
;;
;;    2.3.  EXTENDED LOAD BYTE                          <<RE-ENTRANT>>
;;
;;       PURPOSE:  LOAD A BYTE FROM 0-64K ADDRESS SPACE.
;;
;;       CALLING SEQUENCE:   XGETBYTE
;;       ENTRY:     SBA <-- SOURCE WORD BASE ADDRESS
;;                  A1  <-- SOURCE BYTE OFFSET FROM WORD BASE
;;                LACBY (ALTERNATE) ENTRY
;;                  A2 = SOURCE WORD BASE ADDR(INSTEAD OF SBA)
;;
;;       EXIT:      A2  <-- BYTE
;;                  A1  <-- UNCHANGED
;;                  CY  <-- 0 = LEFT BYTE, 1 = RIGHT BYTE
;;-------- ---------- ---------- ---- ----------------------------

    6645   30040 XACBY: LDA     2,SBA     ;EXTENDED CORE ACCESS BYTE
    6646  121220 LACBY: MOVZR   1,0
    6647  113003        ADD     0,2,SNC
    6650     425        JMP     ACLBY
    6651   21000 ACRBY: LDA     0,0,2     ;ACCESS RIGHT BYTE
    6652   30064        LDA     2,C377
    6653  113400        AND     0,2
    6654    1400        JMP     0,3
```

```
;                    << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------------------- ------------------------------------------
;;
;;    2.4.   ACCESS NEXT NON-BLANK INPUT BYTE (INBYTE)    <<NON RE-ENTRANT>>
;;
;;        PURPOSE:   FETCH THE NEXT NON-BLANK INPUT BYTE FROM THE
;;                   REGNANT USERS INPUT BUFFER AND UPDATE IBP.(PCB)
;;
;;    2.5.   ACCESS NEXT STRING BYTE (INSTBYTE)           <<NON RE-ENTRANT>>
;;
;;        PURPOSE:   FETCH NEXT BYTE FROM REGNANT USER'S INPUT BUFFER
;;                   AND UPDATE IBP.(PCB) IF (AO) <> 0.
;;
;;    ENTRY:   IBP.(PCB) <-- B(NEXT INPUT BYTE)
;;
;;    2.6.   GET A BYTE (GETBYTE)                         << RE-ENTRANT >>
;;
;;        PURPOSE:   GET BYTE FROM BYTE ADDRESS IN A1
;;
;;    EXIT:    A2 <-- BYTE
;;-------------------------------- ----- --------------------

 6655  54504 ACIB:  STA     3,OTEMP  ;ACCESS NEXT INPUT BYTE AT (IBP)+1
 6656 161000         MOV     3,0      ;(IGNORES ANY SPACE CODE)
 6657   4404         JSR     ACSB
 6660  20056         LDA     0,C240
 6661 142414         SEQ     2,0
 6662   2477         JMP     @OTEMP
 6663  30006 ACSB:  LDA     2,IOP    ;ACCESS NEXT STRING BYTE AT (IBP)+1
 6664  24427         LDA     1,XIOP
 6665 146414         SEQ     2,1      ;FIRST INBYTE THIS USER ?
 6666    413         JMP     ACSB2    ;   YES
 6667  25004 ACSB4: LDA     1,IBP.,2 ;NO
 6670 125400         INC     1,1
 6671 101014         SKZ     0,0      ;STEP IBP UNLESS (AO)=0
 6672  11004         ISZ     IBP.,2
 6673 131222 ACBY:  MOVZR   1,2,SZC  ;ACCESS BYTE AT B(A1)  << GETBYTE ENTRY
 6674    755         JMP     ACRBY
 6675  21000 ACLBY: LDA     0,0,2    ;ACCESS LEFT BYTE
 6676  30023         LDA     2,CM400
 6677 113700         ANDS    0,2      ;BYTE TO A2
 6700   1400         JMP     0,3      ;BYTE ADDRESS IS IN A1

 6701  50412 ACSB2: STA     2,XIOP   ;SAVE IOP IN XIOP TO INDICATE XMEM IOB HAS BEEN READ IN
 6702  40410         STA     0,ACSBT
 6703  54500         STA     3,OURTN
 6704  34103         IOCALL
 6705   7761         IBCOPY           ;COPY IBUF FROM XMEM TO CORE IOB,  IF XMEM AVAIL.
 6706  34475         LDA     3,OURTN
 6707  20403         LDA     0,ACSBT
 6710  30006         LDA     2,IOP
 6711    756         JMP     ACSB4

 6712      0 ACSBT: 0
 6713      0 XIOP:  0               ;PCB WHOSE DATA IN IN SHARED IOB (XMEM CASE)
```

```
;                        << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------- --------- ---------- ------------------------ ----------------
;;
;;     2.7.   OUTBYTE                                  <<NON RE-ENTRANT>>
;;            NOTE:   WILL FAULT IF OUTPUT IS ALREADY IN PROGRESS.
;;
;;           PURPOSE:   STORE A BYTE INTO THE REGNANT USER'S I/O BUFFER
;;                      AND UPDATE OBP.(PCB).   USES TERM-TRANS TABLE IF
;;                      BYTE < 200.
;;           CALLING SEQUENCE:    OUTBYTE
;;
;;           ENTRY:   A2 <-- BYTE
;;           EXIT:    A0 <-- BYTE
;;                    OBP.(PCB) +<-- 1
;;---------- -------- ---------- ----------------------------------------

    6714   20064 STOB:  LDA     0,C377    ;STORE OUTPUT BYTE IN A2 AT (OBP)+1
    6715   40776        STA     0,XIOP    ;MARK SHARED INPUT BUFFER UNOCCUPIED
    6716  143400        AND     2,0
    6717   30006        LDA     2,IOP
    6720   25012        LDA     1,FLW.,2
    6721  125300        MOVS    1,1
    6722  125112        SSP     1,1
    6723     452        JMP     OBFLT     ;OUTPUT IS ACTIVE !?
    6724  101015        SNZ     0,0       ;STORING A ZERO BYTE ?
    6725     411        JMP     OBZBY     ;   YES
    6726   25037        LDA     1,TTN.,2  ;   NO
    6727  125300        MOVS    1,1
    6730  127112        ADDL#   1,1,SZC   ;OUTPUT TRANSLATION IN PROGRESS ?
    6731     431        JMP     OBTRC     ;   YES, CONTINUE IT
    6732   24056        LDA     1,C240    ;   NO
    6733  106032        SGE     0,1       ;POSSIBLE TRANSLATEE ?
    6734     426        JMP     OBTRC     ;   YES
    6735   30006 OBSTO: LDA     2,IOP     ;STORE BYTE IN A0 AT OBP+1
    6736   25005 OBZBY: LDA     1,OBP.,2
    6737   31003        LDA     2,LBA.,2
    6740  125400        INC     1,1
    6741  132020        ADCZ    1,2       ;SET C IF OBP+1 < LBA
    6742   30006        LDA     2,IOP
    6743  125012        MOV#    1,1,SZC   ;IS THERE ROOM ?
    6744   45005        STA     1,OBP.,2  ;   YES (ELSE OVERLAY LAST BYTE)
```

```
;                       << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;- - ------- - ------- ---------- ---------------------------
;;     2.8.   PUTBYTE                                 << RE-RENTRANT >>
;;
;;           STORE THE BYTE IN A0 AT THE BYTE ADDRESS IN A1 (PUTBYTE)
;;           RETURNS WITH A0 PRESERVED
;;
;;  NOTE:  NOT RE-ENTRANT IF 2 USERS STORE INTO THE SAME LOCATION.
;;-------- -- ---------------- ---- -- ----------------------

                        ; ***** NO CODE HERE *****

    6745 131220 STBY:  MOVZR   1,2       ;STORE BYTE IN A0 AT B(A1) << PUTBYTE ENTRY

;LSTBY IS AN ALTERNATE PUTBYTE ENTRY -- STORES BYTE AT ADDRESS IN AC2

    6746  25000 LSTBY: LDA     1,0,2     ;<< ENTRY FROM XSTBY [GET WORD POINTED TO BY (A2)]
    6747  55000        STA     3,0,2     ;SAVE RETURN ADDR.  IN BUF
    6750  34064        LDA     3,C377
    6751 163402        AND     3,0,SZC   ;MASK OFF HIGH BYTE; ODD BYTE TO STORE?
    6752 125300        MOVS    1,1       ;NO, REPOSITION
    6753 167703        ANDS    3,1,SNC
    6754 107301        ADDS    0,1,SKP
    6755 107000        ADD     0,1       ;FORM WORD WITH NEW BYTE
    6756  35000        LDA     3,0,2     ;GET RETURN ADDR FROM BUF
    6757  45000        STA     1,0,2     ;STORE NEW WORD WITH NEW BYTE INTO BUF
    6760   1400        JMP     0,3

    6761      0 OTEMP: 0         ;TEMP FOR OTXT, ACIB, MOVE


    6762  54421 OBTRC: STA     3,OURTN   ;SAVE RETURN ADDRESS
    6763  36417        LDA     3,@I.TTT
    6764 175415        INC#    3,3,SNR   ;IS THERE A $TERMS ?
    6765    403        JMP     OBTRR     ;   NO, STORE IN I/O BUFFER
    6766   5777        JSR     -1,3      ;YES, GO TO $TERMS(YS) OUTPUT PROCESS
                       ;
    6767    746        JMP     OBSTO     ;ACCESS TO OBSTO FROM $TERMn OR $TERMS
                       ;
    6770  30006 OBTRR: LDA     2,IOP     ;$TERMS RETURN
    6771 126120        ADCZL   1,1
    6772 106414        SEQ     0,1       ;ANY CHARACTER TO OUTPUT ?
    6773   4742        JSR     OBSTO     ;   YES
    6774   2407        JMP     @OURTN    ;   NO

    6775  24764 OBFLT: LDA     1,OTEMP   ;OUTBYTE WHILE OUTPUT IS ACTIVE !?
    6776 125220        MOVZR   1,1       ;RETURN ADDRESS FROM OUTTEXT
    6777 161000        MOV     3,0
    7000   6142        TRAPFAULT
    7001 115010        32*K!NOP

    7002    671 I.TTT: INFO+.TTT.        ;POINTER TO TERMINAL TYPE TABLE
    7003      0 OURTN: 0
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-------- ------- ---------- ---------- ---------- ----------
;;    2.8.  MOVE WORDS                                <<NON RE-ENTRANT>>
;;
;;          PURPOSE:  MOVE WORDS IN CORE.  PERFORMS "TUMBLE UP" OR
;;                    "TUMBLE DOWN" DEPENDING ON DIRECTION OF MOVE.
;;
;;          CALLING SEQUENCE:          CALL
;;                                     MOVEWORDS
;;
;;          ENTRY:  A0 <-- SOURCE ADDRESS
;;                  A1 <-- ENDING SOURCE ADDRESS
;;                  A2 <-- DESTINATION ADDRESS
;;
;;          EXIT:   IF (A0=A2)  RETURNS WITH REGISTERS THE SAME
;;                  ELSE IF (A0<A2) MOVES FROM BOTTOM OF LIST FIRST
;;                     A2 <-- FIRST DEST ADDR.
;;                     A3 <-- FIRST SOURCE ADDR.
;;                  ELSE (A0>A2) MOVES FROM TOP OF LIST FIRST
;;                     A2 <-- LAST DEST ADDRESS
;;                     A3 <-- LAST SOURCE ADDRESS
;;--- ------  ------ ---------- ----------------------------------

7004 112415 MOVEX: SNE    0,2       ;Already there (source = destination)?
7005   1400        JMP    0,3       ;  Yes, Nothing to do, Return

7006 106433        SLE    0,1       ;Does source area wraparound address space?
7007   2142        JMP    @TRAPFAULT&377 ;  Yes, TRAP reporting caller as problem area

7010  54751        STA    3,OTEMP   ;Save return address

7011 112433        SLE    0,2       ;Move area down in memory?
7012    450        JMP    MVWDN     ;  Yes, Move down with possibly overlapping src and dest
                                    ;  No, Move up with possibly overlapping areas
```

```
;                       << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

; Move source area to destination area which is at a higher address in
; memory.  The move is performed in blocks of eight words to increase
; efficiency and is performed from end to start in order to correctly
; handle overlapping source and destination areas.
; A0 = Source address, A1 = End of Source address, A2 = Destination address

   7013 135000          MOV     1,3         ;Start move at end of source area to handle overlap
   7014  40413          STA     0,MVWTP     ;Save A0
   7015 106400          SUB     0,1         ;Calc # of words to move minus one
   7016 133000          ADD     1,2         ;Calculate end address of destination area
   7017  20027          LDA     0,C7        ;Load mask of lower 3 bits
   7020 123400          AND     1,0         ;Calc size modulo 8
   7021 116400          SUB     0,3         ;Backup src for first move
   7022 112400          SUB     0,2         ;Backup dest for first move
   7023 103000          ADD     0,0         ;Multiple first cnt by 2
   7024  24434          LDA     1,.MV1UP    ;Load address of 1 word move entry point
   7025 106400          SUB     0,1         ;Calc entry addr for wd cnt
   7026  20431          LDA     0,MVWTP     ;Restore A0
   7027  44430          STA     1,MVWTP     ;Set entry address
   7030   2427          JMP     @MVWTP      ;Start moving

   7031  24030 MV8UP:   LDA     1,C10       ;Load constant eight
   7032 136400          SUB     1,3         ;Calc source address of next block
   7033 132400          SUB     1,2         ;Calc dest address of next block
   7034  25407          LDA     1,7,3       ;Entry to move up 8 words
   7035  45007          STA     1,7,2
   7036  25406          LDA     1,6,3
   7037  45006          STA     1,6,2
   7040  25405          LDA     1,5,3
   7041  45005          STA     1,5,2
   7042  25404          LDA     1,4,3
   7043  45004          STA     1,4,2
   7044  25403          LDA     1,3,3
   7045  45003          STA     1,3,2
   7046  25402          LDA     1,2,3
   7047  45002          STA     1,2,2
   7050  25401          LDA     1,1,3
   7051  45001          STA     1,1,2
   7052  25400 MV1UP:   LDA     1,0,3       ;Entry to move up 1 word
   7053  45000          STA     1,0,2
   7054 162414          SEQ     3,0         ;End of move?
   7055    754          JMP     MV8UP       ;  No, Move next 8 words
   7056   2703          JMP     @0TEMP      ;All done, Return

   7057      0 MVWTP:   0                   ;Temporary storage
   7060   7052 .MV1UP:          MV1UP
   7061   7120 .MV1DN:          MV1DN
```

```
;                       << SI = R91REXQSUBSA;  BO = 1/A.REXM5.7033! >>

; Move source area to destination area which is at a lower address
; in memory.  The move is performed in blocks of 8 to increase
; efficiency and is performed from start to end which correctly
; handles overlapping source and destination areas.
; A0 = Source address, A1 = End of Source address, A2 = Destination address

   7062 115000 MVWDN: MOV     0,3        ;Move source address to A3
   7063  44774         STA     1,MVWTP    ;Save A1
   7064 106400         SUB     0,1        ;Calc # of words to move minus one
   7065  20027         LDA     0,C7       ;Load mask for lower 3 bits
   7066 107400         AND     0,1        ;Calc size module 8
   7067 137000         ADD     1,3        ;Setup src addr for 1st mov
   7070 133000         ADD     1,2        ;Setup dst addr for 1st mov
   7071  20770         LDA     0,.MV1DN   ;Load address of 1 word move down entry
   7072 127000         ADD     1,1        ;Multiply size by 2
   7073 122400         SUB     1,0        ;Calc entry for 1st move
   7074  24763         LDA     1,MVWTP    ;Restore A1
   7075  40762         STA     0,MVWTP    ;Set entry address
   7076   2761         JMP     @MVWTP     ;Start moving

   7077  20030 MV8DN: LDA     0,C10      ;Load constant eight
   7100 117000         ADD     0,3        ;Calc source address of next block
   7101 113000         ADD     0,2        ;Calc destination address of next block
   7102  21771         LDA     0,-7,3     ;Entry to move 8 words
   7103  41371         STA     0,-7,2
   7104  21772         LDA     0,-6,3
   7105  41372         STA     0,-6,2
   7106  21773         LDA     0,-5,3
   7107  41373         STA     0,-5,2
   7110  21774         LDA     0,-4,3
   7111  41374         STA     0,-4,2
   7112  21775         LDA     0,-3,3
   7113  41375         STA     0,-3,2
   7114  21776         LDA     0,-2,3
   7115  41376         STA     0,-2,2
   7116  21777         LDA     0,-1,3
   7117  41377         STA     0,-1,2
   7120  21400 MV1DN: LDA     0,0,3      ;Entry to move 1 word down
   7121  41000         STA     0,0,2
   7122 166414         SEQ     3,1        ;End of move
   7123    754         JMP     MV8DN      ;   No, Move next 8 words
   7124   2635         JMP     @0TEMP     ;   Yes, Return to caller
```

```
;                        << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;--- ------ ------- -------- ------------------------------------
;;     2.9.   MOVE BYTES                                <<NON-RE-ENTRANT>>
;;
;;           PURPOSE:  TO MOVE STRINGS AROUND IN MEMORY
;;           CALLING SEQUENCE:   CALL
;;                               MOVBYTES
;;                               <MODE>
;;
;;                       WHERE <MODE> ::= BIT 15 = SOURCE ADDR REL TO SBA
;;                                        BIT 14 = DEST ADDR REL TO DBA
;;                                        LOWER BYTE = ALTERNATE TERMINATOR
;;
;;
;;           ENTRY:   A0 <-- SOURCE BYTE 1ST OFFSET FROM SOURCE WORD BASE
;;                    A1 <-- SOURCE BYTE LAST OFFSET FROM SOURCE WORD BASE
;;                    SBA <-- SOURCE WORD BASE ADDRESS
;;                    A2 <-- DEST. BYTE OFFSET
;;                    DBA <-- DEST.  WORD BASE ADDRESS
;;
;;           EXIT:    A0 <-- BYTE OFFSET (LAST SOURCE BYTE MOVED)
;;                    A1 <-- #SOURCE BYTES NOT MOVED
;;                    A2 <-- LAST BYTE TRANSFERRED
;;---------- ------------------- ----- ------------------------------

 7125       0          0
 7126   44457 MOBYX:  STA     1,BEBY    ;MOVE BYTES IN CORE
 7127  106423         SUBZ    0,1,SNC
 7130    1401         JMP     1,3       ;SOURCE IS EMPTY
 7131  125400         INC     1,1
 7132   44454         STA     1,BMCNT   ;NUMBER OF SOURCE BYTES
 7133   25400         LDA     1,0,3     ;MODE WORD
 7134  175400         INC     3,3
 7135   54770         STA     3,MOBYX-1
 7136   34040         LDA     3,SBA     ;GET USER BASE ADDRESS
 7137  101220         MOVZR   0,0       ;C=1 ==> PROCESS RIGHT SOURCE BYTE
 7140  125112         SSP     1,1       ;ARE SOURCE ADDRESSES RELATIVE ?
 7141  163000         ADD     3,0       ;   YES, ADD BASE ADDRESS
 7142   40446         STA     0,BMGTW   ;SAVE SOURCE WORD ADDRESS
 7143  101200         MOVR    0,0       ;SOURCE BYTE FLAG --> BIT 15
 7144  151220         MOVZR   2,2       ;DEST.  BYTE FLAG --> CARRY
 7145   34041         LDA     3,DBA     ;A(DEST.  BASE)
 7146  127112         ADDL#   1,1,SZC   ;ARE DEST.  ADDRESSES RELATIVE ?
 7147  173000         ADD     3,2       ;   YES, ADD BASE ADDRESS
 7150   50441         STA     2,BMSTW   ;SAVE DEST.  WORD ADDRESS
 7151   34064         LDA     3,C377
 7152  167400         AND     3,1
 7153   44434         STA     1,BMTMF   ;ALTERNATE TERMINATOR
 7154     405         JMP     MOBYG+4
```

```
;                       << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

 7155  24432 MOBYG: LDA     1,BMTMF
 7156 151014         SKZ     2,2; IS IT A ZERO BYTE . . .
 7157 146415         SNE     2,1         ;OR ALTERNATE TERMINATOR ?
 7160    421         JMP     MOBYD       ;   YES, DONE
 7161  32427         LDA     2,@BMGTW;    NO, GET A WORD
 7162 101103         MOVL    0,0,SNC   ;PROCESSING RIGHT SOURCE BYTE ?
 7163 151301         MOVS    2,2,SKP   ;   NO
 7164  10424         ISZ     BMGTW     ;   YES, ADVANCE SOURCE POINTER
 7165 173400         AND     3,2
 7166  26423         LDA     1,@BMSTW
 7167 101262         MOVCR   0,0,SZC   ;PRESERVE LEFT DEST. BYTE ?
 7170 125300         MOVS    1,1       ;   YES
 7171 167700         ANDS    3,1       ;MOVE PRESERVED BYTE TO LEFT
 7172 147003         ADD     2,1,SNC   ;WORD CORRECT ?
 7173 125300         MOVS    1,1       ;   NO
 7174  46415         STA     1,@BMSTW
 7175 101063         MOVC    0,0,SNC   ;JUST STORED RIGHT BYTE ?
 7176  10413         ISZ     BMSTW     ;   YES, BUMP DEST. POINTER
 7177  14407         DSZ     BMCNT     ;DONE ?
 7200    755         JMP     MOBYG     ;   NO
 7201  24405 MOBYD: LDA     1,BMCNT   ;   YES
 7202  20403         LDA     0,BEBY    ;(A1) = # SOURCE BYTES NOT MOVED
 7203 122400         SUB     1,0       ;(A0) = B(LAST SOURCE BYTE MOVED)
 7204   2721         JMP     @MOBYX-1  ;(A2) = LAST BYTE TRANSFERRED


 7205      0 BEBY:  0       ;B(LAST SOURCE BYTE)
 7206      0 BMCNT: 0       ;SOURCE BYTE COUNTER
 7207      0 BMTMF: 0       ;ALTERNATE TERMINATOR
 7210      0 BMGTW: 0       ;SOURCE WORD ADDRESS
 7211      0 BMSTW: 0       ;DEST.  WORD ADDRESS
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-- ------- ------- ----- -------- -- ------------------------------
;;     4.1.   BINARY MULTIPLY                          <<RE-ENTRANT>>
;;
;;           PURPOSE:  PERFORM A MULTIPLICATION OF 2 16-BIT BINARY NUMBERS
;;
;;           ENTRY:    A0 <-- MULTIPLICAND
;;                     A2 <-- MULTIPLIER
;;
;;           EXIT:     A3,A0 <-- (A0) * (A2)
;;                     A1 <-- UNCHANGED
;;                     A2 <-- UNCHANGED
;;
;;           CALLING SEQUENCE:  BINMULTIPLY
;;
;;           INTERRUPTS ARE DISABLED FOR 88 MEMORY CYCLES (35 MICROSECONDS
;;           ON A POINT 4 MARK 5 CPU)
;;--- -------  ---------------------------- ---------------------------

    7212  60277 BMUL:  INTDS
    7213  54415        STA     3,MDRTN   ;BINARY MULTIPLY
    7214  44415        STA     1,M.A1    ;  A3,A0 = (A0)*(A2)
    7215 126520        SUBZL   1,1       ;  A3 IS TOP HALF
    7216 176400        SUB     3,3       ;  A1 IS UNCHANGED
    7217 101203 BMULP: MOVR    0,0,SNC   ;  A2 IS UNCHANGED
    7220 175201        MOVR    3,3,SKP
    7221 157220        ADDZR   2,3
    7222 127004        ADD     1,1,SZR
    7223    774        JMP     BMULP
    7224 101260        MOVCR   0,0
    7225  24404        LDA     1,M.A1
    7226  60177        INTEN
    7227   2401        JMP     @MDRTN


    7230      0 MDRTN: 0                 ;MULTIPLY & DIVIDE RETURN ADDRESS
    7231      0 M.A1:  0                 ;MULTIPLY TEMP.  STORE




;--- ------- - ------- ----------------------------------------------
;
;  NOTE:    THIS ROUTINE (BMUL) IS OVERLAID WHEN $EIS IS ACTIVE
;
;---- -------  ------------------------------------------------------
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;--- -------- ---------- ----------------------------------------------
;;     4.2.   BINARY DIVIDE                                    <<RE-ENTRANT>>
;;
;;           PURPOSE:   TO PERFORM A BINARY DIVISION OF 2 16 BIT BINARY #'S
;;
;;           CALLING SEQUENCE:    BINDIVIDE
;;
;;           ENTRY:     A0 <-- DIVISOR
;;                      A1 <-- DIVIDEND
;;
;;           EXIT:      A0 <-- UNCHANGED
;;                      A1 <-- REMAINDER
;;                      A2 <-- 0
;;                      A3 <-- (A1)/(A0) = QUOTIENT
;;
;;           INTERRUPTS ARE DISABLED FOR A MAXIMUM OF 92 MEMORY CYCLES
;;           (100000/1), A MINIMUM OF 8 CYCLES (IF (A0)>(A1) )
;;---------- - -------- ---------- ---------- ----------------------------

    7232 101015 BDIV:  SNZ     0,0        ; A3 = (A1) / (A0)
    7233   2142        JMP@    TRAPF&377; A1 = REMAINDER
    7234 152520        SUBZL   2,2        ; A0 IS UNCHANGED (MUST BE >0)
    7235 101123 BDVL1: MOVZL   0,0,SNC    ; A2 = 0
    7236 151401        INC     2,2,SKP
    7237 101201        MOVR    0,0,SKP
    7240 106433        SLE     0,1
    7241 150401        NEG     2,2,SKP
    7242    7/3        JMP     BDVL1

    7243  6027/        INTDS
    7244  54764        STA     3,MDRTN
    7245 176401        SUB     3,3,SKP
    7246 101220 BDVL2: MOVZR   0,0
    7247 106423        SUBZ    0,1,SNC
    7250 175121        MOVZL   3,3,SKP
    7251 175141        MOVOL   3,3,SKP
    7252 107000        ADD     0,1
    7253 151404        INC     2,2,SZR
    7254    7/2        JMP     BDVL2

    7255  6017/        INTEN
    7256   2752        JMP     @MDRTN    ;AND RETURN


;---- ------ - ---------- -- -------- ----- ----------
;
; NOTE:  THIS ROUTINE (BDIV) IS OVERLAID IF $EIS IS ACTIVE
;
;---- --- --- ---------- ----------------------------------------------
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-------  --------  --------  ----------------  ------------------------
;;      4.3.   DUMMY FOR DECIMAL ARITHMETIC                <<RE-ENTRANT>>
;;
;;         PURPOSE:   TO INTERCEPT "DECIMAL" CALLS WHEN THERE IS NO
;;                    $DEC OR $DAU INSTALLED.
;;
;;         CALLING SEQUENCE:        DECIMAL
;;
;;         ENTRY:    A0 <-- OPERATION (SEE SUBR. MANUAL)
;;
;;         EXIT:     ERRF <-- #0
;;                   A2 <-- 0
;;----------  ---------------------------  ---  -  ---------------------

     7257    7266          DECX5
     7260    7266          DECX5
     7261   24026 DECX:    LDA     1,C6       ;NO-OP "DECIMAL ARITHMETIC"
     7262   30031          LDA     2,C11
     7263  106414          SEQ     0,1        ;INPUT, OR . . .
     7264  112415          SNE     0,2        ;FORMATTED OUTPUT ?
     7265  175400          INC     3,3        ;   YES, SKIP RETURN
     7266  102400 DECX5:   SUB     0,0
     7267  126400          SUB     1,1
     7270  152400          SUB     2,2
     7271   54076          STA     3,ERRF     ;SET ERROR FLAG
     7272    1400          JMP     0,3
```

```
;                       << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------- - ------------------------- ----- ----- -- ---------
;;
;;     5.1.   START INPUT
;;
;;          ENTRY:    AT STI FOR STANDARD INPUT
;;                    AT SSI FOR SPECIAL INPUT
;;
;;           EXIT:    IF IOP=RUP  BUMP TO SWAP THE USER OUT UNTIL THE INPUT IS COMPLETE
;;                    ELSE        NON-SKIP RETURN TO CALLER
;;
;;        PURPOSE:    STI PERFORMS A STANDARD 'INPUT' FUNCTION FOR THE PORT SPECIFIED BY IOP USING
;;                    THE FULL PORT I/O BUFFER FOR THE INPUT.
;;                    SSI PERFORMS A SPECIAL 'INPUT' FUNCTION FOR THE PORT SPECIFIED BY IOP WHERE THE
;;                    LENGTH OF THE BUFFER AND THE TIMEOUT INTERVAL ARE GIVEN AS PARAMETERS IN
;;                    THE CALL.
;;
;;                    FOR BOTH ENTRIES THE PORT IS SET INTO THE 'INPUT-ACTIVE'
;;                    STATE AND IF OUTPUT IS NOT CURRENTLY ACTIVE A
;;                    CALL TO QCHARACTER IS MADE TO GET THE INPUT GOING.
;;
;;   CALLING SEQUENCE:      STINPUT FOR STANDARD START INPUT
;;
;;                  OR:      (A0)=DESIRED BUFFER LENGTH (<= 0 ==> STANDARD)
;;                           IF BIT 14 SET, RING BELL ON BUFFER FULL
;;                           (A1)=TIME OUT FOR INPUT
;;                           SPINPUT
;;
;;-------- - ------------------------- ---- ------------------

7273 102000 STI:   ADC     0,0       ;START INPUT -- STANDARD
7274 126400        SUB     1,1       ;NO TIME OUT
                   ;
7275  6027/ SSI:   INTDS               ;START INPUT -- SPECIAL
7276  30005        LDA     2,RUP     ;GET REGNANT USER'S PCB ADDRESS
7277  55014        STA     3,URA.,2  ;SAVE RETURN ADDRESS
7300  34006        LDA     3,IOP     ;GET ADDRESS OF SPECIFIED PCB
7301 156414        SEQ     2,3       ;STARTING REGNANT USER ?
7302    404        JMP     SSISI     ;   NO
7303  30073        LDA     2,ESCF    ;   YES
7304 151014        SKZ     2,2       ;ESCAPE PRESSED ?
7305    445        JMP     SSIBU     ;   YES, BUMP WITHOUT STARTING INPUT

7306  45426 SSISI: STA     1,PDC.,3  ;SET OR RESET TIME OUT
7307 101415        INC#    0,0,SNR   ;LEN=-1?
7310 102400        SUB     0,0       ;   YES, USE 0
7311  24415        LDA     1,C40K    ;MASK FOR FIXED LENGTH FIELD MODE
7312  35415        LDA     3,ABN.,3
7313  30412        LDA     2,VCABN
7314 157400        AND     2,3       ;CLEAR ABN FXD LEN FIELD AND TIMEOUT STATUS
7315 107404        AND     0,1,SZR   ;FIXED LEN FIELD MODE?
7316 156000        ADC     2,3       ;   YES, SET ABN
7317 122400        SUB     1,0       ;RESET LEN OUT
7320  30006        LDA     2,IOP
7321  55015        STA     3,ABN.,2  ;SET ABN
```

```
;                       << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

 7322  25002          LDA     1,FBA.,2 ;B(I/O BUFFER)
 7323 100536          SZN     0,0      ;USING THE DEFAULT BUFFER SIZE ?
 7324 123022          ADDZ    1,0,SZC  ;   NO, CALCULATE NEW END OF BUFFER
 7325 102000          ADC     0,0      ;   YES, OR BUFFER WRAPAROUND
 7326  35003          LDA     3,LBA.,2 ;ORIGINAL LBA (SET BY SIR)
 7327 116433          SLE     0,3      ;NEW LIMIT LEGAL (DOESN'T EXCEED ORIGINAL SIZE) ?
 7330 161000          MOV     3,0      ;   NO, LIMIT BUFFER SIZE TO THAT OF ORIGINAL BUFFER
 7331  25012          LDA     1,FLW.,2
 7332  34051          LDA     3,C100
 7333 137414          AND#    1,3,SZR  ;IS INPUT ALREADY ACTIVE ?
 7334   6142          TRAPFAULT        ;   YES -- VERY BAD !!
 7335  41006          STA     0,LIB.,2 ;NO, SET BYTE POINTER
 7336 167000          ADD     3,1      ;SET INPUT-ACTIVE BIT
 7337  45012          STA     1,FLW.,2 ;RESTORE PORT FLAGS
 7340 102000          ADC     0,0
 7341  42416          STA     0,@.XIOP ;MARK SHARED INPUT BUFFER UNASSIGNED
 7342 125300          MOVS    1,1
 7343  20411          LDA     0,VSTINP
 7344 125113          SSN     1,1      ;IS OUTPUT ACTIVE ?
 7345   6103          QCHARACTER       ;   NO, CAUSE INPUT TO BE STARTED
 7346  60177          INTEN
 7347  34005          LDA     3,RUP    ;GET REGNANT USER'S PCB ADDRESS
 7350 156414          SEQ     2,3      ;STARTED INPUT ON REGNANT USER'S PORT ?
 7351   3414          JMP     @URA.,3  ;   NO, RETURN TO CALLER
 7352  35414 SSIBU:   LDA     3,URA.,3 ;   YES, BUMP USER
 7353   2117          JMP     @BUMPU&377

 7354    402 VSTINP:          QCSIN    ;START INPUT TASK
 7355 177557 VCABN:           -1-PORTTOUT-20
 7356  40000 C40K:            40000
 7357   6713 .XIOP:           XIOP
```

```
;                        << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------
;;
;;      5.2.   START OUTPUT
;;
;;           ENTRY:    A3 <-- RETURN ADDRESS
;;                     OBP.(PCB) <-- BYTE ADDR OF LAST BYTE TO OUTPUT
;;                     FBA.(PCB) <-- BYTE ADDR OF FIRST BYTE TO OUTPUT
;;                     SND.(PCB) <-- ADDRESS OF "SEND" SUBROUTINE IN
;;                                   MUX DRIVER
;;
;;           EXIT:     RETURNS AT POINT OF CALL + 1
;;
;;           PURPOSE:  TO INITIATE OUTPUT TO THE PORT IDENTIFIED BY IOP.
;;
;;-------------------------------------------------------------------

 7360  30006 STO:   LDA     2,IOP      ;START OUTPUT
 7361  20073         LDA     0,ESCF
 7362 101014         SKZ     0,0        ;ESCAPE PRESSED ?
 7363   1400         JMP     0,3        ;   YES, DON'T START OUTPUT
 7364  54425         STA     3,STOA3    ;SAVE RETURN ADDRESS
 7365  6027/         INTDS
 7366  21012         LDA     0,FLW.,2
 7367  24053         LDA     1,C200
 7370 107414         AND#    0,1,SZR    ;OUTPUT ALREADY ACTIVE ?
 7371    416         JMP     STOR       ;   YES, LEAVE IT ALONE
 7372 123000         ADD     1,0        ;NO, SET "OUTPUT ACTIVE"
 7373  41012         STA     0,FLW.,2
 7374  34103         IOCALL
 7375   7762         OBCOPY             ;COPY OUTPUT INTO XMEM (IF ANY), AND RESET POINTERS
 7376  21012         LDA     0,FLW.,2
 7377  25011         LDA     1,TOB.,2
 7400 125015         SNZ     1,1        ;OUTPUT CHANNEL STILL OCCUPIED (FROM ECHO)
 7401 103112         ADDL#   0,0,SZC    ;   OR IS X-OFF FLAG SET ?
 7402    405         JMP     STOR       ;      YES, DON'T START OUTPUT NOW
 7403 102000         ADC     0,0        ;NO, TELL DRIVER TO START OUTPUT
 7404   7032         JSR     @SND.,2    ;(RE-ENABLES INTERRUPTS)
 7405 101415         INC#    0,0,SNR    ;REQUEST FIRST CHARACTER ?
 7406   6103         QCHARACTER         ;   YES, QUEUE UP THE REQUEST
 7407  6017/ STOR:   INTEN
 7410   2401         JMP     @STOA3

 7411      0 STOA3:  0                  ;RETURN ADDRESS
```

```
;                       << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------- -------- ------ ---------- ----------------------------
;;
;;     5.3.   WAIT FOR OUTPUT NOT ACTIVE
;;
;;          ENTRY:   A3 <-- RETURN ADDRESS
;;
;;          EXIT:    IF (OUTPUT ACTIVE) GOES TO BUMP VIA "BUMPUSER"
;;                   ELSE
;;                        IF (PROCESSOR IGNORED ESCAPE FLAG)
;;                             THEN GOES TO BUMP
;;------------ ----- ------------------------------ ------------------

     7412  30005 WONAX: LDA     2,RUP     ;WAIT FOR OUTPUT NOT ACTIVE
     7413  20073        LDA     0,ESCF
     7414 101014        SKZ     0,0       ;DID PROCESSOR IGNORE ESC FLAG ?
     7415   2117        JMP     @BUMPU&377;   YES, BUMP USER
     7416  21012        LDA     0,FLW.,2 ;NO
     7417 101300        MOVS    0,0
     7420 101112        SSP     0,0       ;IS OUTPUT ACTIVE ?
     7421   2117        JMP     @BUMPU&377;   YES, BUMP USER
     7422  22404        LDA     0,@.OBIN ;   NO, PICK UP ADDRESS OF OBINIT
     7423  40402        STA     0,.+2
     7424   2401        JMP     @.+1      ;GO THERE WITH A2=RUP & A3 PRESERVED
     7425      0        0

     7426  10024 .OBIN: OBINIT!177400+QCHAX
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-----------  ------------------------------  ------------------
;;
;;    6.1   CLEAR ALL DATA CHANNELS  (ALCLR)      <<NON RE-ENTRANT>>
;;
;;        CALLING SEQUENCE:      CALL
;;                               ALLCLEAR
;;        ENTRY:   SHOULD BE CALLED BY REGNANT USER
;;        EXIT:    FOR EACH DATA CHANNEL IN THE REGNANT USER'S DATA FILE
;;                 TABLE, THE CHANNEL IS CLEARED.
;;-----------  ------------------  ------------  ---------  ----------

     7427     603 .NDCH: INFO+NDCH.
     7430       0        0
     7431       0 UTEMP: 0
     7432   54777 ALCLR: STA     3,.-1      ;CLEAR ALL DATA CHANNELS
     7433   22774        LDA     0,@.NDCH
     7434   40774        STA     0,ALCLR-2
     7435   20773        LDA     0,ALCLR-2
     7436  100400        NEG     0,0
     7437  100000        COM     0,0
     7440    6106        CHANNEL
     7441      27        CLEAR
     7442    6142        TRAPFAULT          ;ILLEGAL CHANNEL NUMBER !?
     7443  114010        30*K!NOP
     7444   14764        DSZ     ALCLR-2
     7445     770        JMP     ALCLR+3
     7446    2763        JMP     @ALCLR-1
```

```
;                        << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------------
;;
;;     6.2  UNLOCK A RECORD IN A CHANNEL              <<NON RE-ENTRANT>>
;;
;;       CALLING SEQUENCE:      CALL
;;                              UNLOCK
;;
;;       ENTRY:         A0 <-- CHANNEL #
;;                      MUST BE CALLED BY REGNANT USER
;;       EXIT:
;;                      A0 <-- CHANNEL STATUS (FROM STS IN CHANNEL)
;;                      A2 <-- A (CHANNEL) = A (DFT ENTRY)
;;
;;
;;       PURPOSE:       TO RESET THE LOCK BIT IN "STS" WORD OF A CHANNEL.
;;                      THIS ALLOWS USER TO KEEP RECORD AROUND, WITHOUT
;;                      WRITING IT, BUT LET OTHER USERS READ IT.
;;-------------------------------------------------------------------------

 7447   54762 UNLKX: STA     3,UTEMP   ;UNLOCK RECORD IN CHANNEL #(A0)
 7450  101113         SSN     0,0       ;NEGATIVE CHANNEL . . .
 7451    4410         JSR     CHCHX     ;OR CHANNEL NOT OPEN ?
 7452    2757         JMP     @UTEMP    ;   YES, NON-SKIP RETURN
 7453   21003         LDA     0,STS,2   ;   NO
 7454  101100         MOVL    0,0
 7455  101220         MOVZR   0,0       ;CLEAR LOCK STATUS BIT
 7456   41003         STA     0,STS,2
 7457   10752         ISZ     UTEMP     ;SKIP RETURN
 7460    2751         JMP     @UTEMP
```

```
;                        << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------------
;;     6.3  CHECK IF CHANNEL IS IN USE FOR REGNANT USER      <<RE-ENTRANT>>
;;
;;     CALLING SEQUENCE:      CALL
;;                            CHKCHANEL
;;
;;     ENTRY:       A0 <-- CHANNEL #
;;
;;     EXIT:        A0 <-- CHANNEL #
;;                  A2 <-- A(CHANNEL) = A (DFT ENTRY)
;;                  IF (OPEN) SKIP
;;                  ELSE (NOT OPEN) DON'T SKIP
;;
;;     PURPOSE:     TO DETERMINE IF A GIVEN CHANNEL IS OPEN
;;-------------------------------------------------------------------------

 7461 126120 CHCHX: ADCZL   1,1         ;CHECK CHANNEL #(A0)
 7462 106033        SLS     0,1         ;CHANNEL -1 OR -2 ?
 7463    404        JMP     .+4         ;   YES
 7464  26743        LDA     1,@.NDCH    ;   NO
 7465 106033        SLS     0,1
 7466   1400        JMP     0,3         ;ILLEGAL CHANNEL NUMBER
 7467  30005        LDA     2,RUP
 7470  25025        LDA     1,DFT.,2
 7471 111120        MOVZL   0,2         ;CHANGE IF CHM1 IS CHANGED    ***
 7472 151120        MOVZL   2,2
 7473 151120        MOVZL   2,2
 7474 133000        ADD     1,2
 7475  25001        LDA     1,FDA,2
 7476 125015        SNZ     1,1
 7477   1400        JMP     0,3         ;CHANNEL NOT IN USE
 7500   1401        JMP     1,3         ;CHANNEL IN USE
```

```
;                    << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-------- ----------------------------------------------------------
;;
;;    6.4  CHECK PROTECTION ON FILE                <<NON RE-ENTRANT>>
;;
;;         CALLING SEQUENCE:     CALL
;;                               <PROTECT-INDEX>
;;              WHERE <PROTECT INDEX> ::= CHKRP - "CHECK READ PROTECT"
;;                                        CHKCP - "CHECK COPY PROTECT "
;;                                        CHKWP - "CHECK WRITE PROTECT"
;;
;;          ENTRY:   A0 <-- ACNT(HEADER)
;;                   A1 <-- TYPE (HEADER)
;;
;;                   FOR CHKCP ENTER AT CHKPB
;;                   FOR CHKWP ENTER AT CHKPB+1
;;                   FOR CHKRP ENTER AT CHKPB+2
;;
;;        PURPOSE:    TO DETERMINE IF A FILE IS PROTECTED FROM A
;;                   PARTICULAR TYPE OF USER ACCESS
;;
;;        EXIT:      IF (PROTECTED) THEN DON'T SKIP
;;                   ELSE (ACCESS OK) SKIP
;;---------- ----------------------------------------------------------

 7501 125100 CHKPB: MOVL    1,1        ;CHECK COPY PROTECT BIT
 7502 125100        MOVL    1,1        ;CHECK WRITE PROTECT BIT
 7503  44726        STA     1,UTEMP    ;CHECK READ PROTECT BIT
 7504  30005        LDA     2,RUP
 7505  25010        LDA     1,ACT.,2
 7506 131000        MOV     1,2
 7507 112400        SUB     0,2
 7510 151124        MOVZL   2,2,SZR    ;SAME ACCOUNT NUMBER ?
 7511 151125        MOVZL   2,2,SNR
 7512   1401        JMP     1,3        ;   YES, ACCESS GRANTED
 7513  30402        LDA     2,.+2      ;   NO
 7514 147401        AND     2,1,SKP    ;PRIV LEVEL OF USER
 7515 140000        140000
 7516 113400        AND     0,2        ;PRIV LEVEL OF FILE
 7517 125137        MOVZL#  1,1,SBN    ;USER PRIV = 3   . . .  OR
 7520 132433        SLE     1,2        ;USER PRIV > FILE PRIV ?
 7521    411        JMP     CHKP2      ;   YES
 7522  20707        LDA     0,UTEMP    ;   NO
 7523 146414        SEQ     2,1        ;USER PRIV = FILE PRIV ?
 7524    403        JMP     .+3        ;   NO
 7525 103100 CHKP1: ADDL    0,0        ;   YES, SHIFT TYPE WORD
 7526 101100        MOVL    0,0
 7527 103112        ADDL#   0,0,SZC    ;PROTECTED ?
 7530   1400        JMP     0,3        ;   YES
 7531   1401        JMP     1,3        ;   NO, ACCESS GRANTED
```

```
;                       << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

     7532 125112 CHKP2: SSP     1,1         ;USER PRIV > 1 ?
     7533   1401        JMP     1,3         ;   YES, ACCESS GRANTED
     7534  30005        LDA     2,RUP       ;   NO, USER PRIV = 1
     7535  25010        LDA     1,ACT.,2    ;       AND FILE PRIV = 0
     7536  30402        LDA     2,.+2
     7537 147401        AND     2,1,SKP     ;USER'S GROUP NUMBER
     7540  37700        37700
     7541 143400        AND     2,0         ;FILE'S GROUP NUMBER
     7542 122415        SNE     1,0         ;SAME GROUP ?
     7543   1401        JMP     1,3         ;   YES, ACCESS GRANTED
     7544  20665        LDA     0,UTEMP     ;   NO
     7545    760        JMP     CHKP1       ;CHECK AS IF SAME PRIV
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-------- ---------- ---------- ----------
;;
;;      6.7  POLYFILE VOLUME LOOK UP              <<NON REENTRANT>>
;;
;;  ON ENTRY:
;;       AC0     = VOLUME NUMBER
;;       AC2     = VOLUME ZERO HEADER BLOCK BUFFER ADDRESS
;;
;;  NONSKIP RETURN IF VOLUME NOT FOUND WITH:
;;      AC2     =  (Unchanged)
;;      AC3     =   177777                    --> INVALID VOLUME #
;;              =  A(VOL INFO TABLE ENTRY)    --> VOLUME NOT DEFINED
;;
;;  ELSE SKIP RETURN WITH:
;;       AC0     = VOLUME LOGICAL UNIT
;;       AC1     = RDA OF VOLUME HEADER
;;       AC2     = (Unchanged)
;;       AC3     = A(VOL INFO TABLE ENTRY)
;;-- ---------- ---------- -- ----------

 7546  54424 VLKPX: STA     3,VLRET   ;Save Return
 7547 176000         ADC     3,3
 7550  24420         LDA     1,CMVLN   ;Maximum Volume Number
 7551 106433         SLE     0,1       ;Legal Volume # ?
 7552   2420         JMP     @VLRET    ;   No - Non-skip with AC3 = 177777
 7553  34416         LDA     3,CDVIT   ;   Yes - Get Offset to VIT from Start of Header
 7554 157000         ADD     2,3       ;Start of VIT
 7555 105120         MOVZL   0,1       ;Makes Volume Number times 2
 7556 107000         ADD     0,1       ;Makes Volume Number times 3
 7557 137000         ADD     1,3       ;A(VITE) for requested volume
 7560  21400         LDA     0,VLU,3   ;VLU for requested volume
 7561 101015         SNZ     0,0       ;Is this volume defined?
 7562   2410         JMP     @VLRET    ;   No - Undefined Volume
 7563  24052         LDA     1,C177    ;   Yes - Extract Logical Unit Number
 7564 123400         AND     1,0       ;to be returned in AC0
 7565  25402         LDA     1,HRDA,3  ;and get RDA to be returned in AC1
 7566  10404         ISZ     VLRET     ;Bump to cause SKIP RETURN and
 7567   2403         JMP     @VLRET    ;Return to caller.

 7570     47 CMVLN:  MVOLN             ;Highest Legal Volume Number
 7571    200 CDVIT:  DVIT              ;Offset from Start of Header to VIT
 7572      0 VLRET:  0                 ;RETURN ADDRESS



 7573      0 VLFTY:  0                 ;Requested Volume Type
 7574      0 VLFVN:  0                 ;Current (previous) Volume Number
 7575 160000 VLFMK:  160000            ;Volume type Mask
 7576      0 VLRTN:  0                 ;Return Address Storage
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-------- --- ------- ----------------------------------------------
;;      6.8  FIND NEXT VOLUME OF POLYFILE       <<NON REENTRANT>>
;;
;;  CALLING SEQUENCE:
;;
;;          AC0 = Requested Type in VLU format (Bits 12 thru 0 ignored)
;;          AC1 = Volume Zero Header Block Address
;;          AC2 = A(Last VITE Examined); Or Zero
;;          CALL
;;          VOLFIND
;;          Error Return
;;          Normal return
;;
;;     NOTE:  On entry, IF AC2 <> 0 then
;;                      Set AC0 = VLU of VITE pointed to by AC2
;;
;;  RETURN IS NON-SKIP IF NOT FOUND
;;
;;  ELSE SKIP RETURN WITH :
;;            AC0 = Volume Type Word (VLU)
;;            AC1 = Volume Header's RDA (HRDA)
;;            AC2 = A(VITE)
;;            AC3 = Volume Number
;;-------------------- ------- ----------------------------------

  7577  54777 VLFNX: STA     3,VLRTN  ;Save return address
  7600 151014        SKZ     2,2      ;Is this a continuation?
  7601  21000        LDA     0,VLU,2  ;  Yes - Get type from last VITE
  7602  34773        LDA     3,VLFMK
  7603 163400        AND     3,0      ;Extract Volume Type
  7604  40767        STA     0,VLFTY  ;and save as Requested Type.
  7605 176000        ADC     3,3      ;Prepare to start from the top.
  7606 151015        SNZ     2,2      ;Is this a continuation ?
  7607  54765        STA     3,VLFVN  ;  No - Initialize Volume Number
  7610 131000        MOV     1,2      ;A(Volume Zero Header Block)
  7611  10763 VLFD1: ISZ     VLFVN    ;Bump Volume Number
  7612 100010        NOP
  7613  20761        LDA     0,VLFVN  ;Get Next Volume Number
  7614   4732        JSR     VLKPX    ;Call VOLUME LOOKUP (VOLLK)
  7615    415        JMP     VLFD2    ;Volume doesn't exist !
  7616  21400        LDA     0,VLU,3  ;Get Volume VLU
  7617  24756        LDA     1,VLFMK
  7620 123400        AND     1,0      ;Extract Volume's type
  7621  24752        LDA     1,VLFTY  ;Get Requested type
  7622 122414        SEQ     1,0      ;Does this Volume match request ?
  7623    766        JMP     VLFD1    ;  No - Not this one.
  7624 171000        MOV     3,2      ;  Yes - put A(VITE) into AC2
  7625  21000        LDA     0,VLU,2  ;Get full VLU for this volume,
  7626  25002        LDA     1,HRDA,2 ;it's RDA,
  7627  34745        LDA     3,VLFVN  ;and it's Volume Number.
  7630  10746        ISZ     VLRTN    ;Establish Skip Return
  7631   2745        JMP     @VLRTN   ;And return with found volume.

  7632 175415 VLFD2: INC#    3,3,SNR  ;Legal Volume Number ?
  7633   2743        JMP     @VLRTN   ;  No - No more to check.
  7634    755        JMP     VLFD1    ;  Yes - Go check next volume.
```

```
;                       << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------
;;     7.1.   IS A2 A LETTER?                          <<RE-ENTRANT>>
;;
;;         PURPOSE:   DETERMINE IF THE CONTENTS OF A2 IS THE ASCII
;;                    REPRESENTATION OF A LETTER
;;
;;         CALLING SEQUENCE:   ISA2LETTER
;;
;;         ENTRY:     A2 <-- VALUE
;;         EXIT:      IF (NOT A LETTER ) THEN NON-SKIP
;;                       A2 <-- UNCHANGED
;;                    ELSE (LETTER) SKIP
;;                       A2 <-- INDEX OF LETTER
;;                       A1 <-- CONSTANT  << (A1) + (A2) = LETTER >>
;;-------------------------------------------------------------------

    7635    332         332
    7636    340         340
    7637    372         372
    7640  20777 IA2L:   LDA     0,.-1     ;IS (A2) A LETTER ?
    7641  24062         LDA     1,C300
    7642 142432         SGR     2,0
    7643 146432         SGR     2,1
    7644   1400         JMP     0,3       ;   NO
    7645  20770         LDA     0,IA2L-3  ;   MAYBE
    7646 142432         SGR     2,0       ;UPPER CASE LETTER ?
    7647    404         JMP     .+4       ;   YES
    7650  24766         LDA     1,IA2L-2  ;   NO
    7651 146432         SGR     2,1       ;LOWER CASE LETTER ?
    7652   1400         JMP     0,3       ;   NO
    7653 132400         SUB     1,2       ;   YES, CONVERT TO LETTER NUMBER
    7654   1401         JMP     1,3
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-- -------- ---------------------- ---------- ----------------
;;      7.2.   IS A2 A DIGIT?                                <<RE-ENTRANT>>
;;
;;           PURPOSE:   TO DETERMINE IF A2 IS A DIGIT
;;                      (I.E.  THE ASCII REPRESENTATION OF A DIGIT)
;;
;;           CALLING SEQUENCE:   ISA2DIGIT
;;
;;           ENTRY:     A2 <-- VALUE
;;
;;           EXIT:      IF (GOOD) THEN SKIP
;;                         A0 <-- UNCHANGED
;;                         A1 <-- CONSTANT << ASCII ZERO >>
;;                         A2 <-- UNCHANGED
;;                      ELSE (NOT A DIGIT) NON-SKIP
;;                         A2 <-- UNCHANGED
;;---- ----- --- ------------------------------------ ---------------

     7655  24061 IA2D:  LDA     1,C271    ;IS (A2) A DIGIT ?
     7656 146433        SLE     2,1
     7657   1400        JMP     0,3       ;   NO
     7660  24060        LDA     1,C260
     7661 146032        SGE     2,1
     7662   1400        JMP     0,3       ;   NO
     7663   1401        JMP     1,3       ;   YES
```

```
;                       << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>
;;-------------------------------------------------------------------------
;;      7.3  CHANGE OR CHECK A FLAG                              <<REENTRANT>>
;;
;;           PURPOSE:    EXAMINES OR ALTERS A BIT FLAG AT A SPECIFIED
;;                       LOCATION IN MEMORY
;;           CALLING SEQUENCE:     FLAGCHANGE
;;                                 <COMMAND>+<DISPLACEMENT>+<SKIP>
;;                                 MASK
;;                 WHERE:
;;                 <COMMAND>   ::=  SET,RESET,TOGGLE,<OMITTED>
;;                 <SKIP>      ::=   SKIPZ,SKIPO,<OMITTED>
;;                                   (SKIPZ = SKIP IF ALL MASKED BITS = 0)
;;                                   (SKIPO = SKIP IF ANY MASKED BITS = 1)
;;                 <DISPLACEMENT> ::= OFFSET FROM PTR IN A2, TO FLAGWORD
;;
;;            ENTRY:   A2 <-- POINTER TO TABLE
;;
;;            EXIT:    A2 <-- PTR TO FLAGWORD
;;----------- -------------------------------------------------------------

7664  21400 FLAGX: LDA     0,0,3     ;CHANGE OR CHECK A FLAG
7665  24064        LDA     1,C377
7666 107400        AND     0,1       ;FLAG DISPLACEMENT
7667 133000        ADD     1,2
7670  6027/        INTDS
7671  25000        LDA     1,0,2     ;CURRENT FLAG WORD
7672 101113        SSN     0,0       ;"SET" OR "RESET" ?
7673    407        JMP     FLGX1     ;   NO
7674 103100        ADDL    0,0       ;   YES
7675  21401        LDA     0,1,3
7676 100000        COM     0,0
7677 107402        AND     0,1,SZC   ;"SET" ?
7700 106000        ADC     0,1       ;   YES
7701    407        JMP     FLGX2

7702 103113 FLGX1: ADDL#   0,0,SNC   ;"TOGGLE" ?
7703    406        JMP     FLGX2+1   ;   NO
7704  21401        LDA     0,1,3     ;   YES
7705 107414        AND#    0,1,SZR   ;IS THE BIT SET ?
7706 106401        SUB     0,1,SKP   ;   YES, RESET IT
7707 107000        ADD     0,1       ;   NO, SET IT
7710  45000 FLGX2: STA     1,0,2     ;STORE RESULT BACK
7711  60177        INTEN
7712  21401        LDA     0,1,3
7713 107400        AND     0,1
7714  21400        LDA     0,0,3
7715 103100        ADDL    0,0
7716 101113        SSN     0,0       ;SKIP IF RESULT IS ONE ?
7717    404        JMP     .+4       ;   NO
7720 125015        SNZ     1,1       ;   YES
7721   1402        JMP     2,3
7722   1403        JMP     3,3

7723 103112        ADDL#   0,0,SZC   ;SKIP IF RESULT IS ZERO ?
7724 125014        SKZ     1,1       ;   YES
7725   1402        JMP     2,3
7726   1403        JMP     3,3
```

```
;                       << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-------- ---------- ---------- -- -- --- ----------
;;        7.4   RELEASE FIXED BUFFERS
;;
;;        PURPOSE:   TO CLEAR THE CONTEXT OF THE FIXED BUFFERS
;;                   WRITING BSA IF BSACF IS SET AND CLEARING
;;                   THE "IN-FIXED-BUFFER" FLAG IN THE POOL ENTRY
;;
;;        CALLING SEQUENCE:    CALL
;;                             XFIXBUFFERS
;;
;;        ENTRY AND EXIT CONDITIONS NOT MEANINGFULL
;;
;;---------- - --- ----- ----------------------------------------

7727       0          0
7730   547// RLSFB:  STA     3,.-1
7731    6424          JSR     @.CBSA    ;CLEAR BSA IF CHANGED
7732   30420          LDA     2,L.BSA
7733   20024          LDA     0,C4
7734   40417          STA     0,RLSFC
7735   35000 RLSF1:   LDA     3,0,2
7736  175415          INC#    3,3,SNR   ;BUFFER USED ?
7737     407          JMP     RLSF2     ;   NO GO TO NEXT
7740   21402          LDA     0,P.FL,3  ;   YES
7741   24413          LDA     1,RLSFM
7742  107400          AND     0,1
7743   45402          STA     1,P.FL,3  ;CLEAR FIXED BUFFER FLAG
7744  102000          ADC     0,0
7745   41000          STA     0,0,2     ;RESET (TO -1) THE POOL ENTRY POINTER
7746  151400 RLSF2:   INC     2,2
7747   14404          DSZ     RLSFC     ;RELEASE(D) ALL 4 BUFFERS ?
7750     765          JMP     RLSF1     ;   NO, DO NEXT BUFFER
7751    2756          JMP     @RLSFB-1  ;   YES, EXIT

7752    3417 L.BSA:   P.BSA
7753       0 RLSFC:   0                 ;ITERATION COUNTER
7754 1677/7 RLSFM:   -1-INFIXED        ;MASK TO CLEAR FIXED BUFFER FLAG
7755    3431 .CBSA:   CBSAX
```

```
;                        << SI = R91REXGSUBSA;  BO = 1/A.REXM5.7033! >>

;;-------  ------------------------------------------------------------
;;      7.5 RELEASE SPECIFIED FIXED BUFFER
;;
;;           ENTRY:   (AO) <-- # OF BUFFER TO BE RELEASED
;;                    CALL
;;                    SFIXBUFFER
;;
;;            EXIT:   SPECIFIED BUFFER HAS BEEN RELEASED
;;
;;          PURPOSE:  TO RELEASE ONLY ONE FIXED FUNCTION BUFFER
;;                    AT A TIME, IE.  TO DISASSOCIATE IT WITH THE
;;                    DISC BLOCK IT PREVIOUSLY CONTAINED.
;;
;;--  --------  ---------  --------  ------  ------------------

     7756   54751 RSFBX: STA     3,RLSFB-1;RELEASE A SELECTED FIXED BUFFER
     7757  101015        SNZ     0,0       ;RELEASING BSA ?
     7760    67/5        JSR     @.CBSA    ;   YES, CLEAR IT IF IT'S DIRTY
     7761   307/1        LDA     2,L.BSA
     7762  113000        ADD     0,2       ;COMPUTE OFFSET TO CORRECT AREA TO RELEASE
     7763  102520        SUBZL   0,0
     7764   40767        STA     0,RLSFC   ;PRESET THE ITERATION COUNT
     7765     750        JMP     RLSF1     ;COMPLETE THE RELEASE (RETURN TO CALLER FORM RLSF2)
```

```
;                        << SI = R91REXGSUBSA; BO = 1/A.REXM5.7033! >>

;;-  ------- ---------------- -------- ---------- ------------
;;      7.6 RELATIVE JSR                                  <<REENTRANT>>
;;
;;            PURPOSE:  TO PROVIDE A MECHANISM WHEREBY A DISCSUB OR DRIVER
;;                      CAN JUMP TO ANY ADDRESS IN CORE WITHOUT
;;                      NEEDING AN ABSOLUTE ADDRESS OR USING
;;                      MORE THAN TWO WORDS.
;;
;;    CALLING SEQUENCE:
;;          RELJMPRET         ;EXECUTION RESUMES AT
;;          XXXX-.            ;  XXXX WITH AC3=YYYY
;;    YYYY:
;;
;;    AC0, AC1 & AC2 ARE UNCHANGED; C is toggled if displ. < 0
;;
;;---------- --------------------.------------ ---------------

 7766       0 DEST:  0                    ;DESTINATION ADDRESS
 7767       0 R0:    0                    ;AC0

 7770  6347/ XRJSR: SKPBN   CPU           ;INTERRUPTS ENABLED ?
 7771    412        JMP     XRJIH        ;   NO
 7772  60277        INTDS                ;   YES, DISABLE THEM
 7773  40774        STA     0,R0         ;SAVE AC0
 7774  21400        LDA     0,0,3        ;LOAD DISPLACEMENT TO DESTINATION
 7775 163000        ADD     3,0          ;COMPUTE DESTINATION
 7776  40770        STA     0,DEST
 7777  20770        LDA     0,R0         ;RESTORE AC0
10000 175400        INC     3,3          ;CORRECT RETURN
10001  60177        INTEN                ;RE-ENABLE INTERUPTS
10002   2764        JMP     @DEST        ;JUMP TO DESTINATION

10003  40764 XRJIH: STA     0,R0         ;SAVE AC0
10004  21400        LDA     0,0,3        ;LOAD DISPLACEMENT TO DESTINATION
10005 163000        ADD     3,0          ;COMPUTE DESTINATION
10006  40760        STA     0,DEST
10007  20760        LDA     0,R0         ;RESTORE AC0
10010 175400        INC     3,3          ;CORRECT RETURN
10011   2755        JMP     @DEST



        .EOT  ;REX "GSUB" FOR "IRIS" RESIDENT EXECUTIVE
```

```
;                              << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>
;  29 OCT 86,  RB.

;;-------------------------------- -- ------ -- ----------------------
;;
;;                  P R O C E S S     C H A R A C T E R S
;;
;;
;;       Major Components:
;;
;;             QCHARACTER
;;             IIB ACCESS ROUTINES
;;             IOB ACCESS ROUTINES
;;             PC (THE MOST MAJOR COMPONENT)
;;             SIGNAL TASK
;;
;;-------- --- ------------------------------------------------------

;  15 NOV 85,  RB.:  MADE IIB GET/PUT BYTE STANDARD ROUTINES
;   1 MAR 86,  RB.:  IIB'S IN XMEM
;   5 APR 86,  RB.:  IOB'S IN XMEM
;   9 APR 86,  RB.:  PSEUDO-INTERRUPT IN QCHARACTER
;  13 MAY 86,  RB.:  CORRECT IOB HANDLING, ADD TWM PROVISIONS FOR DETACHED PORTS...
;  30 MAY 86,  RB.:  ADD CIRCULAR I/O ROUTINES
```

```
;                       << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

; FIXED NEGATIVE OFFSETS FROM QCHARACTER ENTRY POINT:
; NOTE:  THE ROUTINE POINTERS MAY GET OVERLAID BY AN EXTENDED MEMORY DRIVER

  10012  10233          PIOBSAVE          ;-40; SAVE REGNANT IOB IN XMEM
  10013  10233          PIOBREST          ;-37; RESTORE REGNANT IOB FROM XMEM
  10014  10234          PCBINIT           ;-36; INITIALIZE CIRCULAR BUFFER
  10015  10241          PCBACCESS         ;-35; PREPARE XMEM CIRCULAR BUFFER FOR CORE ACCESS
  10016  10250          PCBAPPEND         ;-34; APPEND LATEST CORE CONTENT TO XMEM CIRCULAR BUFFER
  10017  10255          PCBLINK           ;-33; LINK IN NEXT CIRCULAR IOB SEGMENT IN XMEM
  10020  10220          PIBCNT            ;-32; RETURN # BYTES CURRENTLY IN INPUT BUFFER
  10021  10224          PIBINIT           ;-31; INIT INPUT BUFFER
  10022  10224          PIBRESET          ;-30; RESET INPUT BUFFER
  10023  10224          PIBDONE           ;-27; PROCESS INPUT COMPLETION
  10024  10231          POBINIT           ;-26; INIT OUTPUT BUFFER
  10025  10233          PIBLINK           ;-25; LINK IN NEXT INPUT BUFFER SEGMENT, IF ANY
  10026  10233          PIBUNLINK         ;-24; UNLINK CURRENT INPUT BUFFER SEGMENT, IF POSSIBLE
  10027  10233          POBLINK           ;-23; LINK IN NEXT OUTPUT BUFFER SEGMENT, IF ANY
  10030   6745          PIBPUTBYTE        ;-22; PUT BYTE INTO INPUT BUFFER
  10031   6673          PIBGETBYTE        ;-21; GET BYTE FROM INPUT BUFFER
  10032   6673          POBGETBYTE        ;-20; GET BYTE FROM OUTPUT BUFFER
  10033  10233          PIBCOPY           ;-17; COPY XMEM INPUT BUFFER (IF ANY) TO REGNANT IOB
  10034  10227          POBCOPY           ;-16; COPY REGNANT IOB TO XMEM OUTPUT BUFFER (IF ANY)

  10035  10170 .IIBP:   PIIBPUTBYTE       ;-15; PUT BYTE INTO IIB AT GIVEN BYTE POINTER
  10036  10201 .IIBG:   PIIBGETBYTE       ;-14; GET A BYTE FROM IIB AT GIVEN BYTE POINTER
  10037  10155          PIIBINBYTE        ;-13; INPUT NEXT BYTE FROM IIB
  10040  10130          PIIBROUTBYTE      ;-12; OUTPUT BYTE TO IIB AT REVERSE END
  10041  10104          PIIBOUTBYTE       ;-11; OUTPUT BYTE TO IIB, 237 IF FULL

  10042      0                 0          ;-10; START OF SHARED PERIPHERAL IOB, IF XMEM, ELSE 0
  10043      0                 0          ; -7; FBA OF CORE IOB IF IOB'S ARE IN XMEM, ELSE 0

  10044      0 .CHQ:           0          ; -6; POINTER TO CHARACTER QUEUE      \  SET
  10045      0 .CHQE:          0          ; -5; POINTER TO END OF CHAR. QUEUE    ) UP
  10046      0 CHQIP:          0          ; -4; CHARACTER QUEUE INPUT POINTER    ) BY
  10047      0 CHQOP:          0          ; -3; CHARACTER QUEUE OUTPUT POINTER /  SIR

  10050      0 QCOVCOUNT:      0          ; -2; CHARACTER QUEUE OVERFLOW COUNT
  10051  11530                 IHTTBL     ; -1; TABLE OF INPUT HANDLER TABLES

                ; ***** NO CODE HERE *****
```

```
;                       << SI = R9OREXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;- --------  --------  ---------  ----------------  ---------------
;;
;;     QUEUE CHARACTER
;;
;;       PURPOSE:   QUEUE INTERRUPT TASK REQUESTS, INPUT CHARACTERS, AND OUTPUT
;;                  CHARACTER REQUESTS FOR PROCESSING BY PC.
;;
;;       ENTRY:     THIS IS A GLOBAL SUBROUTINE ACCESSIBLE TO ANY MODULE
;;                  BY EXECUTING A QCHARACTER INSTRUCTION WITH A CHAR OR
;;                  TASK IDENTIFER IN AC0 AND A PCB POINTER OR TASK PARAM
;;                  IN AC2.  A0 = CHAR (MAY HAVE MSB SET)
;;
;;       EXIT:      TO CALLER, NON-SKIP, WITH A2 PRESERVED
;;
;;       NOTE:      ENABLES INTERRUPTS
;;                  SEE PC "DESCRIPTION OF CHARACTER QUEUE"
;;
;;- ----------------------------------------------------------------

                    ; ***** NO CODE HERE *****

10052  60277 QCHAX: INTDS                ;QUEUE A CHARACTER FOR PROCESSING
10053  427/3        STA     0,@CHQIP
10054 145000        MOV     2,1
10055  30771        LDA     2,CHQIP     ;IDENTIFY PORT CHAR CAME FROM
10056 151400        INC     2,2
10057  45000        STA     1,0,2       ; CHARACTER IN IIB
10060 151400        INC     2,2
10061  20764        LDA     0,.CHQE
10062 142033        SLS     2,0         ;END OF QUEUE ?
10063  30761        LDA     2,.CHQ      ;   YES, (IT'S CIRCULAR)
10064  20763        LDA     0,CHQOP
10065 142415        SNE     2,0         ;IS QUEUE FULL ?
10066    410        JMP     QCHOV       ;   YES
10067  50757        STA     2,CHQIP     ;   NO
10070 131000 QCHA2: MOV     1,2
10071  26411        LDA     1,@XIRUPT
10072 125415        INC#    1,1,SNR     ;CALLED FROM BELOW PC ?
10073   2410        JMP     @XPSINTRPT;   YES, GIVE PSEUDO-INTERRUPT & RETURN
10074  6017/        INTEN               ;NO,  RETURN DIRECTLY
10075   1400        JMP     0,3


10076  10752 QCHOV: ISZ     QCOVCOUNT;CHARACTER QUEUE OVERFLOW !?
10077    7/1        JMP     QCHA2
10100  14750        DSZ     QCOVCOUNT;LIMIT AT 177777
10101    767        JMP     QCHA2

10102    665 XIRUPT:        INFO+IRUPT
10103  12137 XPSINTRPT:    PSINTRPT
```

```
;                        << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;--------- ---------------- ------------------------------------------
;;     DESCRIPTION OF INTERMEDIATE INPUT BUFFER (IIB)
;;
;;       TEMPORARY STORAGE AREA FOR INPUT CHARACTERS WHEN
;;       ANY OF THE FOLLOWING CONDITIONS EXIST:
;;
;;         1.  OUTPUT ENABLED
;;         2.  INPUT NOT ENABLED
;;         3.  TOB NOT EMPTY
;;         4.  EXPECTING CURSOR
;;         5.  IIB NOT EMPTY
;;
;;       ALSO RECEIVES BASIC'S CHAIN STATEMENT STRING,
;;       AS WELL AS SYSCO'S (CALL 98)
;;
;;
;;       CIRCULAR (POINTERS ARE IN PCB EXTENDER)
;;                     FBA --> FIRST BYTE OF BUFFER - 1
;;                     LBA --> LAST BYTE OF BUFFER
;;                     IBP --> LAST BYTE PUT IN BY IIBOUTBYTE OR IIBROUTBYTE
;;                     OBP --> LAST BYTE TAKEN OUT BY IIBINBYTE
;;
;;                     IBP = OBP        MEANS BUFFER IS EMPTY
;;                     IBP = OBP - 1    MEANS BUFFER IS FULL
;;                     (IE.  MAX.  DATA THAT IIB CAN HOLD = IIB SIZE - 1)
;;
;;                     AT IPL TIME, SIR SETS IBP = OBP = LBA.
;;                     THEY ARE NEVER LEFT AT FBA, BUT MAY BE LEFT AT LBA.
;;
;;       IBP AND OBP MAY BE LOOKED AT BY $MMUX, ETC., TO TELL IF IIB
;;       IS EMPTY, OR FULL, OR HOW MUCH ROOM IS LEFT -- BUT THEY MAY NOT
;;       BE MODIFIED BY ANYONE EXCEPT IIBINBYTE, IIBOUTBYTE, IIBROUTBYTE
;;       (OR PCESC/PCCC TO CLEAR IIB)
;;
;;--- ------ ---------- ------------- -- --------------------
```

```
;                       << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;- -------- - ------------ ------------ - ------ ----------------
;;
;;          OUTPUT CHAR TO IIB              ***** RE-ENTRANT *****
;;
;;          PURPOSE:  ALLOWS A STANDARD MECHANISM FOR PUTTING A BYTE
;;                    INTO THE NEXT AVAILABLE CELL IN IIB
;;
;;          CALLING SEQUENCE:
;;                    (A2) = PCB
;;                    (A0) = BYTE TO BE STORED
;;                    IOCALL
;;                    IIBOUTBYTE
;;
;;          EXIT:     A2 PRESERVED
;;                    (A0) PUT INTO NEXT IIB CELL UNLESS IIB IS FULL,  IN WHICH CASE
;;                         A 237 IS PUT INTO IIB (ECHOES A BELL)
;;
;;          NOTE:     SEE PC "DESCRIPTION OF IIB"
;;
;;--- -------- - ---------------- -------- ----------------------------

        10104 PIIBOUTBYTE:
10104   60277         INTDS
10105   54421         STA     3,PIIA3
10106   35017         LDA     3,PCX.,2 ;SAVE CHAR.  IN IIB  << FROM PCIN2
10107   25404         LDA     1,IBP.,3
10110   31403         LDA     2,LBA.,3
10111  132033         SLS     1,2      ;END OF CIRCULAR BUFFER ?
10112   25402         LDA     1,FBA.,3 ;   YES, START OVER
10113  125400         INC     1,1
10114   31405         LDA     2,OBP.,3
10115  132414         SEQ     1,2      ;BUFFER FULL ?
10116     403         JMP     PISTA    ;   NO
10117   25404         LDA     1,IBP.,3 ;   YES, STAY AT LAST BYTE
10120   20407 PIIPB:  LDA     0,C237   ;ENTER "RING BELL" CODE
10121   45404 PISTA:  STA     1,IBP.,3
10122   31400         LDA     2,PCB.,3 ;GET A(PCB)
10123    6712         JSR     @.IIBPUTBYTE;PUT THE BYTE INTO IIB
10124   60177 PIIAX:  INTEN
10125    2401         JMP     @PIIA3


10126       0 PIIA3:          0
10127     237 C237:           237
```

```
;                        << SI = R9OREXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;-----------  ------------------  ----------  ----------  ----------------------
;;          OUTPUT CHAR TO IIB AT REVERSE END       ***** RE-ENTRANT *****
;;
;;          PURPOSE:  ALLOWS A STANDARD MECHANISM FOR PUTTING A BYTE
;;                    INTO IIB AHEAD OF ANY PENDING TYPE-AHEAD
;;
;;          CALLING SEQUENCE:
;;                    (A2) = PCB
;;                    (A0) = BYTE TO BE STORED
;;                    IOCALL
;;                    IIBROUTBYTE
;;
;;          EXIT:     A2 PRESERVED
;;                    (A0) PUT INTO IIB AT NEXT IIBINBYTE POSITION
;;                         IF IIB IS FULL, THE **LAST** TYPE-AHEAD CHARACTER
;;                         IS OVERLAID WITH A 237 (ECHOES A BELL)
;;
;;---  --------  ------  ------------------  ----------  ----------------------

          10130 PIIBROUTBYTE:
  10130   60277          INTDS
  10131   54775          STA     3,PIIA3
  10132   35017          LDA     3,PCX.,2
  10133   25405          LDA     1,OBP.,3
  10134    6701          JSR     @.IIBPUTBYTE;PUT BYTE INTO IIB AT OBP
  10135   35017          LDA     3,PCX.,2
  10136   15405          DSZ     OBP.,3    ;BACK UP OBP
  10137   21405          LDA     0,OBP.,3
  10140   25402          LDA     1,FBA.,3
  10141  106432          SGR     0,1       ;UNDERSHOOT ?
  10142   21403          LDA     0,LBA.,3  ;   YES, WRAP AROUND TO THE TOP
  10143   41405          STA     0,OBP.,3
  10144   25404          LDA     1,IBP.,3
  10145  106414          SEQ     0,1       ;DID WE OVERFILL IIB ?
  10146     756          JMP     PIIAX     ;   NO, RETURN
  10147  124400          NEG     1,1       ;YES, BACK UP IBP
  10150  124000          COM     1,1
  10151   21402          LDA     0,FBA.,3
  10152  122432          SGR     1,0       ;IS IT NOW BELOW BEGINNING OF IIB ?
  10153   25403          LDA     1,LBA.,3  ;   YES, WRAP AROUND TO TOP
  10154     744          JMP     PIIPB     ;PUT A 237 CODE THERE
```

```
;                        << SI = R9OREXPCHRSC; BO = 1/A.REXM5.7033! >>

;;- - ------  ---------------------- -------- ----------------------
;;          INPUT NEXT BYTE FROM IIB              ***** RE-ENTRANT *****
;;
;;          PURPOSE:  ALLOWS A STANDARD MECHANISM FOR GETTING A BYTE
;;                    FROM THE NEXT CELL IN IIB
;;
;;          CALLING SEQUENCE:
;;                    (A2) = PCB
;;                    IOCALL
;;                    IIBINBYTE
;;
;;          EXIT:     A2 PRESERVED
;;                    (A0) = NEXT BYTE FROM IIB
;;
;;          CALLING RULES:
;;                    CALLER MUST MAKE SURE IIB IS NOT EMPTY
;;
;;--   ------  ---------------------- ----- -------- ---------

          10155 PIIBINBYTE:
   10155  60277          INTDS
   10156  54750          STA     3,PIIA3
   10157  35017          LDA     3,PCX.,2
   10160  25405          LDA     1,OBP.,3 ;CURRENT POSITION
   10161  21403          LDA     0,LBA.,3 ;END
   10162 122033          SLS     1,0      ;END OF CIRCULAR BUFFER ?
   10163  25402          LDA     1,FBA.,3 ;   YES, WRAP AROUND
   10164 125400          INC     1,1
   10165  45405          STA     1,OBP.,3
   10166   6650          JSR     @.IIBGETBYTE;GET THE BYTE FROM IIB
   10167    735          JMP     PIIAX    ;ENABLE INTERRUPTS AND RETURN
```

```
;                       << SI = R9OREXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------------
;;
;;          PUT BYTE INTO IIB              ***** NOT RE-ENTRANT *****
;;          GET BYTE FROM IIB
;;
;;          PURPOSE:  THESE ARE THE ONLY TWO ROUTINES ACTUALLY ACCESSING THE IIB
;;                    ITSELF (AS OPPOSED TO ACCESSING IIB POINTERS)
;;
;;          CALLING SEQUENCE:
;;                    (A2) = PCB
;;                    (A1) = IIB BYTE POINTER
;;                    (A0) = BYTE TO BE STORED
;;                    IOCALL
;;                    IIBPUTBYTE
;;
;;                    (A2) = PCB
;;                    (A1) = IIB BYTE POINTER
;;                    IOCALL
;;                    IIBGETBYTE (BYTE IS RETURNED IN A0)
;;
;;          EXIT:     A2 PRESERVED
;;
;;          CALLING RULES:  MUST BE CALLED WITH INTERRUPTS DISABLED
;;                    INTENDED TO BE CALLED ONLY BY IIBINBYTE/OUTBYTE/ROUTBYTE
;;
;;-------------------------------------------------------------------------

          10170 PIIBPUTBYTE:
10170   54423           STA     3,PIIB3
10171   50423           STA     2,PIIB2
10172   31015           LDA     2,ABN.,2 ;GET ABN WORD
10173   34422           LDA     3,VUCIIB ;GET UCIIBFLAG MASK
10174  173414           AND#    3,2,SZR  ;IIB IN UPPER CORE?
10175  131241           MOVOR   1,2,SKP  ;   YES, SET MSB
10176  131220           MOVZR   1,2      ;   NO, RESET MSB
10177    6417           JSR     @.LSTBYTE;ALTERNATE ENTRY TO PUTBYTE
10200     411           JMP     PIIBX

          10201 PIIBGETBYTE:
10201   54412           STA     3,PIIB3
10202   50412           STA     2,PIIB2
10203   35015           LDA     3,ABN.,2 ;GET ABN WORD FROM PCB
10204   30411           LDA     2,VUCIIB ;GET UCIIBFLAG MASK
10205  173404           AND     3,2,SZR  ;IIB IN UPPER CORE ?
10206  152620           SUBZR   2,2      ;   YES, BASE BYTE ADDR IS 100K
10207    6410           JSR     @.LACBYTE;USE BASE BYTE ADDR IN AC2
10210  141000           MOV     2,0
10211   30403 PIIBX:    LDA     2,PIIB2
10212    2401           JMP     @PIIB3

10213       0 PIIB3:    0
10214       0 PIIB2:    0
10215     100 VUCIIB:   UCIIBFLAG
10216    6746 .LSTBYTE: LSTBY    ;ALTERNATE PUTBYTE ENTRY
10217    6646 .LACBYTE: LACBY    ;ALTERNATE ENTRY POINT TO GETBYTE
```

```
;                        << SI = R9OREXPCHRSC; BO = 1/A.REXM5.7033! >>

;;- -------- ---------------------- --------------------------
;;
;;              IOB ACCESS ROUTINES (DEFAULT)
;;
;;- ---------- ---------------------- ------ ---- ------------------

          10220 PIBCNT:                       ;CALC.  # BYTES IN INPUT BUFFER
10220     21004         LDA     0,IBP.,2
10221     25002         LDA     1,FBA.,2
10222    122400         SUB     1,0
10223      1400         JMP     0,3

          10224 PIBINIT:                      ;INITIALIZE INPUT BUFFER
          10224 PIBRESET:                     ;RESET INPUT BUFFER
          10224 PIBDONE:                      ;INPUT IS DONE
10224     21002         LDA     0,FBA.,2
10225     41004         STA     0,IBP.,2
10226      1400         JMP     0,3

          10227 POBCOPY:                      ;COPY REGNANT IOB TO XMEM & RESET POINTERS
10227     21005         LDA     0,OBP.,2
10230     41007         STA     0,LOB.,2
          10231 POBINIT:                      ;INTIALIZE OUTPUT BUFFER
10231     21002         LDA     0,FBA.,2
10232     41005         STA     0,OBP.,2
          10233 PIOBSAVE:                     ;SAVE IOB IN XMEM
          10233 PIOBREST:                     ;RESTORE IOB FROM XMEM
          10233 PIBCOPY:                      ;COPY INPUT BUFFER FROM XMEM TO CORE IOB
          10233 PIBLINK:                      ;LINK IN ANOTHER SEGMENT (CAN'T)
          10233 PIBUNLINK:                    ;UNLINK CURRENT SEGMENT
          10233 POBLINK:                      ;LINK IN ANOTHER SEGMENT
10233      1400         JMP     0,3

           6673 PIBGETBYTE=   ACBY
           6745 PIBPUTBYTE=   STBY
           6673 POBGETBYTE=   ACBY
```

```
;                       << SI = R9OREXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------- ------ ----------------------------------------
;;
;;              CIRCULAR IOB ACCESS ROUTINES (DEFAULT)
;;
;;------- ------- --------- ------------------ ---- ---------------

          10234 PCBINIT:                    ;INITIALIZE CIRCULAR BUFFER
  10234   21003         LDA     0,LBA.,2
  10235   41005         STA     0,OBP.,2
  10236   41007         STA     0,LOB.,2
  10237   41004         STA     0,IBP.,2
  10240     405         JMP     PCBA2

          10241 PCBACCESS:                  ;PREPARE CIRCULAR BUFFER FOR ACCESS
  10241   21005         LDA     0,OBP.,2
  10242   25002         LDA     1,FBA.,2
  10243  106415         SNE     0,1        ;IS OBP AT LEFT END OF CIRCULAR BUFFER ?
  10244   21003         LDA     0,LBA.,2   ;   YES, THEN USE RIGHT END
  10245   41006 PCBA2:  STA     0,LIB.,2   ;SET LIB
  10246   15006         DSZ     LIB.,2     ;   = OBP - 1
  10247    1400         JMP     0,3

          10250 PCBAPPEND:                  ;APPEND LATEST INPUT TO OUTPUT
  10250   21004         LDA     0,IBP.,2
  10251   25007         LDA     1,LOB.,2
  10252  106433         SLE     0,1        ;IS IBP TO THE RIGHT OF LOB ?
  10253   41007         STA     0,LOB.,2   ;   YES, THEN SET LOB = IBP
  10254    1400         JMP     0,3

          10255 PCBLINK:                    ;LINK CIRCULAR OUTPUT BUFFER; IE. WRAP-AROUND
  10255   21007         LDA     0,LOB.,2   ;   (CALLED WITH OBP = LOB)
  10256   25004         LDA     1,IBP.,2
  10257  106415         SNE     0,1        ;IS BUFFER EMPTY ?
  10260    1400         JMP     0,3        ;   YES, NON-SKIP RETURN
  10261   25003         LDA     1,LBA.,2   ;NO
  10262  106414         SEQ     0,1        ;ARE WE AT RIGHT END ?
  10263     403         JMP     PCBL2      ;   NO
  10264   21002         LDA     0,FBA.,2   ;YES, WRAP AROUND
  10265   41005         STA     0,OBP.,2
  10266   25004 PCBL2:  LDA     1,IBP.,2
  10267  106033         SLS     0,1        ;ARE WE TO THE RIGHT OF IBP ?
  10270   25003         LDA     1,LBA.,2   ;   YES, THEN LIMIT IS LBA (ELSE IT'S IBP)
  10271   45007         STA     1,LOB.,2   ;SET LOB
  10272    1401         JMP     1,3        ;SKIP RETURN
```

```
;                        << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------- ------------------------------------------------------
;;
;;    * * * * *  PC  ==  PROCESS CHARACTER QUEUE  * * * * * *
;;
;;   ENTRY:  FROM INTERRUPT SERVICE EXIT (INTRR-1)
;;           TO READ CHARACTER QUEUE NODE (PCSTT)
;;               AC3 <-- CHQOP
;;               INTERRUPTS DISABLED
;;               CHARACTER QUEUE CONTAINS ONE OR MORE NODES FOR PROCESSING
;;
;;   EXIT:   FROM PROCESS CHARACTER QUEUE (PCHAR)
;;           TO INTERRUPT SERVICE EXIT (INTRR)
;;               AC0 <-- 0
;;               INTERRUPTS DISABLED
;;               CHARACTER QUEUE EMPTY
;;
;;     MAIN ROUTINES
;;       1.  (DELETED)
;;       2.  PCSTI - START AUTO INPUT (IF APPROPRIATE)
;;       3.  (DELETED)
;;       4.  PCSI  - START INPUT
;;       5.  MAIN BODY
;;           A. PCHAR     - PROCESS CHAR QUEUE
;;           B. PCSTT     - READ CHAR QUEUE NODE
;;           C. PCSTT+12  - BRANCH ON TASK REQUEST
;;           D. PCSTT+17  - PROCESS INPUT CHAR
;;           E. PCIN1     - CLEAR ^Y FLAG
;;           F. PCIN1+11  - SET IIB FLAG
;;           G. PCIN2     - BRANCH ON CONTROL CODE
;;           H. PCIN-2    - STORE CHAR
;;           I. PCEN-10   - ECHO CHAR
;;           J. PCINC     - GET CHAR FROM IIB AND REPEAT
;;       6.  (DELETED)
;;
;;     TASK AND CONTROL CHARACTER SUBROUTINES
;;       7.  PCCP  - TOGGLE PARITY
;;       8.  PCER  - ERROR
;;       9.  PCCB  - QUEUE SIGNAL TASK
;;      10.  PCCE  - TOGGLE ECHO
;;      11.  PC.CR - CLEAR XON
;;      12.  PCEOM - EOM
;;      13.  PCCH  - BACKSPACE
;;      14.  PCCO  - KILL OUTPUT
;;      15.  PCODN - OUTPUT DONE
;;      16.  PCCS  - PAUSE OUTPUT
;;      17.  PCCQ  - RESUME OUTPUT
;;      18.  PCO   - OUTPUT CHAR REQUEST
;;      19.  PCCX  - CANCEL INPUT LINE
;;      20.  PCTIO - TERMINATE I/O SUBROUTINE
;;      21.  PCTSK - PROCESS INTERRUPT TASK
;;      22.        - DELETED
;;      23.  SEND  - SEND CHAR SUBROUTINE
;;      24.  PCCC  - SYS ESCAPE
;;           PCES  - ESCAPE
;;      25.  PCCY  - BYPASS ERROR BRANCH
```

```
;                        << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------
;;     DESCRIPTION OF CHARACTER QUEUE
;;
;;       CIRCULAR   INPUT POINTER  = CHQIP  \
;;                  OUTPUT POINTER = CHQOP   \  AS DEFINED IN REX
;;                  BEGINNING      = .CHQ    /  QUEUE CHARACTER
;;                  END            = .CHQE  /
;;
;;                  2 WORDS PER CHAR QUEUE NODE
;;                      0 = CHAR!TASK  WORD
;;                      1 = PCB!PARAMETER WORD
;;
;;                                CHAR QUEUE NODE
;;                               /               \
;;                            WORD 0              WORD 1
;;                           /      \            /      \
;;                      INPUT        TASK     PCB        TASK
;;                      CHAR         REQUEST  PNTR       PARM
;;                     /    \
;;                PRINT      CONTROL
;;                          /       \
;;                    IMMEDIATE   DEFERRED
;;
;;     DEFERRED            IMMEDIATE
;;     CONTROL             CONTROL          PRINT       TASK
;;     CODE                CODE             CHAR        REQUEST
;;
;;     201 ^A BS           200 ^@ NULL      240 SPACE   400 INPUT BUFFER FULL
;;     205 ^E ECHO         202 ^B BREAK     241 !       401 OUTPUT BUFFER EMPTY
;;     206 ^F              203 ^C SYS ESC     .         402 START INPUT
;;     207 ^G BELL         204 ^D ESC         .         403 INPUT DONE
;;     210 ^H BS           212 ^J LF          .         404 NOT USED
;;     211 ^I H-TAB        215 ^M CR        375 }
;;     213 ^K V-TAB        217 ^O KILL      376 ~       >404 TASK ADDR
;;     214 ^L FF           220 ^P PARITY                < 0  OUT CHAR REQ
;;     216 ^N              221 ^Q XON
;;     222 ^R TAPE         223 ^S XOFF
;;     224 ^T TAPE         233 ^[ ESC
;;     225 ^U
;;     226 ^V
;;     227 ^W
;;     230 ^X CANCEL
;;     231 ^Y ERR BR
;;     232 ^Z
;;     234 ^\
;;     235 ^]
;;     236 ^^
;;     237 ^_ BELL
```

```
;                        << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;- -------- ------------------- ---------- ------- ----------------
;;      2.  START AUTO INPUT (IF APPROPRIATE)
;;
;;          ENTRY:    GET CHAR (PCI1)
;;                    AC2 <-- PCB
;;                    (TOB) = 0
;;                    IIB EMPTY
;;
;;          EXIT:     PROCESS CHAR QUEUE (PCHAR)
;;                    AC'S   NONE
;;
;;                    START INPUT (PCSI)
;;                    AC2 <-- PCB
;;                    INTERRUPTS DISABLED
;;
;;          PURPOSE:  DETERMINE IF AUTO INPUT MAY START
;;----------  ------------------------------ -------- ----------------

10273  6027/ PCSTI: INTDS                ;START AUTO INPUT, IF APPR.   << FROM PCI1
10274  21004        LDA     0,IBP.,2
10275  25006        LDA     1,LIB.,2
10276 106032        SSGE    0,1          ;IS THERE ROOM IN INPUT BUFFER ?
10277    404        JMP     PCST2        ;   YES
10300  34103        IOCALL               ;NOT IN CURRENT SEGMENT
10301   7753        IBLINK               ;IS ANOTHER SEGMENT AVAILABLE ?
10302    456        JMP     PCHA1        ;   NO, DON'T START AUTO
10303  34103 PCST2: IOCALL               ;YES, SEARCH CHAR. QUEUE
10304  257/3        LDA     1,XCHQE,3
10305  35775        LDA     3,XCHQO,3
10306  22415 PCSQ:  LDA     0,@.CQIP
10307 162415        SNE     3,0          ;ANYTHING IN CHARACTER QUEUE ?
10310    414        JMP     PCSI         ;   NO, THEN IT'S OK TO START INPUT
10311  21401        LDA     0,1,3        ;YES, GET PCB POINTER
10312 112415        SNE     0,2          ;FOR THIS PORT ?
10313    445        JMP     PCHA1        ;   YES, WAIT FOR IT
10314 175400        INC     3,3          ;NO
10315 175400        INC     3,3
10316 166032        SGE     3,1          ;END OF CHARACTER QUEUE ?
10317    767        JMP     PCSQ         ;   NO, CONTINUE SCAN OF CHAR. QUEUE
10320  34103        IOCALL
10321  35772        LDA     3,XCHQB,3;YES, WRAP AROUND TO BEGINNING
10322    764        JMP     PCSQ         ;AND CONTINUE SCAN

10323  10046 .CQIP:        CHQIP
```

RB. 24 Dec 87

```
;                        << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;-----------  ------------------  ------------------------------------
;;      4.  START INPUT (PCSI)
;;
;;          ENTRY:    START AUTO INPUT (PCSQ)
;;                    AC0 <-- CHAR
;;                    AC2 <-- PCB
;;                    (TOB) = 0
;;
;;          EXIT:     PROCESS CHAR QUEUE (PCHAR)
;;                    AC'S   NONE
;;
;;          PURPOSE:  START INPUT FROM DEVICE AND SET APPROPRIATE FLAGS
;;- --------  --------------------  -----  ------------------------------

  10324 102120 PCSI:  ADCZL   0,0       ;TELL DRIVER TO START INPUT  << FROM PCSQ
  10325   7032        JSR     @SND.,2
  10326 101113        SSN     0,0       ;DID IT START INPUT ?
  10327    430        JMP     PCHAR     ;   NO, INTERRUPT MUST BE PENDING
  10330  21012        LDA     0,FLW.,2  ;YES
  10331  24024        LDA     1,C4
  10332 107415        AND#    0,1,SNR   ;XON ENABLED ?
  10333    424        JMP     PCHAR     ;   NO
  10334  24067        LDA     1,C1000   ;YES
  10335 107414        AND#    0,1,SZR   ;HAS X-ON BEEN SENT YET ?
  10336    421        JMP     PCHAR     ;   YES
  10337  20406        LDA     0,CXON    ;NO, SEND IT NOW
  10340   7032        JSR     @SND.,2
  10341   6102        FLAGCHANGE
  10342 140012        SET+FLW.
  10343   1000        1000              ;SET X-ON HAS BEEN SENT FLAG
  10344    413        JMP     PCHAR

  10345     21 CXON:          21        ;X-ON = CTRL-Q

;  POINTERS AND PARAMETERS FOR "MAIN BODY" OF PC:

  10346  11174 XPCO:          PCO
  10347  10662 .PCCP:         PCCP
  10350  12161 ..INR:         INTRR     ;INTERRUPT SERVICE EXIT RETURN
  10351  11271 .PTSK:         PCTSK
  10352  10703 .PPER:         PCPER     ;POTENTIAL PARITY ERROR
  10353  10707 .PCER:         PCER      ;ANY ERROR:  ECHO BELL
  10354  10765 .PICR:         PCICR
  10355    204 C204:          204
  10356    233 C233:          233
```

```
;                        << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>
;;-  ----------------------------------------------------------------
;;     5. MAIN BODY
;;        A. PROCESS CHAR QUEUE (PCHAR)
;;        B. READ CHAR QUEUE NODE (PCSTT)
;;        C. BRANCH ON TASK
;;        D. PROCESS INPUT CHAR
;;        E. CLFAR ^Y FLAG
;;        F. SET IIB FLAG
;;        G. BRANCH ON CONTROL CODE
;;        H. STORE CHAR
;;        I. ECHO CHAR
;;        J. GE! NEXT CHAR. FROM IIB, IF ANY
;;
;;      A.  PROCESS CHAR. QUEUE
;;
;;        ENTRY:    AT COMPLETION OF ANY PC PROCESS
;;
;;      PURPOSE:    RETURN TO INTERRUPT SERVICE EXIT WHEN CHAR QUEUE IS EMPTY
;;-------------------------------------------------------------------

  10357  60277 PCHAR: INTDS              ;PROCESS CHARACTER QUEUE
  10360  34103 PCHA1: IOCALL             ;GET POINTER TO CHARACTER QUEUE POINTERS INTO A3
  10361  21774        LDA    0,XCHQI,3;GET Q INPUT POINTER
  10362  31775        LDA    2,XCHQO,3;GET Q OUTPUT POINTER
  10363 112415        SNE    0,2       ;ANY CHARACTERS TO PROCESS ?
  10364   2764        JMP    @..INR    ;  NO, RETURN TO REGNANT TASK

;;-------------------------------------------------------------------
;;      B. READ CHAR QUEUE NODE
;;
;;         ENTRY:    INTERRUPT SERVICE EXIT (INTRR-1)
;;                   PROCESS CHAR QUEUE (PCHAR)
;;                   AC2 <-- CHQOP
;;                   AC3 <-- IOCALL REFERENCE
;;                   INTERRUPTS DISABLED
;;
;;         PURPOSE:  READ CHAR QUEUE NODE
;;-------------------------------------------------------------------

  10365  60177 PCSTT: INTEN              ;PROCESS CHARACTERS FROM QUEUE
  10366 145400        INC    2,1         ;STEP THE QUEUE POINTER
  10367 125400        INC    1,1
  10370  21773        LDA    0,XCHQE,3;GET Q END POINTER
  10371 122033        SLS    1,0         ;END OF CIRCULAR QUEUE ?
  10372  25772        LDA    1,XCHQB,3;   YES, WRAP AROUND TO BEGINNING
  10373  45775        STA    1,XCHQO,3;UPDATE Q OUTPUT POINTER
  10374  21000        LDA    0,0,2       ;CHARACTER TO BE PROCESSED
  10375  31001        LDA    2,1,2       ;FOR WHICH PCB
  10376  52515        STA    2,@XPPCB
```

```
;                        << SI = R9OREXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------- ---------------- ---------- ---------- ---------------
;;      C.  BRANCH ON TASK
;;
;;          PURPOSE:  GO DIRECTLY TO SPECIAL TASK ROUTINE IF TASK REQUESTED
;;------------------------------ ----------------------------------------

   10377  35015        LDA     3,ABN.,2
   10400  24515        LDA     1,VABITM ;INPUT TYPE MASK
   10401 167700        ANDS    3,1
   10402  34512        LDA     3,.IHTTB ;LOAD ADDRESS
   10403 137000        ADD     1,3
   10404  35400        LDA     3,0,3    ;   OF INPUT TRANSLATION TABLE
   10405  54511        STA     3,.IHTB  ;SAVE IHT ADDRESS FOR LATER USE
   10406 101112        SSP     0,0      ;OUTPUT CHARACTER REQUEST ?
   10407   2737        JMP     @XPCO    ;   YES
   10410  24065        LDA     1,C400   ;NO
   10411 106033        SLS     0,1      ;INTERRUPT TASK ?
   10412   2737        JMP     @.PTSK   ;   YES (A1 MUST = 400)

;;-------- ---------- --------- ---------------------------
;;      D.  PROCESS INPUT CHAR
;;
;;          PURPOSE:  PARITY CHECKING AND CHAR TRANSLATION IF REQ'D
;;-------- -- ---------- ----- ---------------------------------

   10413  25012        LDA     1,FLW.,2 ;PROCESS INPUT CHARACTER
   10414 125112        SSP     1,1
   10415     434       JMP     PCN1A    ;BINARY BYTE MODE
   10416  24053        LDA     1,C200
   10417 107415        AND#    0,1,SNR  ;POTENTIAL PARITY ERROR ?
   10420   2732        JMP     @.PPER   ;   YES, ECHO BELL IF SO
   10421  36476        LDA     3,@.ITTT ;NO
   10422 175415        INC#    3,3,SNR  ;IS THERE A $TERMS ?
   10423     406       JMP     PCIN1    ;   NO
   10424    575        JSR     -3,3     ;YES, GO TO $TERMS TO TRANSLATE CHAR.
   10425     404          JMP  PCIN1    ;NORMAL RETURN, PROCESS CHARACTER
   10426    2726          JMP  @.PICR   ;RETURN TO TERMINATE INPUT (AFTER CURSOR READ)
   10427     730          JMP  PCHAR    ;IGNORE
   10430    2723          JMP  @.PCER   ;PARITY ERROR
```

```
;                       << SI = R9OREXPCHRSC; BO = 1/A.REXM5.7033! >>
;;-------------------------------------------------------------
;;       E.  CLFAR ^Y FLAG
;;
;;          PURPOSE:  CLEAR 'LAST CHAR WAS ^Y FLAG' IF APPROPRIATE
;;-------------------------------------------------------------

   10431  25012 PCIN1:LDA     1,FLW.,2 ;   << FROM PC #5D.
   10432  34065       LDA     3,C400
   10433 137415       AND#    1,3,SNR  ;IS CTRL-Y FLAG SET ?
   10434    415       JMP     PCN1A    ;  NO, THEN PROCEED
   10435  34721       LDA     3,C233   ;YES
   10436 116415       SNE     0,3      ;IS CHARACTER = ESC ?
   10437    412       JMP     PCN1A    ;  YES, LEAVE IT SET
   10440  34715       LDA     3,C204
   10441 116415       SNE     0,3      ;NO, CTRL D THEN?
   10442    407       JMP     PCN1A    ;  YES, LEAVE IT
   10443  60277       INTDS            ;NO, THEN CLEAR CTRL-Y FLAG
   10444  25012       LDA     1,FLW.,2
   10445  34453       LDA     3,CC400
   10446 167400       AND     3,1
   10447  45012       STA     1,FLW.,2
   10450  60177       INTEN

;;-------------------------------------------------------------
;;      F.  SET IIB FLAG
;;
;;         PURPOSE:  DETERMINE IF INPUT SHOULD GO INTO IIB.  IF SO SET CARRY.
;;-------------------------------------------------------------

          10451 PCN1A:                 ;NOW SEE IF CHAR. GOES TO IIB  << FROM #5D., PCIN1
   10451 125300       MOVS    1,1      ;C=1 ==> PUT IT IN IIB
   10452 125103       MOVL    1,1,SNC  ;SET C IF OUTPUT ENABLED
   10453 124102       COML    1,1,SZC  ;SET C IF INPUT NOT ENABLED
   10454    415       JMP     PCIN2
   10455  25011       LDA     1,TOB.,2
   10456 124442       NEGO    1,1,SZC  ;SET C IF TOB NOT EMPTY
   10457    412       JMP     PCIN2
   10460  25037       LDA     1,TTN.,2
   10461  34042       LDA     3,C20
   10462 137414       AND#    1,3,SZR
   10463    405       JMP     PCNIB    ;SET C IF EXPECTING CURSOR
   10464  35017       LDA     3,PCX.,2
   10465  25404       LDA     1,IBP.,3
   10466  35405       LDA     3,OBP.,3
   10467 136414       SEQ     1,3      ;IIB EMPTY ?
   10470 101040 PCNIB:MOVO    0,0      ;  NO, SET C TO SAY CHARACTER IS TO GO INTO IIB
```

```
;                       << SI = R9OREXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;--- ------- ----- ---- ------------------------------------------
;;      G.  BRANCH ON CONTROL CODE
;;
;;          PURPOSE:  EXIT TO CONTROL CHAR SUBROUTINE.
;;--- ----- ---------------------------- ------------------------------

10471  25012 PCIN2: LDA     1,FLW.,2 ; << FROM PCN1A, PCI1 WITH CHAR. FROM IIB, C=0
10472  34420        LDA     3,PCBMK  ;CNTRL CODE & BINARY MODE MASK
10473 167414        AND#    3,1,SZR  ;IN BINARY MODE OR ACTIVATING ON CTRL CODES ?
10474    434        JMP     PCINA    ;   YES, ACCEPT AS IS
10475  24064        LDA     1,C377
10476 106415        SNE     0,1      ;RUBOUT CHARACTER ?
10477    422        JMP     PCRUB    ;   YES
10500  24053        LDA     1,C200   ;NO
10501  34056        LDA     3,C240
10502 106033        SLS     0,1      ;200 <= CHAR < 240 ?
10503 116033        SLS     0,3
10504    424        JMP     PCINA    ;   NO, ACCEPT AS IS
10505  34411        LDA     3,.IHTB  ;LOAD ADDRESS OF INPUT TABLE
10506 136460        SUBC    1,3      ;COMPENSATE FOR 200 OFFSET, PRESERVE CARRY
10507 117000        ADD     0,3      ;INDEX INTO BRANCH TABLE
10510  35402        LDA     3,IHTCTL,3
10511    412        JMP     PCTSP

10512 100010 PCBMK:         100010   ;BINARY MODE OR CONTROL CODE ACTIVATE MASK
10513  10722 XPPCB:         PPCB
10514  11530 .IHTTB:        IHTIBL   ;ADDRESS OF TABLE OF INPUT TABLES
10515   7400 VABITM:        ABITM    ;ABN INPUT TYPE MASK
10516      0 .IHTB:         0        ;SAVE ADDRESS OF CURRENT TABLE HERE
10517    671 .ITTT:         INFO+.TTT.
10520 177377 CC400:         -1-400

10521  34775 PCRUB: LDA     3,.IHTB
10522  35401        LDA     3,IHTRUB,3
10523 175113 PCTSP: SSN     3,3      ;CTRL. CHAR. REQ. IMMED. RESPONSE ?
10524   1400        JMP     0,3      ;   YES
10525 175100        MOVL    3,3      ;NO
10526 175223        MOVZR   3,3,SNC  ;CTRL. CHAR. BEING "ECHOED" ?
10527   1400        JMP     0,3      ;   YES, GO TO CONTROL PROCEDURE

                    ;***NO CODE HERE***
```

```
;                        << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;---  -------  ---------  -------------  ------------------------
;;       H.  STORE CHAR
;;
;;           PURPOSE:  STORE CHAR IN APPROPRIATE BUFFER
;;---  -------  ---------------------------------  ------------------------

10530 101013 PCINA: MOV#    0,0,SNC   ;CHAR. GOING INTO IIB ?  << FROM PCIN2
10531    404        JMP     PCIN      ;  NO
10532  34103        IOCALL            ;YES, USE STANDARD ROUTINE TO OUTPUT BYTE TO IIB
10533   7767        IIBOUTBYTE
10534    623        JMP     PCHAR

10535 105001 PCIN:  MOV     0,1,SKP   ; << FROM BRANCH TABLE, PCCH
10536  24056        LDA     1,C240    ; << FROM BRANCH TABLE TO ECHO SPACE
10537  44456 PCIEC: STA     1,PCTS    ;SAVE CHARACTER TO BE ECHOED  << FROM PCCE
10540  34056        LDA     3,C240
10541  25012        LDA     1,FLW.,2
10542 116032        SGE     0,3       ;PRINTABLE CHARACTER
10543 125112        SSP     1,1       ;  OR BINARY MODE ?
10544    416        JMP     PCACC     ;    YES, ACCEPT CHARACTER AS IS
10545  34052        LDA     3,C177    ;NO
10546 116432        SGR     0,3       ;IS IT AN ASCII CONTROL CHARACTER ?
10547    403        JMP     PCCKF     ;  NO, CHECK IF FUNCTION KEY CODE
10550  34030        LDA     3,C10     ;YES
10551    407        JMP     PCCKK     ;CHECK IF MODE = ACTIVATE ON CONTROL CHARACTERS

10552  24442 PCCKF: LDA     1,C140
10553 106033        SLS     0,1       ;IS THE CHARACTER A FUNC CODE ?
10554 116033        SLS     0,3       ;  (BETWEEN 140 AND 176?)
10555    405        JMP     PCACC     ;  NO, ACCEPT AS IS
10556  34024        LDA     3,C4      ;YES, CHECK IF MODE = FUNCTION KEYS ENABLED
10557  25015        LDA     1,ABN.,2
10560 137404 PCCKK: AND     1,3,SZR   ;IS MODE = WHAT WE ARE LOOKING FOR ?
10561   2432        JMP     @.PILC    ;  YES, ACCEPT AS LAST CHARACTER
10562  25004 PCACC: LDA     1,IBP.,2  ;NO, ACCEPT CHARACTER INTO INPUT BUFFER
10563  35006        LDA     3,LIB.,2
10564 136033        SSLS    1,3       ;IS THERE ROOM FOR IT ?
10565    522        JMP     PCER      ;  NO, ECHO A BEEP
10566 125400        INC     1,1       ;YES
10567  45004        STA     1,IBP.,2
10570  34103        IOCALL
10571   7756        IBPUTBYTE         ;STORE CHARACTER IN BUFFER
10572  32721        LDA     2,@XPPCB
10573  20422        LDA     0,PCTS    ;RETRIEVE CHARACTER TO BE ECHOED
```

RB. 24 Dec 87

```
;                       << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;-------- ---------- ---------------- --- ----- ---------------------------
;;      I.  ECHO CHAR
;;
;;          PURPOSE:  ECHO CHAR USING THE PORT'S DEVICE DRIVER SEND ROUTINE.
;;                    TRANSLATE CHAR BEFORE ECHO IF REQ'D.
;;---------- ---------- ---------------- -- ------------------------------

    10574  25012 PCEN:  LDA     1,FLW.,2 ;       << FROM PCCH, IHTTBL
    10575 125213         SKO     1,1      ;IS ECHO ENABLED ?
    10576    423         JMP     PCNC1    ;   NO
    10577  30523 PCEKO: LDA     2,PPCB   ;YES  << FROM PCCP
    10600  34056         LDA     3,C240
    10601 116033         SLS     0,3      ;PRINTING CHARACTER ?
    10602    414         JMP     PCNCA    ;   YES
    10603 101015         SNZ     0,0      ;NO, NULL BYTE ?
    10604    415         JMP     PCNC1    ;   YES, NOTHING TO ECHO
    10605  36712 PCEK4: LDA     3,@.ITTT ;NO  << FROM PCER
    10606 175415         INC#    3,3,SNR  ;IS THERE A $TERMS ?
    10607    410         JMP     PCNCO    ;   NO, ECHO AS IS
    10610   5776         JSR     -2,3     ;YES, TRANSLATE FOR ECHO
    10611   2541           JMP   @.SEND   ;   ACCESS TO "SEND" FOR $TERMS
    10612    405         JMP     PCNCO    ;NORMAL $TERMS RETURN

    10613  10755 .PILC: PCILC
    10614    140 C140:  140
    10615      0 PCTS:  0

    10616  11033 PCNCA: ISZ     OCC.,2   ;UPDATE OUTPUT COLUMN COUNT & ECHO << FROM PCEKO
    10617   6533 PCNCO: JSR     @.SEND   ;ECHO THE CHAR.

;  CHECK IF OK TO CONTINUE INPUT (OR TO BEGIN)

    10620  25012 PCINC: LDA     1,FLW.,2 ;   << FROM PCCB,PCCE,PCIBF,IHTTBL
    10621  34051 PCNC1: LDA     3,C100   ;   << FROM PCEN, PCEKO, PCO2
    10622 137524         ANDZL   1,3,SZR  ;IS INPUT ENABLED,
    10623 137414         AND#    1,3,SZR  ;AND OUTPUT DISABLED ?
    10624   2527         JMP     @.PCHAR  ;   NO
                                          ;YES

                         ; ***** NO CODE HERE *****
```

```
;                        << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------
;;      J.  GET NEXT CHAR
;;
;;        PURPOSE:   DETERMINE IF INPUT IS TO CONTINUE;
;;                   IF SO, RETRIEVE CHARS FROM IIB, IF ANY.
;;-------------------------------------------------------------------

   10625   21004 PCGNX: LDA     0,IBP.,2 ;   << FROM PCSIN
   10626   25006        LDA     1,LIB.,2
   10627  106032        SSGE    0,1      ;BUFFER SEGMENT FULL ?
   10630     411        JMP     PCI1     ;   NO
   10631   34103        IOCALL           ;YES
   10632    7753        IBLINK           ;IS INPUT BUFFER FULL ?
   10633     402        SKIP
   10634     405        JMP     PCI1     ;   NO, CONTINUE
   10635   21015        LDA     0,ABN.,2 ;YES
   10636   24042        LDA     1,C20
   10637  107415        AND#    0,1,SNR  ;FIXED LENGTH FIELD MODE?
   10640     525        JMP     PCICR    ;   NO, STUFF A <CR>
   10641   21011 PCI1:  LDA     0,TOB.,2 ;YES
   10642  101014        SKZ     0,0      ;OK TO ECHO SOME MORE ?
   10643    2510        JMP     @.PCHAR  ;   NO
   10644   25037        LDA     1,TTN.,2 ;YES
   10645   34042        LDA     3,C20
   10646  137434        ANDZ#   1,3,SZR  ;EXPECTING CURSOR ?
   10647    2504        JMP     @.PCHAR  ;   YES, DON'T START INPUT
   10650   35017        LDA     3,PCX.,2 ;NO
   10651   25405        LDA     1,OBP.,3
   10652   21404        LDA     0,IBP.,3
   10653  122415        SNE     1,0      ;ANYTHING IN IIB ?
   10654    2405        JMP     @.PSTI   ;   NO, (RE)START INPUT
   10655   34103        IOCALL           ;YES, GET BYTE FROM IIB
   10656    7765        IIBINBYTE
   10657  101020        MOVZ    0,0      ;C=0 ==> CHAR. IS FROM IIB
   10660     611        JMP     PCIN2

   10661   10273 .PSTI:         PCSTI
```

RB. 24 Dec 87

```
;                        << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;--- ------- ---------------- --- - --- ---- -------- ---------
;;     7.  TOGGLE PARITY
;;
;;         PURPOSE:  DISABLE/ENABLE PARITY CHECKING
;;
;;--- ------ - ------- - ----- --- ------- --------------------

   10662   6102 PCCP:  FLAGCHECK          ;TOGGLE PARITY CHECK (CTRL P)  << FROM PCPER, IHTTBL
   10663  10012        SKIPZ+FLW.         ;IS OUTPUT ACTIVE ?
   10664    200        200
   10665   2466        JMP     @.PCHAR    ;   YES, IGNORE
          10666        M3L20=  .          ;---------- THE FOLLOWING IS MARK 5 CODE:
   10666   6102        FLAGCHANGE         ;MK5:NO
   10667  40024        TOGGLE+PCW.-FLW.   ;MK5:TOGGLE PARITY INHIBIT BIT
   10670    200        200                ;MK5:
   10671   6102        FLAGCHANGE         ;MK5:
   10672  40000        TOGGLE             ;MK5:AND CHARACTER LENGTH (7 VS. 8 BITS)
   10673     20        20                 ;MK5:
   10674  30426        LDA     2,PPCB     ;MK5:
   10675  20405        LDA     0,VMXPW    ;MK5:
   10676   7032        JSR     @SND.,2    ;MK5:SEND PORT CONTROL WORD TO MUX
          10677        M5L20=  .          ;---------- THE FOLLOWING IS MARK 3 CODE:
          1067/        .LOC M3L20-.*MK3+.
   10677  30423        LDA     2,PPCB     ;MK3:
   10700  21036        LDA     0,PCW.,2   ;MK3:
   10701  24402        LDA     1,C34      ;MK3:
   10702 114000        COM     0,3        ;MK3:
   10703 137400        AND     1,3        ;MK3:EXTRACT LENGTH & PARITY FLAGS
   10704 124000        COM     1,1        ;MK3:
   10705 123400        AND     1,0        ;MK3:
   10706 163000        ADD     3,0        ;MK3:EFFECT COMPLEMENT OF THREE FLAG BITS
   10707  41036        STA     0,PCW.,2   ;MK3:
          1067/        .LOC M5L20-.*MK5+. ;---------- END OF HARDWARE-DEPENDENT CODE
   10677  20402        LDA     0,.+2      ;ECHO A "%" SYMBOL
   10700    67/        JMP     PCEKO
   10701    245        200!"%

   10702 177774 VMXPW: MXPW               ;---------- USED IN MARK 5 ONLY
          10703        .LOC    .-MK3
   10703     34 C34:   34                 ;---------- USED IN MARK 3 ONLY
          10703        .LOC    .-MK5
```

```
;                        << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;--- ------- ------- ------- ------------------------------------
;;     8.  ERROR
;;
;;         PURPOSE:  ENTERED ON BUFFER OVERFLOW OR PARITY ERROR
;;--- ------- ------- ---------- ---------- ---------- ----------

   10703  34613 PCPER: LDA     3,.IHTBL ;POTENTIAL PARITY ERROR  << FROM #5D.
   10704  35400        LDA     3,IHTPAR,3
   10705 116415        SNE     0,3      ;IS IT THE PARITY TOOGLE CHARACTER ?
   10706    754        JMP     PCCP     ;   YES, TOGGLE PARITY
   10707  20027 PCER:  LDA     0,C7     ;NO, ERROR:  ECHO A BELL  << FROM PCACC, IHTTBL
   10710    675        JMP     PCEK4



;;--- ------- ------- ---------- ---------- ---------- ----------
;;     9.  BREAK:
;;         QUEUE SIGNAL TASK
;;
;;         PURPOSE:  TERMINATE PAUSE (IF ANY) AND ALLOW PORT TO REACTIVATE
;;--- ------- ------- ---------- ---------- ---------- ----------

   10711  21015 PCCB:  LDA     0,ABN.,2 ;   << FROM IHTTBL
   10712  24576        LDA     1,VPRTDT
   10713 107414        AND#    0,1,SZR  ;DETACHED PORT ?
   10714    704        JMP     PCINC    ;   YES, IGNORE BREAK
   10715 141000        MOV     2,0      ;NO, SEND SIGNAL (BREAK OR CTRL B)
   10716 126400        SUB     1,1
   10717 152400        SUB     2,2
   10720   6104        QUEUE
   10721  11717        SSIGTASK
   10722      0 PPCB:  0                ;PCB POINTER
   10723  30000        QP.SI             ;PRIORITY OF SIGNAL TASK
   10724      0        0                 ;A3 IGNORED
   10725  30775 JPINC: LDA     2,PPCB
   10726    672        JMP     PCINC
```

```
;                        << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;--- ------- ------- ---------- ----- -------------------------
;;     10.  TOGCLE ECHO
;;
;;          PURPOSE:  DISABLE/ENABLE ECHO
;;--- ---- -- ---------------------- ------ ----------------------

   10727  40425 PCCE:  STA     0,PCTS2   ;SAVE CHARACTER  << FROM IHTTBL
   10730   6102         FLAGCHANGE
   10731  40012         TOGGLE+FLW.       ;TOGGLE ECHO BIT
   10732      1         1
   10733  30767         LDA     2,PPCB
   10734  21012         LDA     0,FLW.,2  ;PICK UP FLW.
   10735  24002         LDA     1,C2
   10736 107404         AND     0,1,SZR   ;TRANSPARENT CTRL-E MODE ?
   10737    661         JMP     PCINC     ;   YES, THROW AWAY
   10740 101212         SKE     0,0       ;NO, IS ECHO NOW ENABLED ?
   10741    404         JMP     PCCE2     ;   YES, DON'T ECHO ANYTHING
   10742  20407         LDA     0,COLON   ;   NO, ECHO A COLON
   10743  11033         ISZ     OCC.,2    ;UPDATE COLUMN COUNTER
   10744   6406         JSR     @.SEND    ;AND ECHO A COLON ":"
   10745  20407 PCCE2: LDA     0,PCTS2   ;RETURN INPUT CHARACTER FOR IOB
   10746 126400         SUB     1,1       ;BUT DON'T ECHO ANYTHING MORE
   10747   2401         JMP     @.+1
   10750  10537         PCIEC

   10751    272 COLON:          200!":
   10752  11314 .SEND:         SEND
   10753  10357 .PCHAR:        PCHAR
   10754      0 PCTS2:         0
```

```
;                    << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------- ------------------- ------ ----------------------
;;     12. EOM
;;
;;         PURPOSE: TERMINATE INPUT
;;-- ------- ------------------- - -- ----- ----- -----------

10755  11004 PCILC: ISZ     IBP.,2     ;INPUT LAST CHARACTER << FROM PCCKK, IHTTBL
10756  25004        LDA     1,IBP.,2
10757  34103        IOCALL
10760   7756        IBPUTBYTE          ;MAY GO TO 1ST GUARD BYTE
10761  30741        LDA     2,PPCB
10762  25012 PCIDN: LDA     1,FLW.,2   ;INPUT IS DONE, TERMINATE << FROM PCTSK, PCIBF
10763 125112        SSP     1,1        ;BINARY MODE ?
10764    431        JMP     PCEOM      ;   YES
10765  25004 PCICR: LDA     1,IBP.,2   ;NO, STUFF A <CR>  << FROM #5D., PCINC, IHTTBL
10766  35006        LDA     3,LIB.,2
10767 136032        SSGE    1,3        ;IS THERE ROOM FOR IT ?
10770    405        JMP     PCIC2      ;   YES
10771  34103        IOCALL             ;NO
10772   7753        IBLINK             ;TRY TO LINK IN ANOTHER IBUFF SEGMENT
10773 100010        NOP                ;   END OF IBUFF --STUFF CR INTO SAME SEGMENTD
10774  25004        LDA     1,IBP.,2
10775 125400 PCIC2: INC     1,1        ;MAY GO TO 2ND GUARD BYTE !
10776  45004        STA     1,IBP.,2
10777  20055        LDA     0,C215
11000  34103        IOCALL
11001   7756        IBPUTBYTE          ;PUT <CR> INTO IBUFF
11002  30720        LDA     2,PPCB
11003  25012        LDA     1,FLW.,2
11004  34430        LDA     3,C1004
11005 137415        AND#    1,3,SNR    ;XON/XOFF ENABLED AND X-ON SENT ?
11006    407        JMP     PCEOM      ;   NO
11007   6102        FLAGCHANGE         ;YES
11010 100012        RESET+FLW.
11011   1000        1000               ;CLEAR X-ON SENT FLAG
11012  30710        LDA     2,PPCB
11013  20420        LDA     0,CXOFF
11014   7032        JSR     @SND.,2    ;AND SEND X-OFF NOW
11015   6102 PCEOM: FLAGCHANGE         ;END OF INPUT (MESSAGE)
11016 100012        RESET+FLW.         ;DISABLE INPUT
11017    100        100
11020  30702        LDA     2,PPCB
11021  34103        IOCALL
11022   7751        IBDONE             ;PROCESS IOB FOR INPUT DONE, INCL. XMEM IF ANY
11023 102400        SUB     0,0
11024  41026        STA     0,PDC.,2   ;RESET TIME OUT
11025  21015 PCIOD: LDA     0,ABN.,2   ;   << FROM PCODN, PCIOR
11026  24462        LDA     1,VPRTDT
11027 107414        AND#    0,1,SZR    ;DETACHED PORT ?
11030    405        JMP     PCDDT      ;   YES
11031   6147        STINTERACTION      ;NO, START AN INTERACTION
11032   2721        JMP     @.PCHAR

11033    223 CXOFF:         223        ;X-OFF, OR CTRL-S
11034   1004 C1004:         1004
```

RB. 24 Dec 87

```
;                       << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

   11035  35030 PCDDT: LDA     3,TON.,2 ;DETACHED PORT
   11036 175014        SKZ     3,3      ;DRIVER PROVIDED ?
   11037   5400        JSR     0,3      ;   YES, CALL IT
   11040   2713        JMP     @.PCHAR

;;-------- - ------- --- ---------- ---------- ----------
;;     13.  BACKSPACE
;;
;;          PURPOSE:  REMOVE LAST CHAR FROM INPUT STREAM
;;-----------  ------------------------------ ------- -----------

   11041  35015 PCCH:  LDA     3,ARN.,2 ;BACKSPACE (CTRL H OR CTRL A)  << IHTTBL
   11042  24447        LDA     1,KABITM
   11043 167700        ANDS    3,1      ;EXTRACT INPUT HANDLER TYPE
   11044  34446        LDA     3,KIHSMRR
   11045 136415        SNE     1,3      ;SMB READ RECORD ?
   11046   2445        JMP     @.PIN    ;   YES, BS IS DATA
   11047  25004        LDA     1,IBP.,2 ;NO
   11050  35002        LDA     3,FBA.,2
   11051  40703        STA     0,PCTS2  ;TEMP SAVE INPUT CHAR.
   11052 136433        SSLE    1,3      ;IS THIS IBUFF SEGMENT EMPTY ?
   11053    405        JMP     PCCH1    ;   NO
   11054  34103        IOCALL           ;YES
   11055   7754        IBUNLINK         ;IS THERE A PREVIOUS SEGMENT ?
   11056    647        JMP     JPINC    ;   NO, IBUFF EMPTY; IGNORE CTRL-H
   11057  25004        LDA     1,IBP.,2 ;YES
   11060  15004 PCCH1: DSZ     IBP.,2   ;BACK UP IBP
   11061  34103        IOCALL           ;GET LAST CHARACTER FROM INPUT BUFFER
   11062   7757        IBGETBYTE
   11063 141000        MOV     2,0
   11064  30636        LDA     2,PPCB
   11065  24054        LDA     1,C205
   11066 106414        SEQ     0,1      ;IS IT A CTRL E ?
   11067    407        JMP     PCCH2    ;   NO
   11070   6102        FLAGCHANGE       ;YES
   11071  60012        TOGGLE+SKIPO+FLW.;TOGGLE ECHO-ENABLE
   11072      1        1                ;IS IT NOW ENABLED ?
   11073    632        JMP     JPINC    ;   NO, ECHO NOTHING
   11074  30626        LDA     2,PPCB   ;YES
   11075  20654        LDA     0,COLON  ;ECHO A COLON IF CTRL A
   11076  24656 PCCH2: LDA     1,PCTS2  ;RECOVER INPUT CHARACTER
   11077 125212        SKE     1,1      ;ARE WE DOING CTRL A ?
   11100    406        JMP     JPEN     ;   YES, ECHO THE LAST CHARACTER
   11101  20030        LDA     0,C10    ;NO, ECHO A BACKSPACE
   11102  25033        LDA     1,OCC.,2 ;DECREMENT OCC IF NOT ZERO
   11103 124404        NEG     1,1,SZR
   11104 124000        COM     1,1
   11105  45033        STA     1,OCC.,2
   11106   2401 JPEN:  JMP     @.+1
   11107  10574        PCEN

   11110     40 VPRTDT:        PORTDTCH
   11111   7400 KABITM:        ABITM
   11112      2 KIHSMRR:       IHSMRR
   11113  10535 .PIN:          PCIN
```

RB. 12-24-87

LDA 1,LIB.,2

NOP

;can = 0 in ma
DMA =

```
;                       << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;--- --- ---- ---------------- ------- ---------- ------------
;;     14.  KILL OUTPUT (PCCO)
;;
;;          PURPOSE:  TERMINATE OUTPUT IN PROGRESS
;;--- -------- ------------- -- ------ ------------------------

   11114   25012 PCCO:  LDA     1,FLW.,2 ;KILL OUTPUT (CTRL O)  << FROM IHTTBL
   11115  121300        MOVS    1,0
   11116  101113        SSN     0,0      ;OUTPUT ACTIVE ?
   11117    2634        JMP     @.PCHAR  ;   NO
   11120  102400        SUB     0,0      ;YES
   11121   41011        STA     0,TOB.,2
   11122    4544        JSR     PCTIO    ;TERMINATE I/O

;;--- -------- ------------- ----- --- ---- --------------------
;;     15.  DISABLE OUTPUT, & START INPUT IF ENABLED
;;
;;          PURPOSE:  SAME
;;--- -------- ------------ -------- --- ------- --------------

   11123    6102 PCODN: FLAGCHANGE       ;OUTPUT DONE  << FROM PCOBE, PCO2
   11124  100012        RESET+FLW.
   11125     200        200              ;CLEAR OUTPUT ACTIVE BIT
   11126   32565        LDA     2,@.PPCB
   11127   34103        IOCALL
   11130    7752        OBINIT           ;INITIALIZE OBUF FOR NEXT USE IN CORE
   11131   25012        LDA     1,FLW.,2
   11132  125300        MOVS    1,1
   11133  127113        ADDL#   1,1,SNC  ;INPUT ENABLED ?
   11134     671        JMP     PCIOD    ;   NO
   11135   34103 PCSIN: IOCALL           ;YES, START INPUT NOW  << PCTSK (402)
   11136    7747        IBINIT           ;INITIALIZE INPUT BUFFER IN XMEM, IF ANY
   11137    2401        JMP     @.+1
   11140   10625        PCGNX

   11141   34103 PCOBE: IOCALL           ;OUTPUT BUFFER EMPTY  << FROM PCTSK (401)
   11142    7755        OBLINK           ;IS THERE AN EXTENSION SEGMENT TO LINK TO ?
   11143     760        JMP     PCODN    ;   NO, OUTPUT IS DONE
   11144     421        JMP     PCCQ2    ;   YES, RESTART OUTPUT
```

```
;                       << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;;--- ------ ---------- ----- --------- - ----- ----------------
;;      16.  PAUSE OUTPUT
;;
;;           PURPOSE:  PAUSE OUTPUT INDEFINITELY
;;--- ------ ---------- ------------------------------ -- ---------

   11145   6102 PCCS:  FLAGCHANGE          ;PAUSE OUTPUT (CTRL S)  << IHTTBL
   11146 140012         SET+FLW.           ;SET XOFF FLAG
   11147  40000         40000
   11150   6102         FLAGCHECK
   11151  10000         SKIPZ              ;OUTPUT ACTIVE ?
   11152    200         200
   11153   4513         JSR     PCTIO      ;   YES, TERMINATE I/O -- $MUX MUST CONT. UNTIL TOB = 0
   11154   2536         JMP     @.PC

;;--- ------ ---------- ------------------------------ -- ---------
;;      17.  RESUME OUTPUT
;;
;;           PURPOSE:  RESUME OUTPUT IF PAUSED
;;--- ------ ---------- ------------------------------ -- ---------

   11155   6102 PCCQ:  FLAGCHANGE          ;RESUME OUTPUT IF PAUSED (CTRL Q)  << IHTTBL
   11156 100012         RESET+FLW.         ;CLEAR XOFF FLAG
   11157  40000         40000
   11160   6102         FLAGCHECK
   11161  20000         SKIPO              ;IS OUTPUT ENABLED ?
   11162    200         200
   11163   2527         JMP     @.PC       ;   NO, IGNORE
   11164  32527         LDA     2,@.PPCB   ;YES
   11165  21011 PCCQ2: LDA     0,TOB.,2   ;   << FROM PCOBE
   11166 101014         SKZ     0,0        ;EXPECTING INTERRUPT FROM MUX ?
   11167   2523         JMP     @.PC       ;   YES, WAIT FOR IT ($MUX MUST LEAVE TOB = 0)
   11170 102000         ADC     0,0        ;NO
   11171   7032         JSR     @SND.,2    ;START OUTPUT
   11172 101414         INC#    0,0,SZR    ;OUTPUT CHARACTER REQUESTED ?
   11173   2517         JMP     @.PC       ;   NO

                        ; NO CODE HERE
```

```
;                       << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------- ------- ------- ------- ---------- ----------
;;     18.  OUTPUT CHAR REQUEST
;;
;;          ENTRY:    BRANCH ON TASK (PCSTT+?)
;;                    AC2 <-- PCB
;;
;;          PURPOSE:  SEND CHAR TO DEVICE USING DEVICE DRIVER SEND ROUTINE.
;;                    AND THE FOLLOWING PRECEDENCE:
;;                         1.  IF TOB NONZERO SEND TOB
;;                         2.  SEND NULL IF OUTPUT DELAY IN PROGRESS
;;                       * 3.  SEND NEXT CHAR FROM IOB
;;
;;          NOTE:  DEVICE DRIVER SEND ROUTINE MUST PUT CHAR IN TOB IF BUSY.
;;                 DEVICE DRIVER INTERRUPT HANDLER SHOULD QUEUE A -1 WHEN DONE.
;;
;;             * IGNORE REQUEST IF OUTPUT NOT ACTIVE OR PAUSED OR END OF BUFFER
;;---------- ------------------------------------------------------------

                        ; NO CODE HERE

  11174 126400 PCO:   SUB     1,1        ;OUTPUT CHARACTER REQUEST  << FROM #5C.
  11175  60277        INTDS
  11176  21011        LDA     0,TOB.,2
  11177  45011        STA     1,TOB.,2
  11200  60177        INTEN
  11201 101014        SKZ     0,0        ;ANYTHING IN TOB ?
  11202    441        JMP     PCO3       ;   YES, RE-SEND IT
         11203 PCO1:                     ;---------- THE FOLLOWING CODE IS FOR MARK 5 ONLY:
  11203  25034        LDA     1,ODC.,2   ;MK5:   NO  << FROM PCO3
  11204 125015        SNZ     1,1        ;MK5:ANY OUTPUT DELAY ?
  11205    414        JMP     PCO2       ;MK5:   NO
  11206  20023        LDA     0,CM400    ;MK5:YES
  11207 125113        SSN     1,1        ;MK5:JUST OUTPUT A RETURN ?
  11210    407        JMP     PCO2A      ;MK5:   NO
  11211  25035        LDA     1,RDE.,2   ;MK5:YES, GET RETURN DELAY VALUE
  11212 107700        ANDS    0,1        ;MK5:
  11213   6436        JSR     @.NULC     ;MK5:CONVERT TO NULL COUNT
  11214  45034        STA     1,ODC.,2   ;MK5:
  11215  20032        LDA     0,C12      ;MK5:LINE FEED
  11216    425        JMP     PCO3       ;MK5:

  11217  15034 PCO2A: DSZ     ODC.,2     ;MK5:END OF DELAY ?
  11220    423        JMP     PCO3       ;MK5:   NO, SEND A NULL
         11221        .LOC    PCO1-.*MK3+.;---------- END OF MARK 5-ONLY CODE
```

```
;                       << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>
                        ; *** NO CODE HERE ***

        11221 M3PC01:                   ;---------- THE FOLLOWING IS MARK 3 CODE:
11221   25034         LDA     1,ODC.,2  ;MK3:
11222  125113         SSN     1,1       ;MK3: JUST OUTPUT A RETURN ?
11223    776          JMP     PC02      ;MK3:   NO
11224  102400         SUB     0,0       ;MK3: YES, CLEAR THE RETURN FLAG
11225   41034         STA     0,ODC.,2  ;MK3:
11226   20032         LDA     0,C12     ;MK3: OUTPUT A LINE FEED
11227     414         JMP     PC03      ;MK3:
        11221         .LOC M3PC01-.*MK5+.;---------- END OF MARK 3 CODE

11221   25012 PC02:   LDA     1,FLW.,2  ;YES
11222   34053         LDA     3,C200
11223  137415         AND#    1,3,SNR   ;OUTPUT ENABLED ?
11224    2424         JMP     @.PNC1    ;   NO, CHECK IF INPUT ACTIVE
11225  127112         ADDL#   1,1,SZC   ;YES, IS IT PAUSED (X-OFF) ?
11226    2464         JMP     @.PC      ;   YES, IGNORE FOR NOW
11227   11005         ISZ     OBP.,2    ;NO
11230   25005         LDA     1,OBP.,2
11231   21007         LDA     0,LOB.,2  ;B(LAST BYTE TO BE PUT OUT)
11232  122432         SGR     1,0       ;END OF THIS OUTPUT SEGMENT ?
11233     405         JMP     PC02B     ;   NO
11234   34103         IOCALL            ;YES
11235    7755         OBLINK            ;IS THERE AN EXTENSION SEGMENT ?
11236     665         JMP     PCODN     ;   NO, WRAP IT UP
11237   25005         LDA     1,OBP.,2  ;YES (IT HAS BEEN LINKED IN)
11240   34103 PC02B:  IOCALL
11241    7760         OBGETBYTE         ;GET NEXT OUTPUT BYTE
11242  141000         MOV     2,0
11243    4451 PC03:   JSR     SEND      ;SEND THE CHARACTER
11244   21011         LDA     0,TOB.,2
11245  101014         SKZ     0,0       ;IS A CHARACTER BUFFERED ?
11246    2444         JMP     @.PC      ;   YES, GO TO PC
11247     734         JMP     PC01      ;   NO, GET ONE INTO TOB

11250   10621 .PNC1:  PCNC1

11251   11470 .NULC:  NULCT
        11252         .LOC    .-MK3     ;NULCT POINTER NOT REQUIRED IN MARK 3
```

NOP ; OBP CAN = -1 IF IN MAPPED SPACE
RDC 3-18-87

```
;                        << SI = R9OREXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;-----------  ---------------------------------------  ---------
;;
;;     19.  CANCEL INPUT LINE
;;
;;--  -------  ------------------  ------  --------  --------------

    11252   4414 PCCX:  JSR     PCTIO     ;TERMINATE I/O  << FROM IHTTBL
    11253 102400        SUB     0,0
    11254  41011        STA     0,TOB.,2
    11255  41033        STA     0,OCC.,2
    11256  34103        IOCALL
    11257   7750        IBRESET           ;RESET INPUT BUFFER POINTERS
    11260  20063        LDA     0,C334    ;SEND BACKSLASH
    11261   4433        JSR     SEND
    11262  20055        LDA     0,C215    ;SEND CR (MAY OVERLAY \ IN TOB)
    11263   4431        JSR     SEND
    11264   2426        JMP     @.PC


;;-  -------  -------  ------------------  ------------------------
;;     20.  TERMINATE I/O SUBROUTINE
;;
;;          ENTRY:    KILL OUTPUT (PCCO)
;;                    PAUSE OUTPUT (PCCS)
;;                    CANCEL INPUT LINE (PCCX)
;;                    ESCAPE (PCCC/PCES)
;;
;;          PURPOSE:  TERMINATE I/O USING DEVICE DRIVER SEND ROUTINE
;;
;;          NOTE:  SEND ROUTINE (AS DEFINED IN PCB) DOES ACTUAL RTN TO CALLER
;;-----  ---------  ---------------------------------------  ------

    11265 177775        MXTA
    11266  20777 PCTIO: LDA     0,.-1     ;TERMINATE I/O
    11267  32424        LDA     2,@.PPCB
    11270   3032        JMP     @SNO.,2
```

```
;                        << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------- -- ---------- ------------------------------ ----------------------
;;     21.  PROCESS INTERRUPT TASK
;;
;;          PURPOSE:  BRANCH TO INTERRUPT TASK
;;-------- ---------- ------------------------------ ----------------------

   11271  34413 PCTSK: LDA     3,PCTMX   ;PROCESS INTERRUPT TASK  << FROM #5C.
   11272 116433         SLE     0,3       ;POINTER GIVEN ?
   11273    413         JMP     PCT3      ;   YES
   11274 122400         SUB     1,0       ;NO (A1 STILL = 400)
   11275   4410         JSR     PCT2

   11276  11455 PCITB: PCIBF   ; 400 = INPUT BUFFER FULL (CURRENT SEGMENT)
   11277  11141         PCOBE   ; 401 = OUTPUT BUFFER (CURRENT SEGMENT) EMPTY
   11300  11135         PCSIN   ; 402 = START INPUT
   11301  10762 .PIDN: PCIDN   ; 403 = INPUT DONE (EG. TIMED OUT)
   11302  11402         PCIOR   ; 404 = I/O RESET
   11303  12470         TFALT   ; 405 = (SPARE)

   11304    405 PCTMX: .-PCITB+377      ;MAX. LEGAL TASK NUMBER

   11305 117001 PCT2:  ADD     0,3,SKP   ;BRANCH TO INTERRUPT TASK
   11306 115001 PCT3:  MOV     0,3,SKP
   11307  35400        LDA     3,0,3
   11310   5400        JSR     0,3
   11311   2401        JMP     @.PC

   11312  10357 .PC:   PCHAR
   11313  10722 .PPCB: PPCB
```

```
;                        << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------------------------------------------------------------------------
;;     23.  SEND SUBROUTINE
;;
;;          ENTRY:    ECHO CHAR (PCEN-10)
;;                    OUTPUT CHAR REQUEST (PCO)
;;                    CANCEL LINE (PCCX)
;;                     ACO <-- CHAR
;;
;;          PURPOSE:  1.  SET LF FLAG AFTER CR
;;                    2.  CHECK FOR SPECIAL DELAYS
;;                    3.  SEND BYTE TO DEVICE DRIVER SEND ROUTINE
;;
;;          NOTE:     RETURN TO CALLER IS FROM DEVICE DRIVER SEND ROUTINE
;;-------------------------------------------------------------------------

11314  32777 SEND:  LDA     2,@.PPCB ;SEND CHARACTER IN AO
11315  25036        LDA     1,PCW.,2
11316 127112        ADDL#   1,1,SZC  ;POINT 4 DMA MUX PORT ?
11317   3032        JMP     @SND.,2  ;   YES, $MMUX DOES DELAYS & PARITY
11320  24064        LDA     1,C377   ;   NO
11321 123400        AND     1,0
11322  25012        LDA     1,FLW.,2
11323 125112        SSP     1,1      ;BINARY MODE ?
11324     442       JMP     SEND2    ;   YES
11325  24055        LDA     1,C215   ;NO
        11326       M3L25=  .        ;MK5:---------- THE FOLLOWING IS MARK 5 CODE ONLY:
11326 106415        SNE     0,1      ;MK5:RETURN ?
11327     435       JMP     SNDCR    ;MK5:   YES
11330  25037        LDA     1,TTN.,2 ;MK5:NO
11331 125113        SSN     1,1      ;MK5:ANY SPECIAL DELAY CHARACTERS ?
11332     434       JMP     SEND2    ;MK5:   NO
11333  30037        LDA     2,C17    ;MK5:YES
11334 147400        AND     2,1      ;MK5:EXTRACT TERMINAL TYPE NUMBER
11335  32420        LDA     2,@PCITT ;MK5:
11336 133000        ADD     1,2      ;MK5:
11337  31000        LDA     2,0,2    ;MK5:GET POINTER TO $TERMN
11340  25006        LDA     1,6,2    ;MK5:
11341  44415        STA     1,SNDTS  ;MK5:TEMP.  STORE 2ND SPEC.  DELAY CHAR.
11342  25005        LDA     1,5,2    ;MK5:
11343 106700        SUBS    0,1      ;MK5:
11344  30023        LDA     2,CM400  ;MK5:
11345 133415        AND#    1,2,SNR  ;MK5:SENDING 1ST SPEC.DELAY CHAR.  ?
11346     411       JMP     SENDO    ;MK5:   YES,  SET DELAY VALUE
11347  24407        LDA     1,SNDTS  ;MK5:NO
11350 106700        SUBS    0,1      ;MK5:
11351 133415        AND#    1,2,SNR  ;MK5:SENDING 2ND SPEC.DELAY CHAR.  ?
11352     405       JMP     SENDO    ;MK5:   YES
11353  32740        LDA     2,@.PPCB ;MK5:NO
11354     412       JMP     SEND2    ;MK5:
```

RDC 8-SEP-87
Q.REXM5.RDCG

SKZ 0,0 ; outputting char (not a
(YES)

```
;                       << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

11355     671 PCITT: INFO+.TTT.          ;MK5:

11356       0 SNDTS: 0                   ;MK5: TEMP STORAGE

11357   54777 SENDO: STA     3,SNDTS     ;MK5:
11360   32733        LDA     2,@.PPCB    ;MK5:
11361    4507        JSR     NULCT       ;MK5: EXPAND TO NULL COUNT
11362   34774        LDA     3,SNDTS     ;MK5:
11363     402        JMP     SNDSP       ;MK5:

11364  126000 SNDCR: ADC     1,1         ;MK5: -1 IN ODC MEANS SEND LF NEXT
11365   45034 SNDSP: STA     1,ODC.,2    ;MK5: SENDING A SPECIAL DELAY CHARACTER
         11366        .LOC  M3L25-.*MK3+.;---------- END OF MARK-5-ONLY CODE


         11366        M5L25=  .           ;---------- THE FOLLOWING IS THE MARK 3 VERSION:
11366  106414        SEQ     0,1         ;MK3: OUTPUTTING A RETURN ?
11367       7        JMP     SEND2       ;MK3:   NO
11370  126000        ADC     1,1         ;MK3: YES, SET FLAG TO SEND LINE FEED
11371   45034        STA     1,ODC.,2    ;MK3:
         11366        .LOC  M5L25-.*MK5+.;---------- END OF MARK 3 CODE


11366  103240 SEND2: ADDOR   0,0         ;SET MSB = 1
11367    3032        JMP     @SND.,2     ;NOW SEND THE CHARACTER AND RETURN TO CALLER
```

```
;                       << SI = R90REXPCHRSC;  BO = 1/A.REXM5.7033! >>

;;--- ------ -- - ---------- - ---------- --- ------------------
;;     24.  ESCAPE
;;
;;          PURPOSE:  1.  TERMINATE I/O IN PROGRESS
;;                    2.  RE-INITIALIZE PCB
;;                    3.  SET ABNORMAL TERMINATION CONTROL BITS IN ABN WORD
;;                    4.  START INTERACTION IF NOT REGNANT
;;--- ------   ---------- ---------- ------------------------------------

   11370   6102 PCCC:  FLAGCHANGE          ;SYSTEM ESCAPE (CTRL-C)  << FROM IHTTBL
   11371 140015        SET+ABN.
   11372      2        2                   ;CTRL-C INDICATOR
   11373    404        JMP     PCES2

   11374   6102 PCES:  FLAGCHANGE          ;ESCAPE FROM CURRENT TASK (ESC)  << IHTTBL
   11375 140015        SET+ABN.
   11376      1        1                   ;ESCAPE INDICATOR
   11377  32714 PCES2: LDA     2,@.PPCB
   11400   2401        JMP     @.ESCLINK;EXECUTE ANY TASKS LINKED INTO EXCAPE PROCESSING

          11401 .ESCLINK:                  ;THERE IS A POINTER TO HERE AT INFO+ESCL.
   11401  11402        PCIOR               ;MAY BE OVERLAID BY AN INTERCEPTOR

   11402  60277 PCIOR: INTDS               ;CONTINUE ESCAPE PROCESSING  << PCTSK (404)
   11403  25012        LDA     1,FLW.,2
   11404  20444        LDA     0,FLWMASK
   11405 123400        AND     1,0         ;TURN OFF VARIOUS I/O CONTROL BITS
   11406 101400        INC     0,0         ;ENABLE ECHO
   11407  41012        STA     0,FLW.,2
   11410 125300        MOVS    1,1
   11411 125113        SSN     1,1         ;OUTPUT ACTIVE ?
   11412    405        JMP     PCIO2       ;   NO
   11413   4653        JSR     PCTIO       ;YES, TERMINATE MUX I/O
   11414  34103        IOCALL
   11415   7752        OBINIT              ;INITIALIZE OUTPUT BUFFER (IN CORE)
   11416    410        JMP     PCIO6

   11417 127113 PCIO2: ADDL#   1,1,SNC     ;IS INPUT ACTIVE ?
   11420    405        JMP     PCIO4       ;   NO
   11421   4645        JSR     PCTIO       ;YES, TERMINATE I/O
   11422  34103        IOCALL
   11423   7751        IBDONE              ;INPUT-DONE PROCESSING
   11424    402        JMP     PCIO6

   11425   4641 PCIO4: JSR     PCTIO       ;TERMINATE I/O (JUST IN CASE)
   11426 102400 PCIO6: SUB     0,0
   11427  41011        STA     0,TOB.,2
   11430  41026        STA     0,PDC.,2
   11431  41034        STA     0,ODC.,2
   11432  35017        LDA     3,PCX.,2    ;CLEAR IIB
   11433  21404        LDA     0,IBP.,3
   11434  41405        STA     0,OBP.,3
```

```
;                       << SI = R90REXPCHRSC;  BD = 1/A.REXM5.7033! >>

   11435  20063         LDA     0,C334
   11436  25037         LDA     1,TTN.,2
   11437  34072         LDA     3,C4000
   11440 137415         AND#    1,3,SNR   ;SILENT ESCAPE ?
   11441   4653         JSR     SEND      ;   NO, ECHO A "\"
   11442  20005         LDA     0,RUP
   11443 112414         SEQ     0,2       ;REGNANT USER ?
   11444   2403         JMP     @.PIOD    ;   NO, DO I/O DONE PROCESSING
   11445  50073         STA     2,ESCF    ;YES, SET ESCAPE FLAG
   11446   2641         JMP     @.PC

   11447  11025 .PIOD:PCIOD
   11450  32436 FLWMASK:       32436


;;-------- ---------- ------------------------------ ----------------------
;;     25.  BYPASS ERROR BRANCH
;;
;;          PURPOSE:  SET "LAST CHAR WAS ^Y" FLAG
;;-------- ---------- ------------------------------ ----------------------

   11451   6102 PCCY:  FLAGCHANGE       ;BYPASS ERROR BRANCH (CTRL Y)  << FROM IHTTBL
   11452 140012         SET+FLW.
   11453    400         400
   11454   2636         JMP     @.PC


;;-------- ---------- ------------------------------ ----------------------
;;      26.    INPUT BUFFER FULL
;;
;;      ENTRY:   PROCESS INTERRUPT TASK (PCTSK)
;;               400 = INPUT BUFFER FULL, MAY NEED EXTENSION SEGMENT LINKED IN
;;               A2 = PCB
;;
;;      PURPOSE:RESPOND TO AUTO-INPUT BUFFER FULL
;;-------- ---------- ------------------------------ ----------------------

   11455  34103 PCIBF:IOCALL           ;INPUT BUFFER SEGMENT IS FULL  << FROM PCTSK (400)
   11456   7753         IBLINK          ;IS THERE AN EXTENSION SEGMENT ?
   11457    402         SKIP
   11460   2406         JMP     @.PINC  ;   YES, CONTINUE INPUT
   11461  21015         LDA     0,ABN.,2 ;NO, INPUT BUFFER IS FULL
   11462  24042         LDA     1,C20
   11463 107414         AND#    0,1,SZR ;IN FIXED LENGTH FIELD INPUT MODE?
   11464   2626         JMP     @.PC    ;   YES, DO NOTHING
   11465   2614         JMP     @.PIDN  ;NO, INPUT IS DONE

   11466  10620 .PINC:         PCINC
```

```
;                    << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

        11467 NULM3=.       ;CODE ON THIS PAGE NOT APPLICABLE TO MARK 3

; NULCT - Routine to convert from 50ths of seconds to a null count.
;
; Entry:
;         Ac1 - 50ths of second count
;         Ac2 - PCB address
;
; Exit:
;         Ac1 - Null count
;         Ac2 - PCB address

   11467       0 NULRT: 0                  ;Return address

   11470    5477/ NULCT: STA     3,NULRT
   11471   21036         LDA     0,PCW.,2 ;Port Control Word
   11472   34027         LDA     3,C7
   11473  163400         AND     3,0      ;Mask for baud rate bits
   11474    4411         JSR     NULTB    ;AC3= table address
;      Baud rate factor table.
;       Note: All elements are scaled by a factor of 2^5 (32).
                        ;                  ;Baud
   11475       7         7                  ;110
   11476      12         12                 ;150
   11477      24         24                 ;300
   11500    2400         2400               ;19200 (Patch to 50 for 600 baud)
   11501     120         120                ;1200
   11502     240         240                ;2400
   11503     500         500                ;4800
   11504    1200         1200               ;9600

   11505  117000 NULTB: ADD     0,3        ;"Index" into table
   11506   21400         LDA     0,0,3      ;Conversion factor
   11507   34042         LDA     3,C20      ;Cycle count (16.)
   11510   54417         STA     3,NULCY    ;Set counter
   11511  176440         SUBO    3,3        ;A1=A0*A1
   11512  125203 NULLP: MOVR    1,1,SNC    ;Skip to add
   11513  175201         MOVR    3,3,SKP    ;Nothing to add, skip
   11514  117220         ADDZR   0,3        ;Add and shift
   11515   14412         DSZ     NULCY      ;Done ?
   11516     7/4         JMP     NULLP      ;  No
   11517  125200         MOVR    1,1        ;=Factor*50ths
   11520  175220         MOVZR   3,3
   11521  125200         MOVR    1,1        ;/2
   11522  125220         MOVZR   1,1        ;/4
   11523  125220         MOVZR   1,1        ;/8
   11524  125220         MOVZR   1,1        ;/16
   11525  125220         MOVZR   1,1        ;/32
   11526    2741         JMP     @NULRT

   11527       0 NULCY: 0                  ;Cycle counter

        11530         .LOC    NULM3-.*MARK3+.;OVERLAY THIS PAGE IF MARK 3
```

```
;                        << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;   THE TABLE BELOW IS A TABLE OF INPUT HANDLER TABLES.   INPUT HANDLER
;   TABLES DEFINE HOW NON-PRINTABLE CHARACTERS ARE TREATED ON INPUT BY PC.
;   A PARTICULAR TABLE IS SELECTED BY THE INPUT HANDLER TYPE FIELD (ABITM)
;   OF THE ABN. WORD OF THE PORT CONTROL BLOCK (PCB).   THAT FIELD IS
;   TREATED AS A WORD OFFSET INTO THE TABLE BELOW.

          11530 IHTTBL:       ;THERE IS A POINTER TO HERE AT QCHARACTER ENTRY POINT - 1
    11530 11550         ITIRS  ;TYPE = 0, STANDARD IRIS INPUT HANDLING
    11531 11612         ITSMB  ;TYPE = 1, SMBASIC INPUT STATEMENT STYLE INPUT HANDLING
    11532 11612         ITSMB  ;TYPE = 2 (IHSMRR), SMBASIC READRECORD STYPE INPUT HANDLING
    11533 11654         ITSIM  ;TYPE = 3, ONLY CONTROL-Q AND CONTROL-S DEFINED
    11534 11550         ITIRS  ;TYPE = 4, UNDEFINED, DEFAULT TO STANDARD IRIS
    11535 11550         ITIRS  ;TYPE = 5, UNDEFINED, DEFAULT TO STANDARD IRIS
    11536 11550         ITIRS  ;TYPE = 6, UNDEFINED, DEFAULT TO STANDARD IRIS
    11537 11550         ITIRS  ;TYPE = 7, UNDEFINED, DEFAULT TO STANDARD IRIS
    11540 11550         ITIRS  ;TYPE = 8, UNDEFINED, DEFAULT TO STANDARD IRIS
    11541 11550         ITIRS  ;TYPE = 9, UNDEFINED, DEFAULT TO STANDARD IRIS
    11542 11550         ITIRS  ;TYPE = 10, UNDEFINED, DEFAULT TO STANDARD IRIS
    11543 11550         ITIRS  ;TYPE = 11, UNDEFINED, DEFAULT TO STANDARD IRIS
    11544 11550         ITIRS  ;TYPE = 12, UNDEFINED, DEFAULT TO STANDARD IRIS
    11545 11550         ITIRS  ;TYPE = 13, UNDEFINED, DEFAULT TO STANDARD IRIS
    11546 11550         ITIRS  ;TYPE = 14, UNDEFINED, DEFAULT TO STANDARD IRIS
    11547 11550         ITIRS  ;TYPE = 15, UNDEFINED, DEFAULT TO STANDARD IRIS

;   THE STRUCTURES BELOW ARE INPUT HANDLER TABLES.   THESE TABLES DEFINE
;   HOW NON-PRINTABLE CHARACTERS (OCTAL 200-237 AND 377) ARE TO BE TREATED
;   ON INPUT FROM INTERACTIVE AND DETACHED PORTS.   THE TABLE TO BE USED IS
;   SELECTED FROM THE TABLE OF TABLES (IHTTBL) BY THE INDEX CONTAINED IN
;   THE ABITM FIELD (INPUT HANDLER TYPE) OF THE ABN. WORD OF THE PCB.

; IN THESE BRANCH TABLES, IF MSB = 0, THE CHARACTER IS PROCESSED
; IMMEDIATELY UPON RECEIPT.   IF MSB = 1 (CAUSED BY @), CHARACTER
; IS SUBJECT TO BEING BUFFERED IN IIB (INTERMEDIATE INPUT BUFFER)
```

```
;                       << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

; ITIRS  -  STANDARD IRIS INPUT HANDLER TABLE

 11550      20 ITIRS:"P-100 ;CONTROL-P TOGGLES PARITY
 11551   10620         PCINC   ;377 = RUBOUT, IGNORE
 11552   10574         PCEN    ;200 = ^@  NULL (ECHO ONLY)
 11553  111041      @  PCCH    ;201 = ^A  BACKSPACE, ECHO CHARACTER
 11554   10711         PCCB    ;202 = ^B  (BREAK)
 11555   11370         PCCC    ;203 = ^C  (SYSTEM ESCAPE)
 11556   11374         PCES    ;204 = ^D  (ESCAPE)
 11557  110727      @  PCCE    ;205 = ^E  (TOGGLE ECHO)
 11560  110536      @  PCIN+1  ;206 = ^F
 11561  110535      @  PCIN    ;207 = ^G  BELL
 11562  111041      @  PCCH    ;210 = ^H  BACKSPACE, ECHO ^H
 11563  110536      @  PCIN+1  ;211 = ^I  HORIZONTAL TAB
 11564   10620         PCINC   ;212 = ^J  LINE FEED (IGNORE)
 11565  110535      @  PCIN    ;213 = ^K  VERTICAL TAB
 11566  110535      @  PCIN    ;214 = ^L  FORM FEED
 11567  110765      @  PCICR   ;215 = ^M  RETURN
 11570  110536      @  PCIN+1  ;216 = ^N
 11571   11114         PCCO    ;217 = ^O  (KILL OUTPUT)
 11572   10662         PCCP    ;220 = ^P  (TOGGLE PARITY CHECK)
 11573   11155         PCCQ    ;221 = ^Q  XON (RESUME OUTPUT IF PAUSED)
 11574  110536      @  PCIN+1  ;222 = ^R  TAPE
 11575   11145         PCCS    ;223 = ^S  XOFF (PAUSE OUTPUT)
 11576  110536      @  PCIN+1  ;224 = ^T  /TAPE
 11577  110536      @  PCIN+1  ;225 = ^U
 11600  110536      @  PCIN+1  ;226 = ^V
 11601  110536      @  PCIN+1  ;227 = ^W
 11602  111252      @  PCCX    ;230 = ^X  CAN (CANCEL LINE)
 11603   11451         PCCY    ;231 = ^Y  (BYPASS ERROR BRANCH IN RUN)
 11604  110536      @  PCIN+1  ;232 = ^Z  (CHANGED TO RETURN BY BASIC & EDIT)
 11605   11374         PCES    ;233 = ^[  ESC (ESCAPE)
 11606  110535      @  PCIN    ;234 = ^\
 11607  110535      @  PCIN    ;235 = ^]
 11610  110535      @  PCIN    ;236 = ^^
 11611  110707      @  PCER    ;237 = ^_  (RING BELL)
```

```
;                       << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>
;   ITSMB  -  SMBASIC INPUT HANDLER TABLE

          11612 ITSMB:
    11612 177777          177777 ;NO PARITY TOGGLE CHARACTER
    11613 110535      @   PCIN   ;377 = RUBOUT,  DATA
    11614 110535      @   PCIN   ;200 = ^@, DATA
    11615 110535      @   PCIN   ;201 = ^A, DATA
    11616 110535      @   PCIN   ;202 = ^B, DATA
    11617 110535      @   PCIN   ;203 = ^C, DATA
    11620 110535      @   PCIN   ;204 = ^D, DATA
    11621 110535      @   PCIN   ;205 = ^E, DATA
    11622 110535      @   PCIN   ;206 = ^F, DATA
    11623 110535      @   PCIN   ;207 = ^G, DATA
    11624 111041      @   PCCH   ;210 = ^H, PERFORM BACKSPACE
    11625 110535      @   PCIN   ;211 = ^I, DATA
    11626 110755      @   PCILC  ;212 = ^J, LINE FEED, INPUT TERMINATOR
    11627 110535      @   PCIN   ;213 = ^K, DATA
    11630 110535      @   PCIN   ;214 = ^L, DATA
    11631 110765      @   PCICR  ;215 = ^M, CARRIAGE RETURN, INPUT TERMINATOR
    11632 110535      @   PCIN   ;216 = ^N, DATA
    11633 110535      @   PCIN   ;217 = ^O, DATA
    11634 110535      @   PCIN   ;220 = ^P, DATA
    11635  11155          PCCQ   ;221 = ^Q, XON, RESUME OUTPUT IF PAUSED
    11636 110535      @   PCIN   ;222 = ^R, DATA
    11637  11145          PCCS   ;223 = ^S, XOFF, PAUSE OUTPUT
    11640 110535      @   PCIN   ;224 = ^T, DATA
    11641 110535      @   PCIN   ;225 = ^U, DATA
    11642 110535      @   PCIN   ;226 = ^V, DATA
    11643 110535      @   PCIN   ;227 = ^W, DATA
    11644 110535      @   PCIN   ;230 = ^X, DATA
    11645 110535      @   PCIN   ;231 = ^Y, DATA
    11646 110535      @   PCIN   ;232 = ^Z, DATA
    11647  11374          PCES   ;233 = ^[, ESCAPE
    11650 110755      @   PCILC  ;234 = ^\, MOTOR BAR   I, INPUT TERMINATOR
    11651 110755      @   PCILC  ;235 = ^], MOTOR BAR  II, INPUT TERMINATOR
    11652 110755      @   PCILC  ;236 = ^^, MOTOR BAR III, INPUT TERMINATOR
    11653 110755      @   PCILC  ;237 = ^_, MOTOR BAR  IV, INPUT TERMINATOR
```

```
;                       << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;  ITSIM  -  SIMPLE INPUT, ONLY CR, CTRL-Q AND CTRL-S RECOGNIZED

          11654 ITSIM:
   11654 177777/          177777 ;NO PARITY TOGGLE CHARACTER
   11655 110535      @    PCIN   ;377 = RUBOUT, DATA
   11656 110535      @    PCIN   ;200 = ^@, DATA
   11657 110535      @    PCIN   ;201 = ^A, DATA
   11660 110535      @    PCIN   ;202 = ^B, DATA
   11661 110535      @    PCIN   ;203 = ^C, DATA
   11662 110535      @    PCIN   ;204 = ^D, DATA
   11663 110535      @    PCIN   ;205 = ^E, DATA
   11664 110535      @    PCIN   ;206 = ^F, DATA
   11665 110535      @    PCIN   ;207 = ^G, DATA
   11666 110535      @    PCIN   ;210 = ^H, DATA
   11667 110535      @    PCIN   ;211 = ^I, DATA
   11670 110535      @    PCIN   ;212 = ^J, DATA
   11671 110535      @    PCIN   ;213 = ^K, DATA
   11672 110535      @    PCIN   ;214 = ^L, DATA
   11673 110765      @    PCICR  ;215 = ^M, CARRIAGE RETURN, INPUT TERMINATOR
   11674 110535      @    PCIN   ;216 = ^N, DATA
   11675 110535      @    PCIN   ;217 = ^O, DATA
   11676 110535      @    PCIN   ;220 = ^P, DATA
   11677  11155           PCCQ   ;221 = ^Q, XON, RESUME OUTPUT IF PAUSED
   11700 110535      @    PCIN   ;222 = ^R, DATA
   11701  11145           PCCS   ;223 = ^S, XOFF, PAUSE OUTPUT
   11702 110535      @    PCIN   ;224 = ^T, DATA
   11703 110535      @    PCIN   ;225 = ^U, DATA
   11704 110535      @    PCIN   ;226 = ^V, DATA
   11705 110535      @    PCIN   ;227 = ^W, DATA
   11706 110535      @    PCIN   ;230 = ^X, DATA
   11707 110535      @    PCIN   ;231 = ^Y, DATA
   11710 110535      @    PCIN   ;232 = ^Z, DATA
   11711 110535      @    PCIN   ;233 = ^[, DATA
   11712 110535      @    PCIN   ;234 = ^\, DATA
   11713 110535      @    PCIN   ;235 = ^], DATA
   11714 110535      @    PCIN   ;236 = ^^, DATA
   11715 110535      @    PCIN   ;237 = ^_, DATA
```

```
;                       << SI = R90REXPCHRSC; BO = 1/A.REXM5.7033! >>

;;-------- -- ---------------------- ----- -------------------------
;;     SIGNAL TASK
;;
;;        ENTRY:    INTERRUPT SERVICE EXIT (INTRA) WHEN SIGNAL TASK
;;                  BECOMES REGNANT
;;                  AC0 <-- PCB;destination port
;;                  AC1 <-- ;first signal parameter
;;                  AC2 <-- ;second signal parameter
;;
;;        CALLING SEQUENCE:
;;                  QUEUE
;;                  SIGNAL (OR SSIGTASK IF CALLING FROM WITHIN REX)
;;                  <PCB POINTER>
;;                  QP.SIGNAL
;;                  <NOT USED>
;;                  <-- RETURN
;;
;;        PURPOSE:  SET PDC IN PCB TO 1 IF "ACTIVATE FROM SIGNAL" FLAG IN
;;                  FLW IS SET.  QUEUE A NULL SIGNAL FROM THE PORT TO
;;                  ITSELF (PCB,PCB,0,0).
;;--- ---- ---------- ---- --------- -- ----------------------------

 11716 100000         @0                  ;CONFLICT FLAG = TRUE
 11717  40404 SSIGT:  STA     0,SDEST
 11720  44404         STA     1,.+4
 11721  50404         STA     2,.+4
 11722   4404         JSR     .+4
 11723      0 SDEST:  0                   ;destination port
 11724      0         0                   ;first signal parameter
 11725      0         0                   ;second signal parameter
 11726 102520         SUBZL   0,0         ;A0 = 1 --> Send signal
 11727  24007         LDA     1,RTP       ;A1 = sending port
 11730 171000         MOV     3,2         ;A2 = parameter list pointer
 11731   6101         CALL
 11732     57         SIGPAUSE
 11733    402         JMP     SIG1        ;BUFFER FULL, ECHO BELL
 11734   6105         DQUEUE

       11735 SIG1:                       ;IF NOT SYSTEM SIGNAL, ECHO BELL
 11735 151112         SSP     2,2         ;IF SYSTEM SIGNAL
 11736  30765         LDA     2,SDEST     ;SEND TO RECEIVER
 11737   6103         QCHARACTER
 11740   6105         DQUEUE


              .EOT  ;REX "PCHR" FOR "IRIS" RESIDENT EXECUTIVE
```

```
;                          << SI = R9OREXMTTYSA; BO = 1/A.REXM5.7033! >>
;  29 JUN 86, RB.

;;----------- - -------- -- --------- ---------- --------------------------
;;
;;        R E X :    M A S T E R     T T Y    H A N D L E R
;;
;+
;;    MAJOR COMPONENTS:
;;
;;         1.   INTERRUPT SERVICE
;;         2.   SEND ROUTINE
;;         3.   POWER-FAIL RESTART
;;
;;    UPDATE RECORD:
;;         6-10-80   (LLL)   DOCUMENTED FOR READABILITY
;;         4-11-82   (rb.)   incorporated Mark 3 / Mark 5 code
;;       11 MAY 86    RB.     SET MSB IN INPUT TO INDICATE NO PARITY ERROR
;;       29 JUN 86    RB.     NEW TOB ACCESS LOGIC
;;
;;    GLOBALS:
;;         SUBROUTINES:
;;            QCHARACTER
;;         DATA:
;;            INFO+LPCA
;;            .INTR    (POINTER TO INTERRUPT RETURN IN INTS)
;;
;;----------- - ------------------------ -- ---------- ----------------------


;  DEFINE DEVICE CODES FOR MARK 3:

               10 PZS=   10      ;PORT ZERO STATUS INPUT OR CONTROL OUTPUT

               11 PZD=   11      ;PORT ZERO DATA INPUT OR OUTPUT
```

```
;                        << SI = R90REXMTTYSA; BO = 1/A.REXM5.7033! >>
;;-- -------- ------ ---------- ------ ---------------------------
;;
;;     1.   INTERRUPT SERVICE
;;
;;          PURPOSE:   PERFORMS THE ACTUAL I-O OPERATION, EVERY TIME THE
;;                     DEVICE INTERRUPTS FOR A NEW CHARACTER
;;
;;          ENTRY:     (INPUT)  VECTORS TO "TTIS"
;;                     (OUTPUT) VECTORS TO "TTOS"
;;                     in Mark 3 both vector to PZIH
;;
;;          EXIT:      BRANCHES TO SYSTEM INTERRUPT RETURN ROUTINE
;;
;;- -------- ------------------- - -- ------------------------------------

; FIRST THE MARK 5 CODE :

          11741 M3L30=.          ;SAVE STARTING LOCATION FOR MARK 3

  11741       2          2                    ;MASK OUT TTI
  11742    1400          JMP     0,3          ;POWER-FAIL RESTART DONE BY TTO
  11743   60610 TTIS:    DIAC    0,TTI        ;MASTER TELETYPE INPUT INTERRUPT
  11744   32421          LDA     2,@I.LPC
  11745   24053          LDA     1,C200
  11746  107415          AND#    0,1,SNR
  11747  123000          ADD     1,0
  11750    2103          JMP     @QCHARACTER&377


  11751       3          3                    ;MASK OUT TTO & TTI IN CASE OF T-OPTION MMUX
  11752     433          JMP     TTPFR        ;TTO POWER-FAIL RESTART ROUTINE
  11753   60211 TTOS:    NIOC    TTO          ;MASTER TELETYPE OUTPUT INTERRUPT
  11754   32411          LDA     2,@I.LPC
  11755  102000 TTOS2:   ADC     0,0          ;REPORT OUTPUT DONE
  11756   25011          LDA     1,TOB.,2
  11757  125015          SNZ     1,1          ;ANYTHING IN TOB ?
  11760    2103          JMP     @QCHARACTER&377;   NO
  11761  102400          SUB     0,0          ;YES
  11762   41011          STA     0,TOB.,2
  11763   65111          DOAS    1,TTO        ;OUTPUT IT
  11764    1400          JMP     0,3

          11765 M5L30=.
```

```
;                    << SI = R90REXMTTYSA;  BO = 1/A.REXM5.7033! >>

; AND NOW THE CORRESPONDING MARK 3 CODE :

          11765 .LOC   M3L30-.*MK3+.     ;RESTORE STARTING LOCATION, IF MARK 3

    11765 102400 PZIH:  SUB     0,0        ;MK 3 PORT ZERO INTERRUPT HANDLER
    11766  6217/        DOBS    0,CPU      ;TURN OFF PORT ZERO INTERRUPT
    11767  327/6        LDA     2,@I.LPC
    11770  60410        DIA     0,PZS      ;READ PORT ZERO STATUS
    11771 101203        MOVR    0,0,SNC    ;INPUT INTERRUPT ?
    11772    406        JMP     PZOUT      ;   NO, MUST BE OUTPUT
    11773  60411        DIA     0,PZD      ;READ PORT 0 DATA INPUT
    11774  24053        LDA     1,C200
    11775 107415        AND#    0,1,SNR
    11776 123000        ADD     1,0        ;SET MSB TO INDICATE NO PARITY ERROR
    11777   2103        JMP     @QCHARACTER&377;QUEUE IT AND RETURN

    12000 101213 PZOUT: SKO     0,0        ;OUTPUT REGISTER EMPTY ?
    12001   1400        JMP     0,3        ;   NO, IGNORE INTERRUPT
    12002 102000        ADC     0,0        ;YES
    12003   6103        QCHARACTER         ;QUEUE OUTPUT-DONE NOTIFICATION
    12004  21011        LDA     0,TOB.,2
    12005 101015        SNZ     0,0        ;IS THERE A CHARACTER IN TOB ?
    12006    405        JMP     PZOU2      ;   NO
    12007  61011        DOA     0,PZD      ;YES, OUTPUT IT
    12010 102400        SUB     0,0
    12011  41011        STA     0,TOB.,2
    12012  20044        LDA     0,C40      ;PREPARE TO REQUEST OUTPUT-DONE INTERRUPT
    12013  25036 PZOU2: LDA     1,PCW.,2
    12014 107000        ADD     0,1
    12015  65010        DOA     1,PZS      ;TURN OUTPUT INTERRUPT REQUEST ON OR OFF
    12016   2111        JMP     @.INTR

          11765 .LOC   M5L30-.*MK5+.     ;---------- END OF HARDWARE-DEPENDENT CODE


    11765    604 I.LPC: INFO+LPCA.         ;LOCATION OF PORT 0 CONTROL BLOCK
```

```
;                         << SI = R90REXMTTYSA; BO = 1/A.REXM5.7033! >>
;;-------------------------------------------------------------------------
;;
;;      2.   SEND ROUTINE
;;
;;      PURPOSE:   SENDS A CHARACTER OUT TO INITIATE AN OUTPUT, OR
;;                 INITIALIZES AN INPUT OR OUTPUT
;;
;;      ENTRY:     A2 <-- A(PCB)
;;                 A0 = -1 --> START OUTPUT
;;                    = -2 --> START INPUT
;;                    >= 0 --> OUTPUT CHARACTER IN A0
;;
;;      EXIT:      RETURN TO CALLER, WITH A2 INTACT
;;                 IN START OUTPUT, LEAVE A0 = -1 TO MEAN NOT IF DEVICE DMA OUTPUT
;;-------------------------------------------------------------------------

        11766         M3L32= .          ;---------- THE FOLLOWING IS MARK 5 CODE:
 11766 105405 TTSND: INC     0,1,SNR    ;MK5:MASTER TTY "SEND" ROUTINE
 11767    403         JMP     TTSNO      ;MK5:  A0 = -1 := START OUTPUT
 11770 103112         ADDL#   0,0,SZC    ;MK5:OTHER CONTROL FUNCTION ?
 11771    411         JMP     TTSNI      ;MK5:  YES
 11772  60277 TTSNO: INTDS              ;MK5:NO, MUST BE OUTPUT CHARACTER
 11773  63411         SKPBN   TTO        ;MK5:OUTPUT BUSY ?
 11774    403         JMP     TTSN1      ;MK5:
 11775  41011         STA     0,TOB.,2   ;MK5:  YES, SAVE CHARACTER
 11776    402         SKIP               ;MK5:
 11777  61111 TTSN1: DOAS    0,TTO      ;MK5:  NO, OUTPUT CHARACTER
 12000  60177         INTEN              ;MK5:
 12001   1400         JMP     0,3        ;MK5:

 12002 125415 TTSNI: INC#    1,1,SNR    ;MK5:START INPUT ?
 12003  60110         NIOS    TTI        ;MK5:  YES
 12004   1400         JMP     0,3        ;MK5:

        12005         M5L32= .          ;---------- THE FOLLOWING IS MARK 3 CODE:
        12005         .LOC M3L32-.*MK3+.
 12005 101415 PZSND: INC#    0,0,SNR    ;MK3:START OUTPUT ?
 12006    413         JMP     PZSTO      ;MK3:  YES
 12007 103112         ADDL#   0,0,SZC    ;MK3:ANY OTHER CONTROL FUNCTION ?
 12010   1400         JMP     0,3        ;MK3:  YES, NO ACTION REQUIRED
 12011  60277         INTDS              ;MK3:NO, OUTPUT THE CHARACTER IN A0
 12012  64410         DIA     1,PZS      ;MK3:
 12013 125200         MOVR    1,1        ;MK3:
 12014 125212         SKE     1,1        ;MK3:OUTPUT BUSY ?
 12015  61011         DOA     0,PZD      ;MK3:  NO, SEND THE CHARACTER
 12016 125213         SKO     1,1        ;MK3:
 12017  41011         STA     0,TOB.,2   ;MK3:  YES, SAVE CHARACTER IN TOB
 12020  60177         INTEN              ;MK3:
 12021  21036 PZSTO: LDA     0,PCW.,2   ;MK3:
 12022  24044         LDA     1,C40      ;MK3:
 12023 123000         ADD     1,0        ;MK3:
 12024  61010         DOA     0,PZS      ;MK3:REQUEST INTERRUPT WHEN OUTPUT IS DONE
 12025   1400         JMP     0,3        ;MK3:

        12005         .LOC M5L32-.*MK5+.;---------- END OF HARDWARE-DEPENDENT CODE
```

```
;                    << SI = R90REXMTTYSA; BO = 1/A.REXM5.7033! >>
;;-------------------------------------------------------------------------
;;
;;    3.    POWER FAIL RESTART  (NOT APPLICABLE IN MARK 3)
;;
;;      PURPOSE:   RE-INITIALIZE THE MASTER TTY AFTER A POWER FAIL
;;                 HAS OCCURED
;;
;;      ENTRY:     A3 <-- RETURN ADDR.
;;
;;      EXIT:      IF (INPUT WAS ENABLED) THEN RESTART INPUT
;;                 ELSE IF (OUTPUT WAS ENABLED ) THEN RESTART OUTPUT
;;-------------------------------------------------------------------------

          12005        M3L35= .          ;---------- THE FOLLOWING IS MARK 5 CODE:

   12005  32760 TTPFR: LDA     2,@I.LPC ;MASTER TTY "POWER FAIL RESTART"
   12006  21012        LDA     0,FLW.,2
   12007 101300        MOVS    0,0
   12010 101112        SSP     0,0       ;OUTPUT ENABLED ?
   12011    741        JMP     TTOS2     ;   YES, RESUME IT
   12012 103112        ADDL#   0,0,SZC   ;INPUT ENABLED ?
   12013  60110        NIOS    TTI       ;   YES
   12014   1400        JMP     0,3

          12015        .LOC M3L35-.*MK3+.;---------- END OF MARK-5-ONLY CODE

                .EOT  ;REX "MTTY" FOR "IRIS" RESIDENT EXECUTIVE
```

```
;                          << SI = R9OREXINTSSA; BO = 1/A.REXM5.7033! >>
; 9 AUG 86, RB.

;;---------- ---------- ---------- ----- -------- --------------------
;;
;;        R E X :    I N T E R R U P T    S E R V I C E
;;
;;    MAJOR COMPONENTS:
;;         1.   INTERRUPT DISTRIBUTOR
;;         2.   INTERRUPT RETURN
;;         3.   INTERRUPT HANDLER FOR UNKNOWN DEVICE
;;         4.   INTERRUPT IGNORER
;;
;;    UPDATE RECORD:
;;         6-10-80   (LLL) DOCUMENTED FOR READABILITY
;;         2- 5-82   (rb.) allow for indirect TRAP 0, added some comments
;;         4-11-82   (rb.) provided for Mark 5 / Mark 3 versions
;;
;;    INTERRUPT STACK DISPLACEMENTS (SAME AS FIRST 8 WORDS IN TASK CONTROL NODE)
;;
;;         0   :   A2
;;         1   :   A1
;;         2   :   A0
;;         3   :   A3
;;         4   :   PC
;;         5   :   BIT 15 <-- CARRY, BITS 14-0 <-- CURR MASK
;;         6   :   SBA
;;         7   :   DBA
;;
;;---------- ------- ---------- ----------------------------------

; SPECIAL INSERT USED FOR NOVA 3 ONLY:

           12015        M3L40= .            ;---------- THIS CODE NOT REQUIRED FOR MARK 3

   12015  34046 INT3A: LDA     3,46         ;CHECK FOR NOP TRAP
   12016  20047         LDA     0,47         ;(FOR NOVA 3 ONLY)
   12017 106415         SNE     0,1          ;INTERRUPTED AT TRAP3 ?
   12020 175401         INC     3,3,SKP      ;   YES
   12021 106015         ADC#    0,1,SNR      ;INTERRUPTED AT TRAP3+1 ?
   12022 165000         MOV     3,1          ;   YES
   12023  45004         STA     1,PC,2       ;SAVE ADDRESS
   12024    447         JMP     INT3B+1

   12025  10046 TRAP3: ISZ     46           ;ignore trapped NOP in Nova 3
   12026   2046         JMP     @46
```

```
;                        << SI = R90REXINTSSA;  BO = 1/A.REXM5.7033! >>
;;-------- ------------ ------------------------------------------
;;
;;     1.   SYSTEM INTERRUPT DISTRIBUTOR
;;
;;     PURPOSE:   1.   DETERMINES WHETHER TO SAVE REGISTERS IN NODE, OR IN
;;                     THE INTERRUPT STACK, AND SAVES THEM.
;;                2.   DETERMINES LOCATION OF APPROPRIATE INTERRUPT HANDLER
;;                     & CALLS IT, MASKING OUT FURTHER INTERRUPTS FROM THE
;;                     SAME DEVICE BEFORE DOING SO.
;;
;;     ENTRY:     ENTERED WHEN AN INTERRUPT OCCURS, WHICH SAVES PC IN
;;                LOCATION 0 AND VECTORS VIA LOCATION 1 TO INTS.
;;
;;     EXIT:      CALLS INTERRUPT HANDLER FOR INTERRUPTING DEVICE.   HANDLER
;;                WILL RETURN TO INTSR.
;;
;;     THEORY:  IRUPT (INTERRUPT FLAG IN INFO TABLE) INDICATES WHETHER NEXT
;;                INTERRUPT'S ACCUMULATORS SHOULD BE SAVED IN THE REGNANT
;;                TASK CONTROL NODE OR ON THE INTERRUPT STACK; IE.  WHEN
;;                IRUPT = -1, THE TCN'S FIRST 8 WORDS ARE AVAILABLE FOR
;;                SAVING RETURN INFORMATION; WHEN IRUPT <> -1, THE TCN'S
;;                FIRST 8 WORDS ALREADY CONTAIN THE TASK'S SAVED STATUS.
;;                INTERRUPTS ARE KEPT OFF FOR 73 MEMORY CYCLES (30 MICROSEC.
;;                ON MARK 5)
;;--- ------ - ------------------------- ---------------------------------

   12027   7727/ TRAPH: 77277   ;MK12 HAS FAULTED, BUT FAULTHANDLER NOT PRESENT

;  The next TWO locations MUST precede INTS for Mark 12

   12030      0          0                    ;PC SAVE
   12031  12027          TRAPH                ;VECTOR TO TRAP HANDLER

          12032 .LOC     M3L40-.*MK3+.        ;---------- END OF NON-MARK 3 CODE

   12032  12407 INTS:    ISZ      @.IRUPT     ;INT HANDLER OR PCHAR INTERRUPTED ?
   12033    413           JMP      INTS1       ;   YES, MUST STACK
   12034  52015           STA      2,@TASKQ    ;NO,  SAVE IN TOP TASK QUEUE NODE
   12035  30015           LDA      2,TASKQ
   12036  45001           STA      1,A1,2
   12037  25005           LDA      1,CPRI,2
   12040    420           JMP      INTS2

   12041    665 .IRUPT: INFO+IRUPT              ;INTERRUPT FLAG IN INFO TABLE
   12042      0 INTSP:          0               ;STACK PTR                  {SET UP BY SIR}
   12043      0 INTSB:          0               ;BEGINNING OF STACK         {SET UP BY SIR}
   12044      0 INTSE:          0               ;END OF INTERRUPT STACK     {SET UP BY SIR}
   12045      0 INTCM:          0               ;CURRENT MASK
```

```
;                       << SI = R90REXINTSSA;  BO = 1/A.REXM5.7033! >>

;  INTERRUPT HANDLER OR PCHAR WAS INTERRUPTED

 12046  52774 INTS1: STA     2,@INTSP ;SAVE STATUS IN INTERRUPT STACK
 12047  30773        LDA     2,INTSP
 12050  45001        STA     1,A1,2
 12051  24773        LDA     1,INTSE
 12052 146033        SLS     2,1      ;STACK OVERFLOW ?
 12053    543        JMP     STKOV    ;   YES
 12054  24030        LDA     1,C10    ;   NO
 12055 147000        ADD     2,1      ;PUSH DOWN THE STACK
 12056  44764        STA     1,INTSP
 12057  24766        LDA     1,INTCM  ;GET THE CURRENT MASK
 12060 127012 INTS2: ADD#    1,1,SZC  ;(A1)15 SAME AS CARRY ?
 12061 127240        ADDOR   1,1      ;   NO, MAKE IT SAME
 12062  45005        STA     1,CPRI,2 ;SAVE CARRY WITH MASK (OR CPRI)
 12063  41002        STA     0,A0,2   ;SAVE REGISTERS
 12064  55003        STA     3,A3,2
 12065  24040        LDA     1,SBA
 12066  45006        STA     1,SSBA,2
 12067  24041        LDA     1,DBA
 12070  45007        STA     1,SDBA,2
 12071  24000        LDA     1,0      ;SAVE ADDRESS
 12072  45004 INT3B: STA     1,PC,2   ;CHANGED TO "JMP INT3A" FOR NOVA 3
 12073  24524        LDA     1,KFLTO  ;SET TRAP AT LOCATION ZERO
 12074  44000        STA     1,0

        12075        M3L42=  .        ;---------- THE FOLLOWING IS MARK 5 CODE:
 12075  63777        SKPDZ   CPU      ;MK5:POWER FAIL INTERRUPT ?
 12076    555        JMP     INTP     ;MK5:   YES
 12077  61477        INTA    0        ;MK5:   NO
 12100  34100 INTS4: LDA     3,.INFO  ;MK5:
 12101 117000        ADD     0,3      ;MK5:GET INTERRUPT HANDLER POINTER
 12102  31530        LDA     2,INVT.,3;MK5:
 12103  25376        LDA     1,-2,2   ;MK5:HANDLER'S MASK BITS
 12104  34741        LDA     3,INTCM  ;MK5:"OR" WITH CURRENT MASK
 12105 174000        COM     3,3      ;MK5:
 12106 167400        AND     3,1      ;MK5:
 12107 166000        ADC     3,1      ;MK5
 12110  44735        STA     1,INTCM  ;MK5:STORE NEW MASK
 12111  66077        MSKO    1        ;MK5:AND OUTPUT IT
 12112  60177        INTEN            ;MK5:
 12113   5000        JSR     0,2      ;MK5:BRANCH TO INTERRUPT HANDLER

        12114        M5L42=  .        ;---------- THE FOLLOWING IS MARK 3 CODE:
        12114        .LOC M3L42-.*MK3+.
 12114 102000        ADC     0,0      ;MK3:MASK OUT ALL FURTHER INTERRUPTS
 12115  62077        MSKO    0        ;MK3:
 12116  30100        LDA     2,.INFO  ;MK3:
 12117   7131        JSR     @INVT.+1,2;MK3:BRANCH TO MUX INTERRUPT HANDLER

        12114        .LOC M5L42-.*MK5+.;---------- END OF HARDWARE-DEPENDENT CODE

                     ; ***** no code here *****
```

```
;                       << SI = R9OREXINTSSA;  BO = 1/A.REXM5.7033! >>
;;--- ------------------- ---------- ---- ------------------------------
;;
;;      2.   INTERRUPT SERVICE RETURN
;;
;;      PURPOSE:   -- SETS UP NEW "CURRENT MASK"
;;                 -- IF (STACK NOT EMPTY) POPS STACK & RETURNS
;;                 -- ELSE IF (CHARS TO BE PROCESSED) GOES TO PCHAR
;;                 -- ELSE IF (IDLING) CAUSES A SCHEDULER SCAN
;;
;;      ENTRY:   RETURN FROM INTERRUPT HANDLER, OR FROM "START"
;;
;;      EXIT:    THE HIGHEST PRIORITY TASK ON THE TASK QUEUE OR INTERRUPT STACK
;;               INTERRUPTS ARE DISABLED FOR 69 MEMORY CYCLES (28 USEC ON PT 4)
;;--- -------  ---------------- ----- ------ ----- ------------------

                        ; ***** no code here *****

          12114         M3L44=  .               ;---------- THE FOLLOWING IS MARK 5 CODE:
   12114  24727  INTSR:  LDA     1,INTSB         ;MK5:RETURN FROM HANDLER
   12115  22500          LDA     0,@.IMSK        ;MK5:GET INITIAL MASK
   12116  60277          INTDS                   ;MK5:
   12117  30723          LDA     2,INTSP         ;MK5:
   12120 146414          SEQ     2,1             ;MK5:IS INTERRUPT STACK EMPTY ?
   12121  21375          LDA 0,CPRI-ISTKF,2      ;MK5:  NO, GET PREVIOUS MASK
   12122  40723          STA     0,INTCM         ;MK5:STORE NEW CURRENT MASK
   12123  62077          MSKO    0               ;MK5:OUTPUT NEW CURRENT MASK
   12124  63777          SKPDZ   CPU             ;MK5:POWER FAIL ?
   12125    526          JMP     INTP            ;MK5:  YES
   12126  61477          INTA    0               ;MK5:NO
   12127 101014          SKZ     0,0             ;MK5:ANOTHER INTERRUPT ?
   12130    750          JMP     INTS4           ;MK5:  YES
          12131          M5L44=  .               ;---------- THE FOLLOWING IS MARK 3 CODE:
          12131          .LOC M3L44-.*MK3+.
   12131  60277          INTDS                   ;MK3:
   12132 102120          ADCZL   0,0             ;MK3:
   12133  62077          MSKO    0               ;MK3:MASK INTERRUPTS BACK IN AGAIN
   12134  63777          SKPDZ   CPU             ;MK3:POWER FAIL ?
   12135  63077          HALT                    ;MK3:  YES
   12136  30704          LDA     2,INTSP         ;MK3:  NO, RESTORE AND RETURN
   12137  24704          LDA     1,INTSB         ;MK3:
          12131          .LOC M5L44-.*MK5+.      ;---------- END OF HARDWARE-DEPENDENT CODE
   12131 146432          SGR     2,1             ;IS INTERRUPT STACK EMPTY ?
   12132    422          JMP     INSCO           ;  YES
   12133  24030          LDA     1,C10           ;  NO, "POP" THE STACK
   12134 132400          SUB     1,2
   12135  50705          STA     2,INTSP
   12136    432          JMP     INTRA

; PSEUDO-INTERRUPT

          12137 PSINTRPT:
   12137  60277          INTDS
   12140  52015          STA     2,@TASKQ
```

```
;                       << SI = R90REXINTSSA;  BO = 1/A.REXM5.7033! >>

    12141  30015          LDA     2,TASKQ
    12142  41002          STA     0,AO,2     ;SAVE CURRENT STATUS IN TASK CONTROL NODE
    12143  45001          STA     1,A1,2     ;   (EXCEPT CARRY)
    12144  55003          STA     3,A3,2
    12145  55004          STA     3,PC,2
    12146  20040          LDA     0,SBA
    12147  41006          STA     0,SSBA,2
    12150  20041          LDA     0,DBA
    12151  41007          STA     0,SDBA,2
    12152 102400          SUB     0,0
    12153  42666          STA     0,@.IRUPT

    12154  34103 INSCO:   IOCALL                ;STACK EMPTY, CHECK CHAR. QUEUE
    12155  25774          LDA     1,XCHQI,3
    12156  31775          LDA     2,XCHQO,3
    12157 132414          SEQ     1,2        ;CHARACTERS TO BE PROCESSED ?
    12160   2434          JMP     @.PCST     ;   YES, PROCESS CHARACTERS
    12161 102000 INTRR:    ADC     0,0        ;NO, RESET STACK FLAG << RETURN FROM PCHAR
    12162  42657          STA     0,@.IRUPT
    12163   2401          JMP     @.INTERCEPT

          12164 .INTERCEPT:                    ;THERE IS A POINTER TO HERE IN INFO AT .INT.
    12164 12165          INTEX                 ;MAY BE OVERLAID BY AN INTERCEPTOR

    12165  30015 INTEX:   LDA     2,TASKQ    ;<< ENTRY FROM DQUE, SLEP3
    12166  25010          LDA     1,TCBP,2   ;SET REGNANT TASK POINTER
    12167  44007          STA     1,RTP
    12170  25006 INTRA:    LDA     1,SSBA,2   ;<< FROM INTSR
    12171  44040          STA     1,SBA
    12172  25007          LDA     1,SDBA,2     .
    12173  44041          STA     1,DBA
    12174  25004          LDA     1,PC,2     ;RESTORE ADDRESS
    12175 125014          SKZ     1,1        ;IDLING ?
    12176    404          JMP     INTST      ;   NO, RESUME INTERRUPTED PROCESS
    12177  20005          LDA     0,RUP      ;   YES
    12200 101015          SNZ     0,0        ;ANY REGNANT USER ?
    12201  25011          LDA     1,TASK,2   ;   NO, FORCE A SCHEDULER SCAN
    12202  44411 INTST:    STA     1,INRPC
    12203  35003          LDA     3,A3,2     ;RESTORE REGISTERS
    12204  21005          LDA     0,CPRI,2
    12205 101100          MOVL    0,0        ;AND CARRY
    12206  21002          LDA     0,AO,2
    12207  25001          LDA     1,A1,2
    12210  31000          LDA     2,A2,2
    12211  60177          INTEN
    12212   2401          JMP     @.+1       ;RETURN TO INTERRUPTED PROGRAM
    12213      0 INRPC:    0                  ;RETURN PC


    12214  10365 .PCST:   PCSTT               ;*** REF *** (PCHR)
    12215    674 .IMSK:   INFO+MASK.


    12216  63377 STKOV:   DOCP    0,CPU       ;STACK OVERFLOW
                          ;keep together so that pressing CONT will produce TRAP 0
    12217    151 KFLTO:   JFLTO   ;allow for TRAP 0 = JMP 0 or JMP @0
```

```
;                       << SI = R9OREXINTSSA; BO = 1/A.REXM5.7033! >>
;;--- --- ----- ------------------------------ ----------------------
;;
;;     3.  UNKNOWN INTERRUPT HANDLER (NOT USED IN MARK 3)
;;
;;      PURPOSE:   TO DISMISS INTERRUPTS FROM DEVICES WHICH ARE NOT
;;                 GEN'D INTO SYSTEM.
;;
;;      ENTRY:     A0 <-- DEVICE CODE
;;--- --- --- ------- ------------------------- ------- ----------

          12220         M3L46= .          ;---------- THE FOLLOWING IS MARK 5 CODE ONLY:

    12220  65000 INTXA: DOA     1,0
    12221  60200 INTXB: NIOC    0

    12222 177777        177777            ;MASK OUT ALL INTERRUPTS
    12223   1400        JMP     0,3       ;NO POWER-FAIL RESTART
    12224   2475 INTX:  LDA     1,INTXB   ;DISMISS UNKNOWN INTERRUPT
    12225 107000        ADD     0,1
    12226  44407        STA     1,NIOX    ;BUILD A 'RESET' INSTR
    12227  24771        LDA     1,INTXA
    12230 107000        ADD     0,1       ;BUILD CONTROLLER 'STOP' INSTR (FOR S.I.)
    12231  44403        STA     1,DOAX
    12232    401        JMP     .+1       ;FORCE RELOAD OF MK12 PREFETCH
    12233 126400        SUB     1,1
    12234  65000 DOAX:  DOA     1,0
    12235  60200 NIOX:  NIOC    0
    12236    656        JMP     INTSR

          12237         .LOC M3L46-.*MK3+.;---------- END OF MARK-5-ONLY CODE

;;--- --- ---- -------- ------------------------------------------------
;;
;;     4.  INTERRUPT IGNORER
;;
;;     Theory:  During system initialization, when interrupts are not
;;     desired but subroutines which enable interrupts are employed,
;;     a vector to the "IGNORE INTERRUPTS" routine is put at
;;     location 1. When an interrupt occurs, the hardware disables further
;;     interrupts, saves the program counter in location 0, and executes
;;     a JMP @1 instruction.  This gets us to the "Interrupt ignorer", which
;;     in effect does a JMP @0 to return to the interrupted program.
;;--- ------  ------- ------------------------ ----------------------

    12237  40410 I.IGN: STA     0,I.IAO   ;save A0
    12240  20000        LDA     0,0       ;pick up the interrupted program counter
    12241  40405        STA     0,I.IPC   ;prepare for resume
    12242  20755        LDA     0,KFLTO   ;restore TRAP 0 @ 0 vector
    12243  40000        STA     0,0
    12244  20403        LDA     0,I.IAO   ;recover A0
    12245   2401        JMP     @.+1      ;return to interrupted program
    12246      0 I.IPC: 0

    12247      0 I.IAO: 0        ;save A0 here

                .EOT  ;REX "INTS" FOR "IRIS" RESIDENT EXECUTIVE
```

```
;                              << SI = R91REXPFRSSA; BO = 1/A.REXM5.7033! >>
;  13 SEP 86, RB.

;;--- -------  ------  -------  -- ----------------------------------------
;;
;;          R E X :   P O W E R    F A I L  --  R E S T A R T
;;
;;     MAJOR COMPONENTS:
;;
;;          1.   POWER FAIL ENTRY
;;          2.   RESTART ENTRY
;;          3.   START I-O SYSTEM
;;
;;     UPDATE RECORD:
;;          6-10-80 (LLL)  DOCUMENTED FOR READABILITY
;;                  (GAD)  REVISED DOCUMENTATION
;;          4- 1-81 (GAD)  INCORPORATED PATCHES
;;          2- 8-82 (RB.)  RESTORE R7 SOV INTERFACE
;;          4-11-82 (RB.)  ADD Mark 5 / Mark 3 PROVISIONS
;;          1-22-86 (rdc)  move RECOVER to SIR
;;        4 MAR 86  RB.    ELIMINATE AUTO-INCREMENT MECHANISM
;;       24 JUL 86  JAS
;;
;;     EXTERNAL REFERENCES:
;;          INTSR    (INTS)
;;          INTX     (INTS)
;;          INTS     (INTS)
;;          SCHED    (SCHD)
;;          INTSB    (INTS)
;;          INTSP    (INTS)
;;
;;     GLOBAL REFERENCES:
;;          SUBROUTINES:
;;             BINMULTIPLY
;;          DATA:
;;             INFO TABLE
;;             20
;;             TASKQ
;;
;;--- -------  ------  -------  -- ----------  ----------------------------
```

```
;                       << SI = R91REXPFRSSA; BO = 1/A.REXM5.7033! >>

;;--- ------ ------- ---------- ----- ------------------------------------
;;
;;     1.  POWER FAIL INTERRUPT
;;
;;     ENTRY:   THIS ROUTINE IS ENTERED FROM INTS WHEN A
;;              POWER FAIL INTERRUPT OCCURS.
;;
;;     EXIT:    NO EXIT (HALTS), UNTIL AUTO RESTART TAKES OVER, OR
;;              OPERATOR INTERVENES.
;;
;;     PURPOSE:  THIS ROUTINE SETS UP LOCATION 0 AND 1 WITH THE FOLLOWING:
;;                    0:   JMP @.+1
;;                    1:   A(PFRST)
;;
;;               THUS WHEN POWER COMES BACK UP AND THE MACHINE
;;               BEGINS EXECUTING AT LOCATION 0, IT WILL
;;               ENTER THE POWER FAIL RESTART ROUTINE.
;;-- ---------- ------- ------------------------ - ------------------------------

          12250        M3L50= .          ;---------- THE FOLLOWING CODE IS NOT USED IN MARK 3

   12250    675         INFO+PFRF.
   12251   2401         JMP     @.+1
   12252  12267         PFRST

          12253 INTP:  ;POWER FAIL INTERRUPT
   12253  62677         IORST             ;STOP EVERYTHING
   12254   4471         JSR     PFHDW     ;RE-ESTABLISH SYSTEM HARDWARE FUNCTION
   12255  24774         LDA     1,INTP-2  ;SET UP THE 'POWER UP' ROUTINE
   12256  44000         STA     1,0
   12257  24773         LDA     1,INTP-1
   12260  44001         STA     1,1
   12261  30100         LDA     2,.INFO
   12262  31076         LDA     2,LINKT,2
   12263  31000         LDA     2,PFDN,2
   12264 151014         SKZ     2,2       ;EXTERNAL POWER-FAIL INTERRUPT HANDLER ?
   12265   5000         JSR     0,2       ;   YES, USE IT
   12266  73077         DOC     2,CPU     ;HALT SYSTEM AND WAIT FOR 'POWER UP'


;  IF CPU WAS NOT IN 'AUTO' MODE, SET UP BY POSITION OF POWER SWITCH, THE
;  SYSTEM WILL COME UP HALTED AT 0.  PRESSING "CONT" WILL BRING UP THE SYSTEM.

;  IF THE POWER-FAIL INTERRUPT WAS A FALSE ALARM, IT WILL HALT HERE.
;  PRESSING "CONT" WILL THEN ALSO BRING UP THE SYSTEM (EXCEPT ON A MARK 12).

                       ; *** NO CODE HERE ***
```

```
;                    << SI = R91REXPFRSSA;  BO = 1/A.REXM5.7033! >>

;;--- ------  ---------------------------- ----------------------------
;;
;;  2.   POWER FAIL RESTART
;;
;;       ENTRY:    THIS ROUTINE IS ENTERED WHEN A POWER UP AUTO RESTART
;;                 OCCURS.  (A POWER DOWN MUST HAVE OCCURRED FIRST, SO
;;                 THAT LOCATIONS 0 AND 1 WILL BE SET UP FOR THIS
;;                 ENTRY POINT.)
;;
;;       EXIT:     WHEN ALL DEVICES HAVE BEEN INITIALIZED, BRANCHES TO
;;                 "START" TO START THE SYSTEM.
;;
;;       PURPOSE:  To re-initialize all I/O devices after a power failure.
;;
;;       Algorithm:
;;            1.  Initialize all disc drivers
;;            2.  Wait for all disc drives to be "Ready"
;;            3.  If time-out, HALT for operator intervention
;;            4.  Recalibrate all disc drives.
;;            5.  Use Power-Fail-Restart routines of all I/O drivers
;;            6.  Fall into "START" to start the I/O system
;;--- ------  ---------------------------- ----------------------------

                        ; *** NO CODE HERE ***
                                          ;---------- THE FOLLOWING CODE IS NOT USED IN MARK 3:

 12267   4456 PFRST: JSR     PFHDW     ;INITIALIZE SYSTEM HARDWARE
 12270   4513        JSR     IADRV     ;PREPARE THE DISK DRIVERS
 12271  20510        LDA     0,RTEM2   ;GET MAX RESTART DELAY
 12272 101015        SNZ     0,0       ;WILL ALL DRIVES RESTART ?
 12273    432        JMP     PFRSH     ;   NO. ASK FOR ASSISTANCE
 12274  31401        LDA     2,SPED.,3;   YES
 12275   6116        BINMULTIPLY       ;ADJUST DELAY FOR CPU SPEED
 12276  40503        STA     0,RTEM2   ;LS PORT
 12277 175400        INC     3,3
 12300  54502        STA     3,RTEM3   ;MS PORT
 12301   4533        JSR     SLUTP     ;SET UP POINTER TO THE LU TABLE
 12302  36475 PFRSL: LDA     3,@RTEMP  ;RECOVER A(LUFIX)
 12303  30474        LDA     2,RTEMP
 12304 174537        SGZ     3,3       ;LU ACTIVE ?
 12305    406        JMP     PFSLO     ;   NO
 12306  31001        LDA     2,.LVR.,2;YES, GET A(LUVAR)
 12307  63777        SKPDZ   CPU       ;ANOTHER POWER FAILURE ?
 12310    743        JMP     INTP      ;   YES
 12311   7774        JSR     @SLUR,3   ;IS THIS DRIVE ACCESSABLE YET ?
 12312    407        JMP     PFSL1     ;    NO, CONTINUE DELAY COUNTDOWN
 12313  10464 PFSLO: ISZ     RTEMP     ;DRIVE IS ACCESSABLE, ADVANCE TO NEXT LUT
 12314  10463        ISZ     RTEMP
 12315  10462        ISZ     RTEMP
 12316  14462        DSZ     RTEM1     ;ALL DRIVES READY ?
 12317    763        JMP     PFRSL     ;   NO
 12320    406        JMP     PFRSR     ;yes, recalibrate them
```

```
;                    << SI = R91REXPFRSSA;  BO = 1/A.REXM5.7033! >>

                                        ;---------- THE FOLLOWING CODE IS NOT USED IN MARK 3:

12321  14460 PFSL1: DSZ     RTEM2      ;END OF DELAY TIME DONE ?
12322    760        JMP     PFRSL      ;   NO
12323  14457        DSZ     RTEM3      ;   MAYBE
12324    756        JMP     PFRSL      ;   NO
12325  67077 PFRSH: DOC     1,CPU        ;YES, TIMED OUT!! * READY ALL DISCS, PRESS CONTINUE *
12326   4517 PFRSR: JSR     RECAL
12327  20415        LDA     0,.INV
12330  40417        STA     0,RTEMP
12331  10416 PFRS0: ISZ     RTEMP      ;RESTART ALL DEVICES 01 - 76
12332  32415        LDA     2,@RTEMP
12333  24410        LDA     1,.INTZ
12334 146415        SNE     2,1        ;DEVICE 77 ?
12335    414        JMP     PFDSC      ;   YES
12336  24404        LDA     1,.IN.X    ;   NO
12337 146414        SEQ     2,1        ;ANY DEVICE HERE ?
12340   5377        JSR     -1,2       ;   YES, USE ITS PFRST ROUTINE
12341    770        JMP     PFRS0

12342  12224 .IN.X: INTX
12343  12360 .INTZ: INTZ               ;Interrupt vector table entry for dev. 0 & 77
12344    730 .INV:  INFO+INVT.

       12345 PFHDW:                     ;INITIALIZE SYSTEM HARDWARE
12345  22403        LDA     0,@XNIOP
12346  60377        NIOP    CPU
12347   1400        JMP     0,3

12350    705 XNIOP: NIOP.+INFO

12351  32405 PFDSC: LDA     2,@XINDRVRFLG;GET "IN DISK DRIVER" STATE (A(TASK NODE))
12352  24405        LDA     1,XSRETRYCOUNT;A (TO START DISK TRANSFER WIA DISK DRIVER)
12353 151014        SKZ     2,2        ;IN DISK DRIVER WHEN POWER FAIL OCCURED?
12354  45004        STA     1,PC,2     ;   YES, SET PC TO RE-ISSUE DISK TRANSFER
12355    404        JMP     START      ;   NO, DON'T CHANGE PC


12356   5302 XINDRVRFLG:   INDRVRFLG;"IN DISK DRIVER" FLAG ZERO=NO, NON-ZERO=YES
12357   5304 XSRETRYCOUNT: SRETRYCOUNT;ADDR TO RE-ISSUE A CALL TO THE DISK DRIVER

12360  67377 INTZ:  DOCP    1,CPU      ;INTERRUPT NOT ACKNOWLEDGED !? (OR DEVICE 77 INTERRUPT)

       12361        .LOC M3L50-.*MK3+.;---------- END OF NON-MARK 3 CODE
```

```
;                            << SI = R91REXPFRSSA; BO = 1/A.REXM5.7033! >>
;;-----------------------------------------------------------------------------
;;
;;      3.   START I-O SYSTEM
;;
;;           ENTRY:    FROM SIR, OR POWER-FAIL RESTART ROUTINE.
;;
;;           EXIT:     BRANCHES TO "INTSR" IN THE INTERRUPT SERVICE
;;                     MODULE.
;;
;;           PURPOSE:   RESETS INTERRUPT SERVICE VECTOR TO POINT TO
;;                      "INTS", CLEARS POWER FAIL FLAG, & INTERRUPT STACK.
;;-----------------------------------------------------------------------------

   12361   60277 START: INTDS                ;START THE I/O SYSTEM
   12362   20413        LDA     0,IRSV
   12363   40001        STA     0,1          ;RESET INTERRUPT SERVICE VECTOR
   12364  102400        SUB     0,0
   12365   42411        STA     0,@I.INT ;INITIALIZE IRUPT FLAG
   12366   42662        STA     0,@INTP-3;CLEAR POWER FAIL FLAG
           12367 .LOC   .-MK3            ;OMIT PREVIOUS INSTRUCTION IN MARK 3
   12367   22404        LDA     0,@.INSB ;CLEAR INTERRUPT STACK
   12370   42404        STA     0,@.INSP
   12371    2401        JMP     @.+1         ;DO A RETURN FROM INTERRUPT (SIMULATED)
   12372   12114        INTSR

   12373   12043 .INSB: INTSB
   12374   12042 .INSP: INTSP
   12375   12032 IRSV:  INTS
   12376     665 I.INT: INFO+IRUPT           ;INTERRUPT BEING PROCESSED FLAG
```

```
;                        << SI = R91REXPFRSSA; BO = 1/A.REXM5.7033! >>
;;-----------------------------------------------------------------------------
;;
;;     SUBROUTINE:   INITIALIZE ALL DRIVERS
;;
;;     ENTRY:        A3 <-- RETURN ADDRESS
;;
;;     EXIT:         A3 <-- A(INFO TABLE)
;;                   RTEM2 = Max. restart delay for any drive, or
;;                         = 0 means requires manual intervention
;;
;;     PURPOSE:      FOR EACH DRIVER ON SYSTEM, CALLS THE INITIALIZE
;;                   ROUTINE WITHIN THE DRIVER
;;-----------------------------------------------------------------------------

   12377        0 RTEMP:0       ;A(LUT)
   12400        0 RTEM1:0       ;(MILU)
   12401        0 RTEM2:0       ;MAX RESTART DELAY/DELAY COUNT
   12402        0 RTEM3:0       ;RETURN

   12403   54777  IADRV:STA     3,RTEM3   ;INITIALIZE ALL DISC DRIVERS
   12404   20440        LDA     0,I.IGV   ;pointer to "Ignore interrupts"
   12405   40001        STA     0,1       ;set up benign interrupt service
   12406  102520        SUBZL   0,0       ;GENERATE A ONE
   12407   40772        STA     0,RTEM2   ;INIT TO ONE IN CASE OF MULTIPAL POWER FAILS
   12410    4424        JSR     SLUTP     ;SET UP LU TABLE POINTERS
   12411   36766  IADR2:LDA     3,@RTEMP  ;LUFIX POINTER
   12412   10765        ISZ     RTEMP     ;POINT RTEMP AT LUVAR
   12413  174537        SGZ     3,3       ;ACTIVE LOGICAL UNIT ?
   12414     412        JMP     IADRL     ;   NO
   12415   21764        LDA     0,PFRD,3  ;   YES, PICK UP RESTART DELAY
   12416   24763        LDA     1,RTEM2   ;PREV PFRD VALUE
   12417  125015        SNZ     1,1       ;SAVE LARGEST PFRD
   12420     404        JMP     IADR4
   12421  101014        SKZ     0,0
   12422  106433        SLE     0,1
   12423   40756        STA     0,RTEM2
   12424   20415  IADR4:LDA     0,.CRLA   ;give driver CRLA pointer
   12425    7773        JSR     @IDRV,3   ;INITIALIZE THE DRIVER
   12426   10751  IADRL:ISZ     RTEMP     ;ADVANCE LU + POINTER TO NEXT LUT
   12427   10750        ISZ     RTEMP
   12430   14750        DSZ     RTEM1     ;END OF TABLE ?
   12431     760        JMP     IADR2     ;   NO
   12432   34100  IADRR:LDA     3,.INFO   ;   YES
   12433    2747        JMP     @RTEM3

   12434   22407  SLUTP:LDA     0,@IMILU  ;SET UP LUT POINTERS
   12435   40743        STA     0,RTEM1   ;ALLOWED MAX # LU'S INSTALLED
   12436   22404        LDA     0,@ILUT
   12437   40740        STA     0,RTEMP   ;A(LUT)
   12440    1400        JMP     0,3       ;RETURN

   12441    6571  .CRLA:CRLAX             ;pntr to Convert Real to Logical Address
   12442     661  ILUT: INFO+.LUT.        ;pointer to LU table
   12443     602  IMILU:INFO+MILU.        ;max.no.installed LU's
   12444   12237  I.IGV:I.IGN             ;VECTOR TO "IGNORE INTERRUPTS"
```

PDC 17-FEB-8?

LDA 2,@RTEMP; GIVE DR POINTER T

```
;                        << SI = R91REXPFRSSA; BO = 1/A.REXM5.7033! >>
;;--- ------- ------ ---------- ------- ----------------------
;;
;;    SUBROUTINE:  RECALIBRATE ALL DISCS
;;
;;     ENTRY:      A3 <-- RETURN ADDRESS
;;
;;     EXIT:       A3 <-- A(INFO TABLE)
;;
;;     PURPOSE:    CALLS RECALIBRATE ROUTINE IN THE APPROPRIATE SOV
;;                 FOR EACH ACTIVE UNIT.
;;
;;--- ------   ------ ------- ------ ---------------------------------

   12445  54735 RECAL: STA     3,RTEM3   ;RECALIBRATE ALL DISCS
   12446   4766         JSR     SLUIP     ;SET UP LUT PTRS
   12447  36730 RCAL2: LDA     3,@RTEMP  ;LUFIX POINTER
   12450  10727         ISZ     RTEMP
   12451  32726         LDA     2,@RTEMP  ;LUVAR POINTER
   12452  10725         ISZ     RTEMP
   12453  10724         ISZ     RTEMP
   12454 126000         ADC     1,1       ;set RECAL code for SEEK rtne.
   12455 174536         SZN     3,3       ;ACTIVE UNIT ?
   12456   7777         JSR     @SEEK,3   ;   yes, recalibrate it
   12457  14721         DSZ     RTEM1     ;DONE ?
   12460    767         JMP     RCAL2     ;    NO
   12461    751         JMP     IADRR     ;yes, return with A3 = .INFO

                .EOT   ;REX "PFRS" FOR "IRIS" RESIDENT EXECUTIVE R9.0
```

```
;                        << SI = R91REXFALTSA; BO = 1/A.REXM5.7033! >>
; 29 JUN 86, RB.

;;-------------------------------------------------------------------------
;;
;;                R E X :   F A U L T   H A N D L I N G
;;
;;       MAJOR COMPONENTS:
;;
;;            1.   TRAPFAULT ROUTINE
;;            2.   FAULT START TASK
;;            3.   FAULT WRITE TASK
;;
;;       NOTE:   ALSO ASSOCIATED WITH THIS MODULE IS THE FAULT DISCSUB,
;;               AND THE FAULT PRINT PROCESSOR
;;
;;       GLOBALS:    TASKQ
;;                   INFO TABLE
;;
;;       EXTERNALS:  GTNOD (MEMM)
;;
;;       UPDATE RECORD:
;;                   10-15-80   (GAD)  CLARIFY COMMENTS & CORRECT
;;                    1-13-81   (GAD)  CHANGED SOURCE NAME
;;                    4-01-81   (GAD)  INCORPORATED PATCHES
;;                    8-13-81   (GAD)  UPGRADED FAULT WRITE TASK
;;                   11 NOV81   (RMS) ADDED CODE TO CLEAN UP BUFFER POOL
;;                    1-25-82   (RB.)  NEW DATAPUMP, NODE DEFINITIONS
;;-------------------------------------------------------------------------

;                              OVERVIEW:
;
; TRAPFAULT OCCURS IN SOME TASK (NOT LEGAL IN INTERRUPT HANDLERS),
;     AND JSR'S TO TFALT,
; OR ELSE  JMP 0  OR  JMP @0  OCCURS WHICH IS VECTORED TO FALTO
; TFALT OR FALTO OPERATE WITH INTERRUPTS DISABLED.  THEY GET A FREENODE
;     (FAULT DATA NODE) AND SAVE ACCUMULATORS, TIME, ETC.  IN IT, THEN
;     QUEUE THE FAULT START TASK AT PRIORITY 77777 (MAX).  INTERRUPTS
;     ARE RE-ENABLED AFTER COMPLETION OF THE QUEUE.
; FSTSK EXTRACTS THE FAULTING TCN (TASK CONTROL NODE) AND ANY ADDITIONAL
;     TCN'S AUXLINKED TO IT -- EXCEPT IF TCN IS THE SCHEDULER IT CANNOT
;     BE XQUEUED, SO IT IS COPIED TO A FREENODE INSTEAD.  IT AUXLINKS
;     THE WHOLE CHAIN TO THE ORIGINAL FAULT DATA NODE.  IT CLEARS THE
;     DISCSUB STACK.  THEN IT QUEUES THE FAULT WRITE TASK AT PRIORITY 2
;     (JUST ABOVE SCHEDULER).
; FWTSK USES DATAPUMP TO READ IN THE FAULT DISCSUB AND JUMPS TO IT.
; FAULT DISCSUB WRITES THE DATA FROM THE FAULT DATA NODE AND ITS AUXLINKS
;     TO THE FAULT HISTORY FILE ON DISC.  IF THE FAULT TCN CHAIN ENDS AT
;     THE SCHEDULER (IE. THERE WAS A REGNANT USER), IT SETS UP THE
;     REGNANT USER'S PROCESSOR TO BE FAULTPRINT
; FAULTPRINT PRINTS THE FAULT MESSAGE ON THE USER'S TERMINAL, THEN EXITS
;     TO SCOPE.
```

```
;                        << SI = R91REXFALTSA;  BO = 1/A.REXM5.7033! >>

;;-------  ---------------- - ----- ----- ----------------------------
;;
;;      1.   TRAPFAULT ROUTINE
;;
;;      PURPOSE:   THIS ROUTINE ACQUIRES A NODE AND SAVES THE REGISTER
;;                 CONTENTS AT THE TIME OF THE FAULT IN IT, AS WELL
;;                 AS THE EXACT TIME THE TRAP OCCURRED.  THEN IT QUEUES UP THE
;;                 FAULT START TASK AT TOP PRIORITY AND FALLS INTO IT.
;;
;;      ENTRY:     A3 <-- A(TRAPFAI T+1) POINTS TO FAULT-TYPE WORD.
;;                 A0 <-- CONTENTS OF A0 AT TIME OF TRAP
;;                 A1 <-- CONTENTS OF A1 AT TIME OF TRAP
;;                 A2 <-- CONTENTS OF A2 AT TIME OF TRAP.
;;
;;      EXIT:      QUEUES THE FAULT-START TASK WITH A2 --> FAULT DATA NODE.
;;-- ---------- ---------- ---- ----------------------------------------

;  CONTENT OF FAULT DATA NODE:

;       F.A0=   A0          ;CONTENTS OF A0 AT TIME OF TRAP
;       F.A1=   A1          ;CONTENTS OF A1 AT TIME OF TRAP
;       F.A2=   A2          ;CONTENTS OF A2 AT TIME OF TRAP
;       F.A3=   PC          ;CONTENTS OF A3 AT TIME OF TRAP, OR TRAP LOC+1
;       F.CA=   CPRI        ;CARRY BIT
;       F.FL=   A3          ;"TRAP AT 0" FLAG
;       F.TW=   27          ;FAULT TYPE WORD
;       F.HR=   32          ;HRS AT TIME OF FAULT
;       F.TS=   33          ;TENTH-SECONDS AT TIME OF FAULT
;       F.SA=   11          ;DISCSUB STARTING ADDRESS
;       F.DA=   12          ;DISC ADDRESS OF CURRENT DISCSUB
;       F.KW=   13          ;DISCSUB KEYWORD - NUMBER + FLAGS
;       TCBP                ;FAULTING TASK'S TASK CONTROL BLOCK POINTER
;       AUXL                ;LINK TO FAULTING TASK CONTROL NODE(S)


  12462       0 FLTFL: 0        ;  -1 ==> AC3 DESTROYED BY TRAP CALL
                                ;   0 ==> TRAP 0 @ 0
  12463       0 FLTA3: 0        ;TRAP'S A3

  12464   60277 FALTO: INTDS              ;TRAP AT LOCATION ZERO:
  12465   54776         STA     3,FLTA3   ;SAVE CONTENTS OF A3 AT TIME OF TRAP
  12466  176460         SUBC    3,3       ;SET FLAG FOR TRAP AT 0, WITHOUT CHANGING CARRY
  12467     404         JMP     TFLT2     ;ENTER TRAP ROUTINE

  12470   60277 TFALT: INTDS              ;NORMAL TRAPFAULT ENTRY
  12471   54772         STA     3,FLTA3   ;SET UP A(TRAPWORD)
  12472  176000         ADC     3,3       ;SET FLAG TO INDICATE A3 WAS DESTROYED
  12473   54767 TFLT2: STA     3,FLTFL   ;<<  ENTRY FROM FALTO
```

```
;                       << SI = R91REXFALTSA;  BO = 1/A.REXM5.7033! >>

    12474   6430        JSR     @GTND1      ;GET A WORK NODE (DOES NOT INTEN)
                                            ;PRESERVES CARRY, SAVES A2 IN A2 CELL OF NODE
    12475  41002        STA     0,F.A0,2    ;CONTENTS OF A0 AT TIME OF TRAP
    12476  45001        STA     1,F.A1,2    ;CONTENTS OF A1 AT TIME OF TRAP
    12477  20763        LDA     0,FLTFL
    12500  41003        STA     0,F.FL,2    ;"TRAP AT ZERO" FLAG
    12501 102560        SUBCL   0,0
    12502  41005        STA     0,F.CA,2    ;CARRY BIT AT TIME OF TRAP
    12503  34760        LDA     3,FLTA3
    12504  55004        STA     3,F.A3,2    ;CONTENTS OF A3 OR TRAP LOC + 1
    12505  21400        LDA     0,0,3
    12506  41027        STA     0,F.TW,2    ;TRAPWORD FOLLOWING TRAPFAULT CALL
    12507  34100        LDA     3,.INFO
    12510  21440        LDA     0,HRS.,3
    12511  41032        STA     0,F.HR,2    ;HRS.  AT TIME OF TRAP
    12512  21441        LDA     0,TSC.,3
    12513  41033        STA     0,F.TS,2    ;TENTH-SECS.  AT TIME OF TRAP
    12514  20007        LDA     0,RTP
    12515  41010        STA     0,TCBP,2    ;FAULTING TASK'S TCB/PCB POINTER
    12516  20015        LDA     0,TASKQ
    12517  41035        STA     0,AUXL,2    ;FAULTING TASK'S TCN
    12520   6433        JSR     @XQLINK     ;QLINK THE FAULT-START TASK
    12521  12554 .FSTS: FSTSK               ;FSTSK IS WHERE WE RETURN FROM QLINK
    12522 177777        -1      ;USE REGNANT TCB
    12523  77777        77777   ;TOP PRIORITY
    12524   2734 GTND1: GTNOD    ;A3 NOT USED

;  DOUBLE TRAP:

    12525  21025 FLTD2: LDA     0,N.T1,2    ;FAULT DATA NODE OF FIRST TRAP
    12526  30015 FLTDB: LDA     2,TASKQ
    12527  41027        STA     0,N.T3,2    ;N.T3 = FIRST TRAP
    12530  62677 FLTDT: IORST               ;  ** DOUBLE TRAP **
    12531  30015        LDA     2,TASKQ
    12532  31025        LDA     2,N.T1,2    ;LATEST TRAP'S DATA NODE
    12533   4410        JSR     FLTLD       ;LOAD REG'S FROM FAULT DATA NODE
    12534  77077        77077               ;SPECIAL HALT
    12535  30015        LDA     2,TASKQ     ;GET POINTER TO CURRENT TASK'S
    12536  31027        LDA     2,N.T3,2    ;FIRST TRAP'S DATA NODE
    12537   4404        JSR     FLTLD       ;LOAD REGS FROM OTHER TRAP
    12540  77077        77077               ;SPECIAL HALT
    12541    767        JMP     FLTDT       ;REPEAT

    12542      0        0
    12543  54777 FLTLD: STA     3,FLTLD-1;LOAD REG'S AND CARRY FROM DATA NODE
    12544  21005        LDA     0,F.CA,2
    12545 101100        MOVL    0,0         ;CARRY
    12546  21002        LDA     0,F.A0,2    ;AC0
    12547  25001        LDA     1,F.A1,2    ;AC1
    12550  35004        LDA     3,F.A3,2    ;FAULT ADDR+1
    12551  31000        LDA     2,F.A2,2    ;AC2
    12552   2770        JMP     @FLTLD-1

    12553   2475 XQLINK:        QLINK
```

```
;                        << SI = R91REXFALTSA;  BO = 1/A.REXM5.7033! >>

;;---  ---- ---- ---------------------- -- ----------------------------
;;
;;    2.   FAULT-START TASK
;;
;;          ENTRY:   FROM THE TFALT ROUTINE AS A RESULT OF A 'TRAPFAULT' CALL
;;                   OR A JMP TO LOC ZERO IN CORE.   A2 POINTS TO A NODE
;;                   WITH ALL THE AVAILABLE DATA FOR THE FAULT START TASK TO
;;                   USE.
;;
;;           EXIT:   TO SYSTEM VIA A DQUEUE.  THE FAULT WRITE TASK IS QUEUED AT
;;                   A PRIORITY OF 2 TO DO THE NECESSARY HISTORY FILE
;;                   MANIPULATIONS ETC.  ETC.
;;
;;        PURPOSE:   DETERMINE IF FAULT OCCURRED IN A PROCESSOR OR TASK OR
;;                   DISCSUB AND DO INITIAL CLEANUP AS APPROPRIATE.  COLLECT
;;                   THE FAULTING TCN AND ANY TCN'S AUXLINKED TO IT,
;;                   AND AUXLINK THEM ALL TO THE FAULT DATA NODE.
;;
;;--- ------ ---------------------- ----- -------- ----------------------

                                                  ;NO CONFLICT FLAGS NEEDED BEC.  CALLED BY QLINK
12554   34015 FSTSK: LDA     3,TASKQ    ;FAULT-START
12555   51425         STA     2,N.T1,3   ;SAVE A(FAULT DATA NODE)
12556   51426         STA     2,N.T2,3   ;ALSO SAVE AS LAST AUXL JUST IN CASE
12557   31035         LDA     2,AUXL,2   ;FAULTING TCN
12560   21011 FSTS2: LDA     0,TASK,2   ;CHECK FOR DOUBLE TRAP
12561   24740         LDA     1,.FSTS
12562  106415         SNE     0,1        ;IS NEXT TASK FAULT-START ?
12563     742         JMP     FLTD2      ;   YES, ** DOUBLE TRAP **
12564   24473         LDA     1,.FWIS
12565  106415         SNE     0,1        ;IS NEXT TASK FAULT-WRITE ?
12566     737         JMP     FLTD2      ;   YES, ** DOUBLE TRAP **
12567   24473         LDA     1,.FLSC
12570  106415         SNE     0,1        ;DID A PROCESSOR FAULT ?
12571     411         JMP     FSTS3      ;   YES (CANNOT EXTRACT SCHEDULER NODE)
12572    6101         CALL               ;   NO, EXTRACT FAULTING NODE
12573  100026         XQUEUE
12574   34015         LDA     3,TASKQ
12575   51426         STA     2,N.T2,3   ;SAVE A(LAST FAULT TCN)
12576   31035         LDA     2,AUXL,2
12577  151014         SKZ     2,2        ;IS THE TCN WE JUST XQ'D LINKED FURTHER ?
12600     760         JMP     FSTS2      ;   YES, XQUEUE THAT ONE TOO
12601     423         JMP     FSTS4      ;   NO
```

```
;                       << SI = R91REXFALTSA;  BO = 1/A.REXM5.7033! >>

12602  30005 FSTS3: LDA     2,RUP      ;CHECK FOR TRAP IN "FAULTPRINT"
12603 151015        SNZ     2,2        ;ANYONE REGNANT ?
12604    405        JMP     FST3A      ;   NO
12605  35025        LDA     3,DFT.,2   ;YES
12606  21776        LDA     0,FRR+CHM1,3
12607 111001        MOV     0,2,SZR    ;SUMMARY PRINT IN PROGRESS ?
12610    716        JMP     FLTDB      ;   YES, ** DOUBLE TRAP **
12611   6107 FST3A: FREENODE           ;NO, OBTAIN A NODE
12612  34015        LDA     3,TASKQ
12613  35426        LDA     3,N.T2,3   ;FAULT DATA NODE OR LAST TCN
12614  51435        STA     2,AUXL,3   ;AUXLINK THE NEW NODE TO IT
12615  34100        LDA     3,.INFO
12616  21456        LDA     0,..STN.,3;ADDRESS OF SCHED TCN
12617  24445        LDA     1,C33      ;NUMBER OF WORDS TO MOVE (PRESERVE LAST 4)
12620 107000        ADD     0,1
12621   6101        CALL               ;COPY FAULTING TCN TO THE NEW NODE
12622 100015        MOVEWORDS
12623  34015        LDA     3,TASKQ
12624  31425 FSTS4: LDA     2,N.T1,3   ;A(FAULT DATA NODE)
12625 102400        SUB     0,0        ;PREPARE TO CLEAN UP DISCSUB STACK
12626  41435        STA     0,AUXL,3   ;SEVER AUXL FOR TROUBLE-FREE DQUEUE
12627  41011        STA     0,F.SA,2
12630  41012        STA     0,F.DA,2
12631  34100        LDA     3,.INFO
12632  35457        LDA     3,..STK,3
12633  21400        LDA     0,XSTK,3   ;.STK
12634  25401        LDA     1,XSTP,3   ;STKP
12635 106415        SNE     0,1        ;TRAP IN A DISCSUB ?
12636    420        JMP     FSTS5      ;   NO
12637  41401        STA     0,XSTP,3   ;RESET STACK
12640  25406        LDA     1,XCDN,3
12641  45013        STA     1,F.KW,2   ;    KEYWORD
12642  20064        LDA     0,C777
12643 123400        AND     1,0        ;GET DSUB NUMBER FROM KEYWORD
12644  36422        LDA     3,@..SAT   ;POINTER TO SAT
12645 117000        ADD     0,3
12646  25400        LDA     1,0,3      ;SA
12647  45011        STA     1,F.SA,2   ;SAVE SA IN NODE
12650  36415        LDA     3,@..DAT   ;POINTER TO DAT
12651 117000        ADD     0,3
12652  21400        LDA     0,0,3      ;DISC ADDRESS
12653 101113        SSN     0,0        ;MEMORY RESIDENT (WILL BE COMPLEMENTED IF MEM RES)
12654 100000        COM     0,0        ;   YES, GET TRUE DISC ADDRESS
12655  41012        STA     0,F.DA,2   ;SAVE DA
12656   6104 FSTS5: QUEUE              ;QUEUE FAULT-WRITE TASK
12657  12670 .FWTS: FWTSK              ;START ADDR
12660 177777        -1                 ;USE REGNANT TASK'S TCBP
12661      2        QP.FW              ;PRIORITY
12662   1447 .FLSC: SCHED              ;A3 IS NOT USED
12663   6105        DQUEUE

12664     33 C33:   33
12665   5765 ..DAT: .DAT    ;POINTER TO DAT
12666   5766 ..SAT: .SAT    ;POINTER TO SAT
```

RDC 29-JUN-87

SSP

```
;                        << SI = R91REXFALTSA;  BO = 1/A.REXM5.7033! >>

;;-----------  ------------------ -- ------ --- --------------------------
;;
;;    3.   FAULT-WRITE-TASK
;;
;;    PURPOSE:   TO CALL FAULT DISCSUB WHICH WRITES OUT FAULT INFORMATION
;;               TO THE FAULT HISTORY FILE.
;;
;;    ENTRY:     A2 = A(LOOSE-NODE WHICH CONTAINS FAULT DATA)
;;
;;    EXIT:      DISCSUB DEQUEUES SELF
;;---- --------  -------------------------- ------------------------------

12667       0          0                    ;CONFLICT FLAGS (NO CONFLICTS)
12670   34015 FWTSK:LDA     3,TASKQ
12671   51425          STA     2,N.T1,3 ;SAVE FAULT DATA NODE PNTR FOR DISCSUB
12672    6101          CALL
12673  100017          XFIXBUFFERS      ;DETACH FIXED BUFFERS FROM POOL
12674  102400          SUB     0,0
12675   24411          LDA     1,FRDA   ;RDA OF FAULT DISCSUB
12676   30013          LDA     2,.SSA
12677    6110          DATAPUMP         ;READ DISCSUB INTO SSA
12700      20          D.READ           ;WITHOUT GOING THROUGH POOL
12701  100000          @0
12702    5003          JSR     3,2      ;ENTER FAULT DISCSUB (DEQUEUES SELF)
                       ;
                       ;  *** RDA TABLE MUST BE HERE:
                       ;
12703       0 RDATB:0       ;CURRENT RDA OF HISTORY BLOCK (SET UP BY SIR)
12704       0          0       ;LAST RDA OF HISTORY FILE (SET UP BY SIR)
12705       0          0       ;RDA OF FAULT PRINT PROCESSOR (SET UP BY SIR)
12706       0 FRDA:  0       ;RDA OF FAULT DISCSUB (SET UP BY SIR)


12707   13671 PATSP:.BLK    BPS-3400-. ;BEGIN PATCH SPACE OVERLAP TEST

        26400          .LOC    BPS-2200-1400
26400   77377          77377   ;FIRST  BLOCK FOR LU/Os LUVAR & LUFIX

        27000          .LOC    BPS-2200-1000
27000   77377          77377   ;SECOND BLOCK FOR LU/Os LUVAR & LUFIX

        27400          .LOC    BPS-2200-400
27400   77377          77377   ;THIRD  BLOCK FOR LU/Os LUVAR & LUFIX
```

```
;                       << SI = R91REXFALTSA; BO = 1/A.REXM5.7033! >>
; LINKAGE TABLE TO ENABLE SYSL TO PUT REX ADDRESSES INTO SIR AND PLOAD

          52000         .LOC    RLINK

   52000  12224          INTX    ;P.ITX     ;---------- MARK 5 ONLY
          52001 .LOC     .-MK3
   52001  11765          PZIH    ;P.ITX     ;---------- MARK 3 ONLY
          52001 .LOC     .-MK5
   52001  10044          .CHQ    ;P.CHQ
   52002  10045          .CHQE   ;P.CQE
   52003  10046          CHQIP   ;P.CQI
   52004  10047          CHQOP   ;P.CQO
   52005  11766          TTSND   ;P.SND     ;---------- MARK 5 ONLY
          52006 .LOC     .-MK3
   52006  12005          PZSND   ;P.SND     ;---------- MARK 3 ONLY
          52006 .LOC     .-MK5
   52006  12042          INTSP   ;P.INP
   52007  12043          INTSB   ;P.INB
   52010  12044          INTSE   ;P.INE
   52011  12045          INTCM   ;P.INM
   52012   4313          BPMRU   ;P.MRU
   52013   4320          BPLRU   ;P.LRU
   52014   5477          BUF1    ;P.BF1
   52015   2664          FREE    ;P.FRE
   52016  12403          IADRV   ;P.IAD
   52017  12445          RECAL   ;P.REC
   52020  12361          START   ;P.STA
   52021  12470          TFALT   ;P.TFA
   52022  12464          FALTO   ;P.TFO
   52023    723          JMP  INT3A-INT3B,1;P.I3J
   52024  12072          INT3B   ;P.I3B
          52025 .LOC     .-MK3-MK3          ;PREV.  2 POINTERS (FOR NOVA 3) NOT USED IN MARK 3
   52025   1623          .SIGB   ;P.SIG
   52026  12703          RDATB   ;P.RDA
   52027   2756          MINOD   ;P.MND
   52030 177777          -1      ;TERMINATOR


                .EOT    ;REX "FALT" FOR "IRIS" RESIDENT EXECUTIVE
```

. END

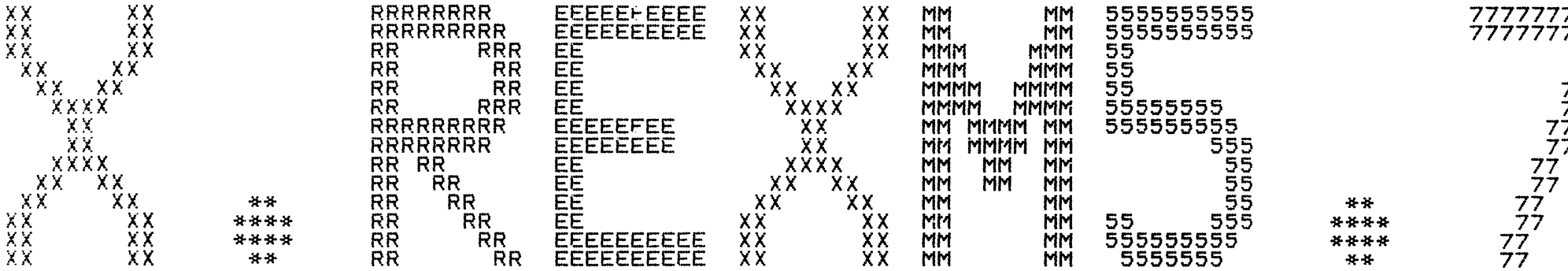
A2TEM	1033	A2TOC	4171	ABNMS	1360	ABPS4	1314
ACIB	6655	ACLBY	6675	ACRBY	6651	ACSB	6663
ACSB4	6667	ACSBT	6712	AF	4001	ALCLR	7432
ATKEY	6270	AWAKX	2470	BDIV	7232	BDVL1	7235
BEBY	7205	BF	4010	BINDI	6115	BINMU	6116
BMGTW	7210	BMSTW	7211	BM1MF	7207	BMUL	7212
BPI	16	BPLRU	4320	BPMRU	4313	BPTRE	4503
BRMSK	1315	BSACF	75	BUF1	5477	BUFFE	4255
BUMP2	1411	BUMP5	1443	BUMPU	6117	BUSYE	5136
C100	51	C1000	67	C1004	11034	C11	31
C13	33	C14	34	C140	10614	C15	35
C160	174	C163	175	C166	176	C17	37
C171	177	C177	52	C1777	70	C2	2
C200	53	C2000	71	C204	10355	C205	54
C233	10356	C237	10127	C240	56	C244	57
C271	61	C3	3	C300	62	C33	12664
C34	10703	C37	43	C377	64	C4	24
C400	65	C4000	72	C40K	7356	C5	25
C600	100	C7	27	C77	50	C774C	22
CALD1	5602	CALD2	5624	CALD3	5644	CALD7	5674
CALLD	5576	CALLF	5736	CALLJ	5676	CALLL	5732
CALLR	5705	CALLS	5542	CALR1	5710	CATMP	5567
CC3	6355	CC400	10520	CCLUD	4125	CDSBN	5771
CH10K	6173	CH20K	6175	CH34K	6176	CH4K	6174
CHAN0	6105	CHAN1	6110	CHAN2	6113	CHAN3	6135
CHAN6	6247	CHAN7	6205	CHAN8	6165	CHANA	6267
CHANJ	6152	CHANN	6106	CHANP	6241	CHANW	6161
CHANY	6103	CHANZ	6101	CHCHX	7461	CHCNP	6204
CHKP1	7525	CHKP2	7532	CHKPB	7501	CHN2A	6130
CHQDP	10047	CHRDC	6200	CHRDM	6203	CHRDP	6202
CKBS2	4205	CKBSA	4175	CKHEA	4676	CKLCM	5201
CLBA0	3455	CLBA1	3456	CLBA3	3457	CLBS2	3451
CLRAX	6553	CM400	23	CMVLN	7570	CNTBL	4770
COPY2	4662	COPYC	4377	COPYF	4633	COPYT	4632
CPNRX	6504	CRLAX	6571	CRLNT	6622	CRLPL	6623
CRPNX	6467	CRST	6427	CXOFF	11033	CXON	10345
C.FLU	4404	C.INF	4523	C.LCG	4625	C.NL	5734
C.UNA	4254	C.VIR	4522	D2RW	5251	D2WRI	5244
DAC	164	DAERR	5134	DAS	165	DAS2	4437
DASEA	4427	DASLO	4442	DASNF	4455	DATAE	5450
DAY	4	DAYC1	1654	DAYC2	1663	DAYCK	1653
DBA	41	DBC	172	DBS	173	DECIM	6120
DECX5	7266	DEFRQ	1520	DEN	5772	DEST	7766
DFCW1	5115	DFL2	6401	DFLOO	6371	DFNEX	6422
DFRTN	6347	DFTCA	34106	DFT.N	6354	DFWCW	5125
DIRTY	1000	DISKR	5156	DISKW	5140	DKERR	5130
DMCAL	34110	DOAX	12234	DOLCM	5221	DONTM	5340
DPDRE	5276	DPDWR	5300	DPERO	3757	DPER1	3754
DPERS	4007	DPEX1	4310	DPEX2	4402	DPEX3	4533
DPEXL	4021	DPEXR	4016	DPFIR	3460	DPFKW	4006
DPQA3	3357	DPRET	3741	DPRRE	5272	DPSNF	4471
DPSTO	4467	DPTAB	3510	DPTAS	3536	DPTRA	3774
DPWRE	5274	DQUE	2562	DQUET	2572	DQUEU	6105
DSKEL	5423	DSKER	5422	DSKRE	5307	DSKRW	5165
DW	4	D.BC	26	D.CA	20	D.CW	22
D.HP	24	D.IP	13	D.LF	31	D.LU	16
D.PE	21	D.T1	14	D.T2	15	D.TB	27
D.TO	33	D.TS	30	D.TY	12	D.WR	25

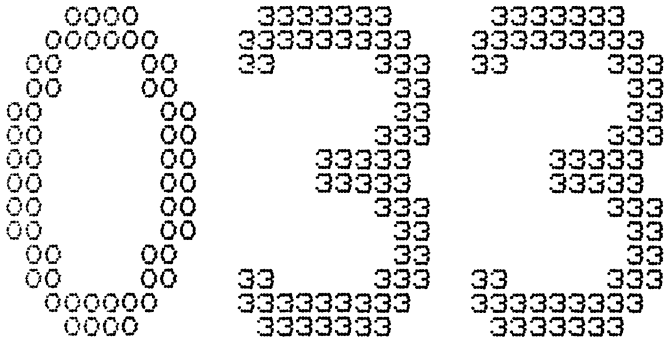
ACBY	6673		
ACSB2	6701		
ASCON	1321		
BDVL2	7246		
BMCNT	7206		
BMULP	7217		
BRKP	1361		
BUMP	1370		
C10	30		
C12	32		
C16	36		
C170K	21		
C20	42		
C215	55		
C260	60		
C334	63		
C40	44		
C6	26		
C777	66		
CALL	6101		
CALLN	6011		
CBSAX	3431		
CDVIT	7571		
CHA8A	6170		
CHAN4	6137		
CHANE	6243		
CHANX	6061		
CHKFF	4405		
CHQIP	10046		
CHRDT	6201		
CKLOO	4271		
CLINK	2557		
COLON	10751		
CPELU	4133		
CRLRA	6624		
C.DIR	4600		
C.PRI	4314		
DA	160		
DAS2L	4441		
DATAP	6110		
DB	166		
DECX	7261		
DFCOU	6356		
DFRCW	5124		
DF.AO	6350		
DMAPC	5303		
DPCER	3533		
DPERR	3756		
DPEXE	4022		
DPLAS	3505		
DPSPE	3746		
DPUMX	3351		
DR	10		
DUMPR	5744		
D.DA	17		
D.LV	32		
D.TB.	4755		
D.XP	23		

ECO	20000	EPAUS	2215	ERRF	76	ESCF	73	ETSF	74
ETSKS	2302	EXITX	1324	FALTO	12464	FF	6000	FFCW	5123
FFCW1	5073	FFSCA	4415	FINDL	6123	FIX	6121	FLAGC	6102
FLAGX	7664	FLCNT	6551	FLGX1	7702	FLGX2	7710	FLOAT	6122
FLRTN	6552	FLTA3	12463	FLTD2	12525	FLTDB	12526	FLTDT	12530
FLTFL	12462	FLTLD	12543	FLUO	6524	FLU1	6535	FLU2	6544
FLUT	6517	FLWMA	11450	FR	4200	FRDA	12706	FREE	2664
FREEG	2675	FREEN	6107	FREEX	2665	FRERT	2662	FREX	2700
FREXG	2705	FRNO2	2710	FRNO3	2726	FRNOD	2707	FST3A	12611
FSTS2	12560	FSTS3	12602	FSTS4	12624	FSTS5	12656	FSTSK	12554
FW	4100	FWTSK	12670	F.AO	2	F.A1	1	F.A2	0
F.A3	4	F.CA	5	F.DA	12	F.FL	3	F.HR	32
F.KW	13	F.SA	11	F.TS	33	F.TW	27	G2	4000
GATHE	5015	GB	1000	GCAPP	5043	GCSCA	5020	GCSTA	5025
GETBY	6124	GETCF	6606	GETLR	4325	GETN2	4351	GETN3	4361
GETNE	4337	GETNL	4340	GOTOT	4042	GOTOU	4043	GOTOX	4041
GOTOY	4045	GOTOZ	4050	GOTO.	4242	GP	2000	GROUP	3666
GTCOP	4371	GTDSU	6032	GTLRU	4327	GTND1	12524	GTNOD	2734
HBAER	3530	HDRER	5007	HDRXX	1101	HRDAY	1722	IA2D	7655
IA2L	7640	IADR2	12411	IADR4	12424	IADRL	12426	IADRR	12432
IADRV	12403	IDLE	1467	IDLEC	2077	IDLEL	2100	IHTTB	11530
ILUER	4123	ILUT	12442	IMILU	12443	INBYT	6125	INDRV	5302
INFIX	10000	INIDS	6031	INPR1	1137	INPRC	1116	INRPC	12213
INSCO	12154	INSTB	6126	INT3A	12015	INT3B	12072	INTCM	12045
INTEX	12165	INTP	12253	INIRA	12170	INTRR	12161	INTS	12032
INTS1	12046	INTS2	12060	INTS4	12100	INTSB	12043	INTSE	12044
INTSP	12042	INTSR	12114	INTST	12202	INTX	12224	INTXA	12220
INTXB	12221	INTZ	12360	IOCAL	34103	IOP	6	IOR	10000
IRSV	12375	ISA2D	6127	ISA2L	6130	ITIRS	11550	ITSIM	11654
ITSMB	11612	I.BPT	4140	I.IAO	12247	I.IGN	12237	I.IGV	12444
I.INT	12376	I.IPC	12246	I.LPC	11765	I.SCO	5267	I.SPC	4627
I.THT	5271	I.TTT	7002	JBRKP	1316	JFLTO	151	JPEN	11106
JPINC	10725	JRSTB	5404	KABIT	11111	KCFPF	5121	KFLTO	12217
KIHSM	11112	KNCNF	5126	KNXMB	4124	K.DIR	4116	LACBY	6646
LACT	3043	LACT1	3061	LACT2	3075	LACT3	3117	LACT4	3120
LAFP	3161	LCALO	3036	LCAL1	3037	LCALR	3035	LCFS	3207
LCGUA	400	LCPA	3223	LCPAL	3230	LCPTN	3204	LDASA	3123
LDAYS	2223	LDFTA	6051	LDFTX	6052	LDMAM	3362	LDMAX	3363
LF	4004	LFMAP	3124	LMAP.	4122	LNOP	3034	LOADD	6131
LOADR	5776	LPART	3273	LPAS	3127	LPAS1	3153	LPAS2	3154
LPCTR	3275	LPIBS	3277	LPIHS	3276	LR	5000	LRELP	3274
LRFLG	3040	LSCON	4117	LSET	3321	LSPT	3241	LSPT1	3251
LSPT2	3263	LSPT3	3267	LSTBY	6746	LSUBR	3202	LSUBX	3203
LUDRE	3125	LUERR	5127	LUPT.	3205	LUSR1	3014	LUSR2	3030
LUSRX	2772	LUTOT	3041	LW	4400	LWAC	3300	LWUP	3166
L.BSA	7752	M1I	100000	M2I	40000	M3L20	10666	M3L25	11326
M3L30	11741	M3L32	11766	M3L35	12005	M3L40	12015	M3L42	12075
M3L44	12114	M3L46	12220	M3L50	12250	M3PCO	11221	M5L20	10677
M5L25	11366	M5L30	11765	M5L32	12005	M5L42	12114	M5L44	12131
MAPRE	5322	MARK3	0	MARK5	1	MARK8	0	MARKI	4527
MCT	5575	MDRTN	7230	MINMO	3506	MINOD	2756	MK3	0
MK5	1	MKDIR	4543	MKEXI	4532	MKIN2	4623	MKINF	4601
MKLAT	4566	MKLUD	4535	MKNOT	4562	MKPRI	4524	MKUNL	4567
MKVIR	4473	MOBYD	7201	MOBYG	7155	MOBYX	7126	MONTH	13
MOVEX	7004	MSK	173777	MSKSU	3511	MST	1500	MV1DN	7120
MV1UP	7052	MV8DN	7077	MV8UP	7031	MVWDN	7062	MVWTP	7057
M.A1	7231	NFREE	2754	NIOX	12235	NODES	2663	NOPKE	4015
NOTPF	4743	NOXME	5224	NO.MO	3507	NULCT	11470	NULCY	11527

NULLP	11512	NULM3	11467	NULRT	11467	NULTB	11505	N.DCH	6050		
N.SUB	36	OBFLT	6775	OBSTO	6735	OBTRC	6762	OBTRR	6770		
OBZBY	6736	OK	3610	OKIFF	4244	OKIFN	4256	OKIFP	4231		
OTEMP	6761	OTXT	6625	OTXT2	6627	OTXT4	6636	OURTN	7003		
OUTBY	6132	OUTTE	6133	PATSP	12707	PB	400	PCACC	10562		
PCBA2	10245	PCBAC	10241	PCBAP	10250	PCBIN	10234	PCBL2	10266		
PCBLI	10255	PCBMK	10512	PCCB	10711	PCCC	11370	PCCE	10727		
PCCE2	10745	PCCH	11041	PCCH1	11060	PCCH2	11076	PCCKF	10552		
PCCKK	10560	PCCO	11114	PCCP	10662	PCCQ	11155	PCCQ2	11165		
PCCS	11145	PCCX	11252	PCCY	11451	PCDDT	11035	PCEK4	10605		
PCEKO	10577	PCEN	10574	PCEOM	11015	PCER	10707	PCES	11374		
PCES2	11377	PCGNX	10625	PCHA1	10360	PCHAR	10357	PCI1	10641		
PCIBF	11455	PCIC2	10775	PCICR	10765	PCIDN	10762	PCIEC	10537		
PCILC	10755	PCIN	10535	PCIN1	10431	PCIN2	10471	PCINA	10530		
PCINC	10620	PCIO2	11417	PCIO4	11425	PCIO6	11426	PCIOD	11025		
PCIOR	11402	PCITB	11276	PCITT	11355	PCN1A	10451	PCNC1	10621		
PCNCA	10616	PCNCO	10617	PCNIB	10470	PCO	11174	PCO1	11203		
PCO2	11221	PCO2A	11217	PCO2B	11240	PCO3	11243	PCOBE	11141		
PCODN	11123	PCPER	10703	PCRUB	10521	PCSI	10324	PCSIN	11135		
PCSQ	10306	PCST2	10303	PCSTI	10273	PCSTT	10365	PCT2	11305		
PCT3	11306	PCTIO	11266	PCTMX	11304	PCTS	10615	PCTS2	10754		
PCTSK	11271	PCTSP	10523	PECOM	4141	PEDON	4167	PELOO	4146		
PFDSC	12351	PFERR	5011	PFHDW	12345	PFINT	4763	PFRSH	12325		
PFRSL	12302	PFRSO	12331	PFRSR	12326	PFRST	12267	PFSL1	12321		
PFSLO	12313	PIB	4	PIBCN	10220	PIBCO	10233	PIBDO	10224		
PIBGE	6673	PIBIN	10224	PIBLI	10233	PIBPU	6745	PIBRE	10224		
PIBUN	10233	PIIA3	10126	PIIAX	10124	PIIB2	10214	PIIB3	10213		
PIIBG	10201	PIIBI	10155	PIIBO	10104	PIIBP	10170	PIIBR	10130		
PIIBX	10211	PIIPB	10120	PIOBR	10233	PIOBS	10233	PISTA	10121		
POBCO	10227	POBGE	6673	POBIN	10231	POBLI	10233	PPCB	10722		
PRC	2000	PRIVA	100000	PSINT	12137	PTSPL	4307	PU	2		
PUTBY	6134	PUTMR	4502	PZD	11	PZIH	11765	PZOU2	12013		
PZOUT	12000	PZS	10	PZSND	12005	PZSTO	12021	P.BSA	3417		
P.CA	5	P.DA	4	P.FL	2	P.HBA	3420	P.HXA	3421		
P.LU	3	P.NX	1	P.PR	0	P.SSA	3422	P..	3412		
QCHA2	10070	QCHAR	6103	QCHAX	10052	QCHOV	10076	QCOVC	10050		
QLINK	2475	QLNKF	2560	QLPC	2556	QR	40	QUE	2306		
QUEO	2336	QUEOA	2350	QUE1	2365	QUE2	2366	QUE4	2403		
QUE5	2412	QUE6	2423	QUE7	2433	QUEF	2361	QUEF2	2362		
QUEUE	6104	QURTN	2305	QW	20	RO	7767	RBLK	3366		
RBLK1	3404	RBLK2	3405	RBQA3	3411	RBTAS	3513	RCAL2	12447		
RD	200	RDATB	12703	RDATE	4761	RDFEF	6327	RDFER	6312		
RDFET	6357	RDFEW	6326	RDFTR	6326	RDFTW	6326	RDR	14000		
READB	6135	RECAL	12445	REGS	5752	RELJM	6136	RESTO	5473		
RETRI	5451	RJSR	6136	RLSF1	7735	RLSF2	7746	RLSFB	7730		
RLSFC	7753	RLSFM	7754	RSFBX	7756	RTEM1	12400	RTEM2	12401		
RTEM3	12402	RTEMP	12377	RTP	7	RUP	5	RVDSU	6041		
RWAFI	5054	RWBC	5346	RWTAS	3535	SA	5767	SAVDI	24		
SA.	5565	SBA	40	SCHO	1452	SCH1	1464	SCH3	1514		
SCH4	1524	SCH4L	1527	SCH5	1501	SCHED	1447	SCTSL	1317		
SD	4040	SDA	5770	SDEST	11723	SELP	1536	SELPO	1545		
SEND	11314	SENDO	11357	SEND2	11366	SETPE	4401	SIG1	11735		
SINT	1561	SINT1	1611	SIODS	1320	SKIFL	4221	SKIFN	4052		
SKIPR	4110	SLEP1	2606	SLEP2	2620	SLEP3	2631	SLEP4	2654		
SLEPQ	2660	SLEPX	2577	SLPCA	1560	SLPQL	2657	SLUTP	12434		
SNDCR	11364	SNDSP	11365	SNDTS	11356	SNODE	2062	SPINP	6146		
SRETR	5304	SSI	7275	SSIBU	7352	SSIGT	11717	SSISI	7306		
SSTAB	5566	START	12361	STBY	6745	STCON	1617	STCTR	1621		

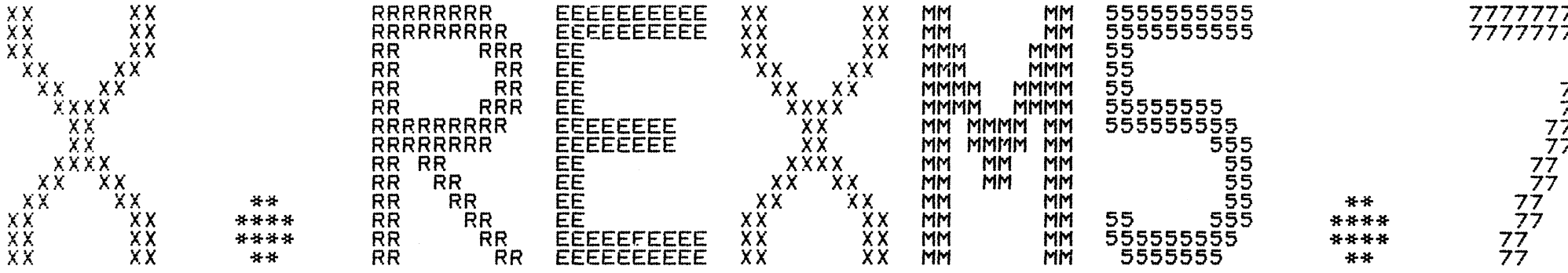
STI	7273	STIMO	1070	STIM1	1164	STIM2	1226
STIM3	1232	STIME	1034	STIMM	1057	STINP	6140
STINT	6147	STIOD	1620	STKOV	12216	STKP	5764
STO	7360	STOA3	7411	STOB	6714	STOR	7407
STORD	6137	STOUT	6141	STSCH	2661	STTSL	1616
SV	1	SVDSU	6033	TASKQ	15	TASKT	2275
TCTSL	2101	TFALT	12470	TFLT2	12473	THEND	1032
THPCB	2076	THZ1	1674	THZ10	1702	THZ2	1713
THZ3	1723	THZ39	1736	THZ4	1744	THZ40	1737
THZ4A	1756	THZ5	1770	THZ5A	2017	THZ5E	2023
THZ5F	2054	THZ6	1751	THZ6A	2056	THZ7	2122
THZ7A	2143	THZ8	2161	THZ8A	2167	THZ9	2203
THZC	2075	TIMEO	5466	TNHZ1	1672	TNHZT	1627
TRANS	4643	TRAP3	12025	TRAPF	6142	TRAPH	12027
TRYNE	4364	TSC	2016	TSPH	1721	TTIS	11743
TTOS	11753	TTOS2	11755	TTPFR	12005	TTSN1	11777
TTSND	11766	TTSNI	12002	TTSNO	11772	UBIT	5571
UF	4002	UN	4020	UNLKX	7447	USE.N	4752
UTEMP	7431	VABIT	10515	VCABN	7355	VCHM1	3320
VCL	4465	VCT	1322	VFF	4464	VFFUF	4463
VFW	5122	VINCL	6177	VIRGI	4000	VLC	5266
VLFD1	7611	VLFD2	7632	VLFMK	7575	VLFNX	7577
VLFTY	7573	VLFVN	7574	VLKPX	7546	VLRET	7572
VLRTN	7576	VMM	5265	VMXPW	10702	VMXSS	2022
VMXTA	1622	VNXMB	5264	VPF	5013	VPRTD	11110
VQIDN	2061	VRO	5270	VSTIN	7354	VTHPD	2060
VUCII	10215	WBLK	3423	WBQA3	3430	WBTAS	3515
WBTCO	3526	WONAX	7412	WR	100	WRITB	6143
WTNOT	5353	WTSYS	5254	XACBY	6645	XDPER	3532
XGETB	6144	XINDR	12356	XIOP	6713	XIRUP	10102
XLC	4120	XLPCA	6466	XMEM	4121	XNIOP	12350
XPCO	10346	XPPCB	10513	XPSIN	10103	XPUTB	6145
XP.BS	4220	XQLIN	12553	XQLINK	1522	XQUE1	2464
XQUEX	2436	XRJIH	10003	XRJSR	7770	XSCON	5574
XSRET	12357	XSTBY	6641	XTNAP	6503	XX	3636
X.DIR	4630	X.INF	4631	YEAR	3702	YY	3721
ZDPER	4403	ZZ	3705	.ABA	14	.AWAK	1735
.BPLR	4317	.BPMR	4311	.BPS	77	.BPS2	1312
.BPS3	1313	.BPS4	1031	.BRKP	150	.BSA	10
.CBSA	7755	.CHQ	10044	.CHQE	10045	.CKLC	5421
.CQIP	10323	.CRLA	12441	.C.PR	4534	.DA	174
.DA3	175	.DAER	4577	.DAT	5765	.DB	176
.DB3	177	.DBYE	1030	.DFEF	6053	.DFER	6057
.DFET	6054	.DFEW	6060	.DFTR	6055	.DFTW	6056
.DKER	5465	.DP1E	5014	.DPEX	3512	.DPRE	5120
.ESCL	11401	.FLSC	12662	.FLTO	152	.FRND	2575
.FSTS	12521	.FWTS	12657	.GTDS	5760	.GTND	2435
.HBA	11	.HRS.	1670	.HXA	12	.IDLL	1476
.IHTB	10516	.IHTT	10514	.IIBG	10036	.IIBP	10035
.IMSK	12215	.INFO	100	.INID	5757	.INSB	12373
.INSP	12374	.INTE	12164	.INTF	2364	.INTR	111
.INTX	2576	.INTZ	12343	.INV	12344	.IN.X	12342
.IRUP	12041	.ITTT	10517	.LACB	10217	.LACT	2762
.LAFP	2764	.LCM	114	.LCPA	2757	.LDAY	1671
.LDRE	2771	.LPAS	2763	.LPCA	6352	.LSET	2761
.LSMM	2767	.LSPT	2766	.LST8	10216	.LTOT	2770
.LUER	4466	.LWAC	2760	.LWUP	2765	.MV1D	7061
.MV1U	7060	.NDCH	7427	.NRET	112	.NULC	11251
.OBIN	7426	.OK	4253	.PC	11312	.PCCP	10347
.PCER	10353	.PCHA	10753	.PCST	12214	.PICR	10354
.PIDN	11301	.PILC	10613	.PIN	11113	.PINC	11466
.PIOD	11447	.PNC1	11250	.PPCB	11313	.PPER	10352
.PSTI	10661	.PTSK	10351	.P.BS	4626	.QLIN	3361
.RVDS	5762	.SAT	5766	.SDFT	6351	.SDSB	5573
.SEND	10752	.SIGB	1623	.SIGE	1625	.SIGN	1624
.SLPQ	1720	.SRET	113	.SSA	13	.STIM	1477
.STK	5763	.SVDS	5761	.UDSB	5572	.XDFX	6353
.XIOP	7357	.XX	4113	.YY	4114	.ZZ	4115
..	0	..DAT	12665	..INR	10350	..SAT	12666

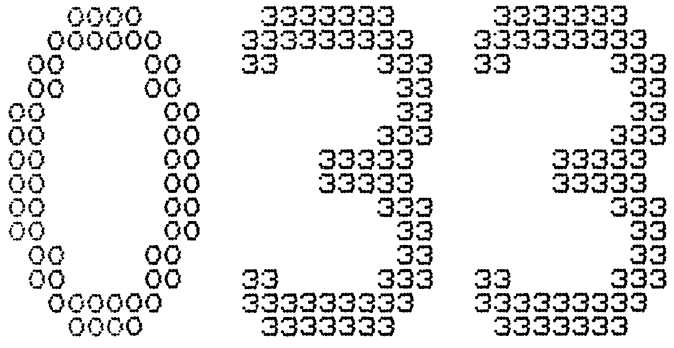




File: 1/X.REXM5.7033 Printed: MAR 2, 1987 12:24:25
File size: 143 blocks Copy 1 of 1

Printed for RDC 9.1





File: 1/X.REXM5.7033 Printed: MAR 2, 1987 12:24:33
File size: 143 blocks Copy 1 of 1

Printed for RDC 9.1

***** J O B S T A T I S T I C S *****

274	TOTAL # DUPLICATE KEYS
0	TOTAL # DIR. RE-ORGS
5,346	TOTAL # KEYS INSERTED
0	TOTAL # ASSEMBLY ERRS

```
..          72.040=  77.012   77.017   77.020   77.023   77.026   77.029
            77.032   77.038   77.041   77.044   77.049   77.052   77.055
            77.058   78.010   78.013   78.016   78.019   78.025   78.028
            78.031   78.034   78.037   78.040   78.043   78.046   78.049
            78.052   78.055   79.012   79.015   79.018   79.021   79.024
            79.027   79.030   79.034   79.037   79.040   79.043   79.046
            79.049   80.008   80.011   80.014   80.017   80.020   80.023
            80.026
..DAT      231.045  231.059:
..INR      180.046: 181.032
..SAT      231.041  231.060:
..STK      231.031
.ABA         4.030:
.AWAK       27.059   27.062:
.BPLR       91.056:  95.022
.BPMR       91.049:  94.023
.BPS         5.058:  15.040   17.007   19.053
.BPS2       17.037   17.052:
.BPS3       17.039   17.053:
.BPS4       14.034:  14.055
.BPT.       87.039
.BRKP        8.009:  17.056
.BSA         4.025:  69.017   70.017   89.029   90.052   98.017
.C.PR       96.019   96.029:
.CBSA      164.023  164.044: 165.024
.CHQ       168.039: 169.037  233.015
.CHQE      168.040: 169.035  233.016
.CKLC      110.056  110.060:
.CQIP      179.036  179.050:
```

```
.CRLA     225.042  225.057:
.DA         8.031:   8.037   56.037
.DA3        8.032:   8.038
.DAER      97.015   97.058:
.DAT      119.045  122.016: 231.059
.DB         8.033:   8.039
.DB3        8.034:   8.040
.DBYE      14.033:  14.051
.DFEF     124.033:
.DFER     124.037:
.DFET     124.034:
.DFEW     124.038:
.DFTR     124.035:
.DFTW     124.036:
.DKER     111.038  111.061  112.007:
.DP1E     102.042  102.046  102.051:
.DPEX      73.040:  76.019
.DPRE     102.037  103.033  104.026  104.031  105.048  105.050: 107.019
          107.050
.ESCL      10.029  201.025  201.027:
.FLSC     230.039  231.055:
.FLTO       8.010    8.011:
.FRND      42.034   42.041:
.FSTS     229.029: 230.033
.FWTS     230.036  231.052:
.GTDS     120.014  120.018  122.011:
```

```
.GTND     36.019   37.057:  41.006
.HBA       4.026:  75.017
.HRS.     26.050   27.007   27.010   27.013:
.HXA       4.027: 120.013
.IDLL     21.055   22.007:
.IHTB    182.019  184.025  184.035: 184.039  189.013
.IHTT    182.016  184.033:
.IIBG    168.031: 173.036
.IIBP    168.030: 171.044  172.031
.IMSK    217.025  218.057:
.IN.X    223.022  223.027:
.INFO      5.059:   8.042   14.059   15.017   15.044   15.058   16.038
          16.054   17.017   18.034   19.057   20.016   21.041   22.011
          23.028   23.036   26.025   28.009   91.016  108.020  129.012
         216.037  216.054  221.039  225.048  229.019  231.018  231.030
.INID    119.020  122.010:
.INSB    224.026  224.031:
.INSP    224.027  224.032:
.INTE     10.026  218.026  218.028:
.INTF     36.052   37.007:
.INTR      6.039: 211.037
.INTX     42.035   42.042:  44.043
.INTZ    223.019  223.028:
.INV     223.015  223.029:
.IRUP    215.041  215.049: 218.017  218.025
.ITTT    182.038  184.036: 186.023
.LACB    174.052  174.061:
```

```
.LACT     53.038:   54.034
.LAFP     53.040:   54.040
.LCM       6.042:   86.032  108.023
.LCPA     53.035:   54.018
.LDAY     26.051    27.014:
.LDRE     53.045:
.LFX.    134.043
.LLU.    134.033
.LPAS     53.039:   54.037
.LPCA    128.028  128.048:  129.033
.LSET     53.037:   54.027
.LSMM     53.043:   54.023   54.029
.LSPT     53.042:   54.022
.LSR.     19.058
.LSTB    174.042  174.060:
.LTOT     53.044:
.LUER     94.016   94.049:
.LUT.    134.026  225.058
.LVR.    134.047  222.048
.LWAC     53.036:   54.026
.LWUP     53.041:   54.043
.MAP.     86.055  108.021
.MV1D    143.053:  144.020
.MV1U    143.022  143.052:
.NDCH    154.018:  154.022  156.026
.NRET      6.040:  122.037
```

```
.SDFT    128.014  128.047: 129.028
.SDSB    115.035: 119.022
.SEND    186.027  186.035  190.026  190.033:
.SIGB     10.017   25.047:  27.018  233.039
.SIGE     25.049:
.SIGN     25.048:  27.019   27.034
.SLPQ     27.043:  27.052
.SRET      6.041:
.SSA       4.028: 119.058  120.017  120.047  232.026
.STIM     22.008:  22.016   22.021
.STK      10.008  119.012  122.014:
.STN.     16.055  231.019
.SVDS    120.008  122.012:
.TTT.    141.057  184.036  200.007
.TXTF     18.049
.UDSB    115.034: 119.027
.UPT.     60.060
.XDFX    128.036  128.049: 129.022
.XIOP    151.022  151.037:
.XX       85.017   86.048:
.YY       85.030   86.049:
.ZZ       85.040   86.050:
A0        31.020   36.020   37.025   40.045   41.035   47.054   48.038
          48.043   81.047   87.018  101.043  105.024  105.043  216.022
         218.008  218.048
A1        36.021   37.026   40.046   41.036   87.020  101.044  105.025
         105.044  215.045  216.011  218.009  218.049
```

```
C1004     191.036  191.061:
C11         5.016:  149.024
C12         5.017:  195.045  196.015
C13         5.018:
C14         5.019:  134.052
C140      185.033  186.031:
C15         5.020:
C16         5.021:
C160        8.037=
C163        8.038=
C166        8.039=
C17         5.022:  199.038
C170K       5.007:
C171        8.040=
C177        5.035:   81.026  159.040  185.027
C1777       5.049:
C2          4.016:   25.017   75.044  111.039  125.030  190.019
C20         5.025:   58.015  101.031  109.032  109.041  183.045  187.023
          187.030  202.049  203.040
C200        5.036:   15.030   15.037  101.045  102.014  152.030  182.035
          184.020  196.020  210.029  211.018
C2000       5.050:
C204      180.051:  183.020
C205        5.037:  192.038
C215        5.038:  191.031  197.021  199.031
C233      180.052:  183.017
C237      171.041  171.050:
```

```
C240        5.039:  139.031  140.036  184.021  185.020  185.022  186.018
C244        5.040:
C260        5.041:  162.028
C271        5.042:  162.025
C3          4.017:  111.036  111.058  112.013  134.036
C300        5.043:   20.028   25.011  161.027
C33       231.020  231.058:
C334        5.044:  197.019  202.007
C34       188.032  188.046:
C37         5.026:   15.060   56.012   58.013   60.045  101.029
C377        5.045:   15.027   17.024   25.026   61.040   90.020   91.014
          100.033  101.059  138.051  140.022  141.024  145.051  163.027
          184.017  199.026
C4          5.011:   27.032   27.039   36.023   40.049   90.054   98.020
          164.025  180.026  185.037
C40         5.027:   14.048   28.018  129.046  132.043  211.033  212.055
C400        5.046:   15.046   15.052  182.022  183.014
C4000       5.051:  202.009
C40K      150.052  151.036:
C5          5.012:   31.040   56.035  106.054  111.027
C6          5.013:  106.058  149.023
C600        8.042=
C7          5.014:  106.046  143.017  144.016  189.017  203.023
C77         5.033:  136.043
C774C       5.008:
C777        5.047:   25.015  119.029  124.044  231.039
CALD1     119.011:  126.047
```

```
CH34K     126.010:  126.043
CH4K      126.008:  126.032
CHA8A     125.045   125.059:
CHAN0     124.061:  125.010   125.031   125.057   125.060
CHAN1     124.046   124.050   125.008:
CHAN2     124.055   124.060   125.011:
CHAN3     125.030:  125.034
CHAN4     125.026   125.033:
CHAN6     125.036   126.023   126.049   127.008:
CHAN7     125.039   126.018:
CHAN8     125.056:  126.042
CHANA     125.046   125.049   127.017   127.019   127.026:
CHANE     125.028   126.045   126.049:  127.011   127.022   127.023
CHANJ     125.044:  125.055
CHANN       6.026=   18.029   154.027
CHANP     126.039   126.046:  126.051
CHANW     125.042   125.047   125.052:  127.018
CHANX       6.027   124.040:
CHANY     124.054   124.059:
CHANZ     124.052   124.057:
CHCHX     130.012   155.028   156.023:
CHCNP     126.016:  126.021   126.036
CHKFF      92.056    92.060    93.035:
CHKP1     157.051:  158.018
CHKP2     157.047   158.007:
CHKPB     130.020   130.021   130.022   157.031:
```

```
CXOFF     191.043  191.060:

CXON      180.032  180.039:

D.BC       71.046=  76.017   86.041  104.021  104.023  107.048  108.007
          110.043  110.047  112.019

D.CA       71.040=  81.037   88.027   88.035   90.051   91.013   93.026
           97.025  104.017  105.026  109.051  110.052  110.054

D.CW       71.042=  76.015   82.038   82.042   92.045   92.053   93.036
          101.039  102.031  104.019  105.013  105.020  105.022  105.031

D.DA       71.039=  81.036   86.020   86.038   87.021   87.033   93.023
           94.025   96.039   97.013   97.023  104.008  104.018  107.013
          107.047  108.028  110.012  110.048  110.050

D.HP       71.044=  93.039  102.020  103.035  103.038  104.009  104.027
          104.028

D.IP       71.035= 102.018  104.012

D.LF       71.049= 107.045  108.054  109.036  110.013  110.021  110.028
          110.034  111.030  111.042

D.LU       71.038=  86.023   87.019   87.031   92.050   93.021   94.014
           94.034   96.037   97.021  107.040  108.050

D.LV       71.050= 107.046  108.008  108.030  108.055  109.050  110.011
          110.029  111.043  111.051  112.010

D.PE       71.041=  76.018   80.041   82.054   88.050   89.040   93.028

D.REA     232.028

D.T1       71.036=  93.040   93.044   93.045  108.027  109.045  110.038
          110.057

D.T2       71.037=  93.042   93.051

D.TB       71.047=  85.031   93.041  102.013  102.034  102.035  103.031
          104.024

D.TB.     101.062  102.011:

D.TO       71.051=  81.016  109.044  110.023

D.TS       71.048=  82.053   82.056   88.029   88.032   88.037   88.038
           88.039   91.023   91.025   91.028   91.029   91.030  110.018
          110.039
```

```
FLTLD     229.042  229.046  229.051:  229.058

FLU        14.054   14.057   16.034    16.042    18.040   125.037   127.008
          127.014  127.016

FLUO      134.033: 134.039

FLU1      134.035  134.043:

FLU2      134.041  134.045  134.051:

FLUT        7.021  134.028: 134.032

FLW.       14.047   16.018   18.043    20.029    25.010    28.043    28.053
           29.015  140.026  151.014   151.020   152.029   152.034   152.037
          153.023  180.025  180.035   182.032   183.013   183.024   183.027
          184.013  185.023  186.014   186.039   188.015   188.020   190.015
          190.018  191.018  191.035   191.040   191.046   192.042   193.013
          193.028  193.033  194.014   194.029   196.019   199.028   201.031
          201.035  202.029  213.022

FLWMA     201.032  202.019:

FMAP       56.057

FNOD.      46.040

FR         72.031=  73.021   79.012    79.015    80.023

FRDA      232.025  232.037:

FREE       46 041:  47.034   47.050    47.052    47.053    48.032    48.033
           48.045  233.030

FREEG      46.048   46.051:

FREEN       6.029= 231.014

FREEX       6.030   46.043:

FRERT      46.039:  46.046   46.053    46.055    46.059    46.061

FREX       46.044   46.055:

FREXG      46.057   46.060:

FRNO2      47.035:  47.048

FRNO3      47.043   47.050:

FRNOD      42.041   46.049   46.058    47.034:
```

```
PIB       4.018:  20.018  20.039  54.030  56.009  56.040  58.022
         59.036   60.042  60.049  61.044
PIBCN   168.016  175.013:
PIBCO   168.027  175.034:
PIBDO   168.019  175.021:
PIBGE   168.025  175.040=
PIBIN   168.017  175.019:
PIBLI   168.021  175.035:
PIBPU   168.024  175.041=
PIBRE   168.018  175.020:
PIBS     64.039
PIBUN   168.022  175.036:
PIHS     64.038
PIIA3   171.030  171.046  171.049:  172.028  173.028
PIIAX   171.045:  172.041  173.037
PIIB2   174.036  174.047  174.054  174.058:
PIIB3   174.035  174.046  174.055  174.057:
PIIBG   168.031  174.045:
PIIBI   168.032  173.026:
PIIBO   168.034  171.028:
PIIBP   168.030  174.034:
PIIBR   168.033  172.026:
PIIBX   174.043  174.054:
PIIPB   171.041:  172.047
PIOBR   168.011  175.033:
PIOBS   168.010  175.032:
```

```
PISTA    171.039  171.042:
PLNK      37.017   37.018   37.019   37.022   37.054   38.036   38.037
          38.040   41.010   41.011   41.014   44.016   44.019   44.020
          44.026   44.032
POBCO    168.028  175.026:
POBGE    168.026  175.042=
POBIN    168.020  175.029:
POBLI    168.023  175.037:
PORTD     29.044  192.058
PORTT     29.022  151.035
PPCB     184.032  186.017  188.025  188.030  189.038:  189.041  190.017
         191.017  191.034  191.042  191.048  192.037   192.045  198.035
PRC      127.033= 127.047  127.053
PRIVA     84.020=  90.060   91.052   91.059
PSINT    169.056  217.056:
PTSPL     91.019   91.041:
PU        72.025=  73.016   77.017   77.026   77.032   77.058   78.016
PUTBY      7.038=
PUTMR     78.044   80.024   95.030:
PZD      209.040= 211.017  211.030  212.050
PZIH      11.010  211.011: 233.013
PZOU2    211.029  211.034:
PZOUT    211.016  211.023:
PZS      209.038= 211.014  211.036  212.047  212.057
PZSND    212.042: 233.021
PZSTD    212.043  212.054:
QCHA2    169.042: 169.051  169.053
```

```
SKIPZ     188.015  194.017
SKNB      109.038  110.026
SLEP1      43.037:  43.043
SLEP2      43.040   44.012:
SLEP3      44.015   44.021:
SLEP4      44.037   44.041:
SLEPQ      27.043   44.045   44.046:
SLEPX      43.030:  130.030
SLPCA      23.041   23.047:
SLPQL      43.035   44.045:
SLUR      108.056  222.051
SLUTP     222.043  225.030  225.051:  226.020
SN.LU      16.057
SND.       29.019  152.043  180.022  180.033  188.027  191.044  194.040
          197.043  199.025  200.031
SNDCR     199.034  200.017:
SNDSP     200.015  200.018:
SNDTS     199.044  199.050  200.009:  200.011  200.014
SNODE       4.033   10.007   30.014:   31.018   31.019   31.020   31.022
           44.047
SPCF.      21.043   98.039
SPED.     222.038
SPINP       7.059=
SRETR     108.052  108.058  109.031:  110.058  223.046
SSBA       40.053  216.025  218.013  218.034
SSI         7.060  150.039:
SSIBU     150.047  151.031:
```

```
SSIGT      36.009  189.037  208.030:
SSISI     150.044  150.049:
SSTAB     115.013  115.016  115.030:
ST         22.009
START     223.042  224.019: 233.033
STAT      101.034
STBY        7.039  141.018: 175.041
STCON      25.020   25.024   25.040:
STCTR      25.028   25.030   25.042:
STI         7.048  150.036:
STIM0      15.008   15.012:
STIM1      15.011   16.013   16.016:
STIM2      16.033   16.037   16.041   16.050:
STIM3      16.021   16.054:
STIME      14.038:  22.008
STIMM      14.050   14.057:
STINP       7.047=  18.057
STINT       7.061=  28.051  191.057
STIOD      25.021   25.041:
STKOV     216.014  218.060:
STKP      119.011  119.050  119.051  120.019  120.028  120.031  120.032
          122.015:
STO         7.050  152.023:
STOA3     152.027  152.047  152.049:
STOB        7.035  137.030  140.022:
STOR      152.032  152.041  152.046:
```

```
VLFD2     160.046  160.060:
VLFMK     159.055: 160.035  160.048
VLFNX     130.035  160.032:
VLFTY     159.053: 160.037  160.050
VLFVN     159.054: 160.040  160.042  160.044  160.056
VLKPX     130.036  159.027: 160.045
VLRET     159.027  159.031  159.039  159.043  159.044  159.048:
VLRTN     159.056: 160.032  160.057  160.058  160.061
VLU       159.037  160.034  160.047  160.054
VMM       108.017  109.007:
VMXPW     188.026  188.044:
VMXSS      29.012:  29.018
VMXTA      25.043:
VNXMB     108.010  109.006:
VPF       101.037  102.032  102.049: 105.037
VPRTD     189.030  191.054  192.059:
VQIDN      29.039  29.045:
VRO       107.017  109.010:
VSTIN     151.024  151.034:
VTHPD      29.025  29.044:
VUCII     174.038  174.049  174.059:
WBLK        7.054  69.054:
WBQA3      69.054  69.059:
WBTAS      69.056  75.016:
WBTCO      75.019  75.026:
WONA       18.047
```

```
XP.BS      89.024   89.045:
XPCO      180.044:  182.021
XPPCB     181.055   184.032:  185.049
XPSIN     169.045   169.056:
XPUTB       7.057=
XQLIN     229.028   229.060:
XQLNK      22.031    22.040:
XQUE1      38.039    38.044    38.051:
XQUEU     230.043
XQUEX      38.029:   39.032    42.033   130.032
XRJIH     166.028   166.039:
XRJSR       7.044   166.027:
XSCON     115.036:  119.038
XSRET     223.039   223.046:
XSTBY       7.058   138.023:
XSTK      231.032
XSTP      231.033   231.036
XTNAP     132.048   133.026:  133.028
XX         78.024:   86.048
YEAR        3.018=    3.031
YY         80.007:   86.049
ZDPER      90.023    92.026    93.032:   97.054
ZZ         79.033:   86.050
```