```
;
; something to return
;
                    .lev t.o
                    .lck all
 031372 121705 5 0 bufch2: jst [bchsch] i
 031373 100000 5 0          skp
 031374 003420 5 0           jmp bufch6              ;stop, not stolen
 031375 045551 5 0          lda bufbch+bufbcl x     ;does bufrt1 match range star
 031376 013434 5 0          era bufrt1
 031377 100040 5 0          sze
 031400 003420 5 0           jmp bufch6              ;no, quit
 031401 045552 5 0          lda bufbch+bufbcl+1 x   ;does bufrt2 match range end?
 031402 013435 5 0          era bufrt2
 031403 100040 5 0          sze
 031404 003420 5 0           jmp bufch6              ;no, quit
 031405 051551 5 0          sta bufbch+bufbcl x     ;clear entry in bufbch
 031406 005434 5 0 bufch3: lda bufrt1
 031407 023435 5 0          cas bufrt2
 031410 101000 5 0           nop                     ;shouldn't happen
 031411 003420 5 0           jmp bufch6              ;done
 031412 024157 5 0          irs maxr                ;bump reassembly count
 031413 021423 5 0          jst buffre
 031414 005434 5 0          lda bufrt1
 031415 015704 5 0          add [buf1]
 031416 011434 5 0          sta bufrt1
 031417 003406 5 0          jmp bufch3

 031420 140040 5 0 bufch6: cra
 031421 011434 5 0          sta bufrt1
 031422 003357 5 0          jmp bufchx

                    .lev t.o
                    .lck all
 031423 000000 5 0 buffre: 0                       ;free up a buffer
 031424 010000 5 0          sta 0                   ;ptr in x
 031425 140040 5 0          cra
 031426 050000 5 0          sta ptrc x              ;clear chain word
 031427 141206 5 0          aoa
 031430 050004 5 0          sta wrdc x 0&usecnt     ;use count of 1
 031431 120115 5 0          jst flushi i            ;so flush will work
 031432 103423 5 0          jmp buffre i

                    .lev var
 031433         V   bufflu: .block 1               ;buffer for flush to steal
 031434         V   bufrt1: .block 1               ;these 2 specify a range to m
 031435         V   bufrt2: .block 1
 031436         V   bufrq1: .block 1               ;these 2 specify a range to s
 031437         V   bufrq2: .block 1
 031440         V   bufbsy: .block 1               ;busy: intermediate buffer ad
 031441         V   bufbs2: .block 1               ;saved bufrq2 value
 031442         V   bufidx: .block 1               ;saved index to free bufbch s
```

```
;
; bchsch: check if buffer is in a stolen range
; input: a=buffer address
; returns: +1 if buffer stolen, x=bufbch index
;             a=2nd range value
;          +2 if buffer not stolen, x is n.g.
;             a=input value
;
                .lev t.o
                .lck all
031443 000000 5 0 bchsch: 0
031444 011470 5 0         sta bchtmp            ; save input value
031445 073703 5 0         ldx [-bufbcl]         ;prepare to loop thru bufbch

031446 045551 5 0 bufsc1: lda bufbch+bufbcl x   ;get 1st word of bufbch pair

031447 101040 5 0         snz                   ; unused pair?
03145C 003460 5 0          jmp bufsc2           ;yes
031451 021471 5 0         jst adrcmp            ;compare entry vs. bchtmp
031452 003460 5 0          jmp bufsc2           ; bchtmp < entry, try next pa
031453 003466 5 0          jmp bufsc3           ; buftmp=entry; in range
031454 045552 5 0         lda bufbch+bufbcl+1 x ;maybe in range, test 2nd val
031455 021471 5 0         jst adrcmp
031456 003466 5 0          jmp bufsc3           ; <entry 2: in range
031457 101000 5 0          nop                  ; =entry 2: just out of range
03146C 024000 5 0 bufsc2: irs 0                 ; bchtmp>entry 2: not in rang
031461 024000 5 0         irs 0                 ; bump x by 2
031462 003446 5 0          jmp bufsc1           ;try next range
031463 025443 5 0         irs bchsch            ; return +2
031464 005470 5 0         lda bchtmp            ; return with input value
031465 103443 5 0         jmp bchsch i          ;not in any range

031466 045552 5 0 bufsc3: lda bufbch+bufbcl+1 x ;in range, return 2nd value
031467 103443 5 0         jmp bchsch i          ;return +1

                .lev var
03147C          V bchtmp: .block 1              ;temp
```

```
;
; adrcmp: compares two addresses
;
; inputs: a=addr1, bchtmp contains addr2
;
; returns: +1 if addr1 > addr2
;          +2 if addr1 = addr2
;          +3 if addr1 < addr2
;
; note: this will work with all addresses up to 64k
;
; "if signs differ, the answer is easy.
;  if the sign is the same, a cas will do." - anom
;
                    .lev t.o
                    .lck all
031471 000000 5 0 adrcmp: 0
031472 011510 5 0         sta addr1              ;save addr1
031473 140320 5 0         csa                    ;c-bit: 1 if addr1 has sign b
031474 005470 5 0         lda bchtmp
031475 101400 5 0         smi
031476 003505 5 0          jmp adr.up            ;go do a1=?, a2=+ case
031477 101001 5 0         ssc                    ; skips if c=1
031500 003504 5 0          jmp adr.pn            ; a1=+, a2=-
; come here if signs are the same, a=addr2
                    adr.nn:
                    adr.pp:
031501 023510 5 0         cas addr1
031502 025471 5 0 adr.np:  irs adrcmp            ;addr2>addr1
031503 025471 5 0          irs adrcmp            ;addr2=addr1
031504 103471 5 0 adr.pn: jmp adrcmp i           ;addr2<addr1

031505 101001 5 0 adr.up: ssc                    ; skip if c=1
031506 003501 5 0          jmp adr.pp            ; both are +
031507 003502 5 0         jmp adr.np             ;a1=-, a2=+, so a1<a2

                    .lev var
031510         V    addr1: .block 1
```

```
;
; buffer bench: keeps track of stolen buffers
;
; format: pairs of values
;
; 1st value: 0 - empty pair
;            else - first buffer of a contiguous stolen block
; 2nd value: first buffer after contiguous stolen block
;            not used when 1st value is zero
;
031511 000000 V   bufbch: .block bufbcl*2,0
031512 000000 V
031513 000000 V
031514 000000 V
031515 000000 V
031516 000000 V
031517 000000 V
031520 000000 V
031521 000000 V
031522 000000 V
031523 000000 V
031524 000000 V
031525 000000 V
031526 000000 V
031527 000000 V
031530 000000 V
031531 000000 V
031532 000000 V
031533 000000 V
031534 000000 V
031535 000000 V
031536 000000 V
031537 000000 V
031540 000000 V
031541 000000 V
031542 000000 V
031543 000000 V
031544 000000 V
031545 000000 V
031546 000000 V
031547 000000 V
031550 000000 V
031551 000000 V
031552 000000 V
031553 000000 V
031554 000000 V
031555 000000 V
031556 000000 V
031557 000000 V
031560 000000 V
031561 000000 V
031562 000000 V
031563 000000 V
031564 000000 V
031565 000000 V
031566 000000 V
031567 000000 V
031570 000000 V
031571 000000 V
```

```
031572 000000 V
031573 000000 V
031574 000000 V
031575 000000 V
031576 000000 V
031577 000000 V
031600 000000 V
031601 000000 V
031602 000000 V
031603 000000 V
031604 000000 V
031605 000000 V
031606 000000 V
031607 000000 V
031610 000000 V
```

```
                                                ; page 0: package hooks area

031611 006172 V   iniC0.hook:     jstini        ; init hooks - vdh
031612 006172 V   iniC1.hook:     jstini
031613 006172 V   iniC2.hook:     jstini
031614 006172 V   iniC3.hook:     jstini        ; stats, sfstats, eestats
031615 006172 V   iniC4.hook:     jstini        ; trace
031616 006172 V   iniC5.hook:     jstini        ; tty
031617 006172 V   iniC6.hook:     jstini        ; ddt
031620 006172 V   iniC7.hook:     jstini        ; hdlc
031621 006172 V   iniD0.hook:     jstini        ; hdh, x25
031622 006172 V   iniD1.hook:     jstini        ; cassette writer
031623 006172 V   iniD2.hook:     jstini
031624 006172 V   iniD3.hook:     jstini
031625 006172 V   iniD4.hook:     jstini
031626 006172 V   iniD5.hook:     jstini
031627 006172 V   iniD6.hook:     jstini
031630 006172 V   iniD7.hook:     jstini

031631 014311 V   bckC0.hook:     jstbck        ; background hooks
031632 014311 V   bckC1.hook:     jstbck
031633 014311 V   bckC2.hook:     jstbck
031634 014311 V   bckC3.hook:     jstbck
031635 014311 V   bckC4.hook:     jstbck
031636 014311 V   bckC5.hook:     jstbck
031637 014311 V   bckC6.hook:     jstbck
031640 014311 V   bckC7.hook:     jstbck
031641 014311 V   bckD0.hook:     jstbck
031642 014311 V   bckD1.hook:     jstbck
031643 014311 V   bckD2.hook:     jstbck
031644 014311 V   bckD3.hook:     jstbck
031645 014311 V   bckD4.hook:     jstbck
031646 014311 V   bckD5.hook:     jstbck
031647 014311 V   bckD6.hook:     jstbck
031650 014311 V   bckD7.hook:     jstbck

031651        V   pcwd00: .block 1             ;package control words
031652        V   pcwd01: .block 1
031653        V   pcwd02: .block 1
031654        V   pcwd03: .block 1
031655        V   pcwd04: .block 1
031656        V   pcwd05: .block 1
031657        V   pcwd06: .block 1
031660        V   pcwd07: .block 1
031661        V   pcwd10: .block 1
031662        V   pcwd11: .block 1
031663        V   pcwd12: .block 1
031664        V   pcwd13: .block 1
031665        V   pcwd14: .block 1
031666        V   pcwd15: .block 1
031667        V   pcwd16: .block 1
031670        V   pcwd17: .block 1
```

```
                .stitl background checksummer
;
; cxsub: background checksum rouitne
;
                .section pg23
                .lev bck
023203 000000 9 cxsub:  0
023204 103203 9         jmp cxsub i             ;&& doesn't work yet
023205 105570 9         lda [tim8s] i           ;use 8 second clock
023206 027251 9         ima cxtold              ;update cxtold
023207 013251 9         era cxtold              ;did clock just change?
023210 101040 9         snz
023211 103203 9          jmp cxsub i            ;no, not time yet.
023212 005251 9         lda cxtold              ;yes, did it become -1?
023213 141206 9         aoa
023214 100040 9         sze
023215 103203 9          jmp cxsub i            ;no, then not time yet.
023216 003231 9         jmp cxblok              ;init subtotal to 0

023217 016052 9 cxloop: sub [1]                 ;ptr to last word
023220 011253 9         sta cxptr               ;of piece
023221 056000 9         sub 0 x                 ;length of piece
023222 140407 9         tca                     ;-length
023223 010000 9         sta 0                   ;for post-indexing
023224 005252 9         lda cxstot              ;get current subtotal
023225 155253 9         add cxptr ix            ;add words in piece
023226 024000 9         irs 0
023227 003225 9          jmp .-2
023230 025254 9         irs cxndx               ;point to next piece
023231 011252 9 cxblok: sta cxstot              ;save subtotal
023232 073254 9         ldx cxndx
023233 044001 9         lda 1 x                 ;addr of end of next piece
023234 100040 9         sze                     ;is it end flag + 1?
023235 003217 9          jmp cxloop             ;no, do next piece
023236 005252 9         lda cxstot              ;yes, get total
;next 3 are for pat00 checksums
023237 140407 9         tca                     ;form checksum
023240 050002 9         sta 2 x                 ;and put in table
023241 101000 9         nop
       023237           .=.-3
;now, the 3 real ones
023237 054002 9         add 2 x                 ;compare with checksum
;        sze                    ;ok?
023240 100000 9          skp                    ;?? for debugging
023241 000000 9 cxnop:   %crash                 ;background found code checksu
023242 005254 9         lda cxndx
023243 014011 9         add three
023244 023571 9         cas [cxend]             ;check for wrap-around
023245 101000 9          nop
023246 005572 9          lda [cxtab]            ;if so, reset to top of list

023247 011254 9         sta cxndx
023250 103203 9         jmp cxsub i

                .lev var
023251         V cxtold: .block 1               ;to detect clock change
023252         V cxstot: .block 1               ;checksum subtotal
023253         V cxptr:  .block 1               ;pointer to next data word
023254         V cxndx:  .block 1               ;current index into cxtab (se
```

```
;
; table for background checksum routine
;
; format: addr-1 of piece 1
;         addr-1 of piece 2 of block 1
;            (=addr+1 of end of piece 1)
;         ...
;
;         addr-1 of piece n of block 1
;         addr+1 of end of piece n
;         0
;         checksum (= -sum of pieces) of block 1
;         addr-1 of piece 1 of block 2
;         ...etc.
;
                .lev var
023255 000004 V   cxtab:  c77-1
023256 000077 V           m30sec+1
023257 000000 V           0
023260 000000 V   cxs2:   0
023261 032777 V           m2irup-1
023262 033142 V           m2irp9
023263 000000 V           0
023264 000000 V   cxs3:   0
023265 007144 V           i2msru-1
023266 007203 V           i2mrch
023267 000000 V           0
023270 000000 V   cxs4:   0
023271 033234 V           delay
023272 033434 V           compar
023273 033512 V           cx22e
023274 000000 V           0
023275 000000 V   cxsa:   0
023276 034002 V           rtset
023277 034021 V           rtoff
023300 034025 V           rton
023301 034034 V           bckno
023302 034057 V           spfini
023303 034230 V           rupdq
023304 034245 V           rupfls
023305 034276 V           rupmst
023306 034337 V           rupq
023307 000000 V           0
023310 000000 V   cxs5:   0
023311 034337 V           rupq
023312 034655 V           cx27b
023313 000000 V           0
023314 000000 V   cxsb:   0
023315 035000 V           rtinc
023316 035106 V           fmdmn
023317 035135 V           retree
023320 035346 V           search
023321 035421 V           router
023322 000000 V           0
023323 000000 V   cxs6:   0
023324 035421 V           router
023325 035572 V           naylps
023326 035603 V           naylpi
023327 035613 V           cas377
```

```
023330 035651 V             bckne
023331 000000 V             0
023332 000000 V   cxs8:     0
023333 036007 V             rupenq
023334 036054 V             fndn
023335 036101 V             spfabt
023336 036250 V             spfalg
023337 000000 V             0
023340 000000 V   cxs9:     0
023341 036250 V             spfalg
023342 036405 V             fndent
023343 036571 V             cx35a
023344 000000 V             0
023345 000000 V   cxs10:    0
                  cxend:
```

```
                .stitl tables and blocks
                .section heap
                .lev var

;spf tables
                lflag:
       041527   lflag1=lflag-1
041530      V   ntb:    .block nnodes+1+1       ;1 extra for last entry's end
041732      V   pdist:  .block nnodes+1
042133      V   lst:    .block nnodes+1

                .lev var
042334      V   ltb:    .block nlines           ;line table for spf

;used by [t.o,bck]
                        ;defplc(/spfrut)
 ;used by [h2i,t.o,tsk,bck]
043074      V   spfrut: .block nnodes+1         ;route use table
```

```
                .stitl end-to-end tables, leader areas, interrupt stack
                .lev var

;** note **
;receive or transmit tables must not overlap 100000 boundary
                        ;defplc(receive message block table)
043275          V   rmblkt: .block rmbl*nmb

                        ;defplc(transmit message block table)
044015          V   tmblkt: .block tmbl*nmb

                        ;defplc(tsbtab - transaction block table)
044445          V   tsbtab: .block tsbl*ntsb        ;table of message transactio

                        ;defplc(reastb - reassembly block table)
044775          V   reastb: .block reasl*nreab      ;reassembly store
```

```
                .stitl DISPLAY DRIVER
                .INCLUDE dpy.m4


;NMFS IMP Display Driver (runs the lights).  In the NMFS IMP, the lights
;look like this:
;
;
;       19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
;       ||  \____/   \                                             /
;       ||  unused    \___________________________________________/
;       ||                  Status of Hosts or Modems 15-0
;       ||
;      Mode
;
;If Mode = 0, bits 0-15 refer to the modem status.  If a given modem's
;bit is off, that modem is down.  If it is on, that modem is up, if it
;is flashing, that modem is up and looped.
;
;If Mode = 1, bits 0-15 refer to the host status.  If a given host's
;bit is off, that hosts is down.  If it is on, that host is up, if it
;is flahsing, that host is tardy.
;
;By the way, the NMFS %LITES instruction (opcode 11) places the A register
;in the bits 0-15 of the lights and the low 4 bits of X in bits 16-19.

                .section pg23
                .lev dpy

023346 140040 8 dpyini: cra                     ;start with everything exting
023347 010000 8         sta 0
02335C 000011 8         %lites

;Endlessly update the lights with modems/hosts status.  If VIEW is set
;non-zero, view that location instead.  Also look for NICE STOP in NCCC.

023351 004101 8 dpylop: lda nccc                ;nice-stop or panic stop?
023352 012024 8         era [ ncc.reld ]
023353 101040 8         snz
023354 003372 8          jmp dpyrld
023355 012072 8         era [ ncc.reld ? ncc.stop ]
023356 100040 8         sze
023357 003375 8          jmp dpylp2

;Here on nice stops.  First, tell all the hosts to send in an IMP going
;down and then to be silent.  After that, sleep for 30 seconds and enter
;the loader with the arg in A.  Note that we leave the NCCC word equal to
;NCC.STOP so that it shows up in the dump.

023360 072032 8         ldx [-nh]               ;for each real host
023361 165573 8 nicel:  irs [ihfl+nh] ix        ;set the LSB (send in nice-st
023362 024000 8         irs 0
023363 003361 8          jmp nicel              ;loop

023364 005574 8         lda [ %30sec ]
023365 000020 8         pcb
023366 000203 8         tpr
023367 000103 8         spr
```

```
023370 100000 8              skp
023371 000000 8               %crash                ;NICE STOP: got poked
023372 001001 8   dpyrld: .inh all                  ;inhibit the world (here on N
023373 004100 8 0          lda ncca                 ;get the arg pointer
023374 103575 8 0          jmp [1001] i             ;enter the loader

023375 004107 8 0 dpylp2:  lda view                 ;anything to view?
023376 100040 8 0          sze
023377 003403 8 0           jmp dpyvew              ;yes, go view it
023400 021414 8 0          jst dpymod               ;do modems
023401 021460 8 0          jst dpyhst               ;then hosts
023402 003351 8 0          jmp dpylop               ;then loop forever

023403 104107 8 0 dpyvew:  lda view i               ;get thing to be viewed
023404 072006 8 0          ldx [ 0 ]                ;clear X
023405 000011 8 0          %lites                   ;update the lights
023406 004056 8 0          lda [ %25.6ms ]          ;sleep for a little while
023407 000020 8 0          pcb
023410 000203 8 0          tpr
023411 000103 8 0          spr
023412 003351 8 0          jmp dpylop
023413 000000 8 0          %crash                ;DPYVEW: got poked
```

```
;Here to do Modem status (MODE bit 19 = 0).  First, we light all lights
;who neighbor entry is non-zero (even if it's us), then we sleep for 50ms,
;then light only those neighbors which aren't us, thus extinguishing looped
;lines.  After sleeping for another 50ms in this mode, we loop back 50 times,
;for a total display time of 5 seconds.

023414 000000 8 0 dpymod: 0
023415 140040 8 0         cra                   ;force hi bits off
023416 011543 8 0         sta dpyhgh

023417 005576 8 0         lda [ -50. ]
023420 011545 8 0         sta dpyitr            ;count the iterations

023421 005577 8 0 dpmlop: lda [ dpmd1 ]         ;get addr of first subr
023422 073566 8 0         ldx [ -ch ]
023423 021511 8 0         jst dpysub            ;do the display
023424 005600 8 0         lda [ dpmd2 ]         ;then the second
023425 073566 8 0         ldx [ -ch ]
023426 021511 8 0         jst dpysub
023427 025545 8 0         irs dpyitr
023430 003421 8 0          jmp dpmlop
023431 103414 8 0         jmp dpymod i          ;return to caller

;First subroutine.  All neighbors which are non-zero.  X=modem number.

023432 000000 8 0 dpmd1:  0
023433 021446 8 0         jst dpmds             ;do common part of MD1/MD2
023434 103432 8 0          jmp dpmd1 i          ;modem doesn't exist, or has
023435 025432 8 0         irs dpmd1
023436 103432 8 0         jmp dpmd1 i

;Second subroutine.  All neighbors except us.

023437 000000 8 0 dpmd2:  0
023440 021446 8 0         jst dpmds
023441 103437 8 0          jmp dpmd2 i
023442 012106 8 0         era mine
023443 100040 8 0         sze
023444 025437 8 0          irs dpmd2
023445 103437 8 0         jmp dpmd2 i

;Subroutine to do most of the work.

023446 000000 8 0 dpmds:  0
023447 145601 8 0         lda [mblock] ix
023450 101040 8 0         snz
023451 103446 8 0          jmp dpmds i          ;if not configured, skip it
023452 010000 8 0         sta 0
023453 044117 8 0         lda line x            ;is line up?
023454 101040 8 0         snz
023455 025446 8 0          irs dpmds            ;if up, prepare to skip
023456 044116 8 0         lda neighb x          ;get neighbor for caller
023457 103446 8 0         jmp dpmds i
```

```
;Here to do Host status (MODE bit 19 = 1).  First, we light all lights
;who HOST entry is not HSTOFF (up or tardy), then we sleep for 50ms,
;then light only those neighbors which are up, thus extinguishing tardy/I-G-D
;hosts.  After sleeping for another 50ms in this mode, we loop back 50 times,
;for a total display time of 5 seconds.

023460 000000 8 0 dpyhst: 0
023461 004055 8 0         lda [ 10 ]            ;get mode bit
023462 011543 8 0         sta dpyhgh            ;set it

023463 005576 8 0         lda [ -50. ]
023464 011545 8 0         sta dpyitr            ;count the iterations

023465 005602 8 0 dphlop: lda [ dphd1 ]         ;get addr of first subr
023466 072032 8 0         ldx [ -nh ]
023467 021511 8 0         jst dpysub            ;do the display
023470 005603 8 0         lda [ dphd2 ]         ;then the second
023471 072032 8 0         ldx [-nh]             ;number of hosts
023472 021511 8 0         jst dpysub
023473 025545 8 0         irs dpyitr
023474 003465 8 0          jmp dphlop
023475 103460 8 0         jmp dpyhst i          ;return to caller

;First subroutine.  All non-down hosts.

023476 000000 8 0 dphd1:  0
023477 144276 8 0         lda hosti ix          ;get host status
023500 100040 8 0         sze 0&hstoff
023501 025476 8 0          irs dphd1
023502 103476 8 0         jmp dphd1 i

;Second subroutine.  All up hosts.

023503 000000 8 0 dphd2:  0
023504 144276 8 0         lda hosti ix
023505 012052 8 0         era [hstup]
023506 101040 8 0         snz
023507 025503 8 0          irs dphd2
023510 103503 8 0         jmp dphd2 i
```

```
;This is the main display subroutine.  Called with A=address of subroutine
;X=iteration number.  These subrcutines should skip if they
;want the light lit.

023511 000000 8 0 dpysub: 0
023512 033550 8 0         stx dpyc                ;save counter
023513 011544 8 0         sta dpysbr              ;save subroutine address
023514 140040 8 0         cra                     ;start display at zero
023515 010000 8 0         sta 0                   ;start at 0
023516 011546 8 0         sta dpydpy

023517 033547 8 0 dpysb1: stx dpyx                ;save X across call
023520 121544 8 0         jst dpysbr i            ;call the subroutine
023521 003526 8 0          jmp dpynxt             ;rotate and do next
023522 073547 8 0         ldx dpyx
023523 044052 8 0         lda bittab x
023524 013546 8 0         era dpydpy
023525 011546 8 0         sta dpydpy
023526 073547 8 0 dpynxt: ldx dpyx                ;get counter
023527 024000 8 0         irs 0
023530 025550 8 0         irs dpyc                ;more to go?
023531 003517 8 0          jmp dpysb1             ;yes

;When all done, pick up the high bits and display it

023532 005546 8 0         lda dpydpy
023533 073543 8 0         ldx dpyhgh
023534 000011 8 0         %lites
023535 004057 8 0         lda [ 2. * %25.6ms ]    ;about 50ms
023536 000020 8 0         pcb
023537 000203 8 0         tpr
023540 000103 8 0         spr
023541 103511 8 0         jmp dpysub i            ;return
023542 000000 8 0         %crash                 ;DPYSUB: got poked

                     .lev var
023543          V    dpyhgh: .block 1             ;holds high 4 bits
023544          V    dpysbr: .block 1             ;addr of current subroutine
023545          V    dpyitr: .block 1             ;iteration counter
023546          V    dpydpy: .block 1             ;display value
023547          V    dpyx:   .block 1             ;display index
023550          V    dpyc:   .block 1             ;display count
```

```
                .stitl ROUTING
                .INCLUDE spf.m4
;routing
```

```
                .stitl spf- modem processing of updates
                .section pg33

;routing update packet field definitions

 ;neth
        100000      rupret=100000                ;1=packet is a retry, must bc
        000200      rupre2=200                   ;copied rupret bit (by m2irup
        070000      rupage=70000                 ;age of information about thi
        007400      rupnn=7400                   ;number of neighbor entries

 ;typh
        001400      ruptyp=1400                  ;subtype of switch type for t

 ;srch
        037400      rupsno=37400                 ;serial number
        020000      rupsho=20000                 ;high order bit of serial num
        000400      rupslo=400                   ;low order bit of serial numb
        000377      rupsrc=377                   ;source imp number

 ;neighbor entry - starts at seqh
        177400      rupdel=177400                ;delay (distance) along this
        000377      rupnei=377                   ;neighbor number at other end

        000003      rupm=artm                    ;bit mask for who should proc
        000017      ruplin=17                    ;bits for modems(m1=1,m2=2,m3
        100000      ruptsk=100000                ;bit for task
        040000      rupfwd=40000                 ;bit for update forwarding pa
```

```
                        .lev m2i
033000 072233 1   m2irup: ldx m2isp                ;get packet pointer
033001 140040 1         cra
033002 050003 1         sta rupm x                 ;clear queue mask
033003 044006 1         lda neth x
033004 007131 1         ana [-1?rupre2]            ;first, clear rupre2 bit
033005 100400 1         spl                        ;retry bit on?
033006 013132 1          era [rupret?rupre2]       ;yes, turn it off, set rupre2
033007 066006 1         ima neth x
033010 056006 1         sub neth x
033011 054010 1         add chkh x
033012 050010 1         sta chkh x                 ;adjust checksum
033013 044006 1         lda neth x                 ;was this a retry?
033014 006061 1         ana [rupre2]               ;(test the copied bit)
033015 101040 1         snz
033016 003023 1          jmp m2irck                ;no, skip down
033017 072232 1         ldx mp                     ;yes, get modem number
033020 044052 1         lda bittab x               ;select bit for mask
033021 072233 1         ldx m2isp
033022 050003 1         sta rupm x                 ;mark for output over input 1
033023 044006 1   m2irck: lda neth x
033024 006050 1         ana [rupage]               ;check age
033025 101040 1         snz                        ;is it zero?
033026 003107 1          jmp m2inac                ;then it is unacceptable (bug
033027 044011 1         lda srch x
033030 011142 1         sta m2ircs                 ;save serial number
033031 141050 1         cal 0&rupsrc               ;get source imp number
033032 011143 1         sta m2iri
033033 010000 1         sta 0
033034 012106 1         era mine                   ;check for dup imp # already
033035 100040 1         sze
033036 003042 1          jmp m2irp0                ;not a duplicate
033037 005144 1         lda iniflg                 ;are we still in init?
033040 101040 1         snz
033041 000000 1          %crash                    ;yes! another imp with same nu
033042 144131 1   m2irp0: lda spfrti ix             ;check age of data base
033043 007133 1         ana [spfage]
033044 101040 1         snz                        ;is our age for this imp zero
033045 003062 1          jmp m2iacc                ;yes, then packet will be acc
033046 144131 1         lda spfrti ix              ;no, must use serial numbers
033047 006047 1         ana [spfsno]
033050 027142 1         ima m2ircs                 ;save current serial number
033051 006047 1         ana [rupsno]
033052 017142 1         sub m2ircs                 ;input serial# - current seri
033053 101040 1         snz                        ;are they equal?
033054 003115 1          jmp m2ira1                ;yes, must be an ack or rebo
033055 100400 1         spl                        ;input < current?
033056 014070 1          add [rupsho+rupsho]       ;yes, add 32.
033057 016067 1         sub [rupsho]               ;no, subtract 32.
033060 101400 1         smi                        ;was difference < 32?
033061 003105 1          jmp m2ina1                ;no, then input not acceptabl
033062 072233 1   m2iacc: ldx m2isp
033063 044006 1         lda neth x
033064 141140 1         icl 0&spfage               ;get new age
033065 011142 1         sta m2ircs
033066 044011 1         lda srch x
033067 006047 1         ana [rupsno]
033070 013142 1         era m2ircs                 ;and serial number
033071 073143 1         ldx m2iri
```

```
033072 152131 1            era spfrti ix          ;update source node's spfrut

033073 007134 1            ana [spfage+spfsno]
033074 152131 1            era spfrti ix
033075 150131 1            sta spfrti ix
033076 121135 1            jst [rupmst] i         ;set queue mask in packet
033077 072233 1            ldx m2isp
033100 050003 1            sta rupm x
033101 072201 1   m2irak:  ldx mbl
033102 005143 1            lda m2iri
033103 121136 1            jst [rtoff] i          ;turn off timer for input lin
033104 100000 1            skp
033105 025145 1   m2ina1:  irs rupobs             ;count an obsolete msg rec'd

033106 101000 1             nop
033107 072233 1   m2inac:  ldx m2isp
033110 044003 1            lda rupm x
033111 101040 1            snz                    ;have any bits been set?
033112 103137 1             jmp [m2ifre] i        ;no, then flush packet
033113 121140 1            jst [rupenq] i         ;yes, set enq, count
033114 103141 1            jmp [m2idun] i

033115 025146 1   m2ira1:  irs rupdp1             ;count dups
033116 101000 1             nop
033117 072233 1            ldx m2isp
033120 044006 1            lda neth x
033121 006061 1            ana [rupre2]
033122 100040 1            sze
033123 025147 1             irs rupdp2            ;count again if retry bit was
033124 101000 1              nop
033125 072201 1            ldx mbl
033126 004052 1            lda [1]
033127 050201 1            sta rup4to x           ;set flag for t.o to tell i2m
033130 003101 1            jmp m2irak

033131 177577 1   .constants
033132 100200 1
033133 000160 1
033134 037560 1
033135 034276 1
033136 034021 1
033137 005434 1
033140 036007 1
033141 005363 1

;confix(-1)
                   m2irp9:                        ;end of checksummed area

                   .lev var
033142         V   m2ircs: .block 1
033143         V   m2iri:  .block 1
033144         V   iniflg: .block 1               ;init sets to -1, first line
033145         V   rupobs: .block 1               ;# rups judged too old
033146         V   rupdp1: .block 1               ;# rups current but already s
033147         V   rupdp2: .block 1               ;# rups current, seen and re
```

```
                .stitl a.r.s. delay package
; *** delay modules for ars

;parameters to be set up from outside
                dparmt:                         ;*** params 100-177 *** delay

033150 000003 V   delshf: 3                     ;right shyfts for each delay

033151 000003 V   smoshf: 3                     ;right shyfts for average del
033152 177176 V   smofrq: -386.                 ;smoother calling frequency
033153 000010 V   thrini: 10                    ;threshold init value
033154 000002 V   thrdcy: 2                     ;threshold decay value
;this table gives the transmission times for various sized
;packets on each of the 4 line speeds.  the first 8 values
;in the table are for 50kb lines, the second 8 for 230kb lines,
;the third 8 for 19.2 kb lines, and the last 8 for 9.6kb lines.
;the first value of each group gives the average time to
;transmit a packet of length 0-7 words on a line of that speed
;(more precisely, the time to transmit a packet with 3.5 data
;words), the second value of each group gives the average time
;to transmit a packet with 8-15 data words (exactly 11.5), etc.
;these values  the time to transmit the 200 bit header,
;but the table is indexed by the number of data words.  this
;table is used by task to fix up artm for the delay calculation.
;it is also used to find out the maximum transmission time for
;a line when determining the value for a retransmission timer.
033155 000063 V   delxmt: 51.                   ;first 8: 50kb line
033156 000115 V           77.
033157 000146 V           102.
033160 000200 V           128.
033161 000232 V           154.
033162 000263 V           179.
033163 000315 V           205.
033164 000346 V           230.
033165 000013 V           11.                   ;second 8: 230kb line
033166 000021 V           17.
033167 000026 V           22.
033170 000034 V           28.
033171 000041 V           33.
033172 000047 V           39.
033173 000054 V           44.
033174 000062 V           50.
033175 000205 V           133.                  ;third 8: 19.2kb line
033176 000310 V           200.
033177 000413 V           267.
033200 000515 V           333.
033201 000620 V           400.
033202 000723 V           467.
033203 001025 V           533.
033204 001130 V           600.
033205 000413 V           267.                  ;last 8: 9.6kb line
033206 000620 V           400.
033207 001025 V           533.
033210 001233 V           667.
033211 001440 V           800.
033212 001645 V           933.
033213 002053 V           1067.
033214 002260 V           1200.
```

```
000045      dparm1=.-dparmt                 ;length
```

```
;variables for delay smoother (averager) and comparer
033215          V   smosum: .block 1                ;sum copy
033216          V   smosmo: .block 1                ;overflow copy
033217          V   smocnt: .block 1                ;packet count copy
033220          V   smotmp: .block 1                ;division temporary
033221          V   smoshn: .block 1                ;shyft counter for average r
033222          V   smocur: .block 1                ;current average delay
033223          V   smobit: .block 1                ;division temp
033224          V   smosvb: .block 1                ;saved b register

033225          V   rupsnd: .block 1                ;flag to generate spf update
```

```
                ;variables for delay adder
033226          V   delayp: .block 1                ;packet pointer
033227          V   delayl: .block 1                ;line number
033230          V   delay1: .block 1                ;sub-total

                ;stats hook for smoother
                        stats.dadeli:
033231 033232   V               jstspf

033232 000000   V   jstspf: 0
033233 103232   V               jmp jstspf i

                ;delay is computed and added to sum, right shyfted first
                        .lev (m2i,i2m)
033234 000000   1   delay:  0
033235 033226   1           stx delayp              ;save x and a
033236 011227   1           sta delayl              ;called with modem block ptr
033237 044121   1           lda itst x              ;compute processing delay
03324C 056003   1           sub artm x              ;artm=arrival time-transmit
033241          1   delays: .block 1                ;divide by power of 2
033242 011230   1           sta delay1
033243 073227   1           ldx delayl
033244 044030   1           lda m2ipcb.delay x
033245 015230   1           add delay1              ;add to subtotal
033246 054211   1           add delsum x
033247 050211   1           sta delsum x            ;add to sum of delays
033250 100001   1           src                     ;add overflow if any
033251 064212   1            irs delsmo x
033252 101000   1            nop
033253 064213   1           irs delcnt x            ;count packet
033254 073226   1           ldx delayp              ;restore x
033255 103234   1           jmp delay i             ;and exit.
```

```
;smoother (averager) for delay
                    .lev t.o
033256 005150 5   smooth: lda delshf          ;shyft parm for delay
033257 006072 5           ana [17]            ;insure it's in range
033260 140407 5           tca
033261 014046 5           add [lgr 0]         ;make shyft instruction
033262 011241 5           sta delays          ;and plant in the code
033263 004314 5           lda smoclk          ;examine clock
033264 015503 5           add [ch]            ;within ch of end?
033265 100400 5           spl
033266 003417 5            jmp smootn         ;no, not time to act
033267 010000 5           sta 0               ;yes, x has line number
033270 145504 5           lda [mblock] ix     ;is this an IMP modem?
033271 101040 5           snz
033272 003417 5            jmp smootn         ;no, skip it
033273 010000 5           sta 0
033274 044117 5           lda line x          ;line up?
033275 100040 5           sze
033276 003427 5            jmp smootk         ;no, report special values
033277 011221 5           sta smoshn          ;zero shyft counter
033300 001001 5           .inh m2i
;preserve integrity of counts
033301 121231 5 1         jst stats.dadeli i  ;** stats
033302 066213 5 1         ima delcnt x        ;clear and copy count of pack
033303 101040 5 1         snz                 ;zero packets summed?
033304 003431 5 1          jmp smoote         ;yes, handle as special case

033305 011217 5 1         sta smocnt          ;save copy
033306 140040 5 1         cra                 ;clear and copy sum of delays
033307 066211 5 1         ima delsum x
033310 011215 5 1         sta smosum
033311 140320 5 1         csa                 ;save sign for overflow fiddl
033312 140040 5 1         cra                 ;clear and copy overflows
033313 066212 5 1         ima delsmo x
033314 000401 5 1         .enb t.o
033315 100001 5           src                 ;was delay sum minus?
033316 016052 5            sub [1]            ;yes, discount "false" overfl
033317 011216 5           sta smosmo          ;(see description of h'well o
033320 005215 5           lda smosum          ;first shyft: reduce to 15
033321 000201 5           iab                 ;significant bits
033322 011224 5           sta smosvb          ;save b
033323 005216 5           lda smosmo
033324 101040 5   smoot0: snz                 ;bits shyfted out of a?
033325 003331 5            jmp smoot1         ;yes, continue
033326 040077 5           lrl 1               ;no, lng rght logical 1
033327 025221 5           irs smoshn          ;and count it
033330 003324 5           jmp smoot0
```

```
033331 011220 5   smoot1: sta smotmp          ;zero division variable
033332 011222 5          sta smocur          ;quotient added on below
033333 140500 5          ssm                 ;bit for division
033334 011223 5          sta smobit
033335 005217 5          lda smocnt          ;if count (divisor) has 16 bi
033336 101400 5          smi
033337 003345 5           jmp smootd         ;(no, it is ok)
033340 040477 5          lgr 1               ;reduce by one bit
033341 011217 5          sta smocnt
033342 005221 5          lda smoshn          ;and compensate shyft counter
033343 016052 5          sub [1]
033344 011221 5          sta smoshn
033345 005220 5   smootd: lda smotmp         ;a & b hold remainder
033346 041077 5          lll 1               ;long left logical 1
033347 011220 5          sta smotmp
033350 023217 5          cas smocnt          ;a .ge. divisor?
033351 101000 5           nop
033352 100000 5           skp                ;yes
033353 003361 5            jmp smoot2        ;no.. go another bit position
033354 017217 5          sub smocnt          ;subtract divisor
033355 011220 5          sta smotmp
033356 005223 5          lda smobit          ;and add bit into quotient
033357 015222 5          add smocur
033360 011222 5          sta smocur
033361 005223 5   smoot2: lda smobit
033362 040477 5          lgr 1               ;shyft current bit
033363 011223 5          sta smobit
033364 100040 5          sze                 ;if done, fall out of loop
033365 003345 5           jmp smootd         ;   no, next loop
033366 005224 5          lda smosvb          ;restore b register
033367 000201 5          iab
033370 005151 5          lda smoshf          ;division done, finish 2nd sh
033371 006072 5          ana [17]
033372 140407 5          tca                 ;negative # places left to sh
033373 015221 5          add smoshn
033374 100400 5          spl                 ;already shyfted far enough?

033375 003403 5           jmp smoot4         ;not yet
033376 101040 5          snz                 ;shyfted exactly enough?
033377 003410 5           jmp smoot6         ;yes, need not shyft
033400 140407 5          tca                 ;shyfted too far, must left s
033401 015505 5          add [lgl 0]
033402 003404 5          jmp smoot5
```

```
033403 014046 5    smoot4: add [lgr 0]          ;build shyft to finish right
033404 011406 5    smoot5: sta smoots
033405 005222 5            lda smocur
033406        5    smoots: .block 1             ;computed shyft instr
033407 011222 5            sta smocur           ;save answer
033410 005222 5    smoot6: lda smocur           ;get propagation + current de
033411 101040 5    smoot8: snz
033412 141206 5             aoa                 ;delay on a line cannot = 0
033413 022007 5            cas [dlinf]          ;mustn't reach max value for
033414 101000 5             nop
033415 005506 5              lda [dlinf-1]      ;it reached... legal maximum

033416 050214 5    smoot7: sta delcur x         ;save away as current delay
033417 024314 5    smootn: irs smoclk           ;tick clock--done?
033420 003426 5             jmp smootx          ;no.. exit
033421 005152 5            lda smofrq           ;yes, reset clock
033422 101400 5            smi                  ;if not negative,
033423 004071 5             lda sign            ;then force very negative
033424 010314 5            sta smoclk           ;also have just finished all
033425 021434 5            jst compar           ;so call comparer
033426 102124 5    smootx: jmp toret i          ;exit

033427 004007 5    smootk: lda [dlinf]          ;line down, max value
033430 003416 5            jmp smoot7

                   .lck m2i
033431 121507 5 1  smoote: jst [lnupsb] i       ;get propagation delay
033432 000401 5 1          .enb t.o
033433 003411 5            jmp smoot8
```

```
                    ;compare module - called by smoother
                                      .lev t.o
033434 000000 5     compar: 0
033435 073510 5             ldx [-ch]
033436 032001 5     compa0: stx ireg
033437 145511 5             lda [mblock+ch] ix      ;does modem exist?
033440 101040 5             snz
033441 003453 5              jmp compl              ;no
033442 010000 5             sta 0
033443 044214 5             lda delcur x            ;compute current-base
033444 056215 5             sub delbas x
033445 100400 5             spl                     ; (absolute value)
033446 140407 5              tca
033447 022315 5             cas thresh              ;compare with current thresho
033450 003462 5              jmp compau             ;too big, must send update
033451 003462 5              jmp compau             ;send if equal, too
033452 072001 5             ldx ireg
033453 024000 5     compl:  irs 0                   ;loop over lines
033454 003436 5              jmp compa0
033455 004315 5             lda thresh              ;none over, so decrement thre
033456 017154 5             sub thrdcy              ;by decay value
033457 101400 5             smi                     ;but not below zero
033460 010315 5              sta thresh
033461 103434 5     compax: jmp compar i            ; quit

033462 073510 5     compau: ldx [-ch]               ;go back and loop over all li
                    compa2:
033463 032001 5             stx ireg
033464 145511 5             lda [mblock+ch] ix      ;does modem exist?
033465 101040 5             snz
033466 003475 5              jmp compl2             ;no
033467 010000 5             sta 0
033470 001001 5             .inh m2i
033471 044214 5 1           lda delcur x
033472 050215 5 1           sta delbas x            ;copy current to base delay
033473 000401 5 1           .enb t.o
033474 072001 5             ldx ireg
033475 024000 5     compl2: irs 0
033476 003463 5              jmp compa2             ;loop over lines
033477 005153 5             lda thrini              ;reset threshold to
033500 010315 5             sta thresh              ;init value
033501 011225 5             sta rupsnd              ;send real spf update
033502 003461 5             jmp compax              ; quit
```

```
033503 000016 5          .constants
033504 046172 5
033505 041500 5
033506 000376 5
033507 022662 5
033510 177762 5
033511 046210 5
;confix(-3)
       033512     cx22e=.
```

```
                        .stitl spf- retransmit timer routines

                        .section pg34
                        .lev var
034000          V       rttemp: .block 1
034001          V       rtino:  .block 1

                        .lev (m2i,t.o,spf)
034002 000000 1         rtset:  0                       ;setup for set or clear retra
034003 032001 1                 stx ireg                ;save x (modem block pointer)
034004 011001 1                 sta rtino               ;save a
034005 040475 1                 lgr 3                   ;impno\8
034006 054205 1                 add rtptrs x
034007 011000 1                 sta rttemp              ;address of word with timer w
034010 005001 1                 lda rtino
034011 006013 1                 ana [7]
034012 041477 1                 lgl 1
034013 010000 1                 sta 0                   ;index into bittab
034014 044052 1                 lda bittab x            ;get one bit we want
034015 052053 1                 era bittab+1 x          ;or in two bit we want
034016 140401 1                 cma                     ;complement mask
034017 107000 1                 ana rttemp i            ;clear our timer
034020 103002 1                 jmp rtset i

034021 000000 1         rtoff:  0                       ;clear a retrans timer
034022 021002 1                 jst rtset               ;get word with our timer clea
034023 111000 1                 sta rttemp i            ;update timers
034024 103021 1                 jmp rtoff i

034025 000000 1         rton:   0                       ;set a retrans timer
034026 021002 1                 jst rtset               ;get wrd with our timer clear
034027 052052 1                 era bittab x
034030 052053 1                 era bittab+1 x          ;set both bits for our timer

034031 111000 1                 sta rttemp i            ;update timers
034032 072001 1                 ldx ireg                ;preserve x
034033 103025 1                 jmp rton i
```

```
;bcknc - find entry in x's nay table that points to father
;returns with x = index for that entry, a with that lflag
;returns+1 if no back link, else +2

                    .lev spf
034034 000000 6   bckno:  0
034035 145617 6           lda [ntb+1] ix
034036 006033 6           ana [ntbidx]          ;must mask off index bits
034037 111620 6           sta [bckne] i
034040 145621 6           lda [ntb] ix
034041 006033 6           ana [ntbidx]
034042 010000 6           sta 0
034043 145621 6   bckno1: lda [lflag] ix
034044 006070 6           ana [lflgf?lflgb]     ;is this line on tree to fath
034045 100040 6           sze
034046 003054 6            jmp bcknx            ;yes, done
034047 024000 6           irs 0                 ;no. checked all of x's lines
034050 105620 6           lda [bckne] i
034051 022000 6           cas 0
034052 003043 6            jmp bckno1           ;no, loop for next line.
034053 100000 6            skp                  ;yes, found no back link. ret
034054 025034 6   bcknx:    irs bckno           ;success, return +2
034055 145621 6           lda [lflag] ix
034056 103034 6           jmp bckno i
```

```
                        .stitl spf- initialization
                        .lev spf

                        spfini:                                 ;init spf database - here ini
034057 073622 6                 ldx [-nnodes-1]                 ;first the node tables
034060 004071 6         spfi1:  lda [spfded]                    ;imps dead to begin with
034061 150132 6                 sta spfrei ix
034062 004044 6                 lda [pdinf] 0&1flag             ;this zeroes lflag, too
034063 151623 6                 sta [pdist+nnodes+1] ix         ;dead imps have inf dist
034064 140040 6                 cra 0&1flag
034065 151624 6                 sta [ntb+nnodes+1+1] ix
034066 151625 6                 sta [lst+nnodes+1] ix
034067 024000 6                 irs 0
03407C 003060 6                  jmp spfi1
034071 005626 6                 lda [nlines+1]                  ;set extra ntb entry to be
034072 111627 6                 sta [ntb+nnodes+1] i            ;too high
034073 073630 6                 ldx [-nlines]                   ;now clear line table
034074 004010 6                 lda [dlinf<<8.]                 ;init all entries as dead,nay
034075 150130 6         spfi2:  sta ltbei ix
034076 024000 6                 irs 0
034077 003075 6                  jmp spfi2
034100 072106 6                 ldx mine                        ;now ixnit entries for this m
034101 140040 6                 cra
034102 150131 6                 sta spfrti ix                   ;reachable, no route,serial#
034103 151624 6                 sta [pdist] ix                  ;distance zero
034104 103631 6                 jmp [spfdb] i                   ;enter mainline SPF
```

```
                        .stitl spf- generate update
                        .lev t.o
034105 140040 5   rupgen: cra                     ;check for routing update to
034106 011672 5           sta rupgnn              ;zero count for later
034107 011660 5           sta rupc                ;init checksum subtotal
034110 127632 5           ima [rupsnd] i
034111 101040 5           snz                     ;send flag set?
034112 102124 5            jmp toret i            ;no, then exit.
034113 105633 5           lda [thrini] i
034114 111634 5           sta [thresh] i          ;reset thresh
034115 105635 5           lda [iniflg] i          ;still in init?
034116 101040 5           snz
034117 102124 5            jmp toret i            ;yes, then exit.
034120 001001 5           .inh fre
034121 121636 5 0         jst [gfree] i           ;attempt to get buffer
034122 003223 5 0          jmp rupnbf             ;no buffers
034123 000401 5 0         .enb t.o
034124 011670 5           sta rupgbf              ;save buffer pointer
034125 010000 5           sta 0
034126 004025 5           lda [seqh]
034127 014000 5           add 0
034130 011671 5           sta rupgbp              ;pointer to delay entries
034131 005637 5           lda [-ch]               ;prepare to loop over all lin
034132 010001 5           sta ireg
034133 072001 5   rupglp: ldx ireg
034134 145640 5           lda [mblock+ch] ix      ;does modem exist?
034135 101040 5           snz
034136 003163 5            jmp rupgl              ;no
034137 010000 5           sta 0
034140 004213 5           lda rtgdwn              ;nice-stopping?
034141 040475 5           lgr 3.                  ;if so, set c-bit (rtgdwn=4)

034142 044120 5           lda lnei x              ;get last neighbor on this li
034143 100040 5           sze                     ;don't report line that's ne
034144 022106 5            cas mine
034145 101040 5             snz
034146 003163 5               jmp rupgl           ;or looped line
034147 025672 5           irs rupgnn
034150 141240 5           icr                     ;no, report real delay values
034151 052215 5           era delbas x
034152 101001 5           ssc 0&rtgdwn            ;(c-bit ==> nice-stopping)
034153 003156 5            jmp rupgl1
034154 141044 5           car                     ;set delay=inf for nice-stop

034155 012007 5           era [dlinf]
034156 141340 5   rupgl1: ica 0&rupdel&rupnei
034157 111671 5           sta rupgbp i
034160 015660 5           add rupc
034161 011660 5           sta rupc                ;add to checksum subtotal
034162 025671 5           irs rupgbp              ;advance pointer to next slot
034163 024001 5   rupgl:  irs ireg                ;increment to next line
034164 003133 5            jmp rupglp             ;loop for another line
034165 001001 5           .inh m2i
034166 072106 5 1         ldx mine
034167 144131 5 1         lda spfrti ix
034170 006047 5 1         ana [spfsno]            ;get current serial number fo
034171 015641 5 1         add [spfsn1+spfage]     ;add one,set max age
034172 152131 5 1         era spfrti ix
034173 007642 5 1         ana [spfsno+spfage]
```

```
034174 152131 5 1          era spfrti ix              ;set in our spfrut entry
034175 150131 5 1          sta spfrti ix
```

```
034176 006047 5 1          ana [spfsno]          ;get serial number
034177 012106 5 1          era mine
034200 073670 5 1          ldx rupgbf
034201 050011 5 1          sta srch x            ;set in packet with our imp r
034202 015660 5 1          add rupc              ;add to checksum subtotal
034203 011660 5 1          sta rupc
034204 005672 5 1          lda rupgnn
034205 141240 5 1          icr 0&rupnn
034206 012050 5 1          era [rupage]          ;set max age
034207 050006 5 1          sta neth x
034210 015660 5 1          add rupc              ;add to checksum subtotal
034211 011660 5 1          sta rupc
034212 072106 5 1          ldx mine
034213 021276 5 1          jst rupmst            ;compute mask, turn on rtimer
034214 000401 5 1          .enb t.o
034215 073670 5            ldx rupgbf
034216 050003 5            sta rupm x            ;set computed mask in packet

034217 021337 5            jst rupq              ;finish building it, then qu
034220 125656 5            irs stats.lrumsi i    ;count # of updates generated
034221 101000 5             nop
034222 102124 5            jmp toret i

                    rupnbf:
034223 000401 5            .enb t.o
034224 025674 5            irs rupxbf            ;count gfree refusals
034225 101000 5             nop
034226 125632 5            irs [rupsnd] i        ;no buffers, set flag back on
034227 102124 5            jmp toret i           ;wait until next t.o call
```

```
                        .stitl spf- rup queue routines
                        .lev (i2m,t.o,spf)
                        .lck all
034230 000000 2 0 rupdq:  0                     ;get packet off rup queue
034231 011675 2 0         sta rupdms            ;bit for modem or task
034232 073643 2 0         ldx [sruq]
034233 044000 2 0 rupdsq: lda 0 x
034234 101040 2 0         snz                   ;is this the end of the queue
034235 103230 2 0          jmp rupdq i          ;yes
034236 010000 2 0         sta 0                 ;no, inspect this buffer
034237 044003 2 0         lda rupm x
034240 007675 2 0         ana rupdms
034241 101040 2 0         snz                   ;is bit on in packet's mask?

034242 003233 2 0          jmp rupdsq           ;no, packet not for us
034243 025230 2 0         irs rupdq             ;yes, skip return, x=packet p
034244 103230 2 0         jmp rupdq i

                        .lev (i2m,t.o,spf)
                        .lck all
034245 000000 2 0 rupfls: 0                     ;dequeue and flush packet if
034246 033676 2 0         stx rupfbf            ;save buffer pointer
034247 140401 2 0         cma                   ;turn off mask bit in packet
034250 046003 2 0         ana rupm x
034251 050003 2 0         sta rupm x
034252 100040 2 0         sze                   ;are we last user?
034253 103245 2 0          jmp rupfls i         ;no, nothing left to do
034254 073643 2 0         ldx [sruq]            ;yes, dequeue
034255 044000 2 0 rupfsq: lda 0 x
034256 101040 2 0         snz                   ;end of queue?
034257 000000 2 0          %crash               ;routing queue broken
034260 023676 2 0         cas rupfbf            ;is this the buffer we want?

034261 100000 2 0          skp
034262 003265 2 0           jmp rupffb          ;yes
034263 010000 2 0         sta 0                 ;no, step through queue
034264 003255 2 0         jmp rupfsq

034265 105676 2 0 rupffb: lda rupfbf i          ;dequeue
034266 101040 2 0         snz
034267 133644 2 0          stx [eruq] i         ;adjust end pointer
034270 066000 2 0         ima 0 x
034271 010000 2 0         sta 0
034272 140040 2 0         cra
034273 050000 2 0         sta ptrc x            ;clear link word
034274 120115 2 0         jst flushi i          ;  and flush it.
034275 103245 2 0         jmp rupfls i
```

```
                .lev (m2i,t.o)
;temp variables usage - to keep things straight.
;ireg, jreg not used since subroutine "rton" is called.
; rupmt1 - to build mask
; rupmt2 - loop counter, -ch to -1
; rupmt3 - modem block pointer
;
034276 000000 1   rupmst: 0                       ;compute mask for an update p
034277 033677 1           stx rupmi
034300 005637 1           lda [-ch]               ;prepare to loop through all
034301 011701 1           sta rupmt2
034302 140040 1           cra
034303 011700 1           sta rupmt1              ;clear mask at beginning
034304 073701 1   rupmlp: ldx rupmt2
034305 145640 1           lda [mblock+ch] ix      ;is this an IMP modem?
034306 101040 1           snz
034307 003326 1            jmp rupmld
034310 010000 1           sta 0
034311 011702 1           sta rupmt3              ;save modem block pointer
034312 044117 1           lda line x              ;yes - check state
034313 101400 1           smi                     ;if line is silent
034314 100100 1            slz                    ;or dead
034315 003326 1             jmp rupmld            ;don't set its bit in mask
034316 073701 1           ldx rupmt2
034317 044070 1           lda bittab+ch x         ;otherwise do
034320 013700 1           era rupmt1
034321 011700 1           sta rupmt1
034322 073702 1           ldx rupmt3
034323 050201 1           sta rup4to x            ;set rup flg for t.o.
034324 005677 1           lda rupmi
034325 121645 1           jst [rton] i            ;and turn on the timer for th
034326 025701 1   rupmld: irs rupmt2
034327 003304 1            jmp rupmlp
034330 140040 1           cra
034331 121655 1           jst fwdset i            ;call forward package
034332 140500 1           ssm 0&ruptsk            ;always set task bit
034333 013700 1           era rupmt1
034334 072137 1           ldx spfpci              ;get ptr to SPF's process
034335 000043 1           gpr                     ;poke it
034336 103276 1           jmp rupmst i

                .lev t.o
034337 000000 5   rupq:   0                       ;finish setup for update pack
034340 044006 5           lda neth x
034341 006074 5           ana [rupnn]             ;get number of neighbors
034342 015646 5           add [(srch-neth+1)<<8.+1] ;length of packet
034343 050004 5           sta wrdc x              ;(plus usecnt of 1)
034344 141140 5           icl                     ;add -(length) to checksum
034345 140407 5           tca
034346 015660 5           add rupc
034347 011660 5           sta rupc
034350 005647 5           lda [swttyp+compat?swtcpt+ruptyp]
034351 050007 5           sta typh x
034352 015660 5           add rupc                ;add to checksum subtotal
034353 140407 5           tca
034354 050010 5           sta chkh x
034355 044003 5           lda rupm x
034356 001001 5           .inh m2i
034357 121650 5 1         jst [rupenq] i
```

```
034360 000401 5 1          .enb t.o
034361 103337 5            jmp rupq i
```

```
                        .stitl spf: retransmission code
                        .lev t.o
                        ruptic:
034362 005637 5                 lda [-ch]               ;tick retransmission timers
034363 011704 5                 sta ruptc2
034364 073704 5         rttic0: ldx ruptc2
034365 145640 5                 lda [mblock+ch] ix      ;this an IMP modem?
034366 101040 5                 snz
034367 003437 5                  jmp rtnxt0             ;no - skip it
034370 011703 5                 sta ruptc1              ;save modem block pointer
034371 010000 5                 sta 0
034372 044205 5                 lda rtptrs x            ;a<-addr first word for this
034373 011657 5                 sta rtrtmp
034374 044117 5                 lda line x
034375 101400 5                 smi
034376 100100 5                  slz
034377 003437 5                   jmp rtnxt0            ;don't tick dead line counter
034400 044207 5                 lda rtclkp x            ;get clock value for this lin
034401 010001 5                 sta ireg
034402 104001 5                 lda ireg i
034403 141206 5                 aoa
034404 100040 5                 sze                     ;is it ready to go off?
034405 003437 5                  jmp rtnxt0             ;no, don't tick this retransm
034406 105657 5         rttic1: lda rtrtmp i
034407 101040 5                 snz                     ;is entire word zero?
034410 003432 5                  jmp rtnxt8             ;yes, no ticking
034411 072032 5                 ldx [-16.]              ;count in doubles
034412 044072 5         rttic1: lda bittab+16. x
034413 052073 5                 era bittab+1+16. x      ;form two bit mask
034414 001001 5                 .inh m2i
034415 107657 5 1               ana rtrtmp i
034416 101040 5 1               snz                     ;is timer zero?
034417 003426 5 1                jmp rtnxt1             ;yes, no ticking
034420 052072 5 1               era bittab+16. x
034421 101040 5 1               snz                     ;is timer now expired?
034422 003442 5 1                jmp rtretr             ;yes, generate retransmission
034423 105657 5 1 rtnxt3: lda rtrtmp i                  ;no, tick it
034424 056072 5 1               sub bittab+16. x
034425 111657 5 1               sta rtrtmp i
034426 000401 5 1 rtnxt1: .enb t.o
034427 024000 5                 irs 0
034430 024000 5                 irs 0                   ;count by twos
034431 003412 5                  jmp rttic1
034432 025657 5         rtnxt8: irs rtrtmp              ;advance to next word
034433 073703 5                 ldx ruptc1              ;x <- modem block pointer
034434 044206 5                 lda rtpend x            ;get pointer to end of ret. w
034435 023657 5                 cas rtrtmp              ;are we at end of ours
034436 003406 5                  jmp rttic1             ;no, tick out this word
034437 025704 5         rtnxt0: irs ruptc2              ;yes, advance to next line
034440 003364 5                  jmp rttic0             ;(CAS < case won't happen)
034441 102124 5                 jmp toret i             ;all done
```

```
                .lev t.o
                .lck m2i
                rtretr:
034442 000401 5 1       .enb t.o
;generate retransmission
034443 033661 5         stx rtrsx               ;index into bittab for expire
034444 005657 5         lda rtrtmp              ;address of word with expired
034445 073703 5         ldx ruptc1              ;modem block pointer
034446 056205 5         sub rtptrs x            ;get impno\8
034447 141206 5         aoa                     ;plus one since rtrsx is neg
034450 041474 5         lgl 4                   ;2*(impno+8)
034451 015661 5         add rtrsx               ;2*impno
034452 040477 5         lgr 1                   ;impno whose timer expired
034453 011662 5         sta rtrino
034454 010000 5         sta 0
034455 001001 5         .inh fre
034456 073643 5 0       ldx [sruq]              ;search routing update queue

034457 044000 5 0 rtnrtl: lda 0 x
034460 101040 5 0       snz                     ;end of queue?
034461 003474 5 0        jmp rtnrtx             ;yes, not found so ok to retr
034462 010000 5 0       sta 0                   ;no, look at the update.
034463 044003 5 0       lda rupm x              ;is this update queued for ta
034464 101400 5 0       smi 0&ruptsk
034465 003457 5 0        jmp rtnrtl             ;no, keep looking
034466 044011 5 0       lda srch x              ;yes, is it from same imp?
034467 141050 5 0       cal 0&rupsrc
034470 013662 5 0       era rtrino
034471 100040 5 0       sze
034472 003457 5 0        jmp rtnrtl             ;no, keep looking
034473 003612 5 0       jmp rtrntk              ;yes, leave timer at 1, exit

034474 140040 5 0 rtnrtx: cra
034475 011660 5 0       sta rupc                ;init checksum subtotal
034476 073662 5 0       ldx rtrino              ;ok to generate retransmissio
034477 144131 5 0       lda spfrti ix
034500 007651 5 0       ana [spfage]            ;extract current age for this
034501 101040 5 0       snz                     ;don't do it if age = 0
034502 003520 5 0        jmp rtretx
034503 011667 5 0       sta rttrue              ;save for neth
034504 141240 5 0       icr 0&rupage            ;line up for update packet
034505 140500 5 0       ssm 0&rupret            ;set retry bit
034506 011663 5 0       sta rtrbfp
034507 145617 5 0       lda [ntb+1] ix          ;compute number of neighbors

034510 157621 5 0       sub [ntb] ix            ;we have in our tables for th
034511 141240 5 0       icr 0&rupnn             ;line up for update packet
034512 022006 5 0       cas [0]                 ;check for 0 neighbors
034513 023652 5 0        cas [(odata1+midh-seqh+2)<<8.] ;and check pkt length
034514 101000 5 0        nop                    ;=0 or .gt. max length pkt +
034515 100000 5 0        skp                    ;<0 or = max length packet +
034516 003522 5 0         jmp rtret0            ;>0 and < max length pkt + 1

                        ;defhlt(/82. spf: gen. retransmission w/bad packet le
034517 120120 5 0       jst hltjst i            ;trap
034520 073661 5 0 rtretx: ldx rtrsx             ;restore x
034521 003423 5 0       jmp rtnxt3              ;and tic this out to zero
```

```
034522 013663 5 0 rtret0: era rtrbfp          ;form complete neth word for
034523 011663 5 0         sta rtrbfp          ;save for later
034524 121636 5 0         jst [gfree] i       ;try to get a buffer
034525 003614 5 0          jmp rtrnbf         ;no buffers, leave timer at
034526 011664 5 0         sta rtrbuf          ;save buffer address
034527 010000 5 0         sta 0
034530 014024 5 0         add [srch]          ;addr of first data loc in pk
034531 027663 5 0         ima rtrbfp          ;get previously setup neth wo
034532 006074 5 0         ana [rupnn]
034533 141140 5 0         icl 0&rupnn
034534 140407 5 0         tca                 ;loop counter for neighbor lo
034535 011665 5 0         sta rtrlc
034536 073662 5 0         ldx rtrino
034537 144131 5 0         lda spfrti ix
034540 006047 5 0         ana [spfsno]
034541 013662 5 0         era rtrino          ;form srch word for packet
034542 111663 5 0         sta rtrbfp i        ;with serial number and imp n
034543 015660 5 0         add rupc
034544 011660 5 0         sta rupc            ;add to checksum subtotal
034545 025663 5 0         irs rtrbfp          ;advance pointer to next word
034546 145621 5 0         lda [ntb] ix
034547 006033 5 0         ana [ntbidx]
034550 010000 5 0         sta 0               ;first neighbor entry for thi
```

```
034551 144127 5 0 rtrlp:  lda ltbi ix          ;check for dead line
034552 141044 5 0         car 0&ltbdst
034553 012010 5 0         era [dlinf<<8.]
034554 101040 5 0         snz
034555 003564 5 0          jmp rtrlp1          ;dead, so skip it.
034556 025667 5 0         irs rttrue           ;count true rupnn
034557 144127 5 0         lda ltbi ix
034560 111663 5 0         sta rtrbfp i         ;copy into packet
034561 015660 5 0         add rupc
034562 011660 5 0         sta rupc             ;add to checksum subtotal
034563 025663 5 0         irs rtrbfp           ;advance packet pointer
034564 024000 5 0 rtrlp1: irs 0
034565 025665 5 0         irs rtrlc            ;loop through all neighbor er
034566 003551 5 0          jmp rtrlp
034567 073703 5 0         ldx ruptc1           ;get modem block pointer in x
034570 005662 5 0         lda rtrino
034571 121645 5 0         jst [rton] i         ;turn on timer for this line

034572 000401 5 0         .enb t.o
034573 064210 5           irs rtrcnt x         ;count retransmissions per li
034574 101000 5            nop
034575 073704 5           ldx ruptc2           ;ruptc2 = modem number - ch
034576 044070 5           lda bittab+ch x
034577 073703 5           ldx ruptc1
034600 050201 5           sta rup4to x         ;set non-0 flag for t.o to tc
034601 073664 5           ldx rtrbuf
034602 050003 5           sta rupm x           ;set mask for this line only

034603 005667 5           lda rttrue           ;set neth
034604 141240 5           icr 0&rupage&rupnn
034605 140500 5           ssm 0&rupret         ;set retry bit
034606 050006 5           sta neth x
034607 015660 5           add rupc
034610 011660 5           sta rupc
034611 021337 5           jst rupq             ;finish setup and queue for o
034612 073661 5   rtrntk: ldx rtrsx
034613 003426 5           jmp rtnxt1           ;return without ticking this


034614 025666 5   rtrnbf: irs rtrfbf          ;count of gfree refusals
034615 101000 5            nop
034616 003612 5           jmp rtrntk


034617 041531 5   .constants
034620 035651 5
034621 041530 5
034622 177577 5
034623 042133 5
034624 041732 5
034625 042334 5
034626 000541 5
034627 041731 5
034630 177240 5
034631 036142 5
034632 033225 5
034633 033153 5
034634 000315 5
034635 033144 5
```

034636 031000 5
034637 177762 5

```
034640 046210 5
034641 000560 5
034642 037560 5
034643 046303 5
034644 046405 5
034645 034025 5
034646 002001 5
034647 121400 5
034650 036007 5
034651 000160 5
034652 042000 5
;confix(-17)
                cx27b:
;end of checksummed area
034655 005651 5 fwdset: jstm2i                  ;package sets to fset
                stats.lrumsi:
034656 034673 5         rupcnt
```

```
                        .lev var
034657          V.  rtrtmp: .block 1
034660          V   rupc:   .block 1            ;checksum subtotal
034661          V   rtrsx:  .block 1
034662          V   rtrino: .block 1
034663          V   rtrbfp: .block 1
034664          V   rtrbuf: .block 1
034665          V   rtrlc:  .block 1
034666          V   rtrfbf: .block 1            ;count of gfree refusals
034667          V   rttrue: .block 1            ;true rupnn counter
034670          V   rupgbf: .block 1
034671          V   rupgbp: .block 1
034672          V   rupgnn: .block 1
034673          V   rupcnt: .block 1            ;count of updates generated
034674          V   rupxbf: .block 1            ;count of gfree refusals
034675          V   rupdms: .block 1
034676          V   rupfbf: .block 1
034677          V   rupmi:  .block 1
034700          V   rupmt1: .block 1            ;temps for rupmst
034701          V   rupmt2: .block 1
034702          V   rupmt3: .block 1
034703          V   ruptc1: .block 1            ;temp for ruptic, rtretr (mod
034704          V   ruptc2: .block 1            ;temp for ruptic, rtretr (loo
```

```
                .stitl spf- incremental changes to tree
;rtinc - process incremental update to line distance
; update has the new length of the line from endpts to endptd
; deltad has the change from its previous value

                .section pg35
                .lev spf
035000 000000 6 rtinc:  0
035001 025670 6         irs rtical              ;count calls to spf
035002 101000 6          nop
035003 072205 6         ldx endptd              ;determine if line already in
035004 121620 6         jst [bckno] i           ;get index for father's entry
035005 003013 6          jmp rtinot             ;no bckno, line not in tree
035006 144127 6         lda ltbi ix             ;is father endpts?
035007 141050 6         cal 0&ltbnay
035010 022206 6         cas endpts
035011 100000 6          skp
035012 003104 6           jmp rtilit            ;yes, then line in tree alrea
035013 004007 6 rtinot: lda [dlinf]             ;no, line not in tree at pres
035014 022210 6         cas update              ;is line now dead?
035015 100000 6          skp
035016 103000 6           jmp rtinc i           ;yes, don't add dead line to
035017 072206 6         ldx endpts
035020 145621 6         lda [pdist] ix          ;compute distance to endptd o
035021 006044 6         ana [pdst]              ;distance to endpts
035022 014210 6         add update              ;plus new line's distance
035023 022044 6         cas [pdinf]             ;keep less than infinite
035024 101000 6          nop
035025 005622 6          lda [pdinf-1]
035026 010207 6         sta deltad
035027 072205 6         ldx endptd
035030 145621 6         lda [pdist] ix
035031 006044 6         ana [pdst]              ;old distance to endptd
035032 140407 6         tca
035033 014207 6         add deltad
035034 010207 6         sta deltad              ;improvement from using new l
035035 101400 6         smi                     ;is new line better than old
035036 103000 6          jmp rtinc i            ;no, then forget about it
035037 145621 6         lda [pdist] ix
035040 006044 6         ana [pdst]              ;old distance to endptd
035041 022044 6         cas [pdinf]             ;was endptd unreachable?
035042 000000 6          %crash                 ;spf cas error
035043 003057 6          jmp rtiadd             ;yes, already severed from tr
035044 121620 6         jst [bckno] i           ;no, must sever old connectio
035045 121623 6          jst [spfabt] i         ;impossible spf abort - rtinc
035046 006044 6         ana [-1?1flg]
035047 151624 6         sta [1flag] ix          ;sever back link
035050 144127 6         lda ltbi ix
035051 141050 6         cal 0&ltbnay
035052 010000 6         sta 0                   ;old father's number
035053 004205 6         lda endptd
035054 120125 6         jst fndni i             ;find father's link to endptd
035055 006044 6         ana [-1?1flg]           ;and sever it
035056 151624 6         sta [1flag] ix          ;endptd disconnected from old
```

```
035057 072206 6   rtiadd: ldx endpts          ;now add endptd's subtree to
035060 004205 6           lda endptd
035061 120125 6           jst fndni i         ;find fwd link from endpts to
035062 140500 6           ssm 0&1flgf         ;make it forward link
035063 151624 6           sta [1flag] ix
035064 072205 6           ldx endptd
035065 004206 6           lda endpts
035066 120125 6           jst fndni i         ;find back link from endptd t
035067 006044 6           ana [-1?1flg]
035070 012037 6           era [1flgb]         ;make it back link
035071 151624 6           sta [1flag] ix
035072 072206 6           ldx endpts
035073 004205 6           lda endptd
035074 001001 6           .inh m2i
035075 021106 6 1         jst fmdmn           ;compute route for this neigh
035076 072205 6 1 rtinrt: ldx endptd
035077 152131 6 1         era spfrti ix       ;update endptd's route
035100 006072 6 1         ana [route]
035101 152131 6 1         era spfrti ix
035102 150131 6 1         sta spfrti ix
035103 000401 6 1         .enb spf
035104 021135 6   rtilit: jst retree          ;line now in tree, call retr
035105 103000 6           jmp rtinc i         ;done now

                  .lev spf
                  .lck m2i
035106 000000 6 1 fmdmn:  0                   ;subroutine to compute route

035107 011655 6 1         sta fmdmi           ;destination imp number
035110 144131 6 1         lda spfrti ix       ;does source have a route?
035111 006072 6 1         ana [route]
035112 100040 6 1         sze
035113 103106 6 1          jmp fmdmn i        ;yes, then use it
035114 005625 6 1         lda [-ch]           ;compute modem number of our
035115 010002 6 1         sta jreg
035116 072002 6 1 fmdmlp: ldx jreg
035117 145626 6 1         lda [mblock+ch] ix  ;does modem exist?
035120 101040 6 1         snz
035121 003127 6 1          jmp fmdbot         ;no
035122 010000 6 1         sta 0
035123 044120 6 1         lda lnei x
035124 013655 6 1         era fmdmi
035125 101040 6 1         snz
035126 003132 6 1          jmp fmdmnr         ;found it
035127 024002 6 1 fmdbot: irs jreg
035130 003116 6 1          jmp fmdmlp         ;keep looking
035131 121623 6 1         jst [spfabt] i      ;impossible spf abort - fmdmn

035132 004072 6 1 fmdmnr: lda [ch+1]          ;number 1-ch
035133 014002 6 1         add jreg
035134 103106 6 1         jmp fmdmn i         ;set route for fmdmi
```

```
;retree - scan subtree of a node
; update distances of all nodes in subtree
; marking them infinite if line is now dead that connects them
; update route to all nodes in subtree
; examine neighbors of subtree nodes and add to ordered list

                .lev spf
035135 000000 6 retree: 0
035136 025671 6         irs retcal              ;count calls to retree
035137 101000 6          nop
035140 073627 6         ldx [-nnodes-1]
035141 145630 6 retini: lda [lst+nnodes+1] ix
035142 007631 6         ana [-1?lstost]         ;nothing on subtree
035143 151630 6         sta [lst+nnodes+1] ix   ;to begin with
035144 024000 6         irs 0
035145 003141 6          jmp retini
035146 004007 6         lda [lstlst]
035147 011667 6         sta head                ;init ordered list empty
035150 011652 6         sta subtre
035151 072205 6         ldx endptd
035152 033654 6         stx scn
035153 145632 6         lda [lst] ix
035154 141044 6         car 0&lstlst
035155 012007 6         era [lstlst]
035156 151632 6         sta [lst] ix            ;subtree list has only endpt
035157 004207 6         lda deltad
035160 022044 6         cas [pdinf]             ;is line dead?
035161 000000 6          %crash                 ;spf cas error
035162 003171 6          jmp retnd              ;yes, distance to it infinite
035163 145621 6         lda [pdist] ix
035164 006044 6         ana [pdst]
035165 014207 6         add deltad              ;else, new distance=old+delta
035166 022044 6         cas [pdinf]             ;check for overflow
035167 101000 6          nop
035170 005622 6          lda [pdinf-1]          ;and reset to max finite valu
035171 153621 6 retnd:  era [pdist] ix
035172 006044 6         ana [pdst]
035173 153621 6         era [pdist] ix
035174 151621 6         sta [pdist] ix          ;new distance to endptd
035175 073654 6 retfst: ldx scn                 ;loop to find subtree of endp
035176 145632 6         lda [lst] ix
035177 141050 6         cal 0&lstlst
035200 027654 6         ima scn                 ;pop scan list
035201 023633 6         cas [nnodes+1]          ;end of list?
035202 003270 6          jmp retsst             ;yes, start search of subtree
035203 000000 6          %crash                 ;spf cas error
035204 011653 6         sta subnod              ;no, put this node on subtree
035205 010000 6         sta 0                   ;at beginning of list
035206 027652 6         ima subtre              ;previous node put on subtree
035207 153632 6         era [lst] ix            ;is who chain should point to
035210 141050 6         cal 0&lstlst
035211 153632 6         era [lst] ix
035212 012062 6         era [lstost]            ;mark on subtree
035213 151632 6         sta [lst] ix            ;and update chain to rest of
035214 144131 6         lda spfrti ix
035215 006072 6         ana [route]
035216 011660 6         sta subnrt              ;remember its routing for lat
```

```
035217 145621 6            lda [pdist] ix
035220 006044 6            ana [pdst]
035221 011657 6            sta subdst              ;and remember its pdist
035222 021572 6            jst naylps              ;setup neighbor search loop
035223 145624 6   retfnl: lda [lflag] ix          ;is this neighbor in the tree
035224 022037 6            cas [lflg] 0&lflgf      ;i.e. is this forward link?
035225 101000 6             nop
035226 003265 6             jmp retfse             ;no, look at next neighbor
035227 144127 6            lda ltbi ix             ;yes, update its distance and
035230 141140 6            icl 0&ltbdst
035231 015657 6            add subdst              ;pdist(subnod)+distance over
035232 022044 6            cas [pdinf]             ;has pdist overflowed?
035233 101000 6             nop
035234 005622 6             lda [pdinf-1]          ;reset to max finite value
035235 011656 6            sta subnd               ;new distance to this neighbo
035236 144127 6            lda ltbi ix
035237 141050 6            cal 0&ltbnay
035240 010000 6            sta 0                   ;neighbor's node number
035241 027654 6            ima scn                 ;put it on scan list
035242 153632 6            era [lst] ix
035243 141050 6            cal 0&lstlst
035244 153632 6            era [lst] ix
035245 151632 6            sta [lst] ix
035246 001001 6            .inh m2i
035247 144131 6 1          lda spfrti ix
035250 006032 6 1          ana [-1?route]
035251 013660 6 1          era subnrt              ;update this neighbor's route
035252 150131 6 1          sta spfrti ix
035253 000401 6 1          .enb spf
035254 004207 6            lda deltad
035255 022044 6            cas [pdinf]             ;if deltad was infinite
035256 000000 6             %crash                ;spf cas error
035257 100000 6             skp                    ;then this nei's pdist should
035260 005656 6              lda subnd             ;else use computed new distan
035261 153621 6            era [pdist] ix
035262 006044 6            ana [pdst]
035263 153621 6            era [pdist] ix
035264 151621 6            sta [pdist] ix
035265 021603 6   retfse: jst naylpi              ;advance to subnod's next nei
035266 003223 6             jmp retfnl            ;not end, continue
035267 003175 6            jmp retfst              ;end, pop scan list

035270 005652 6   retsst: lda subtre
035271 023633 6            cas [nnodes+1]          ;end of subtree list?
035272 021346 6             jst search             ;yes, create new tree
035273 103135 6             jmp retree i           ;and return
035274 011653 6            sta subnod              ;no, remember this subtree no
035275 010000 6            sta 0
035276 145632 6            lda [lst] ix            ;pop list of subtree nodes
035277 141050 6            cal 0&lstlst
035300 011652 6            sta subtre
035301 021572 6            jst naylps              ;prepare to search its neighb
```

```
035302 145624 6   retsnl: lda [lflag] ix
035303 100400 6          spl 0&lflg                ;is line in tree?
035304 003327 6           jmp retsse               ;yes, don't consider it
035305 144127 6          lda ltbi ix
035306 141050 6          cal 0&ltbnay
035307 101040 6          snz                       ;is it a usurped entry?
035310 003327 6           jmp retsse               ;yes, don't consider it
035311 010000 6          sta 0
035312 011656 6          sta subnd                 ;remember for possible router
035313 145632 6          lda [lst] ix
035314 006062 6          ana [lstost]              ;is this neighbor on the subt
035315 100040 6          sze
035316 003327 6           jmp retsse               ;yes, don't consider it eith
035317 004207 6          lda deltad                ;consider new routes for subn
035320 101400 6          smi
035321 101040 6           snz
035322 003332 6            jmp retsrt              ;distance improved
035323 005653 6          lda subnod                ;distance got worse
035324 120125 6          jst fndni i               ;find neighbor's entry for s
035325 005656 6          lda subnd                 ;nei is possible better route
035326 021421 6   retcrt: jst router               ;add routes to ordered list
035327 021603 6   retsse: jst naylpi              ;consider next neighbor of s
035330 003302 6           jmp retsnl               ;not end of entries, continue
035331 003270 6          jmp retsst                ;end, go on to next node on s

035332 145621 6   retsrt: lda [pdist] ix          ;determine if improvement
035333 006044 6          ana [pdst]                ;can possibly effect this nod
035334 011656 6          sta subnd
035335 073653 6          ldx subnod
035336 145621 6          lda [pdist] ix
035337 006044 6          ana [pdst]
035340 023656 6          cas subnd                 ;only if its distance is grea
035341 101000 6           nop
035342 003327 6           jmp retsse
035343 005653 6          lda subnod                ;than that to subnod
035344 072212 6          ldx naylpx                ;in which case, subnod is
035345 003326 6          jmp retcrt                ;possible better route to th
```

```
;search - pick top entry off ordered list and add to tree
;also look at new node's neighbors for possible rerouting

                        .lev spf
035346 000000 6   search: 0
035347 005667 6   schlp:  lda head                ;next entry on list
035350 023633 6          cas [nnodes+1]           ;end of list?
035351 103346 6           jmp search i            ;yes, done
035352 000000 6           %crash                  ;spf cas error
035353 011662 6          sta schnod               ;save new node
035354 010000 6          sta 0
035355 145632 6          lda [lst] ix             ;unchain it from the list
035356 007634 6          ana [-1?lstol]           ;mark it not on list
035357 151632 6          sta [lst] ix
035360 141050 6          cal 0&lstlst
035361 011667 6          sta head                 ;remember new head of list
035362 121620 6          jst [bckno] i            ;is the back link set?
035363 003407 6           jmp schnay              ; no, must be us
035364 144127 6          lda ltbi ix
035365 141050 6          cal 0&ltbnay
035366 011664 6          sta schfth               ;record node's father
035367 010000 6          sta 0
035370 005662 6          lda schnod               ;find father's entry for schn
035371 120125 6          jst fndni i
035372 006044 6          ana [-1?lflg]
035373 012071 6          era [lflgf]
035374 151624 6          sta [lflag] ix           ;make this a forward link
035375 005662 6          lda schnod
035376 073664 6          ldx schfth
035377 001001 6          .inh m2i
035400 021106 6 1        jst fmdmn                ;compute route for schnod
035401 073662 6 1        ldx schnod
035402 152131 6 1        era spfrti ix            ;set new route for schnod
035403 006072 6 1        ana [route]
035404 152131 6 1        era spfrti ix
035405 150131 6 1        sta spfrti ix
035406 000401 6 1        .enb spf
035407 021572 6   schnay: jst naylps              ;setup neighbor search loop
035410 145624 6   schnlp: lda [lflag] ix
035411 041477 6          lgl 1
035412 100400 6          spl 0&lflg&lflgb         ;is this schnod's father?
035413 003416 6           jmp schnxn              ;yes, don't examine this neig
035414 005662 6          lda schnod               ;no, examine route via schnod
035415 021421 6          jst router
035416 021603 6   schnxn: jst naylpi              ;look at next neighbor
035417 003410 6           jmp schnlp              ;not end of neighbors, contin
035420 003347 6          jmp schlp                ;end, continue with next list
```

```
;router - process new route for a node
;if new route is better than old route
;update its distance, remove it from the tree
;and add it to the ordered list removing any
;entry for it than might already be on the list.

                .lev spf
035421 000000 6 router: 0
035422 011665 6         sta rtrs                ;source on new path
035423 144127 6         lda ltbi ix
035424 141050 6         cal 0&ltbnay
035425 011666 6         sta nexnod              ;destination on new path
035426 144127 6         lda ltbi ix
035427 141140 6         icl 0&ltbdst
035430 011663 6         sta rtrdst              ;length of this line
035431 012007 6         era [dlinf]             ;is it dead?
035432 101040 6         snz
035433 103421 6          jmp router i           ;yes, don't consider it
035434 073665 6         ldx rtrs
035435 145621 6         lda [pdist] ix          ;distance to source
035436 006044 6         ana [pdst]
035437 022044 6         cas [pdinf]             ;is distance infinite?
035440 000000 6          %crash                 ;spf cas error
035441 103421 6          jmp router i           ;yes, don't route through dea
035442 015663 6         add rtrdst              ;distance to dest over this l
035443 022044 6         cas [pdinf]             ;has distance overflowed?
035444 101000 6          nop
035445 005622 6          lda [pdinf-1]          ;yes, reset to maximum finit
035446 011663 6         sta rtrdst
035447 073666 6         ldx nexnod
035450 145621 6         lda [pdist] ix          ;old distance to nexnod
035451 006044 6         ana [pdst]
035452 017663 6         sub rtrdst              ;is new route better than old
035453 100040 6         sze
035454 100400 6          spl
035455 103421 6           jmp router i          ;no, then don't consider it
035456 005663 6         lda rtrdst              ;yes, update distance to nexn
035457 153621 6         era [pdist] ix
035460 006044 6         ana [pdst]
035461 153621 6         era [pdist] ix
035462 151621 6         sta [pdist] ix
035463 121620 6         jst [bckno] i           ;now sever nexnod from old fa
035464 003476 6          jmp rtrins             ;no back link to sever
035465 006044 6         ana [-1?1flg]           ;sever old back link
035466 151624 6         sta [1flag] ix
035467 144127 6         lda ltbi ix
035470 141050 6         cal 0&ltbnay            ;must sever old father's fwd
035471 010000 6         sta 0
035472 005666 6         lda nexnod
035473 120125 6         jst fndni i             ;find father's entry for nexn
035474 140100 6         ssp 0&1flgf             ;sever it
035475 151624 6         sta [1flag] ix
035476 004014 6 rtrins: lda minus1
035477 011661 6         sta rtrp                ;flag to detect insert at hea
035500 005667 6         lda head                ;start search of ordered list
```

```
035501 023633 6   rtrin1: cas [nnodes+1]          ;end of list?
035502 003517 6           jmp rtrin1              ;yes, insert here then
035503 000000 6           %crash                  ;spf cas error
035504 010000 6          sta 0                    ;no, check if distance fits h
035505 145621 6          lda [pdist] ix
035506 006044 6          ana [pdst]               ;this node's distance
035507 017663 6          sub rtrdst
035510 101400 6          smi                      ;is nexnod's distance .le. th
035511 003516 6           jmp rtrin0              ;yes, insert here
035512 033661 6          stx rtrp                 ;remember previous entry on l
035513 145632 6          lda [lst] ix
035514 141050 6          cal 0&lstlst             ;next entry on list
035515 003501 6          jmp rtrin1

035516 004000 6   rtrin0: lda 0                   ;insert before this node on l
035517 073666 6   rtrin1: ldx nexnod
035520 167632 6          ima [lst] ix 0&lstlst    ;rest of lst gets set at rtri
035521 027661 6          ima rtrp                 ;remember old lst entry for n
035522 101400 6          smi                      ;is this head of chain insert
035523 003526 6           jmp rtrin2              ;no
035524 033667 6          stx head                 ;yes, set head of chain point
035525 003534 6          jmp rtrin3

035526 026000 6   rtrin2: ima 0                   ;point previous entry at nexn
035527 153632 6          era [lst] ix
035530 141050 6          cal 0&lstlst
035531 153632 6          era [lst] ix
035532 151632 6          sta [lst] ix
035533 073666 6          ldx nexnod
035534 005661 6   rtrin3: lda rtrp                ;nexnod's old lst entry
035535 141044 6          car 0&lstlst
035536 153632 6          era [lst] ix             ;set rest of nexnod's lst ent
035537 007634 6          ana [-1?lstol]
035540 012063 6          era [lstol]              ;marking it on list
035541 151632 6          sta [lst] ix
035542 005661 6          lda rtrp
035543 006063 6          ana [lstol]
035544 101040 6          snz                      ;was it already on the list?

035545 003564 6           jmp rtrin6              ;no
035546 145632 6   rtrin4: lda [lst] ix            ;yes, find old pointer to it

035547 141050 6          cal 0&lstlst             ;which must be after this poi
035550 121635 6          jst [cas377] i           ;diag kdl ****???
035551 023666 6          cas nexnod               ;does this entry point to nex
035552 100000 6           skp
035553 003556 6            jmp rtrin5             ;yes, redirect it
035554 010000 6          sta 0                    ;no, step to next entry on li
035555 003546 6          jmp rtrin4
```

```
035556 005661 6   rtrin5: lda rtrp              ;entry nexnod used to point t
035557 153632 6           era [lst] ix          ;is who this entry should poi
035560 141050 6           cal 0&lstlst
035561 153632 6           era [lst] ix
035562 151632 6           sta [lst] ix
035563 073666 6           ldx nexnod
035564 005665 6   rtrin6: lda rtrs              ;find nexnod's entry for sour
035565 120125 6           jst fndni i
035566 006044 6           ana [-1?lflg]
035567 012037 6           era [lflgb]           ;mark it as back link
035570 151624 6           sta [lflag] ix
035571 103421 6           jmp router i
```

```
                    .stitl spf- useful subroutines
;naylps - setup limits for a neighbor search

                    .lev spf
035572 000000 6   naylps: 0
035573 145636 6           lda [ntb+1] ix          ;end of entries for node
035574 006033 6           ana [ntbidx]
035575 010211 6           sta naylpe
035576 145624 6           lda [ntb] ix
035577 006033 6           ana [ntbidx]
035600 010000 6           sta 0                   ;first entry for node
035601 010212 6           sta naylpx
035602 103572 6           jmp naylps i

;naylpi - increment nay loop index, check for end of loop
;returns +1 if more entries, +2 if end, x=incrmented index

035603 000000 6   naylpi: 0
035604 024212 6           irs naylpx              ;increment index
035605 072212 6           ldx naylpx              ;reload x
035606 004212 6           lda naylpx
035607 022211 6           cas naylpe              ;check for end of entries
035610 121623 6            jst [spfabt] i         ;should not happen?? spf abor
035611 025603 6            irs naylpi             ;end of entries, skip return

035612 103603 6           jmp naylpi i            ;not end, return+1

;diag kdl ****????
;subr to catch lst entry of 377
035613 000000 6   cas377: 0
035614 022007 6           cas [377]
035615 100000 6            skp
035616 000000 6             hlt
035617 103613 6           jmp cas377 i
```

```
                        .lev con
035620 034034 C         .constants
035621 041732 C
035622 037776 C
035623 036101 C
035624 041530 C
035625 177762 C
035626 046210 C
035627 177577 C
035630 042334 C
035631 177377 C
035632 042133 C
035633 000201 C
035634 176777 C
035635 035613 C
035636 041531 C
;end of checksummed area
;confix(-52)

                        .lev var
035651          V       bckne:   .block 1
035652          V       subtre:  .block 1
035653          V       subnod:  .block 1
035654          V       scn:     .block 1
035655          V       fmdmi:   .block 1
035656          V       subnd:   .block 1
035657          V       subdst:  .block 1
035660          V       subnrt:  .block 1
035661          V       rtrp:    .block 1
035662          V       schnod:  .block 1
035663          V       rtrdst:  .block 1
035664          V       schfth:  .block 1
035665          V       rtrs:    .block 1
035666          V       nexnod:  .block 1
035667          V       head:    .block 1
035670          V       rtical:  .block 1               ;calls to rtinc
035671          V       retcal:  .block 1               ;calls to retree
```

```
                        .section pg36
                        .lev var
036000          V   rupqt:  .block 1
036001          V   fndnay: .block 1
036002          V   fndne:  .block 1

;spf counts
036003          V   rlnrec: .block 1        ;count of line entries receiv
036004          V   rpkrec: .block 1        ;count of packets received
036005          V   rqlsum: .block 1        ;sum of queue lengths encount
036006          V   rqlmax: .block 1        ;maximum queue length encount
```

```
                    .stitl spf- subroutines
;rupenq - queue update packet
;compute various counts

                    .lev (m2i,t.o)
036007 000000 1   rupenq: 0
036010 105540 1           lda [eruq] i
036011 010001 1           sta ireg
036012 132001 1           stx ireg i            ;queue packet on rup queue
036013 133540 1           stx [eruq] i
036014 044003 1           lda rupm x
036015 101400 1           smi 0&ruptsk          ;is packet queued for task?
036016 103007 1            jmp rupenq i         ;no, no counting
036017 044006 1           lda neth x            ;yes, get number of lines in
036020 006074 1           ana [rupnn]
036021 141140 1           icl
036022 015003 1           add rlnrec            ;add to # lines received
036023 011003 1           sta rlnrec
036024 025004 1           irs rpkrec            ;count a packet received
036025 101000 1            nop                  ;just to be careful
036026 140040 1           cra
036027 011000 1           sta rupqt             ;clear count of packets on q
036030 073541 1           ldx [sruq]
036031 044000 1   rupeql: lda 0 x               ;count packets on q ahead of
036032 101040 1           snz                   ;end?
036033 003041 1            jmp rupeqa           ;yes
036034 010000 1           sta 0                 ;no, see if this packet has t
036035 044003 1           lda rupm x
036036 100400 1           spl 0&ruptsk
036037 025000 1            irs rupqt            ;yes, count it
036040 003031 1           jmp rupeql            ;continue through queue

036041 005000 1   rupeqa: lda rupqt             ;get count
036042 023542 1           cas [30.]             ;check queue length
036043 021101 1            jst spfabt           ;spfabt: routing queue .gt. 3
036044 101000 1            nop
036045 016052 1           sub [1]               ;don't count newly queued one
036046 023006 1           cas rqlmax            ;max to date?
036047 011006 1            sta rqlmax           ;yes, record it
036050 101000 1            nop
036051 015005 1           add rqlsum            ;also add to sum of lengths
036052 011005 1           sta rqlsum
036053 103007 1           jmp rupenq i
```

```
;fndn - find entry in x's nay table for node in a
;returns with line index in x and lflag in a

                    .lev spf
036054 000000 6   fndn:   0
036055 011001 6           sta fndnay
036056 145543 6           lda [ntb+1] ix          ;get index for end of nei ent
036057 006033 6           ana [ntbidx]
036060 011002 6           sta fndne               ;use to terminate loop
036061 145544 6           lda [ntb] ix
036062 006033 6           ana [ntbidx]
036063 010000 6           sta 0                   ;start of x's neighbor entrie
036064 144127 6   fndnlp: lda ltbi ix
036065 141050 6           cal 0&ltbnay
036066 013001 6           era fndnay
036067 101040 6           snz                     ;is this the entry for fndnay
036070 003077 6            jmp fndnx              ;yes
036071 024000 6           irs 0                   ;keep looking
036072 005002 6           lda fndne               ;get end index
036073 022000 6           cas 0                   ;have we reached end?
036074 003064 6            jmp fndnlp             ;no, continue
036075 101000 6            nop                    ;yes, bad bug
036076 021101 6           jst spfabt              ;spf abort - fndn

                    .lev spf
036077 145544 6   fndnx:  lda [lflag] ix
036100 103054 6           jmp fndn i

;spfabt routine - come here if an inconsistency is detected
;
; The philosophy here is that a routing inconsistency could lead to routing
; loops and uncontrollable congestion, so that it is better to halt the IMP
; than to simply ignore the inconsistency.  Drastic measures, but the only
; safe way to go.  However, it is most often the case that only the data
; base, or a received packet, is bad, and that the imp code is still intact.
; Also, it is quite likely that if the cause is a bad routing message, then
; many IMPs will be in the same mess.  Rather than cause all the IMPs to
; crash into their loaders, we would like to simply restart most IMPs.  We
; will force some IMPs into their loaders, though, so that a dump can be taken.
; The config block declares which IMPs will restart, and which will crash.  A
; modication to this rule was added to prevent an IMP from continuously
; restarting due to a persistent SPF bug:  if the NOC hasn't cleared the
; "crash locations" set from the previous restart, then the IMP should crash
; and not restart when the second spfabt occurs.
;

036101 000000 6   spfabt: 0                       ;centralized spfabt routine
036102 000000 6           %crash                  ;always crash, for now
036103 105545 6           lda [crshp] i           ;was there a recent restart?

036104 100040 6           sze
036105 000000 6            %crash                 ;SPF: abort
036106 105546 6           lda [noc.spfabt] i      ;no
036107 100100 6           slz                     ;conf'ed for crash?
036110 000000 6            %crash                 ;SPF: abort
036111 005101 6           lda spfabt              ;no, do restart
036112 111545 6           sta [crshp] i           ;first set crshp,a,x so
036113 004206 6           lda endpts              ;NOC will know why.
```

sta [crsha] i

```
036115 004205 6          lda endptd
036116 111550 6          sta [crshcd] i
036117 102065 6          jmp [init] i          ;and restart
```

```
                        .stitl spf- tick database ages
                        .lev t.o
036120 105551 5   spfclk: lda [tim8s] i          ;tick spf ages
036121 141206 5           aoa                    ;if 8 second tick about to hz
036122 100400 5           spl
036123 003141 5            jmp spfclx            ;not an 8 second tick
036124 073552 5           ldx [-nnodes-1]        ;loop through all spfrut entr
                        spfclp:
036125 001001 5           .inh m2i
036126 144132 5 1         lda spfrei ix
036127 007553 5 1         ana [spfage]
036130 100040 5 1         sze                    ;is age already zero?
036131 016056 5 1          sub [spfag1]          ;no, then tick down
036132 152132 5 1         era spfrei ix
036133 007553 5 1         ana [spfage]
036134 152132 5 1         era spfrei ix
036135 150132 5 1         sta spfrei ix
036136 000401 5 1         .enb t.o
036137 024000 5           irs 0
036140 003125 5            jmp spfclp
036141 102124 5   spfclx: jmp toret i
```

```
                .stitl spf- field definitions
                .lev spf

;data structures and parameters

 ;ltb - line table
        177400      ltbdst=177400                   ;distance to neighbor along t
        000377      dlinf=377                       ;infinite distance on a line
        000377      ltbnay=377                      ;imp number of neighbor on th

 ;lflag - tree flags
 ;note: lflag is imbedded in node tables ntb,lst & pdist
        140000      lflg=140000                     ;type of link in tree
        100000      lflgf=100000                    ;link is a forward link
        140000      lflgb=140000                    ;link is a backward link

 ;ntb - node index into line table
        037000      ntbunu=37000                    ;unused bits
        000777      ntbidx=777                      ;index into line tables
                                        ;for start of this node's entries

;lst - more node information
        036000      lstunu=36000
        001000      lstol=1000                      ;on list flag 0=false, 1=true
        000400      lstost=400                      ;on subtree flag 0=false, 1=t
        000377      lstlst=377                      ;link to next entry on list

;pdist - distance of entire path to this node from us
        037777      pdst=37777                      ;14-bit distance
        037777      pdinf=37777                     ;infinite distance, max finit

 ;spfrut - new arpanet routing table
        100000      spfded=100000                   ;dead flag, 1=dead(unreachabl
        040000      spfun1=40000                    ;unused bit
        037400      spfsno=37400                    ;latest update serial number
        000400      spfsn1=400                      ;one in serial number
        000200      spfun2=200                      ;unused bits
        000160      spfage=160                      ;age of latest update
        000020      spfag1=20                       ;one in age
        000017      route=17                        ;route to use to this dest.
        000000      spf4us=0                        ;spfrut entry for our imp=0
```

```
                .stitl spf- task processing of update
                .lev spf

036142 000401 6   spfdb:  .enb spf
036143 000103 6            spr                    ;debreak until poked
036144 000000 6             %crash               ;SPFDB: timed out

036145 001001 6   tkspf: .inh all
036146 004071 6 0          lda [ruptsk]           ;select packets for task
036147 121554 6 0          jst [rupdq] i          ;search update queue
036150 003142 6 0           jmp spfdb             ;if nothing left, debreak unt

036151 000401 6 0          .enb spf
036152 032311 6            stx that               ;save packet pointer
036153 044011 6            lda srch x             ;get source imp number
036154 141050 6            cal 0&rupsrc
036155 010206 6            sta endpts
036156 044006 6            lda neth x
036157 006074 6            ana [rupnn]
036160 141140 6            icl                    ;get number of nei entries in
036161 140407 6            tca                    ;negate for loop counter
036162 010313 6            sta thatsq             ;save
036163 072206 6            ldx endpts             ;validate existing neighbor e
036164 121555 6            jst [naylps] i         ;setup search of neighbors
036165 004211 6            lda naylpe             ;save end index
036166 011574 6            sta tnlpsv
036167 004000 6   tksl1: lda 0
036170 023574 6            cas tnlpsv             ;are we at end of entries?
036171 000000 6             %crash               ;spf cas error
036172 003227 6             jmp tksl0             ;yes, start processing packet
036173 144127 6            lda ltbi ix            ;no, examine this entry
036174 141044 6            car 0&ltbdst
036175 012010 6            era [dlinf<<8.]        ;is this a dead entry?
036176 101040 6            snz
036177 003225 6             jmp tksle             ;yes, then it is ok
036200 004313 6            lda thatsq             ;no, see if it is in update
036201 101400 6            smi
036202 003217 6             jmp tksl3             ;if rupnn=0, don't search
036203 010310 6            sta spfsor             ;(used to be SOURCE but TASK
036204 004025 6            lda [seqh]
036205 014311 0            add that
036206 010312 6            sta thatb              ;pointer to line entries
036207 144127 6   tksl2: lda ltbi ix              ;get our entry
036210 112312 6            era thatb i
036211 141050 6            cal 0&ltbnay&rupnei
036212 101040 6            snz                    ;is this a match?
036213 003225 6             jmp tksle             ;yes, advance to our next ent
036214 024312 6            irs thatb              ;no, look at next update entr
036215 024310 6            irs spfsor             ;(SPFSOR is used as loop cou
036216 003207 6             jmp tksl2             ;more update entries
                                                  ;missing entry must be assume
036217 144127 6   tksl3: lda ltbi ix              ;get our missing entry
036220 141050 6            cal
036221 012010 6            era [dlinf<<8.]        ;make it dead
036222 032310 6            stx spfsor             ;save x
036223 021250 6            jst spfalg             ;call spf for this entry
036224 072310 6            ldx spfsor             ;restore x
036225 024000 6   tksle: irs 0                    ;continue with next line ent
036226 003167 6            jmp tksl1              ;note: irs should'nt skip her
```

```
036227 004313 6   tsksl0: lda thatsq          ;packet valid
036230 101400 6          smi
036231 003243 6           jmp tskabt          ;if rupnn=0, don't continue
036232 010310 6          sta spfsor
036233 004025 6          lda [seqh]          ;prepare to call spf for each
036234 014311 6          add that
036235 010312 6          sta thatb
036236 104312 6   tsksl:  lda thatb i         ;get next line entry
036237 021250 6          jst spfalg          ;call spf
036240 024312 6   tskslp: irs thatb          ;advance to next update entry
036241 024310 6          irs spfsor          ;count entries
036242 003236 6           jmp tsksl          ;not last one
                                             ;exit gracefully via tskabt
                  tskabt:
036243 001001 6          .inh m2i
036244 072311 6 1        ldx that
036245 004071 6 1        lda [ruptsk]
036246 121556 6 1        jst [rupfls] i      ;flush packet if last user
036247 003146 6 1        jmp tskspf+1        ;repeat tskspf if more on que
```

```
; ** the spf algorithm **
                    .lev spf
036250 000000 6   spfalg: 0
036251 010210 6           sta update          ;save while we find entry in
036252 141050 6           cal 0&rupnei        ;get neighbor number
036253 010205 6           sta endptd
036254 012206 6           era endpts
036255 101040 6           snz                 ;is line looped?
036256 103250 6            jmp spfalg i       ;yes, don't process
036257 004205 6           lda endptd          ;do an impchk on endpoints
036260 001001 6           .inh all
036261 121557 6 0         jst [impchk] i
036262 121560 6 0          jst [spfabt] i     ;bad, halt spf
036263 004206 6 0         lda endpts
036264 121557 6 0         jst [impchk] i
036265 121560 6 0          jst [spfabt] i     ;bad, halt spf
036266 000401 6 0         .enb spf
036267 072205 6           ldx endptd          ;make sure there is an entry
036270 021405 6           jst fndent
036271 004205 6           lda endptd          ;and one for s to d
036272 072206 6           ldx endpts
036273 021405 6           jst fndent
036274 004210 6           lda update          ;get entry from update packet
036275 141140 6           icl 0&rupdel        ;extract new distance
036276 026210 6           ima update          ;update<-distance, a<-whole e
036277 166127 6           ima ltbi ix 0&ltbdst&ltbnay ;update ltb
036300 141140 6           icl 0&ltbdst
036301 140407 6           tca
036302 014210 6           add update          ;compute improvement (if any)
036303 101040 6           snz
036304 103250 6            jmp spfalg i       ;no change, no processing req
036305 010207 6           sta deltad          ;store change
036306 004210 6           lda update
036307 022007 6           cas [dlinf]         ;is line now dead?
036310 000000 6            %crash             ;spf cas error
036311 003346 6            jmp spf2           ;yes, line dead
036312 072206 6   spf1:   ldx endpts
036313 144131 6           lda spfrti ix
036314 100400 6           spl 0&spfded        ;is source unreachable?
036315 103250 6            jmp spfalg i       ;yes, don't process, just upd
```

```
036316 121561 6           jst [rtinc] i          ;no, go process this update
036317 073552 6           ldx [-nnodes-1]        ;done, do post-pass
036320 032207 6   spfpst: stx deltad             ;save index
036321 145562 6           lda [pdist+nnodes+1] ix ;a<-new distance to this node
036322 006044 6           ana [pdst]
036323 001001 6           .inh m2i
;protect spfrut
036324 022044 6 1         cas [pdinf]            ;is distance infinite?
036325 000000 6 1          %crash                ;spf cas error
036326 003351 6 1          jmp spfpu             ;yes, node is unreachable (dc
036327 144132 6 1         lda spfrei ix          ;no, make sure dead bit off
036330 140100 6 1         ssp 0&spfded
036331 150132 6 1         sta spfrei ix
036332 000401 6 1         .enb spf
036333 024000 6   spfplp: irs 0                  ;loop through all nodes
036334 003320 6            jmp spfpst
036335 073552 6           ldx [-nnodes-1]
036336 024000 6   tskthd: irs 0
036337 144132 6           lda spfrei ix
036340 100400 6           spl
036341 003336 6            jmp tskthd
036342 006072 6           ana [route]
036343 016052 6           sub [1]
036344 111563 6           sta [thd] i            ;thd=this modem for sync time
036345 103250 6           jmp spfalg i


                  .lev spf
036346 004044 6   spf2:   lda [pdinf]            ;if update=dlinf, deltad=pdin
036347 010207 6           sta deltad
036350 003312 6           jmp spf1
```

```
                    .lev spf
                    .lck m2i
036351 144132 6 1 spfpu:  lda spfrei ix       ;node unreachable, sever from
036352 100400 6 1         spl 0&spfded        ;was it already severed?
036353 003332 6 1          jmp spfplp-1       ;yes, then already ok
036354 140500 6 1         ssm 0&spfded        ;no, mark it unreachable now

036355 150132 6 1         sta spfrei ix       ;and sever from tree
036356 000401 6 1         .enb spf
036357 005564 6           lda [nnodes+1]
036360 014000 6           add 0
036361 010000 6           sta 0               ;get positive imp number
036362 121555 6           jst [naylps] i      ;set up to search this node's
036363 145544 6   spfpul: lda [lflag] ix      ;get current link setting
036364 006044 6           ana [-1?lflg]       ;mark it unused
036365 167544 6           ima [lflag] ix
036366 006070 6           ana [lflgb?lflgf]   ;was link back link?
036367 101040 6           snz
036370 003401 6            jmp spfpu2         ;no, continue to next neighbo
036371 144127 6           lda ltbi ix         ;yes, get neighbor number
036372 141050 6           cal 0&ltbnay        ;who is father of this node
036373 010000 6           sta 0
036374 004207 6           lda deltad
036375 015564 6           add [nnodes+1]      ;positive node number of son

036376 120125 6           jst fndni i         ;find entry for son in father
036377 006044 6           ana [-1?lflg]       ;mark it unused, too
036400 151544 6           sta [lflag] ix
036401 121565 6   spfpu2: jst [naylpi] i      ;bump to next neighbor in son
036402 003363 6            jmp spfpul         ;not end, continue
036403 072207 6           ldx deltad          ;end, resume post-pass loop
036404 003333 6           jmp spfplp
```

```
036405 000000 6   fndent: 0                       ;find or create entry for a 
036406 011573 6           sta fnded               ;a=dest. on line
036407 033572 6           stx fndes               ;x=source on line
036410 121555 6           jst [naylps] i          ;prepare to loop thru source'
036411 033576 6           stx fnd1x               ;save index of source's 1st e
036412 004000 6   fnden1: lda 0                   ;get current index
036413 022211 6           cas naylpe              ;check for end of entries
036414 000000 6            %crash                 ;spf cas error
036415 003425 6            jmp fnden2             ;no exact match, go try usurp
036416 144127 6           lda ltbi ix             ;pick up next entry
036417 013573 6           era fnded
036420 141050 6           cal 0&ltbnay
036421 101040 6           snz                     ;exact match?
036422 103405 6            jmp fndent i           ;yes, all done
036423 024000 6           irs 0                   ;no, keep looking
036424 003412 6           jmp fnden1

036425 073572 6   fnden2: ldx fndes              ;start searching for usurpabl
036426 033575 6           stx fndimp              ;init index of current entry'
036427 073576 6           ldx fnd1x               ;start with sources, down to
036430 033577 6   fnden3: stx fndex               ;save current ltb index
036431 073575 6   fnden7: ldx fndimp
036432 145543 6           lda [ntb+1] ix          ;check ntb index
036433 006033 6           ana [ntbidx]
036434 023577 6           cas fndex
036435 100000 6            skp
036436 003475 6             jmp fnden0            ;fndimp too low
036437 073577 6           ldx fndex               ;restore x=ltb index
036440 004000 6           lda 0
036441 023566 6           cas [nlines]            ;check for end of ltb
036442 000000 6            %crash                 ;spf cas error
036443 121560 6            jst [spfabt] i         ;yes, the end: now what?
036444 144127 6           lda ltbi ix             ;pick up next entry
036445 012010 6           era [dlinf<<8.] 0&ltbdst ;dead line?
036446 033571 6           stx fndsv1              ;maybe, save index in case
036447 101040 6           snz
036450 003501 6            jmp spfnw1             ;yes, and nay=0
036451 141044 6           car 0&ltbdst
036452 100040 6           sze                     ;dead, nay not 0?
036453 003472 6            jmp fnden5             ;not dead, keep looking
036454 144127 6           lda ltbi ix             ;dead, must check reverse
036455 141050 6           cal 0&ltbnay
036456 010000 6           sta 0
036457 005575 6           lda fndimp              ;get current entry's source
036460 013573 6           era fnded               ;is it us?
036461 101040 6           snz
036462 003472 6            jmp fnden5             ;yes, then don't steal it
036463 005575 6           lda fndimp              ;no
036464 120125 6           jst fndni i
036465 144127 6           lda ltbi ix
036466 012010 6           era [dlinf<<8.]         ;is reverse also dead?
036467 141044 6           car 0&ltbdst
036470 101040 6           snz
```