```
                    .lev h2i
011617 000000 4   hipkt:  0
011620 144252 4           lda hispi ix          ;copy header
011621 014013 4           add [typh]
011622 011717 4           sta hit2
011623 144263 4           lda hihti ix
011624 111717 4           sta hit2 i 0&typh
011625 025717 4           irs hit2
011626 025717 4           irs hit2 0&chkh
011627 004106 4           lda mine
011630 111717 4           sta hit2 i 0&srch
011631 025717 4           irs hit2
011632 144264 4           lda hihsi ix
011633 111717 4           sta hit2 i 0&seqh
011634 025717 4           irs hit2
011635 144262 4           lda hihpi ix
011636 111717 4           sta hit2 i 0&pkth
011637 025717 4           irs hit2
011640 144266 4           lda hihii ix
011641 111717 4           sta hit2 i 0&dsth
011642 025717 4           irs hit2
011643 144267 4           lda hihli ix
011644 111717 4           sta hit2 i 0&midh
011645 072247 4           ldx hiiocb            ;get ptr to IOCB
011646 044003 4           lda iocb.size x       ;get length
011647 040474 4           lgr 4.                ;in words, not bits
011650 014055 4           add [hdrl]            ; overhead words
011651 011720 4           sta hit3              ;get and save word count
011652 072245 4           ldx hip               ;fix index
011653 144270 4           lda hifpi ix          ;is it first packet?
011654 101040 4           snz
011655 003661 4            jmp hipkt2           ;no
011656 005720 4           lda hit3              ;yes, subtract padding
011657 156253 4           sub hilti ix          ;from received word count
011660 011720 4           sta hit3
011661 144263 4   hipkt2: lda hihti ix
011662 006062 4           ana [ldrtag]
011663 101040 4           snz                   ;are we tagging?
011664 003676 4            jmp hipktg           ;no, skip down
011665 025717 4           irs hit2
011666 140040 4           cra
011667 111717 4           sta hit2 i            ;clear first data word if tag
011670 005720 4           lda hit3
011671 100040 4           sze
011672 003676 4            jmp hipktg
011673 144263 4           lda hihti ix          ;length is 0 - no tagging
011674 012062 4           era [ldrtag]          ;   turn off tag bit
011675 150263 4           sta hihti ix
```

```
011676 144257 4   hipktg: lda hiiti ix          ;get input time
011677 010001 4          sta ireg
011700 144252 4          lda hispi ix           ;get buffer address
011701 010000 4          sta 0
011702 005720 4          lda hit3               ;get computed word count
011703 141240 4          icr
011704 052004 4          era wrdc x
011705 141044 4          car
011706 052004 4          era wrdc x
011707 050004 4          sta wrdc x 0&pktsiz
011710 004001 4          lda ireg               ;get saved input time
011711 050121 4          sta itst x
011712 004245 4          lda hip                ;use physical host no
011713 140500 4          ssm 0&hstmod           ;for input channel
011714 050005 4          sta inch x
011715 072245 4          ldx hip
011716 103617 4          jmp hipkt i

                    .lev var
011717         V  hit2:  .block 1
011720         V  hit3:  .block 1

;generate h-i checksum on packet in x
                    .lev h2i
011721 000000 4   hicksm: 0
011722 020002 4          jst chksum             ;compute checksum
011723 003731 4           jmp hicbdl            ;bad length
011724 140407 4          tca
011725 054010 4          add chkh x
011726 050010 4          sta chkh x             ;true checksum
011727 025721 4          irs hicksm             ;ok, skip return
011730 103721 4          jmp hicksm i

011731 140407 4   hicbdl: tca
011732 141240 4          icr
011733 012245 4          era hip
011734 001001 4          .inh all
                         ;defhlt(/27. hi bad packet length, a=length/host, x=p
011735 120120 4 0        jst hltjst i
011736 044004 4 0        lda wrdc x
011737 141050 4 0        cal 0&pktsiz
011740 013777 4 0        era [(hdrl+2)<<8.]
011741 050004 4 0        sta wrdc x             ;standard control message wor
011742 000401 4 0        .enb h2i
011743 103721 4          jmp hicksm i           ;no-skip return
```

```
                        .section pg0
                        .lev var
000245          V   hip:    .block 1            ;current host number
000246          V   hilp:   .block 1            ;ptr to leader area
000247          V   hiiocb: .block 1            ;ptr to IOCB

000250 040113   V   hirwi:  hirw                ;flag saying whether H2I is w
000251 040067   V   hirsi:  hirs                ;set to 1 by I2H if H2I shoul
000252 040137   V   hispi:  hisp                ;ptr to current packet, 0=non
000253 040164   V   hilti:  hilt                ;padding (0-9) for this host

000254 040210   V   hiwti:  hiwt                ;timestamp of leader (for tar
000255 040234   V   hitbi:  hitb                ;ptr to current TSB, -1=erro
000256 040043   V   hiali:  hial                ;lh=allocate time, rh=count
000257 040017   V   hiiti:  hiit                ;saved input time
000260 040260   V   hibli:  hibl                ;local block number ??
000261 040305   V   hityi:  hity                ;packet type ??
000262 040331   V   hihpi:  hihp                ;packet type ??

000263 040355   V   hihti:  hiht                ;TYPH for packets of this mes
000264 040401   V   hihsi:  hihs                ;SEQH for packets of this mes
000265 040425   V   hihhi:  hihh                ;PKTH for packets of this mes
000266 040451   V   hihii:  hihi                ;DSTH for packets of this mes
000267 040475   V   hihli:  hihl                ;MIDH for packets of this mes
000270 040521   V   hifpi:  hifp                ;non-zero if first packet, ze
000271 040545   V   hibgi:  hibg                ;non-zero if RESETT has kille

                        .section heap
040017          V   hiit:   .block th
040043          V   hial:   .block th
040067          V   hirs:   .block th
040113          V   hirw:   .block th
040137          V   hisp:   .block th
040163          V   b5hisp: .block 1            ;gets IMPDEF'ed - must be aft
040164          V   hilt:   .block th
040210          V   hiwt:   .block th
040234          V   hitb:   .block th
040260          V   hibl:   .block th+1         ;the +1 is for back host 5
040305          V   hity:   .block th
040331          V   hihp:   .block th
040355          V   hiht:   .block th
040401          V   hihs:   .block th
040425          V   hihh:   .block th
040451          V   hihi:   .block th
040475          V   hihl:   .block th
040521          V   hifp:   .block th
040545          V   hibg:   .block th
040571          V   hildra: .block 6*th
```

```
                .section pg12
                .lev bck

;This is the Fake Host to IMP word-at-a-time interface. It
;merely emulates what the hardware would have done.  Enter with
;fake host number in X, word in A.  Call with sign of X set and A=padding if
;this is EOM.

012000 000000 9   gam:    0
012001 011131 9   gamtop: sta gama                ;save caller's data word
012002 004000 9           lda 0                   ;get caller's fake host numbe
012003 011133 9           sta gameom              ;save EOM flag
012004 140100 9           ssp                     ;less sign
012005 011132 9           sta gamx                ;save caller's fake host numb
012006 010000 9           sta 0
012007 145720 9           lda [h2ipct+nh] ix      ;get pointer to correspondin
012010 011130 9           sta gampcb              ;save it, too
012011 145721 9   gam1:   lda [gamsiz] ix         ;do we have an IOCB in progre
012012 100040 9           sze
012013 003033 9            jmp gamgot             ;yes, then continue
012014 073130 9           ldx gampcb              ;no, has H2I posted an IOCB?

012015 044022 9           lda pcb.iocb x
012016 101040 9           snz
012017 003111 9            jmp gamwat             ;not yet, wait and try again


;Use the IOCB in pcb.iocb
;Copy it's size and address into local tables for use.

012020 010000 9           sta 0
012021 044003 9           lda iocb.size x         ;get the size
012022 040474 9           lgr 4.                  ;in words, please
012023 010001 9           sta ireg                ;save for a moment
012024 044004 9           lda iocb.addr x         ;get the address
012025 073132 9           ldx gamx                ;get fake host number
012026 151722 9           sta [gamadr] ix         ;set up the address param
012027 004001 9           lda ireg
012030 151721 9           sta [gamsiz] ix         ;and the size one, too
012031 101040 9           snz                     ;zero size?
012032 003052 9            jmp gamend             ;yes, then finish with iocb

;Proceed to get a word from the transfer.  If the count
;has gone to zero, finish with the IOCB.

012033 073132 9   gamgot: ldx gamx                ;get fake host number back
012034 145722 9           lda [gamadr] ix         ;get the address
012035 010001 9           sta ireg                ;save it
012036 005131 9           lda gama                ;get word
012037 110001 9           sta ireg i              ;store the word so addressed

012040 165722 9           irs [gamadr] ix         ;bump it for next time
012041 145721 9           lda [gamsiz] ix         ;get the remaining size
012042 016052 9           sub [1]
012043 151721 9           sta [gamsiz] ix         ;update count
012044 101040 9           snz                     ;last one?
012045 003052 9            jmp gamend             ;yes, finish with the IOCB
012046 005133 9           lda gameom              ;is this EOM?
012047 100400 9           spl                     ;check sign
```

```
012050 003052 9          jmp gamend              ;yes, finish with IOCB
012051 103000 9          jmp gam i               ;no, return to caller
```

```
;Here when the size was found to be zero.  Finish with the IOCB.  Enter
;with X=fake host number.

012052 145721 9   gamend: lda [gamsiz] ix          ;get size
012053 041474 9           lgl 4.                   ;convert to bits
012054 011134 9           sta gambit               ;save it for a moment
012055 073130 9           ldx gampcb               ;get pointer to PCB
012056 044024 9           lda pcb.put x            ;get ptr to PUT queue for lat
012057 010001 9           sta ireg
01206C 140040 9           cra
012061 066022 9           ima pcb.iocb x           ;poof
012062 101040 9           snz
012063 000000 9            %crash                  ;SUCK: IOCB disapeared during
012064 010000 9           sta 0                    ;get to IOCB
012065 044003 9           lda iocb.size x          ;get total size
012066 017134 9           sub gambit               ;minus amount left = amount
012067 050003 9           sta iocb.size x
01207C 005133 9           lda gameom               ;get EOM flag
012071 100400 9           spl
012072 003107 9            jmp gam3                ;if EOM, go get EOM bit
012073 140040 9           cra
012074 050005 9   gam4:   sta iocb.flags x
012075 004001 9           lda ireg                 ;get to PUT queue
012076 000002 9           enq                      ;send it to the host code
012077 100000 9            skp
01210C 000000 9              %crash                ;SUCK: saw extra IOCB
012101 073130 9           ldx gampcb               ;get pointer to I2H PCB
012102 000043 9           gpr                      ;poke it
012103 073132 9           ldx gamx
012104 140040 9           cra
012105 151721 9           sta [gamsiz] ix          ;set gamsiz to 0 when done wi
012106 103000 9           jmp gam i                ;else just return having writ

012107 004061 9   gam3:   lda [ %c18.eom ]         ;if gameom < 0, set EOM
01211C 003074 9           jmp gam4
```

```
;Here when we need to wait and then go back to GAMTOP.

012111 073132 9   gamwat: ldx gamx                ;get fake host number
012112 005131 9           lda gama                ;get A
012113 151723 9           sta [gamat] ix          ;save it
012114 005133 9           lda gameom              ;get X+EOM bit
012115 151724 9           sta [gamet] ix
012116 005000 9           lda gam                 ;save caller's address
012117 151725 9           sta [gamgt] ix
012120 120111 9           jst doze i              ;go to sleep
012121 145725 9           lda [gamgt] ix
012122 011000 9           sta gam                 ;restore it all
012123 145724 9           lda [gamet] ix
012124 010001 9           sta ireg
012125 145723 9           lda [gamat] ix
012126 072001 9           ldx ireg
012127 003001 9           jmp gamtop

012130        9   gampcb: .block 1
012131        9   gama:   .block 1
012132        9   gamx:   .block 1
012133        9   gameom: .block 1
012134        9   gambit: .block 1                ;holds msg size in bits (gams

                  .section heap
040761        9   gamat:  .block fh               ;holds A during DOZE
040765        9   gamet:  .block fh               ;holds X+EOM during DOZE
040771        9   gamgt:  .block fh               ;hold GAM entry point during
040775        9   gamsiz: .block fh               ;holds IOCB.SIZE during xfer

041001        9   gamadr: .block fh               ;holds IOCB.ADDR during xfer
```

```
                .stitl IMP TO HOST (I2H)
                .INCLUDE i2h.m4
;NMFS notes:



;I2H's PCB:

        000026  i2hpcb.getqh = pcb.ff           ;get queue header
        000032  i2hpcb.putqh = i2hpcb.getqh + qhedr.size ;put queue header
        000036  i2hpcb.iocb = i2hpcb.putqh + qhedr.size ;IOCB
        000044  i2hpcb.swcsr = i2hpcb.iocb + biocb.size ;sw host CSR
        000045  i2hpcb.swpdv = i2hpcb.swcsr + 1 ;sw host PDV PCB address
        000046  i2hpcb.swxdv = i2hpcb.swpdv + 1 ;sw host XDV subr address
        000047  i2hpcb.size = i2hpcb.swxdv + 1  ;size
```

```
;Ready lines, tardiness, upness and downness:
;
;**** Warning: the text below is no longer completely accurate.  It
;              should be updated!

;The interactions between H2I, I2H and the host are somewhat complex. I
;will attempt to illuminate the mechanisms enjoyed by the NMFS IMP.
;
;Both I2H and H2I have two major states for their hosts: UP and DOWN.
;A DOWN host is defined as one whose ready line is down, UP means its
;ready line is up. The notion of "resetting" a host, or that of declaring
;a host "tardy" is simply a transition from UP to DOWN. This transition
;can be made at ANY time (i.e. even when the ready line is up, as in
;the tardy case, below). If one transitions into DOWN state with the
;ready line still UP, one will return to UP state momentarily.
;
;Since we desire I2H and H2I each to be self-contained as possible, each
;implements its own "state machine" (although it takes the form of a
;program as opposed to some transition matrix). First, let's look at
;I2H's state machine:
;
;       DOWN state:
;               If the ready line comes up, we schedule 3 NOPs, etc.
;               When they go out (IHFL, the word containing the command
;               to do special I2H functions is zero), we transition
;               to UP state. Transitioning AFTER the NOPs are sent
;               replaces the old H2I blocking mechanism.
;
;       UP state:
;               If the ready line drops, we set the HOST to "down"
;               (host status is simply a word which is publicly available
;               for reporting, etc.) and transition to DOWN state, resetting
;               the software state information as we go (the routine IHR in
;               this module does most of this). IHR sets HIRS to one
;               which will cause the H2I state machine to reset, also.
;
;               If the host becomes "tardy" (takes too long to perform some
;               transaction), we perform much the same actions as if
;               if the ready line dropped, with two exceptions: first, the
;               public host state word is set to "tardy" as opposed to down.
;
;H2I's state machine is similar:
;
;       DOWN state:
;               If the HOST variable becomes up, we transition into UP state.
;               In other words, we wait for I2H to delcare the host up.
;               Recall that I2H declares the host up when its ready line
;               is raised.
;
;       UP state:
;               If the HIRS becomes set, we reset the H2I variables and
;               transition to DOWN state, then clear HIRS.
```

```
;There is an addition interaction that should be mentioned here.  TASK and
;timecut call RESETR to kill off message blocks.  Sometimes, they kill one
;that I2H is using at a given instant (it's outputting a message on that block

;for example).  It's okay for I2H to continue to output the message in this
;case but it mustn't send back the RFNM because there is no virtual circuit
;to send it on.  When RESETR takes the message block away, set sets the IHIR
;flag (IR = ignore RFNM) which will prevent the tail-end of I2H from calling
;IHRALY to mark RFNM to be sent.

;This is the beginning of I2H. We are started here after INI runs. We
;sleep for time to hold the host down while the IMP brings up its
;trunk lines. This is to prevent over-zealous hosts from trying to make
;connections as soon as the local IMP comes up, only to get lots of
;destination deads while the local IMP is still isolated from the net.

                .section pg12
                .lev i2h

012135 121726 3 i2hini: jst [i2hnmf] i          ;do NMFS initialization
012136 004016 3         lda [ -3 ]              ;number of 30 second interval
012137 010001 3         sta ireg
012140 121727 3 i2hinw: jst [ih30s] i           ;wait for 30 seconds each tim
012141 000103 3         spr
012142 100000 3          skp
012143 000000 3           %crash                ;I2HINI: got poked
012144 024001 3         irs ireg                ;more to wait for?
012145 003140 3          jmp i2hinw             ;yes - do it
012146 121730 3         jst [ihwspr] i          ;no, finish up from the debre

;IHD is a convenient entrance to IHR for setting the host to DOWN state.

012147 004006 3 ihd:    lda [hstoff]            ;then enter IHR with AC=host

                        ;fall into IHR with A=HSTOFF
```

```
                        ;from previous page

;This code resets the host. Enter with A having new value for the
;published "HOST" state (HSTOFF or HSTTDY).  First, we call IHLGET
;until it fails. IHLGET will de-queue TSBs off the host reply queue
;and make them into leaders (see the commentary at IHLGET).

012150 150276 3   ihr:    sta hosti ix          ;set up host state
012151 004014 3           lda [-1]              ;tell H2I to reset sometime,
012152 150251 3           sta hirsi ix

012153 004006 3           lda [%c18.reset]      ;tell the 1822 device to kill
012154 000020 3           pcb
012155 121731 3           jst [i2hxdv] i        ;ready line will be flapped b
012156 072272 3           ldx ihp               ;restore index

012157 121732 3   ihrl:   jst [ihihlp] i        ;set up IHLP
012160 121733 3           jst [ihlget] i        ;try to deq a TSB/leader
012161 100000 3            skp                  ;when it fails, exit the loo
012162 003157 3             jmp ihrl           ;else loop until all TSBs rem

012163 166300 3           ima ihspi ix          ;clear and check the type of
012164 101400 3           smi                   ;if minus, must not be 0 or 1
012165 022052 3           cas [1]               ;what type of xfer?
012166 003171 3            jmp ihrdis           ;minus or .gt. 1, must be buf
012167 003174 3            jmp ihr2             ;if 1, or 0, not a buffer
012170 003174 3            jmp ihr2

;here when the old IHSP wasn't 0 or 1. This means there was a buffer in
;flight out the interface. Call IH2DIS which will pass the buffer to
;the discard host.

012171 010000 3   ihrdis: sta 0                 ;put buffer ptr in X
012172 121734 3           jst [ih2dis] i        ;discard the packet in flight
012173 072272 3           ldx ihp               ;fix index
012174 140040 3   ihr2:   cra                   ;flush all the buffers
012175 121735 3           jst [ihqs] i

;Having completed most of the reset procedure, we now flap our ready line
;and then tell H2I to reset by setting HIRS to -1.

012176 000020 3           pcb
012177 004053 3           lda [%c18.lower]
012200 121731 3           jst [i2hxdv] i
012201 121736 3           jst [ih640] i
012202 000103 3           spr
012203 101000 3            nop
012204 004052 3           lda [%c18.raise]
012205 121731 3           jst [i2hxdv] i
012206 121736 3           jst [ih640] i
012207 000103 3           spr
012210 101000 3            nop
012211 072272 3           ldx ihp

;Now do any NCC command which was waiting

012212 121737 3           jst [ihncc] i
012213 100000 3            skp
012214 003220 3             jmp ihrrw           ;nothing to do, skip it
```

```
012215 121731 3           jst [i2hxdv] i          ;else loop or unloop
012216 140040 3           cra
012217 010101 3           sta nccc                ;then clear command word

;Then just sit waiting for the ready line to come up.

012220 121730 3   ihrrw:  jst [ihwspr] i          ;for SPR, below, XDV, above
012221 121740 3           jst [ihrdyt] i          ;test the ready line
012222 100000 3            skp
012223 003230 3             jmp ihrup             ;if up, declare up and contin
012224 121736 3           jst [ih640] i           ;else look again shortly
012225 000103 3           spr
012226 101000 3            nop
012227 003220 3           jmp ihrrw               ;look again

;When the ready line comes up, schedule 3 NOPs and a reset
;If host was down, then declare host up if ready line is up.  If host was
;tardy, then also wait until a NCP goes out before declaring host up.
;The reason for making it easier for down hosts to be declared up is so
;that looping plugs will work (because H2I waits until I2H declares the
;host up, which wouldn't happen if H2I didn't accept the first looped NOP).

012230 005741 3   ihrup:  lda [740]              ;remind ourselves to send 3 N
012231 150277 3           sta ihfli ix            ;update flags word
012232 144276 3           lda hosti ix            ;was host down (not tardy)?
012233 101040 3           snz                     ;if tardy, then keep down
012234 164276 3            irs hosti ix           ;if only down, then declare i
012235 003254 3           jmp ih0                 ; enter main-line

;Here to declare the host TARDY as well as reset it. Note that we call
;IHWSPR because people who come here are usually doing so because of a
;NMFS I/O timeout form the following code sequence:
;
;  SPR
;   jmp iht                    ;host is tardy..
;  etc.
;
;In addition, if a fake host goes tardy, we crash.

012236 121730 3   iht:    jst [ihwspr] i          ;return from NMFS debreak (i
012237 121742 3           jst [ihfakt] i          ;is this fake?
012240 000000 3             %crash                ;IHT: Fake host went tardy
012241 121740 3           jst [ihrdyt] i          ;the ready line really down?

012242 003147 3            jmp ihd                ;yes, say host is down, not t
012243 004053 3           lda [hsttdy]            ;set up to declare it tardy
012244 003150 3           jmp ihr                 ;and go to reset routine
```

```
;This is the main line of I2H (after initialization). We always try to
;clear up buffers older than 30 seconds. Then we check for interface
;commands which might be need (sending in NOPs, etc.)
;Failing that, we look for leaders to send from TSBs and finally, for
;new messages to start.

;At IH0DB, we debreak and watch the host's ready line. If we awaken from
;our self-set timeout, then we look at ready line. If we awaken from
;a POKE, we look for something to do.

012245 121730 3   ih0db:  jst [ihwspr] i          ;fix up from possible SPR, be
012246 140040 3            cra                    ;tell rest of world that we'r
012247 150300 3            sta ihspi ix           ; ... by clearing IHSP
012250 121743 3            jst [ihrchk] i         ;look for ready line down, et
012251 121736 3            jst [ih640] i          ;set up for slow sleep
012252 000103 3            spr                    ;enter here to debreak
012253 101000 3             nop                   ;pretend we're poked, look ar

;Note: IH0 always calls IHWSPR because some people jump to IH0 after
;a NMFS debreak (see IHFLD0 routines, for example).

012254 121730 3   ih0:     jst [ihwspr] i          ;return from NMFS debreak
012255 121737 3            jst [ihrcc] i          ;any NCC commands waiting?
012256 003147 3             jmp ihd               ;yes, force thru IHR to proce
012257 121743 3            jst [ihrchk] i         ;check the ready line, jump (

012260 004074 3            lda [ldrnlf]           ;always set up first leader w
012261 110273 3            sta ihlp i
012262 024273 3            irs ihlp               ;and pointer leader ptr to se

012263 144277 3            lda ihfli ix           ;any flags to operate on?
012264 100040 3            sze
012265 003655 3             jmp ihfldo            ;yes - go off and do them

;Here when there are no special leader flags set in IHLF.
;Look for TSBs carrying control message leaders.

012266 121733 3            jst [ihlget] i         ;pick up some control message
012267 003304 3             jmp ih1               ;if none of those, try for re

012270 005744 3   ihlo5:   lda [100000 + 16.*5]   ;all control messages are 5 w
                                                   ;and have EOM set in the lead

;Enter here from IHLFD0 routines (like send in NOP, etc.) to have the
;leader area sent to the host. Enter with AC=size of leader (in BITS!)
;and X=host number (IHP). Some JMPs enter at IHL05 for 5-word leader output.

012271 165303 3   ihlout:  irs stats.cnt3i ix      ;**stats
012272 101000 3             nop
012273 121745 3            jst [ihlenq] i         ;output from the leader area

012274 121727 3            jst [ih30s] i          ;set up
012275 000103 3            spr
012276 003236 3             jmp iht               ;took too long, make it tardy
012277 121730 3            jst [ihwspr] i

;declare host up!!!!

012300 004052 3            lda [hstup]            ;when it takes something, dec
```

```
012302 003254 3             jmp ih0                 ;start at top of loop again

                        stats.cnt3i:
012303 040000 3             stats.dummy
```

```
;Here from IH0 when there were no control messages to output. We now
;offer to output a data message.

012304 145746 3   ih1:    lda [shpq] ix          ;look on the priority queue f
012305 101040 3           snz
012306 003311 3            jmp ih1np             ;nothing on priority, try nor
012307 005746 3           lda [shpq]             ;was on priority, let AC=bas
012310 003315 3           jmp ih1got             ;and process it

012311 145747 3   ih1np:  lda [shq] ix           ;normal queue have anything?

012312 101040 3           snz
012313 003245 3            jmp ih0db             ;nothing to do
012314 005747 3           lda [shq]

;here from above with AC=base of queue block for the queue on which a message
;resides. Set IHWQ to point to the queue.

012315 014272 3   ih1got: add ihp                ;offset into the queue block
012316 150301 3           sta ihwqi ix           ;save ptr to it
012317 150302 3           sta ihfpi ix           ;set first packet flag to non
012320 010001 3           sta ireg               ;also, prepare to read the ad
012321 172001 3           ldx ireg i             ;into X

;armed with the packet's address in X, fill in information in the leader
;area. Recall that IHLP points to the second word of our particular leader area

012322 044007 3           lda typh x             ;get flags
012323 006074 3           ana [ldrtfl+ldr1fl] 0&lcreg
012324 110273 3           sta ihlp i             ;fill in second leader word
012325 024273 3           irs ihlp               ;step to fourth word
012326 024273 3           irs ihlp

012327 044011 3           lda srch x             ;pick up source imp
012330 110273 3           sta ihlp i             ;fill in fourth word with it

012331 024273 3           irs ihlp

012332 044015 3           lda midh x             ;get message ID and subtype
012333 110273 3           sta ihlp i             ;into fifth word
012334 024273 3           irs ihlp

012335 044006 3           lda neth x             ;get length
012336 110273 3           sta ihlp i             ;into sixth word

012337 000010 3           %rdclok                ;get time stamp
012340 050121 3           sta itst x             ;install in into the packet

;Now check to see whether its a RAW packet or not.

012341 044015 3           lda midh x             ;get subtype
012342 006072 3           ana [subtyp]
012343 100040 3           sze 0&subunc           ;raw?
012344 003623 3            jmp ihnraw            ;yes, handle differently

                          ;fall into next page
```

```
                        ;from previous page

;At this point we have built much of the leader into IHLDRA. The rest
;of the information which is needed lives in the message block. Note
;that IHLP points to the sixth leader word, from the manipulations above.

012345 044012 3          lda seqh x             ;get receive block ptr
012346 141050 3          cal 0&blknum
012347 015750 3          add [rmblkt]           ;build a pointer to the recei
012350 010000 3          sta 0                  ; ... associated with this VC
012351 044160 3          lda mbfor x            ;get handling type
012352 006075 3          ana [mbhand]
012353 100400 3          spl 0&ldrhpr
012354 013751 3           era [100010]
012355 040664 3          arr 12. 0&ldrhln
012356 052070 3          era mbhst x            ;and source host
012357 141050 3          cal 0&mbfhst
012360 052070 3          era mbhst x
012361 141340 3          ica 0&ldrhnd&ldrhst

;enter IHNSW3 (read: store word 3) with the source host, handling type in AC

012362 026273 3  ihnsw3: ima ihlp
012363 016011 3          sub [3]                ;IHLP was at word 6, point it
012364 026273 3          ima ihlp
012365 110273 3          sta ihlp i             ;put in third leader word
012366 005752 3          lda [16.*6]            ;these leaders are 6. words (
012367 121745 3          jst [ihleng] i         ;queue the leader output
012370 121727 3          jst [ih30s] i          ;set up a 30 second timeout f
012371 000103 3          spr                    ;let it go out
012372 003236 3           jmp iht               ;took too long, make it tardy
012373 121730 3          jst [ihwspr] i         ;return from NMFS debreak
012374 121743 3          jst [ihrchk] i         ;make sure the ready line is
                                                ;go off to IHD if not

;After sending out the leader, get the packet off the working queue (IHWQ).
;This packet is the one which supplied the information needed to build
;the leader, so it must exist. Also, check its checksum. Note that we
;do not DEQ the NMFS IOCB from C18's PUT queue.

012375 144301 3  ihmore: lda ihwqi ix           ;where is the queue?
012376 010000 3          sta 0
012377 121753 3          jst [ihgetq] i         ;deq the packet
012400 100000 3           skp
012401 000000 3            %crash               ;IH: IH queue foulup
012402 004272 3          lda ihp
012403 026000 3          ima 0                  ;A gets packet; X gets ihp
012404 010275 3          sta ihpakt
012405 150300 3          sta ihspi ix
012406 010000 3          sta 0
012407 020002 3          jst chksum             ;check its checksum
                         ;defhlt(/33. IH packet too long for adder)
012410 021633 3           jst ihbad             ;(IHBAD always returns zero f
012411 100040 3          sze                    ;did checksum turn out okay?

                         ;defhlt(/34. IH detected intra-IMP checksum err)
012412 021633 3           jst ihbad             ;(no)

                        ;fall into next page
```

```
                        ;from previous page

;Now compute the amount of time this packet has left before the whole message
;has taken too long to go out (and the host is therefore to be declared
;tardy). TASK writes the completion time for each of the message's packets
;into INCH of the packets. If INCH>=TIMES (the time in slow ticks), then
;the packet is late and we must declare the host tardy.

012413 072275 3          ldx ihpakt              ;get ptr to packet again
012414 044005 3          lda inch x              ;get TASK's time to finish
012415 016152 3          sub times
012416 022006 3          cas [0]
012417 003422 3           jmp ihmor1             ;if still in the future, fin
012420 003236 3           jmp iht                ;else, make host tardy
012421 003236 3           jmp iht

;The packet is now ready for output. We start the offset at DATA, offset
;by any padding words, found in HILT, if this is the first packet

012422 011563 3   ihmor1: sta ihtprs
012423 072272 3          ldx ihp
012424 121754 3          jst [htppf] i           ;count a packet of thruput
012425 121755 3          jst [ihdeq] i           ;deq the IOCB setting up IHIO
012426 072272 3          ldx ihp                 ;make X point back at host nu
012427 144302 3          lda ihfpi ix            ;look at first packet flag
012430 100040 3          sze                     ;first packet? no - use offse
012431 144253 3           lda hilti ix           ;yes, use offset=HILT
012432 140407 3          tca                     ;make negative offset
012433 015756 3          add [data]
012434 014275 3          add ihpakt              ;index into packet's data
012435 072274 3          ldx ihiocb              ;pick up IOCB pointer
012436 050004 3          sta iocb.addr x         ;that's the addr
012437 072275 3          ldx ihpakt              ;get back ptr to packet
012440 044004 3          lda wrdc x              ;get it's length
012441 141140 3          icl 0&pktsiz            ;from LH of WRDC
012442 016055 3          sub [hdrl]              ;subtract off subnet stuff
012443 041474 3          lgl 4.                  ;in bits, for NMFS
012444 010001 3          sta ireg                ;save data size, in bits
012445 072272 3          ldx ihp
012446 144302 3          lda ihfpi ix            ;is this first packet?
012447 100040 3          sze                     ;if no, then use 0 padding
012450 144253 3           lda hilti ix           ;yes, use padding
012451 041474 3          lgl 4.                  ;convert ot bits
012452 014001 3          add ireg
012453 072274 3          ldx ihiocb
012454 050003 3          sta iocb.size x         ;set up size in iocb
012455 072275 3          ldx ihpakt              ;is it the last packet?
012456 044013 3          lda pkth x
012457 006070 3          ana [lstpkt]
012460 100040 3          sze
012461 004071 3           lda [100000]           ;yes, set sign
012462 052011 3          era srch x              ;get foreign IMP
012463 072272 3          ldx ihp                 ;back to host number
012464 150303 3          sta ihlsi ix            ;remember it foreign IMP and
012465 101400 3          smi                     ;was it last packet?
012466 003474 3           jmp ihnotl             ;not last

;on the last packet, call STATs.
```

```
012467 140100 3            ssp                    ;remove last packet flag (lea
012470 121757 3            jst [htpmf] i          ;count a packet of throughput
012471 121564 3            jst stats.hs1i i       ;**call msg cumstat
012472 004061 3            lda [%c18.eom]         ;set the EOM bit in the IOCB
012473 100000 3             skp                   ;skip over the clearing there
012474 140040 3   ihnotl:    cra                   ;entered from above if not la
012475 072274 3            ldx ihiocb             ;get ptr to IOCB
012476 050005 3            sta iocb.flags x       ;copy LASTIOCB flag into IOCB

                           ;fall into next page
```

```
                        ;from previous page

012477 005563 3   ihsett: lda ihtprs
012500 121760 3          jst [st2nmf] i          ;convert slow ticks to NMFS u
012501 000020 3          pcb
012502 000203 3          tpr                     ;set up our timeout
012503 121761 3          jst [ihenq] i           ;enq the IOCB to NMFS (sets T
012504 000103 3          spr
012505 003236 3           jmp iht                ;if timed-out, declare host t
012506 121730 3          jst [ihwspr] i          ;return from NMFS debreak
012507 121743 3          jst [ihrchk] i          ;make sure ready line ok, off

012510 144300 3          lda ihspi ix            ;normal output complete
012511 010275 3          sta ihpakt
012512 072275 3          ldx ihpakt
012513 044004 3          lda wrdc x
012514 141340 3          ica
012515 101400 3          smi 0&trcpkt            ;must we trace him?
012516 003523 3           jmp ihtrdn             ;no
012517 004272 3          lda ihp                 ;yes, output chan #
012520 001001 3          .inh m2i
012521 120145 3 1        jst trace.m2idone i     ;** trace packet
012522 000401 3 1        .enb i2h
012523 072272 3   ihtrdn: ldx ihp
012524 140040 3          cra
012525 166302 3          ima ihfpi ix            ;reset first pkt switch
012526 101040 3          snz                     ;is this the first pkt?
012527 003536 3           jmp ihtrnf             ;no
012530 140040 3          cra                     ;yes, then send RFNM
012531 166305 3          ima ihiri ix            ;is this RFNM to be ignored b
012532 100040 3          sze                     ; ... went away?
012533 003536 3           jmp ihtrnf             ;yes
012534 072275 3          ldx ihpakt
012535 021565 3          jst ihraly              ;mark up rally since host has
012536 072272 3   ihtrnf: ldx ihp
012537 140040 3          cra
012540 150300 3          sta ihspi ix            ;clear out ihsp
012541 001001 3          .inh fre
012542 166304 3 0        ima ihcei ix            ;error in packet?
012543 100400 3 0        spl
012544 003556 3 0         jmp ih6b               ;yes, put on diag q
012545 072275 3 0        ldx ihpakt
012546 120115 3 0        jst flushi i            ;flush packet
012547 024224 3 0        irs nres                ;give back to reassembly
012550 000401 3 0 ih6f1:  .enb i2h
012551 072272 3          ldx ihp
012552 144303 3          lda ihlsi ix
012553 101400 3          smi                     ;last packet?
012554 003375 3           jmp ihmore             ;no, get next
012555 003254 3          jmp ih0                 ;yes

012556 004275 3   ih6b:   lda ihpakt
012557 072275 3          ldx ihpakt
012560 026200 3          ima diagq               ;put on diag q
012561 050000 3          sta 0 x 0&ptrc
012562 003550 3          jmp ih6f1

                        .lev var
012563          V  ihtprs: .block 1              ;temp for ihsett
```

```
                        stats.hs1i:
012564 013016 V                 jsti2h

                        .lev i2h
012565 000000 3         ihraly: 0                       ;here to change rally table
012566 044013 3                 lda pkth x              ;code for multi-pkt mess for
012567 006073 3                 ana [pktnum]            ;on ihq too long in mid-messa
012570 100040 3                 sze                     ;is this first packet?
012571 103565 3                  jmp ihraly i           ;no, the rfnm has been sent a
012572 044015 3                 lda midh x
012573 006072 3                 ana [subtyp]
012574 012011 3                 era three 0&1cunct
012575 101040 3                 snz                     ;is this a raw pkt?
012576 103565 3                  jmp ihraly i           ;yes, does'nt get rfnm
012577 044012 3                 lda seqh x              ;get blk ptr
012600 141050 3                 cal 0&blknum
012601 015750 3                 add [rmblkt]
012602 010001 3                 sta ireg                ;save
012603 044013 3                 lda pkth x              ;get type of reply
012604 006043 3                 ana [mltpkt+pktcod]     ;0=rfnm1, 1=rfnm8, 3=inc
012605 022052 3                 cas [1]                 ;req1?
012606 100000 3                  skp
012607 140040 3                   cra                   ;yes, type=0
012610 022071 3                 cas [100000]            ;mess8?
012611 100000 3                  skp
012612 141206 3                   aoa                   ;yes, type=1
012613 041474 3                 lgl 4                   ;into position for rallyp
012614 012072 3                 era [17]                ;state=3 (reply), mask=3
012615 052012 3                 era seqh x              ;now get mess no
012616 141050 3                 cal 0&mesnum
012617 052012 3                 era seqh x
012620 072001 3                 ldx ireg                ;restore blk ptr
012621 121762 3                 jst [rallyp] i          ;set new rstate, rtype
012622 103565 3                 jmp ihraly i
```

```
;Here when this packet is a RAW packet. We make it look like a MSG1
;(and adjust the checksum accordingly). We then pick up the source host
;from the packet. Note that for RAW packets, the source host is stored
;in SEQH of the packet (where the foreign block number would normally be).

012623 004070 3   ihnraw: lda [lstpkt+msgcod]     ;make it look like MSG1
012624 066013 3          ima pkth x
012625 056013 3          sub pkth x               ;and fix up the checksum
012626 054010 3          add chkh x
012627 050010 3          sta chkh x
012630 044012 3          lda seqh x
012631 141050 3          cal 0&srchst             ;source host is in RH, clear
012632 003362 3          jmp ihnsw3               ;put this in third word of l

;Here when packet to go out appears to have a checksum error.

012633 000000 3   ihbad:  0
012634 011645 3          sta ihbad1               ;save caller's AC for trap
012635 004272 3          lda ihp
012636 013763 3          era [fhpdis]
                         ;defplc(/nop here to turn off i2h checksum check)
012637 101040 3          snz
012640 003653 3           jmp ihbad2              ;don't report discard errors

012641 005633 3          lda ihbad                ;get caller's PC for trap
012642 027645 3          ima ihbad1               ;put PC before JST, get calle
012643 001001 3          .inh all
012644 100000 3 0        skp                      ;now trap to ncc
012645         3 0 ihbad1:  .block 1
012646 120126 3 0             jst hltncc i
012647 000401 3 0        .enb i2h
012650 072272 3          ldx ihp
012651 004014 3          lda [-1]                 ;flag packet to go to diagtt

012652 150304 3          sta ihcei ix
012653 140040 3   ihbad2: cra                     ;and return with AC=zero (see
012654 103633 3          jmp ihbad i              ;return
```

```
;Here from IH0 when IHFL is non-zero indicating some special action is
;needed. Note that IHFL's contents are still in AC at this point. IHFL
;contains bits, each of which specifies some action to perform, as follows:
;
;       1: send in imp going down
;      40: send a NOP
;     100: send a NOP
;     200: send a NOP
;     400: send in IH reset message
;
;Note that there are many bits which cause NOPs to be sent. This is a feature.

;IHLFD0 processes one bit at a time (the least significant one), turns it
;off, and continues. If you want 3 NOPs, you can set the 40, 100, and 200
;bits simultaneously. One reasonable example of IHFL's use can be seen in
;the host reset code (IHR).  Note that fake hosts never come through here.

012655 140407 3   ihfldo: tca                     ;find least significant one b
012656 146277 3           ana ihfli ix            ;this sequence is beyond expl
012657 152277 3           era ihfli ix
012660 166277 3           ima ihfli ix
012661 152277 3           era ihfli ix

;We have now removed the LS 1 from IHFL and put it in AC. Magic.

012662 022056 3           cas [20]                ;start looking for things to
012663 003671 3            jmp ihfg20             ;greater than 20
012664 000000 3            %crash                 ;IHFLD0: bad function
012665 022053 3           cas [2]                 ;less than 20, must be 1,2,4,
012666 003674 3            jmp ihfnop             ;greater than 2 (4,10) both m
012667 000000 3            %crash                 ;IHFLD0: bad function
012670 103764 3            jmp [ihfdwn] i         ;1 =  send imp going down

012671 022061 3   ihfg20: cas [200]               ;here when .gt. 20, check ag
012672 003710 3            jmp ihfrst             ;400 = send reset
012673 101000 3            nop                    ;200, 100 and 40 mean send in

                          ;fall into IHFNOP on next page for 200,100,40
```

```
                        ;from previous page

;Here from the CAS matrix to send a NOP to the host.

012674 004054 3   ihfnop: lda [4] 0&cnop          ;nop code
012675 110273 3          sta ihlp i               ;in second leader word
012676 024273 3          irs ihlp
012677 004000 3          lda 0                    ;get local host
012700 110273 3          sta ihlp i               ;in third leader word
012701 024273 3          irs ihlp
012702 004106 3          lda mine                 ;local imp
012703 110273 3          sta ihlp i               ;in fourth leader word
012704 024273 3          irs ihlp
012705 004010 3          lda [177400]             ;link word for host test ?? H
012706 110273 3          sta ihlp i               ;in fifth leader word
012707 003270 3          jmp ihlo5                ;and output 5-word leader

;Send in a RESET

012710 004025 3   ihfrst: lda [10.] 0&creset     ;send in reset
012711 110273 3          sta ihlp i               ;in second leader word
012712 140040 3          cra                      ;and zero
012713 024273 3          irs ihlp
012714 110273 3          sta ihlp i               ;in third, fourth, fifth lead
012715 024273 3          irs ihlp
012716 110273 3          sta ihlp i
012717 103765 3          jmp [ihfdw2] i
```

```
                        .section pg13
                        .lev i2h
013000 004011 3   ihfdwn: lda [hstigd]          ;get IMP-going-down code
013001 150276 3           sta hosti ix          ;change host state
013002 150251 3           sta hirsi ix          ;and tell H2I to reset, too
013003 004053 3           lda two 0&cimpdn      ;send in imp-going-down
013004 110273 3           sta ihlp i            ;in second leader word
013005 140040 3           cra                   ;and zero
013006 024273 3           irs ihlp
013007 110273 3           sta ihlp i            ;in third and fourth leader w
013010 024273 3           irs ihlp
013011 110273 3           sta ihlp i
013012 105721 3           lda [downms] i        ;and imp going down inf
013013 024273 3   ihfdw2: irs ihlp              ;(enter from IHFRST, too)
013014 110273 3           sta ihlp i            ;in fifth leader word (link)

013015 103722 3           jmp [ihlo5] i         ;output leader

;dummy routine when package hooks not in
;used by stats and trace
013016 000000 3   jsti2h: 0
013017 103016 3           jmp jsti2h i


;IHFAKT will skip if IHP is a real host, non-skip if fake. Preserves everythin

013020 000000 3   ihfakt: 0
013021 026272 3           ima ihp               ;get host number, save old AC
013022 016056 3           sub [nh]              ;compare with lowest numbered
013023 100400 3           spl
013024 025020 3            irs ihfakt           ;if IHP .LT. lowest fake, mus
013025 014056 3           add [nh]              ;fix AC
013026 026272 3           ima ihp               ;and restore IHP, getting old
013027 103020 3           jmp ihfakt i          ;return (maybe skip)

;IH30S will TPR the process for 30 seconds, IH640, for 640 milliseconds.

013030 000000 3   ih30s:  0
013031 005723 3           lda [ %30sec ]
013032 000020 3           pcb
013033 000203 3           tpr
013034 103030 3           jmp ih30s i

013035 000000 3   ih640:  0
013036 005724 3           lda [ %640ms ]
013037 000020 3           pcb
013040 054025 3           add pcb.num x         ;skew by host number to jumbl
013041 000203 3           tpr
013042 103035 3           jmp ih640 i

;Call IHWSPR after de-breaking to NMFS via the SPR instruction. We
;re-compute IHP (which is left in X for caller) and IHLP here.
;Use this routine for i/o completion wakeups.  For timeouts, use
;the similar ihwsp2 routine, following ihwspr.

013043 000000 3   ihwspr: 0
013044 005043 3           lda ihwspr            ;save entry point
013045 000020 3           pcb                   ;get ptr to ourselves
013046 000043 3           gpr                   ;poke ourselves to set REPOKE
```

```
;then go to sleep to force it
```

```
013050 000000 3              %crash                  ;IHWSPR: timed out
013051 011043 3             sta ihwspr               ;restore entry point
013052 044025 3             lda pcb.num x            ;get host number
013053 010272 3             sta ihp                  ;set up IHP
013054 072272 3             ldx ihp                  ;leader IHP in X, too
013055 021066 3             jst ihihlp               ;compute IHLP
013056 103043 3             jmp ihwspr i             ;return

;a version of ihwspr which doesn't clear the repoked bit
;use this version if not waking up from i/o completion
013057 000000 3   ihwsp2: 0
013060 000020 3             pcb
013061 044025 3             lda pcb.num x            ;get host number
013062 010272 3             sta ihp                  ;set up IHP
013063 072272 3             ldx ihp                  ;leader IHP in X, too
013064 021066 3             jst ihihlp               ;compute IHLP
013065 103057 3             jmp ihwsp2 i             ;return

013066 000000 3   ihihlp: 0                          ;subr to compute IHLP
013067 004272 3             lda ihp
013070 041476 3             lgl 2.                   ;compute IHP*4
013071 014272 3             add ihp                  ;*5
013072 014272 3             add ihp                  ;*6 = size of a leader
013073 015725 3             add [ihldra]             ;plus base of leader area
013074 010273 3             sta ihlp                 ;set up leader pointer
013075 072272 3             ldx ihp                  ;return with IHP in X
013076 103066 3             jmp ihihlp i             ;return to caller
```

```
;Subroutine to make sure the host's ready line is still up. If not,
;jump directly to IHD, forcing the host down. Uses A.

013077 000000 3   ihrchk: 0                       ;may not return!!
013100 021103 3           jst ihrdyt              ;host ready line up?
013101 103726 3            jmp [ihd] i            ;no, force it down
013102 103077 3           jmp ihrchk i            ;yes, return

;Subroutine to skip if host ready line is up. Uses A, IREG.

013103 000000 3   ihrdyt: 0
013104 072272 3           ldx ihp
013105 004011 3           lda [ %c18.status ]     ;read host ready line
013106 000020 3           pcb
013107 021500 3           jst i2hxdv
013110 072272 3           ldx ihp                 ;fix up the index
013111 007727 3           ana [ %c18csr.down ! %c18csr.flap ]
013112 101040 3           snz                     ;if both DOWN and FLAP are 0
013113 025103 3            irs ihrdyt             ; ... then bump return addres
013114 103103 3           jmp ihrdyt i            ;and return

;Subroutine to look for NCC commands for this I2H to do.  Returns to +2
;if nothing to do or +1 if XDV needed, XDV code is in A, PCB in X.

013115 000000 3   ihncc:  0
013116 004101 3           lda nccc                ;anything there?
013117 101040 3           snz
013120 003134 3            jmp ihnccn             ;no, don't bother decoding it
013121 004100 3           lda ncca                ;for us?
013122 012272 3           era ihp
013123 100040 3           sze
013124 003134 3            jmp ihnccn             ;no
013125 004101 3           lda nccc                ;get command back
013126 012054 3           era [ ncc.lhst ]        ;loop host?
013127 101040 3           snz
013130 003136 3            jmp ihnccl             ;yes
013131 012052 3           era [ ncc.uhst ? ncc.lhst ]     ;unloop host?
013132 101040 3           snz
013133 003137 3            jmp ihnccu             ;yes
013134 025115 3   ihnccn: irs ihncc               ;not of interest, return +2
013135 103115 3           jmp ihncc i

013136 004014 3   ihnccl: lda [ %c18.loop - %c18.unloop ]
013137 014012 3   ihnccu: add [ %c18.unloop ]
013140 000020 3           pcb
013141 103115 3           jmp ihncc i             ;set up for caller's XDV, ret
```

```
;IHLENQ enq's the leader area in an IOCB for output. Enter with AC=IOCB.SIZE
;(i.e. in bits), sign(AC)=1 if this output should have the LASTIOCB flag set.
;Sets IHSP to 1, indicating leader output.

013142 000000 3   ihlenq: 0
013143 010001 3           sta ireg                  ;save size
013144 021164 3           jst ihdeq                 ;get IOCB for use, ptr in X a
013145 004001 3           lda ireg                  ;get IOCB.SIZE
013146 140100 3           ssp                       ; ... ignoring sign
013147 050003 3           sta iocb.size x
013150 021066 3           jst ihihlp                ;compute IHLP into A
013151 072274 3           ldx ihiocb                ;and store that as IOCB.ADDR

013152 050004 3           sta iocb.addr x
013153 004001 3           lda ireg                  ;get LASTIOCB flag
013154 006071 3           ana sign
013155 100040 3           sze
013156 004061 3            lda [%c18.eom]           ;if sign is set, turn on EOM

013157 050005 3           sta iocb.flags x
013160 021173 3           jst ihenq                 ;enq this IOCB
013161 004052 3           lda [1]                   ;set up IHSP to indicate lea
013162 150300 3           sta ihspi ix
013163 103142 3           jmp ihlenq i              ;and return

;Subr to get the IOCB back from NMFS's PUT queue. Will crash the IMP if
;it's not there. Must not use IREG because IHLENQ (a caller) depends on
;IREG being okay across this routine. Returns IOCB ptr in X and IHIOCB.

013164 000000 3   ihdeq:  0
013165 000020 3           pcb
013166 044024 3           lda pcb.put x             ;get ptr to put queue
013167 000022 3           deq                       ;try to get IOCB
013170 000000 3            %crash                  ;IHDEQ: cant DEQ IOCB
013171 032274 3           stx ihiocb                ;set ptr to IOCB
013172 103164 3           jmp ihdeq i               ;and return to caller

;Subr to ENQ the IOCB (in IHIOCB) onto the NMFS GET queue and poke the
;device. Returns host number in X as a convenience.

013173 000000 3   ihenq:  0
013174 000020 3           pcb
013175 044023 3           lda pcb.get x             ;get ptr to GET qeue
013176 072274 3           ldx ihiocb                ;and to IOCB
013177 066005 3           ima iocb.flags x          ;turn on IOFLAGS.ALWAYS
013200 140500 3           ssm 0&%ioflags.always
013201 066005 3           ima iocb.flags x
013202 000002 3           enq                       ;give it to microcode
013203 100000 3            skp
013204 000000 3             %crash                  ;IHENQ: extra IOCB?
013205 000020 3           pcb                       ;now poke the device
013206 021573 3           jst i2hpdv
013207 072272 3           ldx ihp
013210 103173 3           jmp ihenq i               ;and return

;Subroutine to convert Slow ticks (in AC) to NMFS ticks. A slow tick is
;640ms, a NMFS tick is currently 1.6ms. We must multiply AC*400. which
;is equivelant to AC*256 + AC*128 + AC*16.
```

```
013211 000000 3   st2nmf: 0
013212 041474 3           lgl 4.                ;compute * 16
```

```
013213 010001 3          sta ireg
013214 041475 3          lgl 7.-4.          ;compute * 128
013215 010002 3          sta jreg
013216 041477 3          lgl 8.-7.          ;compute * 256
013217 014002 3          add jreg           ; +      * 128
013220 014001 3          add ireg           ; +      * 16
                                            ;        -----
013221 103211 3          jmp st2nmf i       ;return  * 400
```

```
;Subroutine to do NMFS-related I2H initialization. This is a subroutine only
;for clarity (it has only one caller, near I2HINI). We set up the get and
;put queue pointers and headers, then create the IOCB (and put it on the
;PUT queue).

013222 000000 3   i2hnmf: 0
013223 000020 3           pcb                           ;get PCB pointer
013224 005730 3           lda [ %c18csr.down + %c18csr.flap + %c18csr.impr ]
013225 050044 3           sta i2hpcb.swcsr x            ;init software csr, unused if
013226 004000 3           lda 0                         ;get base
013227 015731 3           add [ i2hpcb.getqh ]          ;get ptr to queue header
013230 050023 3           sta pcb.get x
013231 021243 3           jst i2hnmq                    ;initialize the queue header
013232 014054 3           add [ i2hpcb.putqh - i2hpcb.getqh ] ;to PUT queue, no
013233 050024 3           sta pcb.put x
013234 021243 3           jst i2hnmq                    ;initialize the queue header
013235 014054 3           add [ i2hpcb.iocb - i2hpcb.putqh ] ;to IOCB
013236 026000 3           ima 0                         ;put IOCB ptr in X, PCB in A

013237 015732 3           add [ i2hpcb.putqh ]          ;compute address of put queue
013240 000002 3           enq                           ;equivelant to LDA PCB.PUT X,
013241 103222 3            jmp i2hnmf i                 ;return
013242 000000 3             %crash                      ;I2HNMF: found extra IOCB

;Subroutine (called only from I2HNMF) to init a queue header pointed to by A.
;Preserves A, X.

013243 000000 3   i2hnmq: 0
013244 032001 3           stx ireg                      ;save caller's Index
013245 010002 3           sta jreg                      ;  ... and AC
013246 010000 3           sta 0                         ;put address of queue in X
013247 050000 3           sta q.forw x                  ;point it at self
013250 050001 3           sta q.back x
013251 050002 3           sta q.hedr x
013252 140040 3           cra                           ;length = 0, too
013253 050003 3           sta q.size x
013254 072001 3           ldx ireg                      ;fix index
013255 004002 3           lda jreg                      ;and AC
013256 103243 3           jmp i2hnmq i                  ;return
```

```
;put rally entry, blk ptr in x, messno,type*16+state*4+3(mask)
;in a preserves x

013257 000000 3   rallyp: 0
013260 011333 3           sta ralpt1              ;save info
013261 052250 3           era mbtim x
013262 141044 3           car
013263 052250 3           era mbtim x
013264 056250 3           sub mbtim x             ;get mess-rmess
013265 141140 3           icl
013266 006013 3           ana seven               ;could be previous window of
013267 041677 3           alr 1
013270 140407 3           tca
013271 015335 3           add sftcn4
013272 011311 3           sta ralps1              ;shyft into position
013273 033334 3           stx ralptx              ;save blk ptr
013274 005333 3           lda ralpt1              ;look at state
013275 006054 3           ana four                ;bit is on if state=req or re
013276 101040 3           snz
013277 003304 3            jmp ralp0              ;idle or message, don't set r
013300 005333 3           lda ralpt1
013301 006025 3           ana [12]                ;don't set rally flag
013302 100040 3           sze                     ;if state=req, type=req rcvd,
013303 121733 3            jst [b0setf] i         ;do set otherwise
013304 072016 3   ralp0:  ldx minus3              ;get mask, state, type
013305 005333 3   ralp1:  lda ralpt1
013306 040476 3           lgr 2
013307 027333 3           ima ralpt1
013310 006011 3           ana three
013311        3   ralps1: .block 1
013312 051334 3           sta ralpt1+1 x          ;save mask, state, type
013313 024000 3           irs 0
013314 003305 3            jmp ralp1
013315 073334 3           ldx ralptx              ;restore blk ptr
013316 044340 3           lda mbsta x             ;get rstate
013317 007331 3           ana ralpt1-2            ;use mask
013320 013332 3           era ralpt1-1            ;insert new state
013321 052340 3           era mbsta x
013322 050340 3           sta mbsta x
013323 044430 3           lda mbtyp x             ;get rtype
013324 007331 3           ana ralpt1-2            ;insert new type
013325 013333 3           era ralpt1
013326 052430 3           era mbtyp x
013327 050430 3           sta mbtyp x
013330 103257 3           jmp rallyp i

                  .lev var
013331        V           .block 2                ;mask, state
013333        V   ralpt1: .block 1                ;and type
013334        V   ralptx: .block 1                ;temp x

                  .lev con
013335 041700 C   sftcn4: alr 0                   ;shyft instr constant
```

```
;routine to get tsb block off host reply queue
;expects ihp in x, no-skip returns if nothing on queue,
; ihp in x, skip returns with leader words in leader area,
; and ihp in x if something is on the queue.
;if dest dead, host dead is on queue,
; leaves host status on queue
                .lev i2h
013336 000000 3 ihlget: 0
013337 145734 3         lda [shrq] ix           ;anything on queue?
013340 101040 3         snz
013341 103336 3          jmp ihlget i           ;no, return
013342 025336 3         irs ihlget              ;yes, will skip return
013343 010000 3         sta 0                   ;tsb ptr
013344 044044 3         lda lms x               ;is it dest dead, host dead?

013345 012041 3         era [cdestd]
013346 100040 3         sze
013347 003357 3          jmp ihlgt1             ;no
013350 044220 3         lda lid x
013351 006072 3         ana [subtyp]
013352 012052 3         era one 0&chstd
013353 100040 3         sze
013354 003357 3          jmp ihlgt1             ;no
013355 005735 3         lda [chstst]            ;yes, change to host status
013356 003364 3         jmp ihlgt2              ;and form leader

013357 005734 3 ihlgt1: lda [shrq]              ;pull tsb off queue
013360 014272 3         add ihp
013361 010000 3         sta 0
013362 021413 3         jst ihgetq              ;always succeeds
013363 140040 3         cra                     ;free tsb block
013364 066044 3 ihlgt2: ima lms x               ;get current mess type
013365 141140 3         icl 0&ldrmty
013366 110273 3         sta ihlp i              ;into second word of leader
013367 024273 3         irs ihlp
013370 044044 3         lda lms x               ;get msg type
013371 013735 3         era [chstst]            ;is this type 6 msg.
013372 100040 3         sze                     ;
013373 003377 3          jmp ihthwd             ;no ,set up third wrd
013374 044110 3         lda lcd x               ;yes ,get msg code
013375 141050 3         cal 0&ldrhnd            ;make sure bits 33-40 of thir
013376 050110 3         sta lcd x               ;...word of leader are
013377 044110 3 ihthwd: lda lcd x               ;... cleared
013400 110273 3         sta ihlp i              ;into third word of leader
013401 024273 3         irs ihlp
013402 044154 3         lda ldi x               ;and dest imp
013403 110273 3         sta ihlp i              ;into fourth word of leader
013404 024273 3         irs ihlp
013405 140040 3         cra                     ;clear
013406 066264 3         ima lpk x               ;and get host status
013407 066220 3         ima lid x               ;set, and get message-id
013410 110273 3         sta ihlp i              ;into fifth word of leader
013411 072272 3         ldx ihp
013412 103336 3         jmp ihlget i
```

```
013413 000000 3   ihgetq: 0                    ;ih routine to call getq
013414 001001 3           .inh fre
013415 120121 3 0         jst getqi i
013416 025413 3 0          irs ihgetq
013417 000401 3 0         .enb i2h
013420 103413 3           jmp ihgetq i

013421 000000 3   ih2dis: 0
013422 044013 3           lda pkth x
013423 006032 3           ana [177777?pktcod]
013424 012011 3           era three 0&inccod
013425 050013 3           sta pkth x            ;mark packet as incomplete
013426 004152 3           lda times             ;give it a new 30 secondss
013427 016076 3           sub m30sec            ;for discard
013430 050005 3           sta inch x
013431 105736 3           lda [ehq+fhpdis] i
013432 010001 3           sta ireg
013433 132001 3           stx ireg i
013434 133736 3           stx [ehq+fhpdis] i
013435 103421 3           jmp ih2dis i
```

```
;Queue search subroutine. Enter with age in AC.

013436 000000 3   ihqs:   0                       ;queue search  routine
013437 011477 3           sta ihs1f               ;set initial value for flag
013440 005737 3           lda [shq]               ;get regular queue
013441 021447 3           jst ihs1
013442 005740 3           lda [shpq]              ;get priority queue
013443 021447 3           jst ihs1
013444 072272 3           ldx ihp                 ;restore host #
013445 005477 3           lda ihs1f               ;set up time
013446 103436 3           jmp ihqs i

013447 000000 3   ihs1:   0                       ;discard buffers from queue
013450 014272 3           add ihp
013451 011476 3           sta ihstmp
013452 004272 3           lda ihp
013453 013741 3           era [fhpdis]
013454 101040 3           snz                     ;discard?
013455 103447 3            jmp ihs1 i             ;yes, do not discard discard

013456 105476 3   ihs1a:  lda ihstmp i
013457 101040 3           snz                     ;anything on this queue?
013460 103447 3            jmp ihs1 i             ;no, quit
013461 010000 3           sta 0
013462 005477 3           lda ihs1f               ;=0, flush all, else, flush t
013463 101040 3           snz                     ;discard all?
013464 003472 3            jmp ihs1b              ;yes
013465 004152 3           lda times               ;get time in slow ticks
013466 056005 3           sub inch x              ;no, check time
013467 101400 3           smi                     ;too old?
013470 121742 3           jst [ihfakt] i          ;fake host?
013471 103447 3            jmp ihs1 i             ;not too old or fake host
013472 073476 3   ihs1b:  ldx ihstmp
013473 121743 3           jst [ihgetq] i
013474 121744 3           jst [ih2dis] i
013475 003456 3           jmp ihs1a

                          .lev var
013476         V  ihstmp: .block 1
013477         V  ihs1f:  .block 1                ;0=flush all, else=flush time
```

```
; i2hxdv: called to perform XDVs, with A=code and X=PCB.  For real 1822 hosts,
;         the XDV will be executed.  For software hosts, such as HDH, VDH
;         and the fake hosts, this routine will simulate what the ucode
;         would do for a real 1822 XDV.  Note that for software hosts, a
;         software CSR is kept in the PCB.
;
;         For sw hosts, an XDV will result in a jst via the
;         i2hpcb.swxdv address to a routine supplied by the sw host to
;         handle the abort logic.  SW hosts should use routines in the
;         UTL section when possible, to implement the abort.

;         XDV codes:
;
;             0. abort current IOCB
;             1. raise IMP ready line
;             2. lower IMP ready line
;             3. status, read CSR, clear host flapped bit
;             4. loop
;             5. unloop, then abort
;
;         Come here and return with x=our pcb
;         Usage: jreg is used to hold the xdv code
                .lev i2h
013500 000000 3   i2hxdv: 0
013501 010002 3           sta jreg                  ; save XDV code
013502 044005 3           lda pcb.type x            ; check PCB type
013503 100040 3           sze
013504 003556 3            jmp i2hxd7               ; if not soft, do XDV
013505 004002 3           lda jreg
013506 015745 3           add [i2hxdx]
013507 010001 3           sta ireg
013510 102001 3           jmp ireg i                ;dispatch on type

013511 003517 3   i2hxdx: jmp i2hxd0
013512 003521 3           jmp i2hxd1
013513 003526 3           jmp i2hxd2
013514 003534 3           jmp i2hxd3
013515 003543 3           jmp i2hxd4
013516 003551 3           jmp i2hxd5

013517 021561 3   i2hxd0: jst i2hswh                ;just call the sw host
013520 103500 3           jmp i2hxdv i

013521 044044 3   i2hxd1: lda i2hpcb.swcsr x        ;set IMP ready
013522 006016 3           ana [-1 ? %c18csr.impr] ;clear the bit
013523 050044 3           sta i2hpcb.swcsr x
013524 021561 3           jst i2hswh                ;then call the sw host
013525 103500 3           jmp i2hxdv i

013526 044044 3   i2hxd2: lda i2hpcb.swcsr x        ;set IMP not ready
013527 006016 3           ana [-1 ? %c18csr.impr] ;first clear the bit
013530 012053 3           era [%c18csr.impr]      ;then set it
013531 050044 3           sta i2hpcb.swcsr x
013532 021561 3           jst i2hswh                ;then call the sw host
013533 103500 3           jmp i2hxdv i

013534 044044 3   i2hxd3: lda i2hpcb.swcsr x        ;read CSR
013535 007746 3           ana [-1 ? %c18csr.flapped] ;flapped bit cleared
```

```
013536 066044 3          ima i2hpcb.swcsr x      ;upon reading
```

```
013537 011572 3            sta i2hxdt              ;save read CSR value
013540 021561 3            jst i2hswh              ;then call the sw host
013541 005572 3            lda i2hxdt              ;set A=CSR value
013542 103500 3            jmp i2hxdv i            ;and return

013543 044044 3   i2hxd4:  lda i2hpcb.swcsr x      ;set looped bit
013544 006020 3            ana [-1 ? %c18csr.loop]
013545 012054 3            era [%c18csr.loop]
013546 050044 3            sta i2hpcb.swcsr x
013547 021561 3            jst i2hswh              ;then call the sw host
013550 103500 3            jmp i2hxdv i

013551 044044 3   i2hxd5:  lda i2hpcb.swcsr x      ;unloop and abort
013552 007747 3            ana [-1 ? %c18csr.loop ? %c18csr.flap]
013553 012065 3            era [%c18csr.flap]      ;set flap, clear loop bits
013554 050044 3            sta i2hpcb.swcsr x
013555 003517 3            jmp i2hxd0              ;then do the abort

013556 004002 3   i2hxd7:  lda jreg                ;get XDV code
013557 000403 3            xdv                     ;do XDV
013560 103500 3            jmp i2hxdv i            ;and return

;subr to call the sw host, if PCB address supplied
013561 000000 3   i2hswh:  0
013562 044046 3            lda i2hpcb.swxdv x      ;get address of sw-host xdv r
013563 101040 3            snz
013564 103561 3             jmp i2hswh i           ;if none, just exit
013565 010001 3            sta ireg
013566 004002 3            lda jreg                ;A=xdv code
013567 120001 3            jst ireg i              ;call the subroutine
013570 000020 3            pcb
013571 103561 3            jmp i2hswh i

013572 000000 3   i2hxdt:  0                       ;temp
```

```
;i2hpdv: called to perform PDVs.  If the host is a regular 1822 host, the
;       PDV is simply executed.  If the host is a SW host (software host,
;       such as HDH, VDH or fake) then some of the microcode will be emulated,
;       after which the PCB supplied by the SW host in i2hpcb.swpdv will
;       be GPR'd.  If zero, i2hpdv will just return.
;       The microcode emulation consists of moving an iocb from the I2H get
;       queue to the I2H pcb.iocb, IF the get queue is not empty and pcb.iocb

;       is zero.
;       This routine is called, and returns, with X = PCB.

013573 000000 3   i2hpdv: 0
013574 044005 3           lda pcb.type x          ;check PCB type
013575 100040 3           sze                     ;software host?
013576 003617 3            jmp i2hpd0             ;no do real PDV
013577 044022 3           lda pcb.iocb x          ;check if i/o in progress
013600 100040 3           sze
013601 103573 3            jmp i2hpdv i           ;yes, no need to continue
013602 044023 3           lda pcb.get x
013603 000022 3           deq
013604 103573 3            jmp i2hpdv i           ;no iocb to start, just exit

013605 004000 3           lda 0
013606 000020 3           pcb
013607 050022 3           sta pcb.iocb x
;next, poke sw-host
013610 044045 3           lda i2hpcb.swpdv x      ;yes, check if PCB addr suppl
013611 101040 3           snz
013612 103573 3            jmp i2hpdv i           ;no, just return
013613 010000 3           sta 0
013614 000043 3           gpr
013615 000020 3           pcb                     ;restore x=our pcb
013616 103573 3           jmp i2hpdv i

013617 001003 3   i2hpd0: pdv                     ;real host, do real PDV
013620 103573 3           jmp i2hpdv i
```

```
;This is the IMP to Fake Host word-at-a-time interface. It
;merely emulates what the hardware would have done.  Enter with
;fake host number in X, returns word in A.  Skips if EOM.
;Note the usage of suksiz to determine if an iocb is "opened".

                .lev bck

013621 000000 9   suk:    0
013622 033720 9   suktop: stx sukx                ;save caller's fake host numb
013623 145750 9           lda [i2hpct+nh] ix      ;get pointer to corresponding
013624 011717 9           sta sukpcb              ;save it, too
013625 145751 9           lda [suksiz] ix         ;transfer in progress?
013626 100040 9           sze
013627 003645 9            jmp sukgot             ;yes, skip IOCB processing
013630 073717 9           ldx sukpcb              ;no, has an IOCB arrived?
013631 044022 9           lda pcb.iocb x
013632 101040 9           snz
013633 003710 9            jmp sukwat             ;not yet, wait and try again


;Use the IOCB that has just arrived
;Copy it's size and address into local tables for use.

013634 010000 9           sta 0
013635 044003 9           lda iocb.size x         ;get the size
013636 040474 9           lgr 4.                  ;in words, please
013637 010001 9           sta ireg                ;save for a moment
013640 044004 9           lda iocb.addr x         ;get the address
013641 073720 9           ldx sukx                ;get fake host number
013642 151752 9           sta [sukadr] ix         ;set up the address param
013643 004001 9           lda ireg
013644 151751 9           sta [suksiz] ix         ;and the size one, too

;Proceed to get a word from the transfer.  If the count
;has gone to zero, supply padding and finish with the IOCB.

013645 073720 9   sukgot: ldx sukx                ;get fake host number back
013646 145752 9           lda [sukadr] ix         ;get the address
013647 010001 9           sta ireg                ;save it
013650 104001 9           lda ireg i              ;get the word so addressed
013651 010002 9           sta jreg                ;save it
013652 165752 9           irs [sukadr] ix         ;bump addr for next time
013653 145751 9           lda [suksiz] ix         ;get the remaining size
013654 016052 9           sub [1]
013655 151751 9           sta [suksiz] ix         ;update count first
013656 101040 9           snz                     ;more left?
013657 003662 9            jmp sukend             ;no, finish with the IOCB
013660 004002 9           lda jreg                ;fetch saved data
013661 103621 9           jmp suk i               ;return to caller

;Here when the size was found to be zero.  Finish with the
;IOCB and see if it has EOM.

013662 073717 9   sukend: ldx sukpcb              ;get pointer to PCB
013663 044024 9           lda pcb.put x           ;get ptr to PUT queue for lat
013664 010001 9           sta ireg
013665 140040 9           cra
013666 066022 9           ima pcb.iocb x          ;poof
013667 101040 9           snz
```

```
013670 000000 9          %crash                    ;SUCK: IOCB disapeared during
```

```
013671 010000 9          sta 0                  ;get to IOCB
013672 044005 9          lda iocb.flags x       ;is this one EOM?
013673 006061 9          ana [ %c18.eom ]
013674 026001 9          ima ireg               ;get to PUT queue, save EOM b
013675 000002 9          enq                    ;send it to the host code
013676 100000 9           skp
013677 000000 9            %crash               ;SUCK: saw extra IOCB
013700 073717 9          ldx sukpcb             ;get pointer to I2H PCB
013701 000043 9          gpr                    ;poke it
013702 004001 9          lda ireg               ;was this one EOM?
013703 100040 9          sze
013704 025621 9           irs suk               ;yes, skip return
013705 004002 9          lda jreg               ;if yes, get saved last word

013706 073720 9          ldx sukx
013707 103621 9          jmp suk i
```

```
013710 073720 9   sukwat: ldx sukx                ;get fh number
013711 005621 9          lda suk                 ;save caller's address
013712 151753 9          sta [sukst] ix
013713 120113 9          jst wait i              ;go to sleep
013714 145753 9          lda [sukst] ix
013715 011621 9          sta suk                 ;restore it all
013716 003622 9          jmp suktop

013717        9   sukpcb: .block 1
013720        9   sukx:   .block 1

                  .section heap
041005        9   sukst:  .block fh               ;holds SUK (entry point) duri
041011        9   suksiz: .block fh               ;holds IOCB.SIZE during xfer

041015        9   sukadr: .block fh               ;holds IOCB.ADDR during xfer
```

```
                        .section pg0
000272                9 ihp:     .block 1          ;current host number (0 - TH-
000273                9 ihlp:    .block 1          ;pointer to leader area for t
000274                9 ihiocb:  .block 1          ;pointer to the IOCB associat
000275                9 ihpakt:  .block 1          ;pointer to packet associated

000276 041021         9 hosti:   host              ;pointer to host state (HSTUP
000277 041045         9 ihfli:   ihfl              ;ptr to host flags word (see
000300 041071         9 ihspi:   ihsp              ;ptr to packet, 0=none, 1=lea
000301 041115         9 ihwqi:   ihwq              ;ptr to queue (SHQ, SHPQ) whi
000302 041141         9 ihfpi:   ihfp              ;flag saying this is first of
000303 041165         9 ihlsi:   ihls              ;foriegn imp of curr message
000304 041211         9 ihcei:   ihce              ;flag saying this pkt has bad
000305 041235         9 ihiri:   ihir              ;flag set by RESETR when mess

                        .section heap
041021                9 host:    .block th
041045                9 ihfl:    .block th
041071                9 ihsp:    .block th
041115                9 ihwq:    .block th
041141                9 ihfp:    .block th
041165                9 ihls:    .block th
041211                9 ihce:    .block th
041235                9 ihir:    .block th

041261                9 ihldra:  .block 6*th       ;host output leader areas
041451                9 ihldmy:  .block 6          ;a dummy 6-word leader area u
```

```
                .stitl BACKGROUND
                .INCLUDE bck.m4
;background

;Directed reload not supprted (only reload(0), see NSRT)



                .section pg14

                .lev var
014000        V sumtim: .block 1                ;when this IRS's to 0, do HAC
                .lev bck

      000012    sumfrq=10.                      ;number of background loops b

;background loop

014001 140040 9 backst: cra
014002 026103 9         ima gopoke              ;see if someone wants a proce
014003 010000 9         sta 0
014004 100040 9         sze
014005 000043 9          gpr                    ;if so, go poke it

014006 025000 9         irs sumtim              ;time to do a HACSUM computat
014007 003100 9          jmp bcknos             ;nope, skip it
014010 005642 9         lda [-sumfrq]           ;reset sum interval
014011 011000 9         sta sumtim

014012 073643 9         ldx [configtrunk]       ;compute HACSUM as the sum of
014013 005644 9         lda [configtrunk-hacchk-1]
014014 010001 9         sta ireg
014015 140040 9         cra                     ;start at zero
014016 114000 9 bcksum: add 0 i
014017 024000 9         irs 0
014020 024001 9         irs ireg
014021 003016 9          jmp bcksum             ;sum them all up
014022 111645 9         sta [hacsum] i          ;store it in hacsum

;Now, for each host, copy the HAC words into that host's PCB

014023 072032 9         ldx [-nh]
014024 033255 9 bckhcl: stx bt1
014025 145646 9         lda [h2ipct+nh] ix      ;this host configured?
014026 101040 9         snz
014027 003046 9          jmp bckhcn
014030 010000 9         sta 0
014031 032001 9         stx ireg                ;save PCB address
014032 044025 9         lda pcb.num x           ;get host number
014033 010000 9         sta 0
014034 001001 9         .inh all
014035 121647 9 0       jst [hcnf] i            ;get ptr to config block
014036 000401 9 0       .enb bck
014037 044001 9         lda cf.haccom x
014040 010002 9         sta jreg
014041 044002 9         lda cf.hacmem x
014042 072001 9         ldx ireg                ;back to PCB
014043 050044 9         sta h2ipcb.hacmem x
014044 004002 9         lda jreg
```

```
014045 050045 9            sta h2ipcb.haccom x
014046 073255 9   bckhcn:  ldx bt1
014047 024000 9            irs 0
014050 003024 9             jmp bckhc1

;now, copy line and host data to NOC area
014051 073650 9            ldx [-ch]
014052 004052 9   bnoc00:  lda [1]
014053 151651 9            sta [noc.line+ch] ix     ;init line state to down
014054 145652 9            lda [mblock+ch] ix
014055 101040 9            snz                      ;is this modem configured?
014056 003071 9             jmp bnoc01              ;no
014057 010001 9            sta ireg
014060 015653 9            add [neighb]
014061 010002 9            sta jreg
014062 104002 9            lda jreg i
014063 151654 9            sta [noc.neighb+ch] ix
014064 004001 9            lda ireg
014065 015655 9            add [line]
014066 010002 9            sta jreg
014067 104002 9            lda jreg i
014070 151651 9            sta [noc.line+ch] ix
014071 024000 9   bnoc01:  irs 0
014072 003052 9             jmp bnoc00
014073 073656 9            ldx [-th]
014074 145657 9   bnoc02:  lda [host+th] ix
014075 151660 9            sta [noc.host+th] ix
014076 024000 9            irs 0
014077 003074 9             jmp bnoc02

;come here every background loop
014100 004017 9   bcknos:  lda [-fh]
014101 011255 9            sta bt1
014102 005255 9   bkv:     lda bt1
014103 016017 9            sub [-fh]
014104 010000 9            sta 0
014105 011262 9            sta fakeno               ;for debug
014106 045266 9            lda dztb x
014107 010001 9            sta ireg
014110 102001 9            jmp ireg i               ;resume where jam left off

014111 000000 9   bkx:     0                        ;jam wait (doze)
014112 005111 9            lda bkx
014113 051266 9            sta dztb x
                  nojam:                            ;here if there is no JAM side
014114 045272 9   ttjini:  lda wttb x               ;TTJINI is FHO (TTY) JAM ent
014115 010001 9            sta ireg
014116 102001 9            jmp ireg i               ;resume where suck left off

014117 000000 9   bkw:     0                        ;suck wait (wait)
014120 005117 9            lda bkw
014121 051272 9            sta wttb x
014122 000401 9   nosuck:  .enb bck                 ;here if there's no suck
014123 025255 9            irs bt1
014124 003102 9             jmp bkv
014125 004020 9            lda [-bh]
014126 011255 9            sta bt1
014127 072006 9            ldx zero
014130 133661 9   bky:     stx [backno] i
```

```
014131 045276 9          lda sltb x
014132 010001 9          sta ireg
014133 102001 9          jmp ireg i                ;resume where back hosts left

014134 000000 9   bkz:   0                         ;back host wait (sleep)
014135 000401 9          .enb bck
014136 005255 9          lda bt1
014137 101400 9          smi
014140 000000 9           %crash                   ;spurious background return
014141 173661 9          ldx [backno] i
014142 005134 9          lda bkz
014143 051276 9          sta sltb x
014144 024000 9          irs 0
014145 025255 9          irs bt1
014146 003130 9           jmp bky
014147 004101 9          lda nccc                  ;get NCC command
014150 012013 9          era [ ncc.boot ]          ;panic/reboot?
014151 101040 9          snz
014152 000000 9           %crash                   ;panic boot
014153 012072 9          era [ ncc.boot ? ncc.rest ] ;panic restart?
014154 101040 9          snz
014155 102065 9           jmp [init] i             ;yes, re-init
```

```
014156 004151 9   bkup:   lda time
014157 027263 9           ima wdtold              ;is t.o running?
014160 013263 9           era wdtold
014161 100040 9           sze
014162 003166 9            jmp bkc                ;time changed...yes
014163 025264 9           irs wdtbak
014164 003170 9            jmp bku                ;don't give up yet
014165 000000 9           %crash                  ;background saw timeout stop

014166 005662 9   bkc:    lda [-10000.]           ;give t.o a long time to run

014167 011264 9           sta wdtbak              ;and reset back-wdt
014170 000401 9   bku:    .enb bck
014171 025256 9           irs backx               ;keep count of back loops
014172 100000 9            skp
014173 025257 9             irs backx2            ;(double precision count)
014174 101000 9   knop:      nop
014175 121663 9           jst [cxsub] i           ;call background checksummer


; now, measure time since last background loop, in
; order to compute max-to-date longest loop.  be
; sure to zero maxbck before starting measurement period
;
014176 000010 9           %rdclok                 ; read current time
014177 027261 9           ima bckold              ; update old time
014200 017261 9           sub bckold              ; compute old-new
014201 140407 9           tca                     ; negate to be new-old
014202 023260 9           cas maxbck              ; greater than maxbck?
014203 011260 9            sta maxbck             ; yes, store the new champ
014204 101000 9            nop                    ; equal or less, don't save

; now, call each package (or the jstbck dummy routine)
014205 004032 9           lda [-npaks]
014206 011265 9           sta b.temp              ; b.temp is loop counter
014207 005265 9   bkpaks: lda b.temp              ; compute address of next hoo
014210 015664 9           add [bck00.hook+npaks]
014211 010001 9           sta ireg
014212 104001 9           lda ireg i              ; use it to get hook address

014213 010001 9           sta ireg
014214 120001 9           jst ireg i              ; and "jst" via hook to pkg (
014215 025265 9           irs b.temp
014216 003207 9            jmp bkpaks
;
014217 005306 9           lda modloc              ;get modification flag
014220 101040 9           snz                     ;anything to do?
014221 003236 9            jmp bkmx               ;no, restart back loop
014222 001001 9           .inh all                ;yes, interlock
014223 105306 9 0         lda modloc i            ;get contents
014224 007304 9 0         ana modmsk              ;masked
014225 023303 9 0         cas modold              ;right old value?
014226 003236 9 0          jmp bkmx               ;no, wait
014227 100000 9 0          skp                    ;yes
014230 003236 9 0           jmp bkmx              ;no, wait
014231 013305 9 0         era modnew              ;set in new value (unmasked!)
014232 113306 9 0         era modloc i            ;/with old rest of word
014233 111306 9 0         sta modloc i
014234 140040 9 0         cra                     ;clear flag
```

```
014235 011306 9 0          sta modloc
                      .lev bck
014236 001001 9   bkmx:   .inh all                    ;circulate buffers if stealin
```

```
014237 105665 9 0          lda [bufflu] i          ;check buffer stealing flag
014240 101040 9 0          snz                     ;are we waiting to steal?
014241 003247 9 0           jmp bkmx2              ;no, don't bother to circulat
014242 140040 9 0          cra                     ;yes, so circulate one buffer
014243 120116 9 0          jst getfri i            ;get a buffer
014244 003247 9 0           jmp bkmx2
014245 120115 9 0            jst flushi i          ;then give it back
014246 024224 9 0          irs nres                ;  ... to reassembly
014247 000401 9 0 bkmx2:   .enb bck
014250 000020 9            pcb                     ;poke ourselves
014251 000043 9            gpr
014252 000103 9            spr
014253 000000 9             %crash                 ;BCK: NMFS timeout
014254 003001 9            jmp backst              ;and continue
```

```
                        .lev var
014255                V bt1:    .block 1
014256                V backx:  .block 1                ;no of back loops
014257                V backx2: .block 1                ;(double precision count)
014260                V maxbck: .block 1                ;max-to-date back loop time
014261                V bckold: .block 1                ;time of previous maxbck chec
014262                V fakeno: .block 1
014263                V wdtold: .block 1                ;old time reading
014264                V wdtbak: .block 1                ;back wdt timer
014265                V b.temp: .block 1                ; temp used at bkpaks


;these 3 tables must stay in order!!
                        ;defplc(/dztb - goes with jam)
014266                V dztb:   .block fh
                        ;defplc(/wttb - goes with suck)
014272                V wttb:   .block fh
                        ;defplc(/sltb - goes with sleep [back hosts])
014276                V sltb:   .block bh

                        .lev var
;these modification interloc locations must be in order
014303                V modold: .block 1                ;old value
014304                V modmsk: .block 1                ;mask
014305                V modnew: .block 1                ;new value
014306 000000          V modloc: 0                       ;location (flag)

014307                V downms: .block 1                ;saved imp-going-down info
014310                V bt2:    .block 1                ;temp

                        .lev bck
014311 000000          9 jstbck: 0                       ;dummy routine
014312 103311          9         jmp jstbck i
```

```
                    .stitl back host 0: normal replies
;send off normal replies- rfnm, alloc, inc, and dead
                    .lev bck
                    .lck fre
014313 120114 9 0 b0d:     jst sleep i
014314 140040 9 0 b0d1:    cra                          ;use last buf if necessary
014315 001001 9 0          .inh fre
014316 121666 9 0          jst [getfre] i               ;get buf for our msg
014317 003313 9 0           jmp b0d                     ;none available, wait
014320 004066 9 0          lda [dattyp+datcpt+prirty]
014321 050007 9 0          sta typh x                   ;message type
014322 005634 9 0          lda b0pass
014323 050012 9 0          sta seqh x                   ;mess/block no
014324 005632 9 0          lda b0pkth
014325 050013 9 0          sta pkth x                   ;use no/message code
014326 140040 9 0          cra
014327 050015 9 0          sta midh x                   ;subtype=0
014330 005635 9 0          lda b0stat
014331 050016 9 0          sta data x                   ;host status
014332 105627 9 0          lda back0p i 0&mbimp
014333 050014 9 0          sta dsth x                   ;destination imp, givtsk will
014334 121667 9 0          jst [givtsk] i
                    .ret bck
;give msg to task
014335 024224 9            irs nres                     ;free up reas for mess buf
014336 100000 9            skp
014337 120114 9   b0f:      jst sleep i                 ;wait one background loop
014340 005641 9            lda b0try                    ;anything to do?
014341 101400 9   b0f1:    sma
014342 003337 9             jmp b0f                     ;no, go back to sleep
014343 005670 9            lda [-nmb]                   ;set up full pass counter
014344 011634 9            sta b0pass                   ;to try all message blocks
014345 025641 9            irs b0try                    ;only one?
014346 003351 9             jmp b0g                     ;no, pick up where we left of
014347 005640 9            lda b0flag                   ;yes, pick up at specific one
014350 003356 9            jmp b0g1

014351 005627 9   b0g:     lda back0p                   ;go on to next block
014352 141206 9            aoa
014353 123671 9            cas [b2tble] i               ;wrapped?
                    bk0ini:                             ;here initially
014354 105671 9   back0:    lda [b2tble] i              ;impossible except at init
014355 015670 9             add [-nmb]                  ;wrap
014356 011627 9   b0g1:    sta back0p
014357 010000 9            sta 0
014360 044340 9            lda mbsta x                  ;get msg states
014361 007672 9            ana [52525]                  ;any in reply or request stat
014362 101040 9            snz
014363 003413 9             jmp b0j                     ;no, ignore
014364 004022 9            lda min10                    ;set up position counter
014365 011631 9            sta b0posn
```

```
014366 001001 9            .inh (i2h,tsk)
014367 044250 9 3          lda mbtim x          ;get snapshot of rmess, rstat
014370 011630 9 3          sta b0mess
014371 044430 9 3          lda mbtyp x
014372 011633 9 3          sta b0type
014373 044340 9 3          lda mbsta x
014374 000401 9 3          .enb bck
014375 011635 9   b0h:     sta b0stat
014376 006011 9            ana three            ;get next state
014377 022053 9            cas two              ;what is it?
014400 003425 9             jmp b0l             ;3=reply
014401 003404 9             jmp b0i             ;2=message, loop
014402 100040 9            sze                  ;idle (=0)?
014403 003416 9             jmp b0k             ;no, 1=request
014404 005633 9   b0i:     lda b0type           ;shyft state and type over
014405 040676 9            arr 2
014406 011633 9            sta b0type
014407 005635 9            lda b0stat
014410 040676 9            arr 2
014411 025631 9            irs b0posn           ;and count position
014412 003375 9             jmp b0h             ;loop
014413 025634 9   b0j:     irs b0pass           ;any more blocks to check?
014414 003351 9             jmp b0g             ;yes, loop
014415 003337 9            jmp b0f              ;no, quit and sleep

014416 005633 9   b0k:     lda b0type           ;request, get type
014417 006011 9            ana three
014420 016053 9            sub two
014421 100400 9            spl                  ;all1 or all8? (=2 or =3)
014422 003404 9             jmp b0i             ;no, =0 or =1 (req rcvd), do
014423 011633 9            sta b0type           ;yes,
014424 003433 9            jmp b0m              ;use type codes 0 (all1) or 1

014425 005633 9   b0l:     lda b0type           ;reply, get type
014426 006011 9            ana three            ;use type codes 2-5,
014427 014053 9            add two              ;2=rfnm1,3=rfnm8,4=dead,5=inc
014430 011633 9            sta b0type           ;save type code
014431 021547 9            jst b0a              ;change state
014432 005633 9            lda b0type           ;get type code
014433 012052 9   b0m:     era one              ;flip low order bit
014434 022053 9            cas two              ;makes all1=1, all8=0, rfnm8=
014435 003314 9             jmp b0d1            ;rfnm1=3, dead=5, inc=4, go s
014436 140040 9             cra                 ;make rfnm8=0
014437 101040 9            snz                  ;one pkt guy is 1
014440 004055 9             lda ten 0&mltpkt    ;8 pkt guys 0, make it 8
014441 011637 9            sta b0allc           ;save alloc needed (1 or 8)
```

```
014442 004151 9           lda time
014443 011636 9           sta b0time              ;temp time
014444 005637 9   b0o:    lda b0allc
014445 101100 9           sln                     ;8 pkt guy?
014446 141206 9            aoa                    ;yes, make it 9 for test
014447 001001 9           .inh fre
014450 121673 9 0         jst [maxchk] i          ;enough room?
014451 003503 9 0          jmp b0n                ;no
014452 073674 9 0         ldx [-nreab+1]          ;yes, ignore first reassembly
014453 145675 9 0 b0t:    lda [reastb+nreab+rch] ix ;find free reas block
014454 101040 9 0         snz
014455 003461 9 0          jmp b0u                ;found a free one
014456 024000 9 0         irs 0                   ;loop
014457 003453 9 0          jmp b0t
014460 003503 9 0         jmp b0n                 ;none found, wait some more

014461 005675 9 0 b0u:    lda [reastb+nreab+rch]
014462 014000 9 0         add 0                   ;form ptr to block
014463 010000 9 0         sta 0
014464 005676 9 0         lda [sign-pkth]         ;pkth x at oldmes=<sign>
014465 050000 9 0         sta rch x               ;make it a no-name
014466 005637 9 0         lda b0allc              ;get #alloc-1 in lh
014467 141240 9 0         icr
014470 015627 9 0         add back0p              ;save block number
014471 017677 9 0         sub [rmblkt-200+400]    ;with special flag in rh
014472 050040 9 0         sta rms x
014473 005637 9 0         lda b0allc              ;and take alloc
014474 014221 9 0         add nala
014475 010221 9 0         sta nala
014476 005633 9 0         lda b0type
014477 006052 9 0         ana one                 ;3(rfnm/piggyback)->1(all8)
014500 011633 9 0 b0s:    sta b0type
014501 021547 9 0         jst b0a
                        .ret bck
;change state
014502 003314 9           jmp b0d1                ;rfnm, go send message
```

```
                        .lev bck
                        .lck fre
014503 120114 9 0 b0n:    jst sleep i
                        .ret bck
014504 004151 9           lda time              ;have we been waiting 1/2 sec
014505 017636 9           sub b0time
014506 017700 9           sub [20.]
014507 073627 9           ldx back0p
014510 001001 9           .inh (i2h,h2i,tsk)
014511 100400 9 3         spl
014512 003526 9 3          jmp b0r              ;no
014513 004052 9 3         lda one               ;yes, all1, all8, or rfnm8?
014514 023633 9 3         cas b0type
014515 003542 9 3          jmp b0e              ;all1, give up and reset flag
014516 100000 9 3          skp                  ;all8
014517 003314 9 3           jmp b0d1            ;rfnm8, give up on piggyback
014520 005635 9 3         lda b0stat            ;get all other states
014521 006017 9 3         ana [-4]              ;/ except current
014522 101040 9 3         snz                   ;all others idle?
014523 003542 9 3          jmp b0e              ;yes, try something else for
014524 004011 9 3         lda three             ;no, treat as rfnm8
014525 003500 9 3         jmp b0s               ;without allocate

014526 004011 9 3 b0r:    lda three             ;rfnm or allocate?
014527 023633 9 3         cas b0type
014530 003444 9 3          jmp b0o              ;allocate, wait some more
014531 044340 9 3          lda mbsta x          ;rfnm, are there other rfnms

014532 052430 9 3         era mbtyp x           ;or all8s to be sent?
014533 040677 9 3         arr 1
014534 046340 9 3         ana mbsta x
014535 046430 9 3         ana mbtyp x
014536 007672 9 3         ana [52525]
014537 101040 9 3         snz
014540 003444 9 3          jmp b0o              ;no, wait some more
014541 003314 9 3         jmp b0d1              ;yes, give up on piggyback al

014542 005641 9 3 b0e:    lda b0try             ;give one back to b0try
014543 016052 9 3         sub one
014544 011641 9 3         sta b0try
014545 000401 9 3         .enb bck
014546 003341 9           jmp b0f1              ;and look again
```

```
                    .lev bck
;enter with b0type in a
014547 000000 9   b0a:    0                    ;change state and prepare mes
014550 001001 9           .inh (i2h,tsk)
014551 100040 9 3         sze                  ;all1?
014552 004022 9 3          lda min10           ;no, mess no must be rmess-po
014553 014055 9 3         add ten              ;yes, mess no must be rmess-1
014554 015631 9 3         add b0posn
014555 141240 9 3         icr
014556 015630 9 3         add b0mess
014557 141044 9 3         car
014560 011634 9 3         sta b0pass           ;save mess no
014561 073633 9 3         ldx b0type           ;get type index
014562 045621 9 3         lda b0tabl x         ;get code bits
014563 141050 9 3         cal 0&pktcod
014564 011632 9 3         sta b0pkth           ;save pkt code
014565 045621 9 3         lda b0tabl x         ;get rallyp setting
014566 141140 9 3         icl
014567 013634 9 3         era b0pass           ;incl mess no
014570 073627 9 3         ldx back0p
014571 121701 9 3         jst [rallyp] i       ;put into tables
014572 044160 9 3         lda mbfor x          ;get foreign info
014573 141050 9 3         cal 0&mbfrnb         ;foreign block no
014574 013634 9 3         era b0pass           ;into seqh word
014575 011634 9 3         sta b0pass
014576 044160 9 3         lda mbfor x
014577 006074 9 3         ana [mbfrnu]         ;foreign use no
014600 013632 9 3         era b0pkth           ;into pkth word
014601 011632 9 3         sta b0pkth
014602 121702 9 3         jst [hostno] i
014603 010000 9 3         sta 0                ;put host # in x
014604 121703 9 3         jst [hstatr] i       ;get host status
014605 101000 9 3          nop                 ;don't care if dead or alive

014606 011635 9 3         sta b0stat           ;save status
014607 000401 9 3         .enb bck
014610 103547 9           jmp b0a i

                    .lev (tsk,i2h)
;set rally flag to indicate new entry
;block pointer in x, preserved
014611 000000 3   b0setf: 0
014612 033640 3           stx b0flag           ;save specific
014613 005641 3           lda b0try            ;get -# of entries
014614 101400 3           smi                  ;must be neg
014615 140040 3            cra                 ;or zero
014616 016052 3           sub one              ;subtract one more
014617 011641 3           sta b0try
014620 103611 3           jmp b0setf i
```

```
                .lev con
;state table format:
;left half- next type, next state, mask (=3)
;next types
        000000      b0tr=0*10000                    ;rfnm sent or req1 rcvd or ms
                                    ; rfnm1 to be sent
        010000      b0ta=1*10000                    ;rfnm/all sent or req8 rcvd o
                                    ; rcvd or rfnm8 to be sent
        020000      b0td=2*10000                    ;dead sent or all1 to be sent
                                    ; dead rcvd or dead to be sent
        030000      b0ti=3*10000                    ;inc sent or all8 to be sent
                                    ; inc rcvd or inc to be sent
;next states
        000000      b0si=0*2000                     ;idle
        002000      b0sa=1*2000                     ;request
        004000      b0sm=2*2000                     ;message
        006000      b0sr=3*2000                     ;reply
;mask
        001400      b03m=3<<8.                      ;type, state mask
;right half- packet code for message
014621 005405 C   b0tabl: b0tr+b0sm+b03m+allcod   ;0- all1
014622 011405 C           b0ta+b0si+b03m+allcod   ;1- all8
014623 001404 C           b0tr+b0si+b03m+rfncod   ;2- rfnm1
014624 001404 C           b0tr+b0si+b03m+rfncod   ;3- rfnm8
014625 021406 C           b0td+b0si+b03m+dedcod   ;4- dead
014626 031407 C           b0ti+b0si+b03m+ircod    ;5- inc

                .lev var
014627          V   back0p: .block 1              ;next block pntr
014630          V   b0mess: .block 1              ;saved rmess
014631          V   b0posn: .block 1              ;position counter
014632          V   b0pkth: .block 1              ;temp pkth word
014633          V   b0type: .block 1              ;saved rtype, code type
014634          V   b0pass: .block 1              ;full pass counter, temp seqb
014635          V   b0stat: .block 1              ;saved rstate, temp host stat
014636          V   b0time: .block 1              ;allocate wait timer
014637          V   b0allc: .block 1              ;allocate size
014640          V   b0flag: .block 1              ;specific block ptr
014641          V   b0try:  .block 1              ;0=> nothing to do
                                    ;-1=> do specific block
                                    ;<-1=> do round-robin
```

```
                .section pg15
                .stitl back host 1, 2: message block deletion

;transmit block deletion back host 1
                .lev bck
;common code part 1...sleep, then find block to flush
015000 120114 9   b12a1:  jst sleep i
015001 045110 9   b12b:   lda b1fcnt-1 x          ;get number of free blocks
015002 016054 9           sub four
015003 101400 9           smi                     ;below cutoff? (<2)
015004 003023 9            jmp b12d               ;no
015005 016053 9           sub two                 ;yes-- another -2 for slack
015006 051106 9           sta b1cutc-1 x          ;into cutoff counter
015007 005623 9           lda [-nmb]              ;cycle through all blocks
015010 051112 9           sta b1cycc-1 x
015011 003023 9           jmp b12d                ;for all ages

015012 065112 9   b12c:   irs b1cycc-1 x          ;count next cycle
015013 003025 9            jmp b12e               ;continue looking at current
015014 005623 9           lda [-nmb]
015015 051112 9           sta b1cycc-1 x          ;reset cycle count
015016 065114 9           irs b1cagc-1 x          ;lower age
015017 003025 9            jmp b12e               ;still old enough
015020 140040 9           cra                     ;too young, quit (ages 15-5 c
015021 051106 9           sta b1cutc-1 x          ;kill cutoff counter
015022 003000 9           jmp b12a1

015023 005624 9   b12d:   lda [-11.]             ;oldest possible age
015024 051114 9           sta b1cagc-1 x
015025 045104 9   b12e:   lda back1p-1 x          ;move block pointer
015026 141206 9           aoa
015027 063102 9           cas b1tble-1 x          ;check for wrap
                  bk1ini:                         ;here initially
                  bk2ini:
015030 045102 9   b12st:   lda b1tble-1 x         ;impossible (except at startu
015031 015623 9            add [-nmb]             ;wrap
015032 051104 9           sta back1p-1 x
015033 001001 9           .inh (h2i,tsk,t.o)
;meddling with a block
015034 045104 9 4         lda back1p-1 x
015035 010001 9 4         sta ireg
015036 104001 9 4         lda ireg i 0&mbimp
015037 101040 9 4         snz                     ;is block free?
015040 003051 9 4          jmp b12a               ;yes
015041 045114 9 4         lda b1cagc-1 x          ;no, get current age
015042 072001 9 4         ldx ireg                ;get block ptr
015043 054250 9 4         add mbtim x             ;is it the right age?
015044 006072 9 4         ana [mbage]
015045 012054 9 4         era four
015046 073315 9 4         ldx backno              ;restore t/r flag
015047 101040 9 4         snz
015050 043100 9 4          jmp b1ptr2-1 x         ;yes, dispatch
                  b12a:
015051 000401 9 4         .enb bck
;no, no longer meddling
015052 045106 9           lda b1cutc-1 x          ;are we in cutoff mode?
015053 100400 9           spl
015054 003012 9            jmp b12c               ;yes, run again
015055 003000 9           jmp b12a1               ;no, go to sleep
```

```
;prepare block for reset
                    .lev bck
                    .lck (h2i,tsk,t.o)
015056 073105 9 4 b1e:    ldx back1p
015057 044340 9 4         lda mbmes x              ;look for init or rst
01506C 006037 9 4         ana [mbrst+mbinit]
015061 101040 9 4         snz                      ;is either on?
015062 003065 9 4          jmp b1f                 ;no
015063 073315 9 4         ldx backno               ;yes, ignore
015064 003051 9 4         jmp b12a                 ;and continue
015065 044340 9 4 b1f:    lda mbmes x              ;save allocate count
015066 006042 9 4         ana [mbstp+mballc]
015067 013625 9 4         era [mbrst+mesbts] C&mbinit&mbinct
01507C 050340 9 4         sta mbmes x              ;set rst and inc t.o bits
015071 025111 9 4         irs b1fcnt               ;simulate one more free
015072 120114 9 4         jst sleep i
                    .ret bck
;wait a loop
;common code part 2 -- what to do next?
015073 045106 9   b12i:   lda b1cutc-1 x
015074 101400 9           smi                      ;cutoff mode?
015075 003001 9            jmp b12b                ;no
015076 065106 9           irs b1cutc-1 x           ;yes, increment counter
015077 003012 9            jmp b12c                ;and continue searching
01510C 003001 9           jmp b12b                 ;cutoff satisfied, continue

                    .lev con
015101 003056 C   b1ptr2: jmp b1e                  ;'action' dispatches
015102 003117 C           jmp b2e
;in order!
015103 044105 C   b1tble: tmblkt+nmb               ;block table end + 1
015104 043365 C   b2tble: rmblkt+nmb

                    .lev var
;the following ten variables belong in ordered pairs!
015105         V  back1p: .block 1                 ;current block pointer
015106         V  back2p: .block 1
015107         V  b1cutc: .block 1                 ;cutoff counter
01511C         V  b2cutc: .block 1
                          ;defplc(/free transmit, receive block counters)
015111         V  b1fcnt: .block 1                 ;free block counter
015112         V  b2fcnt: .block 1
015113         V  b1cycc: .block 2                 ;cycle counters
015115         V  b1cagc: .block 2                 ;age counters
```

```
;receive block deletion back host 2
;initial entry into back host 2
;send reset requests
                    .lev bck
                    .lck (i2h,h2i,tsk,t.o)
015117 073106 9 3 b2e:    ldx back2p                ;get blk ptr
015120 044160 9 3         lda mbfor x               ;get frn blk/use number
015121 011154 9 3         sta b2tmp1
015122 121626 9 3         jst [getlus] i            ;get loc   "       "
015123 011155 9 3         sta b2tmp2
015124 121627 9 3         jst [blkage] i            ;reset age to 4
015125 100000 9 3         skp
015126 120114 9 3 b2e1:    jst sleep i
                    .ret bck
015127 140040 9           cra
015130 001001 9           .inh fre
015131 121630 9 0         jst [getfre] i            ;get packet buffer
015132 003126 9 0          jmp b2e1                 ;none available, so sleep
015133 004040 9 0         lda [nettyp+netcpt+prirty]
015134 050007 9 0         sta typh x                ;pkt type
015135 005154 9 0         lda b2tmp1
015136 141050 9 0         cal 0&mbfrnb              ;foreign block number
015137 050012 9 0         sta seqh x
015140 005154 9 0         lda b2tmp1
015141 006074 9 0         ana [mbfrnu]              ;foreign use number
015142 013631 9 0         era [rsqcod]              ;reset request code
015143 050013 9 0         sta pkth x
015144 005155 9 0         lda b2tmp2
015145 050015 9 0         sta midh x                ;local block info
015146 105106 9 0         lda back2p i 0&mbimp      ;foreign imp
015147 050014 9 0         sta dsth x
015150 021215 9 0         jst givtsk
                    .ret bck
;give to task
015151 024224 9           irs nres
015152 025112 9           irs b2fcnt                ;simulate one more free
015153 003073 9           jmp b12i                  ;rejoin common
                    .lev var
015154         V    b2tmp1: .block 1
015155         V    b2tmp2: .block 1                  ;temps
```

```
                    .stitl back host 4: reroute, task reply
;send off packets from a line which just died - reroute
;send off task replies
;also resend off anything refused by task in givtsk
                    .lev bck
015156 120114 9     b4a:    jst sleep i
015157 005175 9             lda back4p              ;get next queue head
015160 141206 9             aoa
015161 023632 9             cas [srrq+2]
                    bk4ini:                         ;here initially
015162 005632 9     back4:  lda [srrq+2]            ;impossible except at init
015163 016053 9             sub two
015164 011175 9             sta back4p
015165 010000 9             sta 0
015166 001001 9             .inh fre
015167 121633 9 0           jst [getq] i            ;get next reroute or task re
015170 003156 9 0            jmp b4a                ;none, sleep
015171 021215 9 0           jst givtsk
                    .ret bck
;give to task (a=0=no chksm)
015172 073175 9             ldx back4p              ;irs nsfs for reroute
015173 165634 9             irs [nsfs-srrq] ix      ;and nres for task reply
015174 003156 9             jmp b4a                 ;sleep

                    .lev var
015175         V    back4p: .block 1                ;reroute or task reply queue


;
; back host 5 located on next core page
;

;get local block/use number/handling type
;block ptr in x, returns value in a, preserves x
                    .lev bck
015176 000000 9     getlus: 0
015177 004000 9             lda 0                   ;local block
015200 017635 9             sub [tmblkt]            ;try transmit block table
015201 022036 9             cas [nmb]               ;compare with normal range
015202 017636 9              sub [rmblkt-tmblkt]    ;>range, must be recieve
015203 101000 9              nop                    ;=max, transmit
015204 100400 9             spl                     ;is it <0?
015205 017636 9              sub [rmblkt-tmblkt]    ;yes, must be recieve
015206 011214 9             sta getlut
015207 044250 9             lda mbtim x             ;now local use no
015210 006073 9             ana [mbusno]
015211 041474 9             lgl 4 0&usenum
015212 013214 9             era getlut              ;now local block no
015213 103176 9             jmp getlus i

                    .lev var
015214         V    getlut: .block 1
```

```
                    .stitl back hosts to task

                    .lev bck
015215 000000 9     givtsk: 0
015216 101040 9             snz
015217 003225 9              jmp givt0              ;not generating a checksum
015220 020002 9             jst chksum             ;compute checksum
015221 000000 9              %crash                ;buffer word count smashed - b
015222 140407 9             tca
015223 054010 9             add chkh x
015224 050010 9             sta chkh x             ;true checksum
015225 005315 9     givt0:  lda backno 0&inpchn
015226 015637 9             add [hstmod+th]
015227 050005 9             sta inch x             ;set up input channel
015230 000010 9             %rdclok                ;set arrival time for delay
015231 050003 9             sta artm x
015232 004000 9             lda 0
015233 073315 9             ldx backno
015234 051323 9             sta hold x             ;save pkt pntr
015235 005215 9             lda givtsk
015236 051316 9             sta givtst x           ;save return addr
015237 045323 9             lda hold x
015240 001001 9             .inh m2i
015241 032001 9 1           stx ireg               ;save index
015242 173640 9 1           ldx [etq] i
015243 032002 9 1           stx jreg
015244 110002 9 1           sta jreg i 0&ptrc
015245 111640 9 1           sta [etq] i            ;put on task queue
015246 072001 9 1           ldx ireg               ;restore index
015247 140040 9 1           cra
015250 151641 9 1           sta [tskflg+th] ix
015251 072136 9 1           ldx tskpci             ;get pointer to task
015252 000043 9 1           gpr                    ;poke it
015253 072001 9 1           ldx ireg               ;restore index
015254 120114 9 1           jst sleep i
                      .ret bck
015255 140040 9             cra
015256 167641 9             ima [tskflg+th] ix
015257 101040 9             snz
015260 000000 9              %crash                ;GIVTSK: TASK didnt take packe
015261 101100 9             sln                    ;task refused it
015262 003277 9              jmp givt4             ;task took it
015263 004000 9             lda 0
015264 012054 9             era four               ;back 4 is special case
015265 001001 9             .inh (i2m,t.o,tsk)
015266 101040 9 2           snz                    ;is it back4?
015267 003302 9 2            jmp givtb4            ;/ yes, check for re-route
015270 105642 9 2 givt2:    lda [erpq] i
015271 010001 9 2           sta ireg
015272 045323 9 2           lda hold x             ;everything else goes onto re
015273 110001 9 2           sta ireg i             ;  and is counted in reassemb
015274 111642 9 2           sta [erpq] i
015275 024220 9 2           irs nrea
015276 000401 9 2 givt3:    .enb bck
015277 045316 9     givt4:  lda givtst x           ;and done. back4 will now han
015300 010001 9             sta ireg
015301 102001 9             jmp ireg i

                    .lck (i2m,t.o,tsk)
```

```
015302 005175 9 2 givtb4: lda back4p          ;from back 4.
015303 013643 9 2         era [srrq]
015304 100040 9 2         sze                 ;was back4 doing reroute?
015305 003270 9 2          jmp givt2          ;no, reply-- handle as above

015306 105644 9 2         lda [errq] i
015307 010001 9 2         sta ireg
015310 045323 9 2         lda hold x          ;yes, different queue and dif
015311 110001 9 2         sta ireg i
015312 111644 9 2         sta [errq] i
015313 024217 9 2         irs nsfa
015314 003276 9 2         jmp givt3

                    .lev var
015315         V    backno: .block 1           ;no of back host currently ac
015316         V    givtst: .block bh          ;rtn addr for back hosts givi
015323         V    hold:   .block bh          ;pkt ptrs for back hosts givi
```

```
                .stitl back host 5
;send off incomplete transmissions if reply overdue, give backs
;when allocates time out, and resets and get-a-blocks when
;retransmit times out
                .lev bck
015330 120114 9   b5a:    jst sleep i             ;sleep for one background loo
015331 140040 9   b5a1:   cra                     ;block needing immediate atte
015332 027622 9           ima back5i              ;and clear indicator
015333 100040 9           sze
015334 003343 9            jmp b5b                ;yes, do it
015335 005621 9           lda back5p              ;no, increment pointer
015336 141206 9           aoa
015337 123645 9           cas [b1tble] i          ;check for wrap
                bk5ini:                           ;here initially
015340 105645 9   back5:   lda [b1tble] i         ;initial start at impossible
015341 015623 9            add [-nmb]             ;wrap
015342 011621 9           sta back5p
015343 010000 9   b5b:    sta 0                   ;blk ptr in x
015344 001001 9           .inh (h2i,t.o,tsk)
;meddling with block
015345 044000 9 4         lda mbimp x             ;block free?
015346 101040 9 4         snz
015347 003330 9 4          jmp b5a                ;yes, loop and sleep
015350 011615 9 4         sta b5dsth              ;save imp # for later
015351 044160 9 4         lda mbfor x
015352 141050 9 4         cal 0&mbfrnb
015353 011613 9 4         sta b5seqh              ;save foreign block no
015354 044160 9 4         lda mbfor x
015355 006074 9 4         ana [mbfrnu]
015356 011614 9 4         sta b5pkth 0&usenum     ;and use number
015357 044340 9 4         lda mbmes x             ;inc t/o max?
015360 006034 9 4         ana [mbinct]
015361 100040 9 4         sze
015362 003477 9 4          jmp b5givb             ;no, check allocate count
015363 121626 9 4         jst [getlus] i          ;yes, get local block and use
015364 011616 9 4         sta b5midh
015365 004040 9 4         lda [nettyp+netcpt+prirty]
015366 011610 9 4         sta b5typh              ;set packet type
015367 044340 9 4         lda mbmes x             ;reset inc t/o
015370 012034 9 4         era [mbinct]
015371 050340 9 4         sta mbmes x
015372 006035 9 4         ana [mbrst+mbinit+mbstp] ;reset, init, or stopped?
015373 101040 9 4         snz
015374 003546 9 4          jmp b5incq             ;no, send inc query
015375 101400 9 4         smi 0&mbrst&mbinit      ;reset on?
015376 003440 9 4          jmp b5gets             ;no, must be get-a-block or s
015377 140100 9 4         ssp 0&mbrst
015400 100040 9 4         sze 0&mbinit            ;both on?
015401 003330 9 4          jmp b5a                ;yes, ignore, loop and sleep

;reset comes here
015402 121646 9 4 b5rest: jst [getall] i          ;suck up all allocates
015403 100000 9 4          skp                    ;no more
015404 003402 9 4           jmp .-2
015405 004025 9 4         lda [rstcod]
```

```
;end code common to all
015406 013614 9 4 b5comn: era b5pkth          ;set pkt code
015407 011614 9 4         sta b5pkth
015410 100000 9 4         skp
015411 120114 9 4 b5com1:  jst sleep i
                  .ret bck
;wait for free buf
015412 140040 9   b5com2: cra
015413 001001 9           .inh fre
015414 121630 9 0         jst [getfre] i      ;get buf
015415 003411 9 0          jmp b5com1         ;none available now
015416 005610 9 0         lda b5typh          ;copy header into buffer
015417 050007 9 0         sta typh x
015420 005613 9 0         lda b5seqh
015421 050012 9 0         sta seqh x
015422 005614 9 0         lda b5pkth
015423 050013 9 0         sta pkth x
015424 005615 9 0         lda b5dsth
015425 050014 9 0         sta dsth x
015426 005616 9 0         lda b5midh
015427 050015 9 0         sta midh x
015430 005617 9 0         lda b5hmem
015431 050016 9 0         sta data x
015432 005620 9 0         lda b5hcom
015433 050017 9 0         sta data+1 x
015434 004052 9 0         lda one             ;ac nonzero so givtsk will ch
015435 121647 9 0         jst [givtsk] i
                  .ret bck
;give to task
015436 024224 9           irs nres            ;and give buf back to reassem
015437 003331 9           jmp b5a1            ;loop, task already slept

                  .lev bck
                  .lck (h2i,t.o,tsk)
015440 022070 9 4 b5gets: cas [mbinit]        ;stopping?
015441 003444 9 4          jmp b5get1         ;yes
015442 003455 9 4          jmp b5getb         ;no, init, send getblk
015443 003546 9 4         jmp b5incq          ;no, 8 allocates, send inc?
```

```
015444 044340 9 4 b5get1: lda mbmes x             ;any messages outstanding?
015445 006007 9 4         ana [mesbts]
015446 012007 9 4         era [mesbts]
015447 100040 9 4         sze
015450 003546 9 4          jmp b5incq             ;yes, send inc query
015451 044340 9 4         lda mbmes x             ;no, change to reset
015452 014067 9 4         add [mbstp] 0&mbinit&mbrst
015453 050340 9 4         sta mbmes x
015454 003402 9 4         jmp b5rest              ;and send reset

;get-a-block comes here
015455 033611 9 4 b5getb: stx b5blkp              ;save block pointer
015456 044160 9 4         lda mbfor x             ;get handling type
015457 006075 9 4         ana [mbhand]
015460 013616 9 4         era b5midh              ;add to local block info
015461 011616 9 4         sta b5midh
015462 044070 9 4         lda mbhst x             ;get host pair
015463 011613 9 4         sta b5seqh              ;goes in seqh
015464 121650 9 4         jst [hostno] i          ;get transmit host
015465 010000 9 4         sta 0
015466 145651 9 4         lda [h2ipct] ix         ;get its PCB
015467 010000 9 4         sta 0
015470 044044 9 4         lda h2ipcb.hacmem x     ;get hac words
015471 011617 9 4         sta b5hmem
015472 044045 9 4         lda h2ipcb.haccom x
015473 011620 9 4         sta b5hcom
015474 073611 9 4         ldx b5blkp              ;restore block ptr
015475 004024 9 4         lda [gtbcod]
015476 003406 9 4         jmp b5comn              ;go to common code
```

```
;give backs come here
015477 044340 9 4 b5givb: lda mbmes x              ;any allocates outstanding?
015500 006070 9 4         ana [mbinit]             ;is init on?
015501 100040 9 4         sze
015502 004067 9 4          lda [mbstp]             ;yes, don't  stp bit
015503 012042 9 4         era [mbstp+mballc]       ;in allocate mask
015504 046340 9 4         ana mbmes x
015505 101040 9 4         snz                      ;any allocates left?
015506 003330 9 4          jmp b5a                 ;no, go back to sleep
015507 033611 9 4         stx b5blkp               ;save block pointer
015510 016064 9 4         sub [mball1]             ;more than 1 allocate?
015511 100040 9 4         sze
015512 003521 9 4          jmp b5giv1              ;yes, give back right away
015513 121650 9 4         jst [hostno] i           ;get transmit host
015514 010000 9 4         sta 0
015515 145652 9 4         lda [hial] ix            ;get its current allocate ti
015516 100400 9 4         spl                      ;time to give back?
015517 003330 9 4          jmp b5a                 ;no, loop and sleep
015520 073611 9 4         ldx b5blkp               ;restore block pointer
015521 121653 9 4 b5giv1: jst [mesget] i           ;get a mess no
015522 003330 9 4          jmp b5a                 ;none available, quit
015523 011613 9 4         sta b5seqh               ;save messno/blockno
015524 121646 9 4         jst [getall] i           ;get allocate
015525 101000 9 4          nop                     ;always skips
015526 140040 9 4         cra 0&subtyp             ;make sure subtype is zero
015527 011616 9 4         sta b5midh
015530 004066 9 4         lda [dattyp+datcpt+prirty]
015531 011610 9 4         sta b5typh               ;packet type
015532 005654 9 4         lda [mltpkt+gvbcod]
015533 013614 9 4         era b5pkth               ;set pkt code
015534 011614 9 4         sta b5pkth
015535 133655 9 4         stx [hibl+th] i          ;fake hi block for tsbput
015536 100000 9 4         skp
015537 120114 9 4 b5giv1:  jst sleep i
                  .ret bck
015540 073656 9           ldx [th]                 ;make believe we're host #8
015541 140040 9           cra
015542 001001 9           .inh h2i
015543 121657 9 4         jst [tsbput] i           ;get and set up tsb block
015544 003537 9 4          jmp b5giv1              ;<1 tsb blocks left, wait
015545 003412 9 4         jmp b5com2
```

```
;incomplete queries come here
015546 140040 9 4 b5incq: cra
015547 011612 9 4         sta b5temp              ;clear for later
015550 044340 9 4         lda mbmes x             ;get messno for inc
015551 025612 9 4 b5incm: irs b5temp              ;1 means oldest, 2 next oldes
015552 040677 9 4         arr 1
015553 100400 9 4         spl                     ;is the oldest possible mess
015554 003551 9 4          jmp b5incm             ;no, count one more backwards
015555 005612 9 4         lda b5temp
015556 141240 9 4         icr                     ;get no of messnos to roll ba
015557 054250 9 4         add mbtim x
015560 141044 9 4         car 0&mesnum
015561 015660 9 4         add [-11*mesnm1]
015562 013613 9 4         era b5seqh
015563 011613 9 4         sta b5seqh
015564 033611 9 4         stx b5blkp              ;save block pointer
015565 173611 9 4         ldx b5blkp i 0&mbimp    ;get imp #
                          ;defhlt(/3. sending inc? to imp=x, messno=a)
015566 120120 9 4         jst hltjst i
015567 013616 9 4         era b5midh              ;get local block number
015570 141044 9 4         car 0&blknum
015571 013616 9 4         era b5midh
015572 121661 9 4         jst [tsbget] i          ;find tsb block
015573 003606 9 4          jmp b5incs             ;not found
015574 044220 9 4         lda lid x               ;mark message as lost in net

015575 006032 9 4         ana [linkno]
015576 012011 9 4         era three 0&clost
015577 050220 9 4         sta lid x
015600 044110 9 4         lda lcd x               ;get message code
015601 006063 9 4         ana [tsbrq1&tsbrq8]     ;pick out requests only
015602 012063 9 4         era [tsbrq1&tsbrq8]     ;reverse sense - mesgs and gv
015603 041477 9 4         lgl 1 0&allusd          ;on if mess used allocate
015604 013610 9 4         era b5typh
015605 011610 9 4         sta b5typh
015606 004055 9 4 b5incs: lda ten 0&qercod
015607 003406 9 4         jmp b5comn
```