```
0001  MUM00

;                                                    RCSL:    43-GL 1
;                                                    AUTHOR:  A P RAVN
;                                                    EDITED:  74.02.08
```

```
; KEYWORD: MUS, MONITOR, LISTING.

; ABSTRACT: MULTIPROGRAMMING UTILITY SYSTEM
; MONITOR PROCEDURES.
;
; ASCII PAPER TAPE:    RCSL  43-GL 2
; REL.BINARY PAPER TAPE: RCSL 43-GL 3
```

```
↑ 0002  MUM00
        000000 INTER=0      ; INTERRUPT MASK

               .NREL        ; RELOCATABLE MODULE
               .TITL MUM00
               .EXTN I8
        000012 .RDX  10     ; DECIMAL THROUGHOUT
        000001 .TXTM 1      ; LEFT TO RIGHT TEXT PACKING

               ; CONTENTS:
               ;   FORMAT DEFINITIONS
               ;   FUNCTION ENTRIES
               ;   MONITOR PROCESS DESCRIPTOR:
               ;     MONITOR TABLE
               ;     INTERRUPT RESPONSE
               ;     MONITOR UTILITY PROCEDURES
               ;     MONITOR FUNCTIONS
```

```
                ;    ******    FORMAT DEFINITIONS    ******

                                        ; ITEM:
000000 .DUSR NEXT= 0                    ;   NEXT ITEM IN A QUEUE OF ITEMS
000001 .DUSR PREV= NEXT+1               ;   PREVIOUS ITEM IN A QUEUE OF ITEMS
000002 .DUSR CHAIN=PREV+1               ;   NEXT ITEM IN A CHAIN OF ITEMS
000003 .DUSR SIZE= CHAIN+1              ;   SIZE OF THE ITEM
000004 .DUSR NAME= SIZE+1               ;   NAME OF THE ITEM (THREE WORDS)

                                        ; PROCESS DESCRIPTOR:
                ;       NEXT            ;   NEXT PROCESS IN A QUEUE OF PROCESSES
                ;       PREV            ;   PREVIOUS PROCESS IN A QUEUE OF PROCESSES
                ;       CHAIN           ;   NEXT PROCESS IN THE PROCESS CHAIN
                ;       SIZE            ;   SIZE OF THE PROCESS DESCRIPTOR
                   ;       NAME          ;    NAME OF THE PROCESS (THREE WORDS)
000007 .DUSR EVENT=NAME+3               ;   EVENT QUEUE HEAD (TWO WORDS)
000011 .DUSR BUFFE=EVENT+2              ;   FREE MESSAGE BUFFER CHAIN HEAD
000012 .DUSR PROG= BUFFE+1              ;   PROGRAM ADDRESS
000013 .DUSR STATE=PROG+1               ;   STATE OF PROCESS
000014 .DUSR TIMER=STATE+1              ;   TIMER COUNT
000015 .DUSR PRIOR=TIMER+1              ;   PRIORITY
000016 .DUSR BREAD=PRIOR+1              ;   BREAK ADDRESS
000017 .DUSR AC0=  BREAD+1              ;   SAVED AC0
000020 .DUSR AC1=  AC0+1                ;   SAVED AC1
000021 .DUSR AC2=  AC1+1                ;   SAVED AC2
000022 .DUSR AC3=  AC2+1                ;   SAVED AC3
000023 .DUSR PSW=  AC3+1                ;   PSW (PROCESS STATUS WORD)
000024 .DUSR SAVE= PSW+1                ;   SAVED LINK
000025 .DUSR 0=    SAVE+1               ; OPTIONAL WORDS:
000025 .DUSR SAVE1=0                    ; WORK LOCATION
000025 .DUSR BUF=  0                    ;   SAVED MESSAGE BUFFER ADDRESS
000026 .DUSR ADDRE=BUF+1                ;   CURRENT VALUE OF ADDRESS
000027 .DUSR COUNT=ADDRE+1              ;   CURRENT VALUE OF COUNT
000030 .DUSR RESER=COUNT+1              ;   RESERVED
000031 .DUSR CONVT=RESER+1              ;   CONVERSION TABLE ADDRESS
000032 .DUSR CLINT=CONVT+1              ;   CLEAR DEVICE INTERRUPT
```

```
                                        ; MESSAGE BUFFER:
                ;       NEXT            ;   NEXT BUFFER IN A QUEUE OF BUFFERS
                ;       PREV            ;   PREVIOUS BUFFER IN A QUEUE OF BUFFERS
                ;       CHAIN           ;   NEXT BUFFER IN A CHAIN OF BUFFERS
                ;       SIZE            ;   SIZE OF THE MESSAGE BUFFER
000004  .DUSR SENDE=SIZE+1              ;   SENDER PROCESS DESCRIPTOR
000005  .DUSR RECEI=SENDE+1             ;   RECEIVER PARAMETER
000006  .DUSR MESS0=RECEI+1             ;   0.MESSAGE
000007  .DUSR MESS1=MESS0+1             ;   1.MESSAGE
000010  .DUSR MESS2=MESS1+1             ;   2.MESSAGE
000011  .DUSR MESS3=MESS2+1             ;   3.MESSAGE
000012  .DUSR BSIZE=MESS3+1             ;   SIZE OF MESSAGE BUFFER

                                        ; PROGRAM DESCRIPTOR:
000000  .DUSR PSPEC=0                   ;   SPECIFICATION OF PROGRAM:
                                        ;     B0: OWN
                                        ;     B1: REENTRANT
                                        ;     B2: PAGE ZERO USER
                                        ;     B(8:15) PROCESS COUNT
000001  .DUSR PSTAR=PSPEC+1             ;   START ADDRESS
                ;       CHAIN           ;   NEXT PROGRAM IN A CHAIN OF PROGRAMS
                ;       SIZE            ;   SIZE OF THE PROGRAM AREA
                ;       NAME            ;   NAME OF THE PROGRAM (THREE WORDS)
```

```
                                        ; ZONE DESCRIPTOR:
000000 .DUSR ZNAME=0                    ;   NAME (THREE WORDS)
       ;     SIZE                       ;   SIZE OF THE ZONE DESCRIPTOR
000004 .DUSR ZMODE=SIZE+1               ;   MODE OF OPERATION
000005 .DUSR ZKIND=ZMODE+1              ;   KIND OF DOCUMENT
000006 .DUSR ZMASK=ZKIND+1              ;   MASK FOR GIVE UP
000007 .DUSR ZGIVE=ZMASK+1              ;   GIVE UP ADDRESS
000010 .DUSR ZFILE=ZGIVE+1              ;   FILE COUNT
000011 .DUSR ZBLOC=ZFILE+1              ;   BLOCK COUNT
000012 .DUSR ZCONV=ZBLOC+1              ;   CONVERSION TABLE ADDRESS
000013 .DUSR ZBUFF=ZCONV+1              ;   BUFFER ADDRESS
000014 .DUSR ZSIZE=ZBUFF+1              ;   SIZE OF BUFFER
000015 .DUSR ZFORM=ZSIZE+1              ;   FORMAT OF RECORD
000016 .DUSR ZLENG=ZFORM+1              ;   LENGTH OF RECORD
000017 .DUSR ZFIRS=ZLENG+1              ;   FIRST OF RECORD (BYTE ADDRESS)
000020 .DUSR ZTOP= ZFIRS+1              ;   TOP OF RECORD (BYTE ADDRESS)
000021 .DUSR ZUSED=ZTOP+1               ;   USED SHARE
000022 .DUSR ZSHAR=ZUSED+1              ;   SHARE LENGTH (IN BYTES)
000023 .DUSR ZREM= ZSHAR+1              ;   REMAINING BYTES IN SHARE
                                        ; AUXILLIARY WORDS:
000024 .DUSR Z0=   ZREM+1               ;   AUX 0
000025 .DUSR Z1=   Z0+1                 ;   AUX 1
000026 .DUSR Z2=   Z1+1                 ;   AUX 2
000027 .DUSR Z3=   Z2+1                 ;   AUX 3
000030 .DUSR Z4=   Z3+1                 ;   AUX 4
000031 .DUSR Z5=   Z4+1                 ;   AUX 5
000006 .DUSR ZAUX= 6                    ;   NUMBER OF AUXILLIARY WORDS
000032 .DUSR Z=    Z0+ZAUX              ; OPTIONAL WORDS:

                                        ; SHARE DESCRIPTOR:
000000 .DUSR SUPER=0                    ;   OPERATION (0,MESSAGE)
000001 .DUSR SCOUN=SUPER+1              ;   COUNT (1,MESSAGE)
000002 .DUSR SADDR=SCOUN+1              ;   ADDRESS (2,MESSAGE)
000003 .DUSR SSPEC=SADDR+1              ;   SPECIAL (3,MESSAGE)
000004 .DUSR SNEXT=SSPEC+1              ;   NEXT SHARE
000005 .DUSR SSTAT=SNEXT+1              ;   STATE OF SHARE
000006 .DUSR SFIRS=SSTAT+1              ;   FIRST SHARED (BYTE ADDRESS)
000007 .DUSR SSIZE=SFIRS+1              ;   SIZE OF SHARE DESCRIPTOR

       ;    ******   END OF FORMAT DEFINITIONS   ******
```

```
↑ 0006  MUM00
                ;    ******   MONITOR FUNCTION ENTRIES    ******

        000000 .LOC  0          ; MONITOR START IN WORD 0

00000 000000         0          ;  0: SAVED PC AT INTERRUPT, ALSO WORK CELL
00001 000067'        A0         ;  1: INTERRUPT RESPONSE ADDRESS

00002 000365'F2:     A20        ;  2: WAIT
00003 000365'F3:     A20        ;  3: WAIT INTERRUPT
00004 000365'F4:     A20        ;  4: SEND MESSAGE
00005 000365'F5:     A20        ;  5: WAIT ANSWER
00006 000365'F6:     A20        ;  6: WAIT EVENT
00007 000365'F7:     A20        ;  7: SEND ANSWER
00010 000365'F8:     A20        ;  8: SEARCH ITEM
00011 000365'F9:     A20        ;  9: CLEAN PROCESS
00012 000365'F10:    A20        ; 10: BREAK PROCESS
00013 000365'F11:    A20        ; 11: STOP PROCESS
00014 000365'F12:    A20        ; 12: START PROCESS
00015 000365'F13:    A20        ; 13: RECHAIN
00016 000000 WORK:   0          ; 14: PAGE ZERO LOCATION
00017 000000 LINK:   0          ; 15: PAGE ZERO LOCATION

                FUNCTION:       ; MONITOR FUNCTION ENTRIES:
00020 000000         0          ; 16: AUTO-INCREMENTING LOCATION
00021 000000         0          ; 17: AUTO-INCREMENTING LOCATION
00022 000420'        A22        ; 18: WAIT(DELAY,DEVICE,FIRST,SECOND,BUF)
00023 000436'        A23        ; 19: WAIT INTERRUPT(DEVICE,DELAY)
00024 000456'        A24        ; 20: SEND MESSAGE(ADDR,NAME ADDR,BUF)
00025 000511'        A25        ; 21: WAIT ANSWER(FIRST,SECOND,BUF)
00026 000520'        A26        ; 22: WAIT EVENT(FIRST,SECOND,BUF)
00027 000530'        A27        ; 23: SEND ANSWER(FIRST,SECOND,BUF)
00030 000535'        A28        ; 24: SEARCH ITEM(CHAIN,NAME ADDR,ITEM)
00031 000541'        A29        ; 25: CLEAN PROCESS(PROC)
00032 000577'        A30        ; 26: BREAK PROCESS(PROC,ERROR NUMBER)
00033 000614'        A31        ; 27: STOP PROCESS(PROC)
00034 000630'        A32        ; 28: START PROCESS(PROC)
00035 000634'        A33        ; 29: RECHAIN(OLD,NEW,ELEM)
00036 000000         0          ; 30: AUTO-DECREMENTING LOCATION
00037 000000         0          ; 31: AUTO-DECREMENTING LOCATION

        006002 .DUSR WAIT=              JSR@   F2
        006003 .DUSR WAITINTERRUPT=     JSR@   F3
        006004 .DUSR SENDMESSAGE=       JSR@   F4
        006005 .DUSR WAITANSWER=        JSR@   F5
        006006 .DUSR WAITEVENT=         JSR@   F6
        006007 .DUSR SENDANSWER=        JSR@   F7
        006010 .DUSR SEARCHITEM=        JSR@   F8
        006011 .DUSR CLEANPROCESS=      JSR@   F9
        006012 .DUSR BREAKPROCESS=      JSR@   F10
        006013 .DUSR STOPPROCESS=       JSR@   F11
        006014 .DUSR STARTPROCESS=      JSR@   F12
        006015 .DUSR RECHAIN=           JSR@   F13

                ;    ******   END OF MONITOR FUNCTION ENTRIES    ******
```

```
↑ 0007  MUM00
                ;   ******   MONITOR PROCESS DESCRIPTOR   ******

                ; THE MONITOR PROCESS DESCRIPTOR HOLDS THE MONITOR TABLE, THE
                ; MONITOR CODE, THE SYSTEM UTILITY PROCEDURES, AND THE INPUT/
                ; OUTPUT UTILITY PROCEDURES.

                ;   ******   MONITOR TABLE   ******

        000040 .DUSR M=     .       ; MONITOR PROCESS DESCRIPTOR:
        000040 .DUSR CUR=   .       ;
 00040  000040              CUR     ;  NEXT: FIRST PROCESS IN RUNNING QUEUE
 00041  000040              CUR     ;  PREV: LAST PROCESS IN RUNNING QUEUE
                                    ;        HEAD OF RUNNING QUEUE AND HEAD OF
                                    ;        PROCESS CHAIN.

 00042  000000              0       ; CHAIN: PROCESS DESCRIPTOR OF FIRST
                                    ;        PROCESS IN PROCESS CHAIN.
 00043  000000            0         ;   SIZE: NOT USED
 00044  000000              0       ;   NAME: NOT USED
        000045 .DUSR TABLE=.
 00045  000370              DEVTA   ;     +1: DEVICE TABLE
                                    ;         CONTAINS A WORD FOR EACH DEVICE
                                    ;         NUMBER HOLDING PROCESS DESCRIPTORS
                                    ;         FOR INTERRUPT REQUESTING PROCESSES.
        000046 .DUSR TOPTA=.
 00046  000457              TOPDE   ;     +2: TOP OF DEVICE TABLE
                    DFIRS:
 00047  000047              DFIRS   ; EVENT: FIRST PROCESS IN DELAY QUEUE
 00050  000047              DFIRS   ;     +1: LAST PROCESS IN DELAY QUEUE
                                    ;         HEAD OF DELAY QUEUE
 00051  000000              0       ;   BUFFER: NOT USED
        000052 .DUSR PFIRS=.
 00052  000000              0       ;  PROG: FIRST IN PROGRAM CHAIN

 00053  000000              0       ; STATE: MONITOR STATE: ALWAYS ZERO
        000054 .DUSR RUNNI=.
 00054  000040              CUR     ; TIMER: RUNNING QUEUE
        000054 .DUSR PROCE=RUNNI    ;        REFERENCE TO HEAD OF RUNNING QUEUE AND
        000054 .DUSR MONIT=RUNNI    ;        HEAD OF PROCESS CHAIN: MONITOR PROCESS

 00055  000000              0       ; PRIOR: MONITOR PRIORITY: LOWEST POSSIBLE: ZER
        000056 .DUSR EXIT=.
 00056  000127'              A1     ; BREAD: MONITOR EXIT
        000057 .DUSR EFIRS=.
 00057  000000              0       ;   AC0: FIRST IN ENTRY CHAIN
        000060 .DUSR FFIRS=.
 00060  000367              247     ;   AC1: LAST IN FREE CORE
        000061 .DUSR DELAY=.
 00061  000047              DFIRS   ;   AC2: DELAY QUEUE
 00062  000062              JMP .   ;   AC3: PC.MONITOR: JMP .+0
 00063  000144              .-1*2   ;   PSW: PSW.MONITOR: PC.MONITOR*2
 00064  000000 WORK1:       0       ;   MONITOR WORK LOCATIONS
 00065  000000 WORK2:       0       ;
```

```
↑ 0008  MUM00
                ; PAGE ZERO VARIABLES:

        000066 .DUSR FREQU=.      ; FREQUENCY OF RTC:
  00066 000000               0    ;   0: 50 HZ, 1: 10 HZ, 2: 100 HZ, 3: 1000 HZ
        000067 .DUSR MASK= .      ;
  00067 000000               INTER ; INTERRUPT MASK
        000070 .DUSR CORES=.      ;
  00070 000000               0    ; CORE SIZE
        000071 .DUSR PROGR=.      ;
  00071 000050         PFIRS-CHAIN ; REFERENCE TO HEAD OF PROGRAM CHAIN
  00072 000640'        A56         ; CLINT: CLEARDEVICE .
  00073 000143'        A2          ; CLINT.RTC: REAL TIME CLOCK
        000074 .DUSR RTIME=.      ; REAL TIME COUNT (DOUBLE WORD)
  00074 000000               0    ;
  00075 000000               0    ;
                ; 3 WORDS NOT USED
  00076 000000               0
  00077 000000               0
  00100 000000               0
                ; PAGE ZERO CONSTANTS:

  00101 100000 B1B0:    1B0
  00102 040000 B1B1:    1B1
  00103 020000 B1B2:    1B2
  00104 010000 B1B3:    1B3
  00105 004000 B1B4:    1B4
  00106 002000 B1B5:    1B5
  00107 001000 B1B6:    1B6
  00110 000400 B1B7:    1B7
  00111 000200 B1B8:    1B8
  00112 000100 B1B9:    1B9
  00113 000040 B1B10:   1B10
  00114 000020 B1B11:   1B11
  00115 000010 B1B12:   1B12
  00116 000004 B1B13:   1B13
  00117 000002 B1B14:   1B14
  00120 000001 B1B15:   1B15

        000101 .DUSR  B1T=    B1B0
        000101 .DUSR .1B0=    B1B0
        000102 .DUSR .1B1=    B1B1
        000103 .DUSR .1B2=    B1B2
        000104 .DUSR .1B3=    B1B3
        000105 .DUSR .1B4=    B1B4
        000106 .DUSR .1B5=    B1B5
        000107 .DUSR .1B6=    B1B6
        000110 .DUSR .1B7=    B1B7
        000111 .DUSR .1B8=    B1B8
        000112 .DUSR .1B9=    B1B9
        000113 .DUSR .1B10=   B1B10
        000114 .DUSR .1B11=   B1B11
        000115 .DUSR .1B12=   B1B12
        000116 .DUSR .1B13=   B1B13
        000117 .DUSR .1B14=   B1B14
        000120 .DUSR .1B15=   B1B15
```

```
↑ 0009  MUM00
                ; STATUS BITS:

        000101 .DUSR SDISC=.1B0      ; DISCONNECTED
        000102 .DUSR SUFFL=.1B1      ; OFF LINE
        000103 .DUSR SBUSY=.1B2      ; BUSY
        000104 .DUSR SDEV1=.1B3      ; DEVICE MODE 1
        000105 .DUSR SDEV2=.1B4      ; DEVICE MODE 2
        000106 .DUSR SDEV3=.1B5      ; DEVICE MODE 3
        000107 .DUSR SILLE=.1B6      ; ILLEGAL
        000110 .DUSR SEOF= .1B7      ; EOF
        000111 .DUSR SBLOC=.1B8      ; BLOCK ERROR
        000112 .DUSR SDATA=.1B9      ; DATA LATE
        000113 .DUSR SPARI=.1B10     ; PARITY ERROR
        000114 .DUSR SEM=  .1B11     ; END MEDIUM
        000115 .DUSR S12=  .1B12     ; BIT 12
        000116 .DUSR SNOTP=.1B13     ; NOT PROCESSED
        000117 .DUSR STIME=.1B14     ; TIMER
        000120 .DUSR S15=  .1B15     ; BIT 15

                ; CONTROL BITS:

        000111 .DUSR CERAS=.1B8      ; ERASURE
        000112 .DUSR CDISC=.1B9      ; DISCONNECT
        000113 .DUSR CPOSI=.1B10     ; POSITIONING
        000114 .DUSR CTERM=.1B11     ; TERMINATION
        000115 .DUSR CCONV=.1B12     ; CONVERSION
        000116 .DUSR CRESE=.1B13     ; RESERVATION

00121 000003 C3:      3
00122 000005 C5:      5
00123 000006 C6:      6
00124 000007 C7:      7
00125 000011 C9:      9
00126 000012 C10:     10
00127 000014 C12:     12
00130 000015 C13:     13
00131 000017 C15:     15
00132 000030 C24:     24
00133 000031 C25:     25
00134 000050 C40:     40
00135 000060 C48:     48
00136 000070 C56:     56
00137 000074 C60:     60
00140 000077 C63:     63
00141 000170 C120:    120
00142 000177 C127:    127
00143 000377 C255:    255
00144 177775 CM3:     -3
00145 177774 CM4:     -4
00146 177760 CM16:    -16
00147 177400 CM256:   -256
```

```
↑ 0010  MUM00
        000055 .DUSR .0=       PRIORITY+M
        000120 .DUSR .1=      .1B15
        000117 .DUSR .2=      .1B14
        000121 .DUSR .3=       C3
        000116 .DUSR .4=      .1B13
        000122 .DUSR .5=       C5
        000123 .DUSR .6=       C6
        000124 .DUSR .7=       C7
        000115 .DUSR .8=      .1B12
        000125 .DUSR .9=       C9
        000126 .DUSR .10=      C10
        000127 .DUSR .12=      C12
        000130 .DUSR .13=      C13
        000131 .DUSR .15=      C15
        000114 .DUSR .16=     .1B11
        000132 .DUSR .24=      C24
        000133 .DUSR .25=      C25
        000113 .DUSR .32=     .1B10
        000134 .DUSR .40=      C40
        000135 .DUSR .48=      C48
        000136 .DUSR .56=      C56
        000137 .DUSR .60=      C60
        000140 .DUSR .63=      C63
        000112 .DUSR .64=     .1B9
        000141 .DUSR .120=     C120
        000142 .DUSR .127=     C127
        000111 .DUSR .128=    .1B8
        000143 .DUSR .255=     C255
        000110 .DUSR .256=    .1B7
        000107 .DUSR .512=    .1B6
        000106 .DUSR .1024=   .1B5
        000105 .DUSR .2048=   .1B4
        000104 .DUSR .4096=   .1B3
        000103 .DUSR .8192=   .1B2
        000102 .DUSR .16384=  .1B1
        000101 .DUSR .32768=  .1B0
        000144 .DUSR .M3=      CM3
        000145 .DUSR .M4=      CM4
        000146 .DUSR .M16=     CM16
        000147 .DUSR .M256=    CM256

        000116 .DUSR .NAME=.4     ; NAME.PROC
        000124 .DUSR .EVEN=.7     ; EVENT.PROC
        000124 .DUSR .EDOC=.EVEN  ; DOCUMENT.ENTRY
        000123 .DUSR .MESS=.6     ; MESS.BUF
        000124 .DUSR .SSIZ=.7     ; SIZE OF SHARE DESCRIPTOR
        000150 .DUSR .Z=    .     ;
  00150 000032                Z   ; OPTIONAL.ZONE
        000127 .DUSR .RTC= .12    ; RTC DEVICE NUMBER
        000126 .DUSR .NL=  .10    ;
        000126 .DUSR .LF=  .10    ;
        000130 .DUSR .CR=  .13    ;
        000127 .DUSR .FF=  .12    ;
        000112 .DUSR CUR2=.64     ; MONITOR PROCESS*2
```

```
↑ 0011  MUM00
                ; REFERENCES:

   00151 000122'R00:  A00    ; INTERRUPT ACCEPT
   00152 000170'R6:   A6     ; POWER RESTART POINT
   00153 000206'R100: A100   ; REMOVE PROCESS(CUR,STATE)
   00154 000207'R10:  A10    ; REMOVE PROCESS(STATE)
   00155 000210'R11:  A11    ; REMOVE(ELEM)
   00156 000220'R12:  A12    ; LINK PROCESS(PROC)
   00157 000232'R13:  A13    ; LINK(HEAD,ELEM)
   00160 000242'R14:  A14    ; RECHAIN(OLD,NEW,ELEM)
   00161 000260'R15:  A15    ; SEARCH(CHAIN,NAME ADDR,ITEM)
   00162 000304'R160: A160   ; SETBUF(CUR,BUF)
   00163 000322'R17:  A17    ; DELIVER ANSWER(BUF)

                ; REFERENCES TO REENTRANT SYSTEM UTILITY PROCEDURES:

   00164 000000 R34:  0      ; NEXT OPERATION(MODE,COUNT,BUF)
   00165 000000 R35:  0      ; RETURN ANSWER(STATUS)
   00166 000640'R36:  A36    ; CLEAR(DEVICE)
   00167 000000 R340: 0      ; WAIT OPERATION
   00170 000645'R361: A361   ; SETINTERRUPT(DEVICE)
   00171 000000 R37:  0      ; SET RESERVATION(MODE)
   00172 000000 R38:  0      ; SET CONVERSION(MODE)
   00173 000000 R39:  0      ; CONBYTE(BYTE)
   00174 000000 R40:  0      ; GETBYTE(ADDR,BYTE)
   00175 000000 R41:  0      ; PUTBYTE(ADDR,BYTE)
   00176 000000 R42:  0      ; MULTIPLY(OP1,OP2,RESULT)
   00177 000000 R43:  0      ; DIVIDE(DIVIDEND,DIVISOR,QUOTIENT,REMAINDER)

        006164 .DUSR NEXTOPERATION=   JSR@   R34
        006167 .DUSR WAITOPERATION=   JSR@   R340
        006165 .DUSR RETURNANSWER=    JSR@   R35
        006170 .DUSR SETINTERRUPT=    JSR@   R361
        006171 .DUSR SETRESERVATION=  JSR@   R37
        006172 .DUSR SETCONVERSION=   JSR@   R38
        006173 .DUSR CONBYTE=         JSR@   R39
        006174 .DUSR GETBYTE=         JSR@   R40
        006175 .DUSR PUTBYTE=         JSR@   R41
        006176 .DUSR MULTIPLY=        JSR@   R42
        006177 .DUSR DIVIDE=          JSR@   R43

        002164 .DUSR .NEXTOPERATION=  JMP@   R34
        002165 .DUSR .RETURNANSWER=   JMP@   R35
        002166 .DUSR .CLEARDEVICE=    JMP@   R36
        100166 .DUSR CLEAR= @R36
        002170 .DUSR .SETINTERRUPT=   JMP@   R361
        002171 .DUSR .SETRESERVATION= JMP@   R37
        002172 .DUSR .SETCONVERSION=  JMP@   R38
        002173 .DUSR .CONBYTE=        JMP@   R39
        002174 .DUSR .GETBYTE=        JMP@   R40
        002175 .DUSR .PUTBYTE=        JMP@   R41
        002176 .DUSR .MULTIPLY=       JMP@   R42
        002177 .DUSR .DIVIDE=         JMP@   R43
```

```
↑ 0012  MUM00
          006000 .DUSR GOS=    JSR@    0
          002000 .DUSR GOT=    JMP@    0
          102414 .DALC SEQ=    SUB#    0,0 SZR       ; SKIP IF S=D
          102415 .DALC SNE=    SUB#    0,0 SNR       ; SKIP IF S<>D
          102033 .DALC SLS=    ADCZ#   0,0 SNC       ; SKIP IF S<D
          102433 .DALC SNG=    SUBZ#   0,0 SNC       ; SKIP IF S<=D
          102032 .DALC SNL=    ADCZ#   0,0 SZC       ; SKIP IF S>=D
          102432 .DALC SGR=    SUBZ#   0,0 SZC       ; SKIP IF S>D
          102414 .DALC INE=    SEQ     0,0           ; IF S<>D THEN EXECUTE
          102415 .DALC IEQ=    SNE     0,0           ; IF S=D THEN EXECUTE
          102033 .DALC INL=    SLS     0,0           ; IF S>=D THEN EXECUTE
          102433 .DALC IGR=    SNG     0,0           ; IF S>D THEN EXECUTE
          102032 .DALC ILS=    SNL     0,0           ; IF S<D THEN EXECUTE
          102432 .DALC ING=    SGR     0,0           ; IF S<=D THEN EXECUTE
```

```
↑ 0015  MUM00
                ; REFERENCES TO REENTRANT INPUT/OUTPUT UTILITY PROCEDURES:

00200 000000 R50:   0       ; GETREC(ZONE,BYTES,ADDR,SPAN)
00201 000000 R51:   0       ; PUTREC(ZONE,BYTES,ADDR,SPAN)
00202 000000 R52:   0       ; WAIT TRANSFER(ZONE)
00203 000000 R528:  0       ; REPEATSHARE(ZONE)
00204 000000 R53:   0       ; TRANSFER(ZONE,OP,LENGTH)
00205 000000 R54:   0       ; INBLOCK(ZONE)
00206 000000 R55:   0       ; OUTBLOCK(ZONE)
00207 000000 R56:   0       ; INCHAR(ZONE,CHAR)
00210 000000 R57:   0       ; BACKSPACE(ZONE)
00211 000000 R580:  0       ; OUTSPACE(ZONE)
00212 000000 R58:   0       ; OUTCHAR(ZONE,CHAR)
00213 000000 R590:  0       ; OUTNL(ZONE)
00214 000000 R59:   0       ; OUTEND(ZONE,CHAR)
00215 000000 R60:   0       ; OUTTEXT(ZONE,ADDR)
00216 000000 R61:   0       ; OUTOCTAL(ZONE,VALUE)
00217 000000 R70:   0       ; SETPOSITION(ZONE,FILE,BLOCK)
00220 000000 R71:   0       ; CLOSE(ZONE,RELEASE)
00221 000000 R72:   0       ; OPEN(ZONE,OP)
00222 000000 R700:  0       ; WAITZONE(ZONE)
00223 000000 R82:   0       ; INNAME(ZONE,NAMEADDR);
00224 000000 R83:   0       ; MOVE(PARAMADDR)
00225 000000 R84:   0       ; INTPRETE
00226 002227 R85:   JMP@ .+1; INTGIVEUP
00227 000000        0       ;
00230 002231 R86:   JMP@ .+1; INTBREAK
00231 000000        0       ;
00232 000000 R90:   0       ; BINDEC(WORD,ADDR,CUR);
00233 000000 R91:   0       ; DECBIN(ADDR,WORD,CUR);

        006232 .DUSR BINDEC=          JSR@   R90
        006233 .DUSR DECBIN=          JSR@   R91
        006200 .DUSR GETREC=          JSR@   R50
        006201 .DUSR PUTREC=          JSR@   R51
        006202 .DUSR WAITTRANSFER=    JSR@   R52
        006204 .DUSR TRANSFER=        JSR@   R53
        006205 .DUSR INBLOCK=         JSR@   R54
        006206 .DUSR OUTBLOCK=        JSR@   R55
        006207 .DUSR INCHAR=          JSR@   R56
        006211 .DUSR OUTSPACE=        JSR@   R580
        006212 .DUSR OUTCHAR=         JSR@   R58
        006213 .DUSR OUTNL=           JSR@   R590
        006214 .DUSR OUTEND=          JSR@   R59
        006215 .DUSR OUTTEXT=         JSR@   R60
        006216 .DUSR OUTOCTAL=        JSR@   R61
        006217 .DUSR SETPOSITION=     JSR@   R70
        006220 .DUSR CLOSE=           JSR@   R71
        006221 .DUSR OPEN=            JSR@   R72
        006223 .DUSR INNAME=          JSR@   R82
        006222 .DUSR WAITZONE=        JSR@   R700
        006224 .DUSR MOVE=            JSR@   R83
        006225 .DUSR INTPRETE=        JSR@   R84
```

```
↑ 0014  MUM00
        002200 .DUSR .GETREC=           JMP@    R50
        002201 .DUSR .PUTREC=           JMP@    R51
        002202 .DUSR .WAITTRANSFER=     JMP@    R52
        002203 .DUSR .REPEATSHARE=      JMP@    R526
        002204 .DUSR .TRANSFER=         JMP@    R53
        002205 .DUSR .INBLOCK=          JMP@    R54
        002206 .DUSR .OUTBLOCK=         JMP@    R55
        002207 .DUSR .INCHAR=           JMP@    R56
        002211 .DUSR .OUTSPACE=         JMP@    R580
        002212 .DUSR .OUTCHAR=          JMP@    R58
        002213 .DUSR .OUTNL=            JMP@    R590
        002214 .DUSR .OUTEND=           JMP@    R59
        002215 .DUSR .OUTTEXT   =       JMP@    R60
        002216 .DUSR .OUTOCTAL=         JMP@    R61
        002217 .DUSR .SETPOSITION=      JMP@    R70
        002220 .DUSR .CLOSE=            JMP@    R71
        002221 .DUSR .OPEN=             JMP@    R72

        000226 .DUSR INTGIVEUP=                 R85
        000230 .DUSR INTBREAK=                  R86

        000234 .DUSR MZSTART=.                  ;  MONITOR PAGE ZERO START:
        000367 .LOC  247                        ; START OF PROCESS TABLE
        000367 .DUSR MZLIM=.
  00367 000367          .                       ; BASE PROCESS:
  00370 000000          0                       ; FIRST
  00371 000000          0                       ; SECOND
  00372 000000          0                       ; THIRD
  00373 000000          0                       ; FOURTH
  00374 000000          0                       ; FIFTH
  00375 177777          -1                      ; TOP OF PROCESS LIST
  00376 063530          SKPBZ       24          ; WHILE(BUSY MTO) DO;
  00377 000376          JMP         .-1         ; GOTO SYSTEM START;
  00400 002401          JMP@        .+1         ;
  00401 177777          I8                      ;

                ; DEVICE TABLE:
                ; THIS TABLE IS USED TO SELECT THE PROPER PROCESS DESCRIPTOR
                ; WHEN A DEVICE ISSUES AN INTERRUPT. THE TABLE IS CLEARED BY
                ; THE MONITOR INITIALIZATION.

                .NREL
        000370 .DUSR DEVTA=256-8                ; DEVICE TABLE:
        000457 .DUSR TOPDE=DEVTA+63-8           ; TOP OF DEVICE TABLE:
        000067 .BLK  63-8                       ;

                ;   ******   END OF MONITOR TABLE   ******
```

```
↑ 0015  MUM00
                ;    ******    INTERRUPT RESPONSE    ******

                ; INTERRUPT RESPONSE:
                ; THE ACCUMULATORS ETC ARE SAVED IN THE PROCESS DESCRIPTOR
                 ; OF THE CURRENT PROCESS.

00067'054016 A0:    STA     3    WORK       ; INTERRUPT RESPONSE:
00070'034040        LDA     3    CUR        ;
00071'041417        STA     0    AC0,3      ;    AC0.CUR:= AC0;
00072'045420        STA     1    AC1,3      ;    AC1.CUR:= AC1;
00073'051421        STA     2    AC2,3      ;    AC2.CUR:= AC2;
00074'020000        LDA     0    0          ;
00075'101100        MOVL    0,0             ;
00076'041423        STA     0    PSW,3      ;    PSW.CUR:= WORD(0)*2+CARRY;
00077'020016        LDA     0    WORK       ;
00100'041422        STA     0    AC3,3      ;    AC3.CUR:= AC3;

                ; GET THE DEVICE NUMBER OF THE NEAREST DEVICE ON THE BUS
                ; WHICH IS ISSUING AN INTERRUPT, AND SELECT A PROCESS
                ; DESCRIPTOR FROM THE DEVICE TABLE.
00101'063777        SKPDZ       CPU        ;    IF POWER FAULT THEN
00102'000463        JMP         A5         ;      GOTO POWER FAILURE;
00103'065477        INTA    1               ;    DEV:= INTERRUPTING DEVICE;
00104'034045        LDA     3    TABLE      ;
00105'137000        ADD     1,3             ;
00106'031400        LDA     2   +0,3        ;    PROC:= DEVICE TABLE(DEV);
00107'151220        MOVZR   2,2             ;    PROC:= PROC SHIFT -1;
00110'141140        MOVOL   2,0             ;    DEVTA(DEV):= PROC SHIFT 1+1;
00111'041400        STA     0   +0,3        ;
                ;AC1=DEVICE AC2=PROC MUST NOT BE DESTROYED BY CLINT.PROC
00112'007032        JSR        @CLINT,2     ;    CLEAR INTERRUPT.PROC
00113'021013        LDA     0    STATE,2    ;
00114'107014        ADD#    0,1 SZR         ;    IF DEV=
00115'106415        SUB#    0,1 SNR         ;           ABS(STATE.PROC) THEN
00116'000402        JMP        .+2          ;      BEGIN
00117'000410        JMP         A1         ;
00120'034045        LDA     3    TABLE      ;
00121'137000        ADD     1,3             ; INTERRUPT ACCEPT:
00122'015400 A00:   DSZ         +0,3        ;    DECR(DEVTA(DEV));
00123'011023        ISZ         PSW,2       ;
00124'011023        ISZ         PSW,2       ;    PSW.PROC:= PSW.PROC+2;
00125'004463        JSR         A11         ;    REMOVE(PROC);
00126'004472        JSR         A12         ;    LINK PROCESS(PROC);
                                            ;      END
                ; THIS IS THE STANDARD MONITOR EXIT. THE ACCUMULATORS ETC
                ; ARE RESTORED AND THE INTERRUPT SYSTEM IS ENABLED AGAIN.
00127'102400 A1:    SUB     0,0             ; INTERRUPT RETURN:
00130'040000        STA     0    0          ;    CATCH JUMP TO ZERO;
00131'034040        LDA     3    CUR        ;
00132'021423        LDA     0    PSW,3      ;    CARRY:= PSW.CUR(15);
00133'101220        MOVZR   0,0             ;    RETURN ADDR:= PSW.CUR(0:14);
00134'040016        STA     0    WORK       ;
00135'021417        LDA     0    AC0,3      ;    AC0:= AC0.CUR;
00136'025420        LDA     1    AC1,3      ;    AC1:= AC1.CUR;
00137'031421        LDA     2    AC2,3      ;    AC2:= AC2.CUR;
00140'035422        LDA     3    AC3,3      ;    AC3:= AC3.CUR;
00141'060177        INTEN                   ;    INTERRUPT ENABLE;
00142'002016        JMP@        WORK       ;    RETURN;
```

```
                ; AT EACH TIMER INTERRUPT
                ; TIMER COUNT IS DECREASED BY ONE FOR ALL DELAYED PROCESSES.
                ; WHEN TIMER COUNT FOR A PROCESS BECOMES ZERO, IT IS STARTED
                ; BY LINKING IT TO RUNNING QUEUE.

                A2:                             ; REAL TIME CLOCK:
00143'010075            ISZ         RTIME+1     ;
00144'000403            JMP         .+3         ;
00145'010074            ISZ         RTIME+0     ;   RTIME:=RTIME+1;
00146'000401            JMP         .+1         ;
00147'020066            LDA     0   FREQUENCY   ;
00150'061114            DOAS    0   RTC         ;   START(RTC,FREQUENCY);
00151'030047            LDA     2   DFIRST      ;   PROC:= FIRST IN DELAY QUEUE;
00152'021000 A3:        LDA     0   NEXT,2      ; NEXT DELAYED:
00153'040016            STA     0   WORK        ;   NEXT:= NEXT.PROC;
00154'024061            LDA     1   DELAY       ;
00155'146415            IEQ     2,1             ;   IF PROC=DELAY QUEUE HEAD THEN
00156'002056            JMP@        EXIT        ;     EXIT;
00157'015014            DSZ         TIMER,2     ;   IF DECR(TIMER COUNT.PROC)<>0 THEN
00160'000403            JMP         A4          ;     GOTO GET NEXT;
00161'004427            JSR         A11         ;   REMOVE(PROC);
00162'004436            JSR         A12         ;   LINK PROCESS(PROC);
                A4:                             ; GET NEXT:
00163'030016            LDA     2   WORK        ;   PROC:= NEXT;
00164'000766            JMP         A3          ;   GOTO NEXT DELAYED;

                ; AT POWER FAULT THIS CODE WILL BE ENTERED, AND THE
                ; FIRST TWO INSTRUCTIONS WILL BE EXECUTED. WHEN POWER COMES
                ; BACK THE REMAINING WILL BE EXECUTED.

                A5:                             ; POWER FAILURE:
00165'020420            LDA     0   A8          ;
00166'040000            STA     0   0           ;   WORD(0):=JMP@ POWER RESTART
00167'063077            HALT                    ;   HALT;

                A6:                             ; POWER RESTART:
00170'062677            IORST                   ;   RESET;
00171'020067            LDA     0   MASK        ;
00172'062077            MSKO    0               ;   MASK OUT(MASK);
00173'034045            LDA     3   TABLE       ;
00174'024046            LDA     1   TOPTA       ; FOR DEV:=8 STEP 1 UNTIL 63 DO
00175'021410 A7:        LDA     0   6,3         ;   DEVTA(DEV):=
00176'101220            MOVZR   0,0             ;     DEVTA(DEV) SHIFT -1
00177'101120            MOVZL   0,0             ;                SHIFT  1;
00200'041410            STA     0   6,3         ;
00201'175400            INC     3,3             ;
00202'136414            SUB#    1,3 SZR         ;
00203'000772            JMP         A7          ;
00204'000737            JMP         A2          ;   GOTO REAL TIME CLOCK;
00205'002152 A8:        JMP@        R6          ; CONSTANT;

                ;   ******   END OF INTERRUPT RESPONSE   ******
```

```
                ;   ******   MONITOR UTILITY PROCEDURES   ******

                ; PROCEDURE REMOVE PROCESS(PROC,STATE);
                ; REMOVES THE PROCESS FROM A QUEUE AND SETS ITS STATE.
                ;         CALL:                 RETURN:
                ; AC0     STATE                 DESTROYED
                ; AC1                           UNCHANGED
                ; AC2     PROC                  PROC
                ; AC3     LINK                  DESTROYED

00206'030040 A100: LDA     2   CUR        ;   PROC:= CUR;

                A10:                      ; REMOVE PROCESS:
00207'041013       STA     0   STATE,2    ;   STATE.PROC:= STATE;
                                          ;   REMOVE(PROC);
                                          ;   RETURN;

                ; PROCEDURE REMOVE(ELEM);
                ; REMOVES A GIVEN ELEMENT FROM A QUEUE.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1                           UNCHANGED
                ; AC2     ELEM                  ELEM
                ; AC3     LINK                  DESTROYED

00210'054000 A11:  STA     3   0          ; REMOVE:
00211'035000       LDA     3   NEXT,2     ;
00212'057001       STA@    3   PREV,2     ;   NEXT.PREV.ELEM:= NEXT.ELEM;
00213'021001       LDA     0   PREV,2     ;
00214'041401       STA     0   PREV,3     ;   PREV.NEXT.ELEM:= PREV.ELEM;
00215'051000       STA     2   NEXT,2     ;   NEXT.ELEM:= ELEM;
00216'051001       STA     2   PREV,2     ;   PREV.ELEM:= ELEM;
00217'002000       JMP@        0          ;   RETURN;

                ; PROCEDURE LINK PROCESS(PROC);
                ; LINKS A PROCESS TO THE RUNNING QUEUE AS THE LAST PROCESS
                ; AMONG PROCESSES OF SAME PRIORITY.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1                           DESTROYED
                ; AC2     PROC                  PROC
                ; AC3     LINK                  DESTROYED

00220'054000 A12:  STA     3   0          ; LINK PROCESS:
00221'126400       SUB     1,1            ;
00222'045013       STA     1   STATE,2    ;   STATE.PROC:= RUNNING(=0);
00223'034054       LDA     3   RUNNING    ;   HEAD:= RUNNING QUEUE;
00224'021015       LDA     0   PRIORITY,2;
                A120:                     ; NEXT PRIORITY:
00225'035400       LDA     3   NEXT,3     ;   HEAD:= NEXT.HEAD;
00226'025415       LDA     1   PRIORITY,3;   IF PRIORITY.PROC
00227'106432       ING     0,1            ;       <=PRIORITY.HEAD THEN
00230'000775       JMP         A120       ;     GOTO NEXT PRIORITY;
00231'000403       JMP         A130       ;   LINK(HEAD,PROC);
                                          ;   RETURN;
```

```
↑ 0018  MUM00
                ; PROCEDURE LINK(HEAD,ELEM);
                ; LINKS A GIVEN ELEMENT TO THE END OF A QUEUE.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1     HEAD                  HEAD
                ; AC2     ELEM                  ELEM
                ; AC3     LINK                  HEAD

00232'054000 A13:   STA     3   0           ; LINK:
00233'135000        MOV     1,3             ;
00234'021401 A130:  LDA     0   PREV,3      ;   OLD PREV:= PREV.HEAD;
00235'051401        STA     2   PREV,3      ;   PREV.HEAD:= ELEM;
00236'055000        STA     3   NEXT,2      ;   NEXT.ELEM:= HEAD;
00237'041001        STA     0   PREV,2      ;   PREV.ELEM:= OLD PREV;
00240'053001        STA@    2   PREV,2      ;   NEXT.PREV.ELEM:= ELEM;
00241'002000        JMP@        0           ;   RETURN;

                ; PROCEDURE RECHAIN(OLD,NEW,ELEM);
                ; EXCLUDES THE ELEMENT FROM THE OLD CHAIN AND INCLUDES IT
                ; IN THE NEW CHAIN.
                ;         CALL:                 RETURN:
                ; AC0     OLD                   CHAIN.NEW
                ; AC1     NEW                   NEW
                ; AC2     ELEM                  ELEM
                ; AC3     LINK                  CUR

00242'054000 A14:   STA     3   0           ; RECHAIN:
00243'115000 A141:  MOV     0,3             ;   HEAD:= OLD;
00244'021402        LDA     0   CHAIN,3     ;   OLD:= CHAIN.OLD;
00245'101005        MOV     0,0 SNR         ;   IF OLD=0 THEN
00246'000545        JMP         A213        ;     GOTO MONITOR ERROR 3;
00247'112414        INE     0,2             ;   IF OLD<>ELEM THEN
00250'000773        JMP         A141        ;     GOTO RECHAIN;
00251'021002        LDA     0   CHAIN,2     ;
00252'041402        STA     0   CHAIN,3     ;   CHAIN.HEAD:= CHAIN.ELEM;
00253'135000        MOV     1,3             ;
00254'021402        LDA     0   CHAIN,3     ;
00255'041002        STA     0   CHAIN,2     ;   CHAIN.ELEM:= CHAIN.NEW;
00256'051402        STA     2   CHAIN,3     ;   CHAIN.NEW:= ELEM;
00257'000423        JMP         A152        ;   RETURN;
```

```
↑ 0019  MUM00
                ; PROCEDURE SEARCH(CHAIN,NAME ADDR,ITEM);
                ; SEARCHES THE CHAIN FOR AN ITEM WITH A GIVEN NAME AND
                ; DELIVERS IT IF PRESENT, AND A ZERO IF THE NAME IS NOT
                ; FOUND IN THE CHAIN.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1     CHAIN                 DESTROYED
                ; AC2     NAME ADDR             ITEM
                ; AC3     LINK                  CUR

00260'054000 A15:  STA     3   0              ; SEARCH:
00261'135000       MOV     1,3                ;   ITEM:= CHAIN;
             A150:                            ; NEXT ITEM:
00262'035402       LDA     3   CHAIN,3        ;   ITEM:= CHAIN.ITEM;
00263'175005       MOV     3,3 SNR            ;   IF ITEM=0 THEN
00264'000415       JMP         A151           ;     RETURN;
00265'021404       LDA     0   +0+NAME,3      ;
00266'025000       LDA     1   +0,2           ;
00267'106414       INE     0,1                ;   IF 0.NAME.ITEM<>0.NAME ADDR THEN
00270'000772       JMP         A150           ;     GOTO NEXT ITEM;
00271'021405       LDA     0   +1+NAME,3      ;
00272'025001       LDA     1   +1,2           ;
00273'106414       INE     0,1                ;   IF 1.NAME.ITEM<>1.NAME ADDR THEN
00274'000766       JMP         A150           ;     GOTO NEXT ITEM;
00275'021406       LDA     0   +2+NAME,3      ;
00276'025002       LDA     1   +2,2           ;
00277'106414       INE     0,1                ;   IF 2.NAME.ITEM<>2.NAME ADDR THEN
00300'000762       JMP         A150           ;     GOTO NEXT ITEM;
00301'171000 A151: MOV     3,2                ;
00302'034040 A152: LDA     3   CUR            ;
00303'002000       JMP@        0              ;   RETURN;

                ; PROCEDURE SETBUF(PROC,BUF);
                ; SAVES BUF AND THE TWO FIRST BUFFER WORDS IN THE GIVEN
                ; PROCESS DESCRIPTOR.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1     PROC                  PROC
                ; AC2     BUF                   BUF
                ; AC3     LINK                  PROC

00304'024040 A160: LDA     1   CUR            ;   PROC:= CUR;

00305'054000 A16:  STA     3   0              ; SETBUF:
00306'135000       MOV     1,3                ;
00307'051421       STA     2   AC2,3          ;   AC2.PROC:= BUF;
00310'021006       LDA     0   MESS0,2        ;
00311'041417       STA     0   AC0,3          ;   AC0.PROC:= MESS0.BUF;
00312'021007       LDA     0   MESS1,2        ;
00313'041420       STA     0   AC1,3          ;   AC1.PROC:= MESS1.BUF;
00314'021005       LDA     0   RECEIVER,2     ;
00315'100503       NEGL    0,0 SNC            ;   IF RECEIVER.BUF<=0 THEN
00316'002000       JMP@        0              ;     RETURN;
00317'011423       ISZ         PSW,3          ;
00320'011423       ISZ         PSW,3          ;   PSW.PROC:= PSW.PROC+2;
00321'002000       JMP@        0              ;   RETURN;
```

```
                ; PROCEDURE DELIVER ANSWER(BUF);
                ; DELIVERS AN ANSWER FROM A RECEIVER AND STARTS THE SENDER
                ; IF IT IS WAITING FOR AN ANSWER OR AN EVENT.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1                           DESTROYED
                ; AC2     BUF                   DESTROYED
                ; AC3     LINK                  DESTROYED

00322'054017 A17:  STA     3   LINK       ; DELIVER ANSWER:
00323'004665       JSR         A11        ;   REMOVE(BUF);
00324'035004       LDA     3   SENDER,2    ;   PROC:= SENDER.BUF;
00325'054016       STA     3   WORK        ;   SAVE(PROC);
00326'021005       LDA     0   RECEIVER,2;
00327'100400       NEG     0,0            ;   RECEIVER.BUF:=
00330'041005       STA     0   RECEIVER,2;       -RECEIVER.BUF;
00331'021413       LDA     0   STATE,3     ;   COMMENT: ANSWER NOW PENDING;
00332'142404       SUB     2,0 SZR        ;   IF STATE.PROC<>BUF THEN
00333'000407       JMP         A171       ;     GOTO DELIVER EVENT;
                                          ; RELEASE BUFFER:
00334'041005       STA     0   RECEIVER,2;   RECEIVER.BUF:= 0;

                A170:                     ; START PROC:
00335'024016       LDA     1   WORK        ;   RESTORE(PROC);
00336'004747       JSR         A16        ;   SETBUF(PROC,BUF);
00337'131000       MOV     1,2            ;
00340'034017       LDA     3   LINK       ;   LINK PROCESS(PROC);
00341'000657       JMP         A12        ;   RETURN;

                A171:                     ; DELIVER EVENT:
00342'024016       LDA     1   WORK       ;   RESTORE(PROC);
00343'020124       LDA     0  .EVENT      ;   HEAD:= EVENT QUEUE HEAD.PROC;
00344'107000       ADD     0,1            ;
00345'004665       JSR         A13        ;   LINK(HEAD,BUF);
00346'034016       LDA     3   WORK       ;   RESTORE(PROC);
00347'021413       LDA     0   STATE,3    ;
00350'101415       INC#    0,0 SNR        ;   IF STATE.PROC=-1 THEN
00351'000764       JMP         A170       ;     GOTO START PROC;
00352'101127       MOVZL   0,0 SBN        ;   IF STATE.PROC>-2
                                          ;   OR STATE.PROC=180 THEN
00353'002017       JMP@        LINK       ;     RETURN;
00354'025423 A172: LDA     1   PSW,3      ; EVENT AFTER WAIT:
00355'020116       LDA     0  .4          ;
00356'107000       ADD     0,1            ;   PSW.PROC:= PSW.PROC+4;
00357'045423       STA     1   PSW,3      ;
00360'145000       MOV     2,1            ;
00361'171000       MOV     3,2            ;
00362'004626       JSR        .A11        ;   REMOVE(PROC);
00363'131000       MOV     1,2            ;
00364'000751       JMP         A170       ;   GOTO START PROC;

                ;   ******   END OF MONITOR UTILITY PROCEDURES   ******
```

```
↑ 0021  MUM00
                ;   ******   MONITOR FUNCTIONS   ******

                ; MONITOR FUNCTION ENTRY:
                ; ALL MONITOR FUNCTIONS ARE HANDLED HERE AFTER THE INITIAL
                ; CALL.

                A20:                            ; MONITOR FUNCTION ENTRY:
00365'060277            INTDS                   ;   INTERRUPT DISABLE;
00366'054000            STA     3   0           ;   WORD(0):= LINK;
00367'034040            LDA     3   CUR         ;
00370'041417            STA     0   AC0,3       ;   AC0.CUR:= AC0;
00371'045420            STA     1   AC1,3       ;   AC1.CUR:= AC1;
00372'051421            STA     2   AC2,3       ;   AC2.CUR:= AC2;
00373'055422            STA     3   AC3,3       ;   AC3.CUR:= CUR;

00374'030000            LDA     2   0           ; UNPACK CALL:
00375'145100            MOVL    2,1             ;
00376'045423            STA     1   PSW,3       ;   PSW.CUR:= LINK*2+CARRY;
00377'031377            LDA     2  -1,2         ;   CALL:= (-1).LINK;
00400'024151            LDA     1  .15          ;
00401'133400            AND     1,2             ;   INDEX:= CALL(12:15);
00402'025020            LDA     1   FUNCTION,2;
00403'044016            STA     1   WORK        ;
00404'024124            LDA     1  .EVENT       ;
00405'167000            ADD     3,1             ;   AC1:= EVENT QUEUE HEAD.CUR;
00406'031421            LDA     2   AC2,3       ;   AC2:= AC2.CUR;
00407'002016            JMP@        WORK        ;   GOTO FUNCTION(INDEX);

                ; WHEN THE CODE TO HANDLE THE FUNCTION IS ENTERED WE HAVE:
                ;
                ; AC0   AC0.CUR
                ; AC1   EVENT QUEUE HEAD.CUR
                ; AC2   AC2.CUR
                ; AC3   CUR

                ; MONITOR CALL ERRORS:
                ; GENERALLY THE ERROR NUMBERS ARE NEGATIVE AND ARE GENERATED
                ; VIA THE FOLLOWING ENTRIES. AT ERROR RETURN WE HAVE:
                ; SAC0   ERR NO
                ; SAC1   DESTROYED
                ; SAC2   CUR
                ; SAC3   LINK

00410'000411'A210:          .+1                 ; MONITOR CALL ERRORS:
00411'004403 A211: JSR        A21               ; MONITOR ERROR 1: ERR NO:= -1,  OR
00412'004402 A212: JSR        A21               ; MONITOR ERROR 2: ERR NO:= -2,  OR
00413'004401 A213: JSR        A21               ; MONITOR ERROR 3: ERR NO:= -3;

00414'020774 A21:  LDA     0   A210              ;
00415'162400            SUB     3,0             ;
00416'030040            LDA     2   CUR         ;   PROC:= CUR;
00417'000560            JMP         A30         ;   GOTO BREAK PROCESS;
```

```
↑ 0022  MUM00
                ; FUNCTION WAIT(DELAY,DEVICE,FIRST,SECOND,BUF);
                ; WORKS AS A COMBINATION OF THE FUNCTIONS, WAIT INTERRUPT
                ; AND WAIT EVENT. THE RETURN PARAMETERS HAVE TWO POSSIBILITIES,
                ; THE FIRST USED IN CASE OF TIME OUT OR INTERRUPT, THE SECOND
                ; USED IN CASE OF AN EVENT.
                ;         CALL:               RETURN:                 LINK
                ; SAC0    DELAY               DELAY OR FIRST            +0: TIME OUT
                ; SAC1    DEVICE              DEVICE OR SECOND          +1: INTERRUPT
                ; SAC2    BUF                 CUR OR BUF                +2: ANSWER
                ; SAC3    LINK                CUR                       +3: MESSAGE

                ; PRECONDITION:
                ;   BUF=0 OR BUF IN EVENT QUEUE OF THE CALLING PROCESS.
                ;   DEVICE MUST BE ZERO OR IN THE RANGE 8-62.

                A22:                            ; WAIT:
00420'151005            MOV     2,2 SNR         ;   IF BUF=0 THEN
00421'131000            MOV     1,2             ;     BUF:= EVENT QUEUE HEAD.CUR;
00422'031000            LDA     2   NEXT,2      ;   BUF:= NEXT.BUF;
00423'146415            IEQ     2,1             ;   IF BUF=EVENT QUEUE HEAD.CUR THEN
00424'000405            JMP         A220        ;     GOTO SET STATE;
00425'054016            STA     3   WORK        ;   SAVE(CUR);
00426'024056            LDA     1   EXIT        ;
00427'044017            STA     1   LINK        ;   LINK:= EXIT;
00430'000724            JMP         A172        ;   GOTO EVENT AFTER WAIT;
                A220:                           ; SET STATE:
00431'111000            MOV     0,2             ;   DELAY:= AC0.CUR;
00432'021420            LDA     0   AC1,3       ;   STATE:=-DEVICE.CUR
00433'100405            NEG     0,0 SNR         ;   IF STATE=0 THEN
00434'102120            ADCZL   0,0             ;   STATE:= -2
00435'000402            JMP         A230        ;

                ; FUNCTION WAIT INTERRUPT(DEVICE,DELAY);
                ; CHECKS IF AN INTERRUPT IS PENDING, IF NOT THEN
                ; THE PROCESS IS LINKED TO THE DELAY QUEUE. IF DEVICE IS ZERO
                ; ONLY THE DELAY FUNCTION IS ACTIVE.
                ;         CALL:               RETURN:                 LINK
                ; SAC0                        UNCHANGED                 +0: TIME OUT
                   ; SAC1   DEVICE              DEVICE                    +1: INTERRUPT
                ; SAC2    DELAY               CUR
                ; SAC3    LINK                CUR

                ; PRECONDITION:
                ;   DEVICE MUST BE ZERO OR IN THE RANGE 8-62.

00436'021420    A23:    LDA     0   AC1,3       ; WAIT INTERRUPT:
00437'051414    A230:   STA     2   TIMER,3     ;   TIMER COUNT.CUR:= DELAY;
00440'055421            STA     3   AC2,3       ;   AC2.CUR:= CUR;
00441'171000            MOV     3,2             ;   PROC:=CUR;
00442'025420            LDA     1   AC1,3       ;   DEV:= DEVICE.CUR;
00443'125005            MOV     1,1 SNR         ;   IF DEV<>0 THEN
00444'000406            JMP         A231        ;    BEGIN
00445'034045            LDA     3   TABLE       ;
00446'137000            ADD     1,3             ;
00447'031400            LDA     2   +0,3        ;     INT:=DEVTA(DEV);
00450'151222            MOVZR   2,2 SZC         ;     IF INT(15) THEN
00451'002151            JMP@        K00         ;       GOTO INTERRUPT ACCEPT;
                A231:                           ;     END;
```

```
↑ 0025  MUM00
  00452'006153          JSR@      R100        ;   REMOVE PROCESS(CUR,STATE);
  00453'024061          LDA    1  DELAY       ;   HEAD:= DELAY QUEUE HEAD;
  00454'034056          LDA    3  EXIT        ;   LINK(HEAD,OLD);
  00455'002157          JMP@      R13         ;   EXIT;

                ; FUNCTION SEND MESSAGE(ADDR,NAME ADDR,BUF);
                ; SENDS A MESSAGE TO A GIVEN NAMED PROCESS.
                ;         CALL:                 RETURN:
                ; SAC0                          UNCHANGED
                ; SAC1    ADDR                  ADDR
                ; SAC2    NAME ADDR             BUF
                ; SAC3    LINK                  CUR

  00456'034056 A24:   LDA    3  EXIT        ; SEND MESSAGE:
  00457'054017          STA    3  LINK        ;   LINK:= EXIT;
  00460'024054          LDA    1  PROCESS     ;   CHAIN:= PROCESS CHAIN;
  00461'006161          JSR@      R15         ;   SEARCH(CHAIN,NAME ADDR,PROC);
  00462'145005          MOV    2,1 SNR        ;   IF PROC=0 THEN
  00463'000727          JMP        A212       ;     GOTO MONITOR ERROR 2;
  00464'050016          STA    2  WORK        ;   SAVE(PROC);
  00465'031411          LDA    2  BUFFER,3    ;   BUF:= BUFFER,CUR;
  00466'151005 A241:  MOV    2,2 SNR        ; REP:
  00467'000724          JMP        A213       ;   IF BUF=0 THEN ERROR(-3);
  00470'021005          LDA    0  RECEIVER,2  ;
  00471'101005          MOV    0,0 SNR        ;   IF RECEIVER.BUF=0 THEN
  00472'000403          JMP        A242       ;    GOTO SET MESSAGE;
  00473'031002          LDA    2  CHAIN,2     ;   BUF:=CHAIN.BUF;
  00474'000772          JMP        A241       ;   GOTO REP;
                A242:                         ; SET MESSAGE:
  00475'051421          STA    2  AC2,3       ;   AC2.CUR:=BUF;
  00476'045005          STA    1  RECEIVER,2  ;   RECEIVER.BUF:= PROC;
  00477'035420          LDA    3  AC1,3       ;
  00500'021400          LDA    0  +0,3        ;
  00501'041006          STA    0   MESS0,2    ;   MESS0.BUF:= 0.AC1.CUR;
  00502'021401          LDA    0  +1,3        ;
  00503'041007          STA    0   MESS1,2    ;   MESS1.BUF:= 1.AC1.CUR;
  00504'021402          LDA    0  +2,3        ;
  00505'041010          STA    0   MESS2,2    ;   MESS2.BUF:= 2.AC1.CUR;
  00506'021403          LDA    0  +3,3        ;
  00507'041011          STA    0   MESS3,2    ;   MESS3.BUF:= 3.AC1.CUR;
  00510'000632          JMP        A171       ;   GOTO DELIVER EVENT;

                ; FUNCTION WAIT ANSWER(FIRST,SECOND,BUF);
                ; WAITS FOR THE ANSWER TO A GIVEN MESSAGE.
                ;         CALL:                 RETURN:
                ; SAC0                          FIRST
                ; SAC1                          SECOND
                ; SAC2    BUF                   BUF
                ; SAC3    LINK                  CUR

  00511'141000 A25:   MOV    2,0            ; WAIT ANSWER:
  00512'006153          JSR@      R100        ;   REMOVE PROCESS(CUR,BUF);
  00513'031013          LDA    2  STATE,2     ;   RESTORE(BUF);
  00514'021005          LDA    0  RECEIVER,2  ;
  00515'101102          MOVL   0,0 SZC        ;   IF RECEIVER.BUF<0 THEN
  00516'004604          JSR        A17        ;     DELIVER ANSWER(BUF);
  00517'002056          JMP@      EXIT        ;   EXIT;
```

```
↑ 0024  MUM00
                ; FUNCTION WAIT EVENT(FIRST,SECOND,BUF);
                ; DELAYS THE CALLING PROCESS UNTIL AN EVENT ARRIVES IN
                ; ITS QUEUE FOLLOWING A GIVEN BUFFER.
                ;         CALL:                 RETURN:               LINK
                ; SAC0                          FIRST                   +0: ANSWER
                ; SAC1                          SECOND                  +1: MESSAGE
                ; SAC2    BUF                   BUF
                ; SAC3    LINK                  CUR

                ; PRECONDITION:
                ;   BUF=0 OR BUF IN EVENT QUEUE OF THE CALLING PROCESS.

                A26:                            ; WAIT EVENT:
00520'151005            MOV     2,2 SNR         ;   IF BUF=0 THEN
00521'131000            MOV     1,2             ;     BUF:= EVENT QUEUE HEAD.CUR;
00522'031000            LDA     2   NEXT,2      ;   BUF:= NEXT.BUF;
00523'102000            ADC     0,0             ;
00524'034056            LDA     3   EXIT        ;   IF BUF=EVENT.CUR THEN
00525'146415            IEQ     2,1             ;     REMOVE PROCESS(CUR,WAIT EVENT=-1)
00526'002153            JMP@        R100        ;   ELSE SETBUF(CUR,BUF);
00527'002162            JMP@        R160        ;   EXIT;

                ; FUNCTION SEND ANSWER(FIRST,SECOND,BUF);
                ; DELIVERS AN ANSWER TO THE SENDER.
                ;         CALL:                 RETURN:
                ; SAC0    FIRST                 FIRST
                ; SAC1    SECOND                SECOND
                ; SAC2    BUF                   BUF
                ; SAC3    LINK                  CUR

                ; PRECONDITION:
                ;   BUF IN EVENT QUEUE OF THE CALLING PROCESS.

                A27:                            ; SEND ANSWER:
00530'041006            STA     0   MESS0,2      ;   MESS0.BUF:= AC0.CUR;
00531'021420            LDA     0   AC1,3       ;
00532'041007            STA     0   MESS1,2     ;   MESS1.BUF:= AC1.CUR;
00533'034056            LDA     3   EXIT        ;   DELIVER ANSWER(BUF);
00534'002163            JMP@        R17         ;   EXIT;

                ; FUNCTION SEARCH ITEM(CHAIN,NAME ADDR,ITEM);
                ; IF THE CHAIN CONTAINS AN ITEM WITH THE GIVEN NAME, THE ITEM
                ; ADDRESS IS DELIVERED, OTHERWISE A ZERO IS DELIVERED.
                ;         CALL:                 RETURN:
                ; SAC0                          UNCHANGED
                ; SAC1    CHAIN                 CHAIN
                ; SAC2    NAME ADDR             ITEM
                ; SAC3    LINK                  CUR

00535'025420    A28:    LDA     1   AC1,3      ; SEARCH ITEM:
00536'006161            JSR@        R15         ;   SEARCH(CHAIN,NAME ADDR,ITEM);
00537'051421            STA     2   AC2,3       ;   AC2.CUR:= ITEM;
00540'002056            JMP@        EXIT        ;   EXIT;
```

```
↑ 0025  MUM00
                        ;. FUNCTION CLEAN PROCESS(PROC);
                    ; MESSAGES TO THE PROCESS ARE ANSWERED WITH STATUS = NOT PROCESSED
                    ; ANSWERS TO THE PROCESS ARE RELEASED. MESSAGES FROM THE PROCESS
                    ; ARE RELEASED AND THE RECEIVERS ARE BREAKED WITH ERROR NUMBER=1.
                    ; FINALLY THE PROCESS IS BREAKED WITH ERROR NUMBER=0.
                    ;       CALL:                   RETURN:
                    ; SAC0                           UNCHANGED
                    ; SAC1                           UNCHANGED
                    ; SAC2    PROC                   PROC
                    ; SAC3    LINK                   CUR

                    A29:                            ; CLEAN PROCESS:
00541'050064                STA     2   WORK1      ;   SAVE(PROC);
00542'035011                LDA     3   BUFFER,2   ; CLEAN BUFFER CHAIN:
                                                    ;   BUF:=BUFFER.PROC;
00543'161005        A292:   MOV     3,0 SNR        ; NEXT BUF:
00544'000417                JMP         A291       ;   IF BUF=0 THEN GOTO CLEAN EVENT QUEU
00545'031405                LDA     2   RECEIVER,3 ;   RECEIVER:= RECEIVER.BUF;
00546'151005                MOV     2,2 SNR        ;   IF RECEIVER=0 THEN
00547'000412                JMP         A293       ;    GOTO NEXTBUF;
00550'102400                SUB     0,0            ;
00551'041405                STA     0   RECEIVER,3 ;   RECEIVER.BUF:=0;
00552'054016                STA     3   WORK       ;
00553'102520                SUBZL   0,0            ;
00554'151113                MOVL#   2,2 SNC        ;   IF RECEIVER>0 THEN
00555'004423                JSR         A301       ;   BREAK IT(RECEIVER,1);
00556'030016                LDA     2   WORK       ;   RESTORE(BUF);
00557'006155                JSR@        R11        ;   REMOVE(BUF);
00560'034016                LDA     3   WORK       ;
00561'035402        A293:   LDA     3   CHAIN,3    ;   BUF:=NEXT.BUF;
00562'000761                JMP         A292       ;   GOTO NEXT BUF;
00563'030064        A291:   LDA     2   WORK1      ;
00564'031007        A290:   LDA     2   EVENT,2    ; CLEAN EVENT QUEUE:
00565'021001                LDA     0   PREV,2     ;   BUF:= EVENT.PROC;
00566'142405                SUB     2,0 SNR        ;   IF BUF=PREV.BUF THEN
00567'000407                JMP         A295       ;    GOTO FINIS;
00570'102400                SUB     0,0            ;
00571'041006                STA     0   MESS0,2    ;   MESS0.BUF:=
00572'041007                STA     0   MESS1,2    ;   MESS1.BUF:=0;
00573'006163                JSR@        R17        ;   DELIVER ANSWER(BUF);
00574'030064                LDA     2   WORK1      ;
00575'000767                JMP         A290       ;   GOTO CLEAN EVENT QUEUE;
                    A295:                           ; FINIS:
00576'030064                LDA     2   WORK1      ;
                                                    ;   ERR NO:=0;
                                                    ;   BREAK PROCESS(PROC,0);
```

```
↑ 0026  MUM00
                ; FUNCTION BREAK PROCESS(PROC,ERROR NUMBER);
                ; ERROR NUMBER SHOULD BE GREATER THAN ZERO. THE BREAKED PROCESS IS
                ; STARTED AT ITS BREAK ADDRESS WITH THE FOLLOWING ACCUMULATOR
                ; CONTENTS:
                ;   AC0   ERROR NUMBER
                ;   AC1   UNCHANGED
                ;   AC2   PROC
                ;   AC3   PSW.PROC//2 (ITS OLD PROGRAM COUNTER)
                ;         CALL:                 RETURN:
                ; SAC0    ERROR NUMBER          ERROR NUMBER
                ; SAC1                          UNCHANGED
                ; SAC2    PROC                  PROC
                ; SAC3    LINK                  CUR

                A30:                            ; BREAK PROCESS:
00577'034056            LDA     3   EXIT        ;   LINK:= EXIT;
00600'054017    A301:   STA     3   LINK        ; BREAK IT:
00601'041017            STA     0   AC0,2       ;   AC0.PROC:= ERR NO;
00602'051021            STA     2   AC2,2       ;   AC2.PROC:= PROC;
00603'021023            LDA     0   PSW,2       ;
00604'101220            MOVZR   0,0             ;
00605'041022            STA     0   AC3,2       ;   AC3.PROC:= PSW.PROC//2;
00606'021016            LDA     0   BREADDR,2   ;
00607'101120            MOVZL   0,0             ;
00610'041023            STA     0   PSW,2       ;   PSW.PROC:= BREAK ADDR.PROC*2;
00611'006155            JSR@        R11         ;   REMOVE(PROC);
00612'006156            JSR@        R12         ;   LINK PROCESS(PROC);
00613'002017            JMP@        LINK        ;   RETURN;

                ; FUNCTION STOP PROCESS(PROC);
                ; THE PROCESS IS
                ; REMOVED IF PLACED IN A QUEUE, AND ITS STATE IS SET TO STOPPED.
                ; IF IT WAS WAITING FOR EVENT OR ANSWER, PSW IS DECREASED BY
                ; TWO, ENSURING THE CALL TO BE REPEATED IF THE PROCESS IS
                ; STARTED AGAIN.
                ;         CALL:                 RETURN:
                ; SAC0                          UNCHANGED
                ; SAC1                          UNCHANGED
                ; SAC2    PROC                  PROC
                ; SAC3    LINK                  CUR

00614'102620    A31:    SUBZR   0,0             ; STOP PROCESS:
00615'025013            LDA     1   STATE,2     ;   OLD:= STATE.PROC;
00616'006154            JSR@        R10         ;   REMOVE PROCESS(PROC,1B0);
00617'125415            INC#    1,1 SNR         ;   IF OLD =-1 THEN
00620'000405            JMP         A310        ;     GOTO FUNCTION;
00621'020112            LDA     0   .64         ;
00622'125134            MOVZL#  1,1 SZR         ;   IF OLD=0 OR OLD=1B0
00623'106532            SUBZL#  0,1 SZC         ;     OR OLD<64 THEN
00624'002056            JMP@        EXIT        ;     EXIT;
00625'015023    A310:   DSZ         PSW,2       ; FUNCTION:
00626'015023            DSZ         PSW,2       ;   PSW.PROC:= PSW.PROC-2;
00627'002056            JMP@        EXIT        ;   EXIT;
```

```
                ; FUNCTION START PROCESS(PROC);
                ; IF STATE OF THE PROCESS IS STOPPED, THE PROCESS IS LINKED
                ; TO RUNNING QUEUE.
                ;         CALL:                 RETURN:
                ; SAC0                          UNCHANGED
                ; SAC1                          UNCHANGED
                ; SAC2    PROC                  PROC
                ; SAC3    LINK                  CUR

00630'021013 A32:  LDA     0    STATE,2     ; START PROCESS:
00631'103245       ADDUR   0,0  SNR         ;   IF STATE.PROC=1B0 THEN
00632'006156       JSR@         R12         ;     LINK PROCESS(PROC);
00633'002056       JMP@         EXIT        ;   EXIT;

                ; FUNCTION RECHAIN(OLD,NEW,ELEMENT);
                ; EXCLUDES THE ELEMENT FROM THE OLD CHAIN AND INCLUDES IT IN
                ; THE NEW CHAIN.
                ;         CALL:                 RETURN:
                ; SAC0    OLD                   OLD
                ; SAC1    NEW                   NEW
                    ; SAC2   ELEM                     ELEM
                ; SAC3    LINK                  CUR

00634'025420 A33:  LDA     1    AC1,3       ; RECHAIN:
00635'034056       LDA     3    EXIT        ;   RECHAIN(OLD,NEW,ELEM);
00636'002160       JMP@         R14         ;   EXIT;
```

```
                ; PROCEDURE CLEAR(DEVICE);
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1     DEVICE                DEVICE
                    ; AC2                           UNCHANGED
                ; AC3     LINK                  DESTROYED

00637'060200            NIOC        0           ; CONSTANT
00640'020777 A36:   LDA     0    .-1            ; CLEAR DEVICE:
00641'123000        ADD     1,0                 ;   INSTRUCTION:= NIOC 0+DEVICE;
00642'040401        STA     0 .+1               ;
00643'060200        NIOC        0               ;   EXECUTE(INSTRUCTION);
00644'001400        JMP        +0,3             ;   RETURN;



                ; PROCEDURE SET INTERRUPT(DEVICE);
                ;         CALL:                  RETURN:
                ; AC0                            DESTROYED
                ; AC1     DEVICE                 DEVICE
                ; AC2                            UNCHANGED
                ; AC3     LINK                   CUR

00645'060277 A361:  INTDS                       ; SET INTERRUPT:
00646'054016        STA     3    WORK           ;  DISABLE;
00647'004771        JSR         A36             ;   CLEAR(DEVICE);
00650'034045        LDA     3    TABLE          ;
00651'137000        ADD     1,3                 ;
00652'020040        LDA     0    CUR            ;
00653'101120        MOVZL   0,0                 ;
00654'041400        STA     0    +0,3           ;   DEVTA(DEV):= CUR*2;
00655'115220        MOVZR   0,3                 ;
00656'060177        INTEN                       ;
00657'002016        JMP        @WORK            ; RETURN;

                ;   ******   END OF MONITOR FUNCTIONS   ******

                .END
```