```
;It's possible that there is no IOCB. This can happen if
;I2MTO just ran and did the CM1.RESET which will post another interrupt
;(on top of the timeout one in progress). Thus, then I2MTO debreaks, it
;will run again immediately, this time with no IOCB. It should be noted
;that if desired, one could now eliminate the I2M busy check made by
;anyone who wants to poke I2M. It is almost "okay" just to poke I2M
;whenever you want since it recognizes the case where it is poked but
;has no IOCB to do the next output. The tricky part in making that change
;is that I2M must re-issue the TPR for the debreak time but it CANNOT
;simply use its original value. Instead, it has to compute the amount
;of time left and TPR for that long otherwise pokes could hold off the
;timecut forever. Since we're retaining the busy flag for now, this
;cant happen.

 006242 044024 2   i2mnc:  lda pcb.put x          ;get ptr to PUT queue
 006243 000022 2           deq                    ;try to pick off just-complet
 006244 003532 2            jmp i2mdb2            ;if there was no IOCB, just d
 006245 032244 2           stx oiocb              ;save ptr to IOCB we just rem

 006246 001001 2           .inh all               ;KDL feels this .INH needs to
                                                  ;in the code, EAH not convinc
 006247 072243 2 0         ldx ochn               ;get modem number
 006250 145665 2 0         lda [mblock] ix
 006251 010000 2 0         sta 0
 006252 010202 2 0         sta ombl
 006253 044117 2 0         lda line x             ;check line state
 006254 101400 2 0         smi                    ;..to see if silent or dead
 006255 100100 2 0          slz                   ;
 006256 100000 2 0            skp                 ;yes
 006257 003272 2 0             jmp i2ms2          ;no ,continue

;The line is silent or dead. Kill any routing updates which were scheduled
;to dc out this modem.

 006260 140040 2 0         cra                    ;clear send rup flag
 006261 050204 2 0         sta sndrup x
 006262 050135 2 0         sta i2mrup x
 006263 072243 2 0 i2ms1:  ldx ochn
 006264 044052 2 0         lda bittab x           ;see if any updates for
 006265 121667 2 0         jst [rupdq] i          ;..this line
 006266 003271 2 0          jmp i2ms0             ;no more, enter main-line of
 006267 121670 2 0         jst [rupfls] i         ;yes ,dequeue and flush
 006270 003263 2 0         jmp i2ms1              ;..if last user
```

```
;Here after killing off any un-sent routing updates on a now-dead line.
;We check for the special case where M2I has ack'ed our current output
;packet. If it has, we flush it here. Throughout this code, I2MNXT points
;to the "next" (current might be more appropriate) packet to be sent. I
;suspect this global exists mostly so M2I can detect the case where it
;is ack'ing the current packet. If so, it seems to set I2MACK so we
;can flush it after the output completes.

006271 072202 2 0 i2ms0:  ldx ombl
006272 140040 2 0 i2ms2:  cra
006273 066142 2 0         ima i2mnxt x          ;clear but remember last pack
006274 010001 2 0         sta ireg

006275 140040 2 0         cra                   ;was last sent buffer acked?

006276 066143 2 0         ima i2mack x
006277 101040 2 0         snz
006300 003305 2 0          jmp i2mhip           ;no, do hi priority check
006301 072001 2 0         ldx ireg              ;yes, get saved address and
006302 120115 2 0         jst flushi i          ;go flush
006303 024223 2 0         irs nsfs              ;give back to sf
006304 072202 2 0         ldx ombl

;Now check to see if there are any "high-priority" actions which we need
;to perform. I2MRAR(modem) contains zero if no, non-zero if some special
;action is needed. I2MRAR is (conceptually) the "or" of a bunch of slow
;but rare possible actions. In one test, we can determine that none of
;them need doing (or checking). Saves oodles of CPU time in I2M...

006305 000401 2 0 i2mhip: .enb i2m
006306 044140 2           lda i2mrar x          ;any high pri. checks?
006307 100040 2           sze
006310 102051 2            jmp [i2mrrr] i

                          ;fall into I2MLIP
```

```
                        ;from previous page

 ;Here to check for "low-priority" actions. Normally, we come here directly
 ;from I2MHIP because I2MRAR was zero, indicating nothing special needed. Here
 ;at I2MLIP, we look for packets to output on first the I2M priority queue
 ;(SMPQ), then on the I2M non-priority (normal) queue (SMQ). If no data packets

 ;exist, we check for the need to send a NULL.

006311 001001 2   i2mlip: .inh m2i
006312 005671 2 1         lda [smpq]              ;calculate ptr to pri q
006313 014202 2 1         add ombl
006314 010000 2 1         sta 0
006315 120122 2 1         jst mgetqi i            ;new priority packet to send
006316 100000 2 1          skp
006317 003347 2 1           jmp i2mgnu            ;yes

006320 005672 2 1         lda [smq]               ;no, try non-priority
006321 014202 2 1         add ombl                ;set x-reg to pointer
006322 010000 2 1         sta 0                   ;..to reg q
006323 120122 2 1         jst mgetqi i            ;new reg packet to send ?
006324 100000 2 1          skp
006325 003347 2 1           jmp i2mgnu            ;yes

006326 000401 2 1         .enb i2m
006327 072202 2           ldx ombl                ;no, try null
006330 044110 2           lda snull x             ;time to send null ?
006331 100040 2           sze                     ;no
006332 103673 2            jmp [i2mnul] i         ;yes

 ;We get to here when we have absolutely nothing to do. We re-queue the IOCB
 ;back on the PUT queue (because we're not going to use it, after all) and
 ;debreak for eternity, or until someone GPR's us (like TASK).

006333 000020 2   i2midl: pcb
006334 044024 2           lda pcb.put x           ;get PTR to PUT queue
006335 072244 2           ldx oiocb               ;get PTR to IOCB
006336 000002 2           enq                     ;put it back
006337 100000 2            skp
006340 000000 2             %crash                ;I2MIDL: saw extra IOCB
006341 140040 2           cra                     ;enter here if you want a deb
006342 072202 2           ldx ombl                ;fix X (who knows what it mig
006343 050137 2           sta i2mbsy x
006344 000103 2           spr
006345 000000 2             %crash                ;I2MIDL: NMFS timeout
006346 003210 2           jmp i2m0                ;back to top of loop
```

```
;Here from I2MLIP+few with X=buffer which needs to be sent (it came from
;either MPQ or MQ, see I2MLIP for details). Set up I2MNXT to point to it.
;Also, make sure I2MREF is not minus1. (I2MREF=-1 means it is being
;re-transmitted).

                    .lck m2i
006347 000401 2 1 i2mgnu: .enb i2m
006350 004000 2           lda 0                 ;got a new packet
006351 072202 2           ldx ombl
006352 050142 2           sta i2mnxt x          ;set I2MNXT(modem)=packet ptr
006353 010000 2           sta 0
006354 010240 2           sta i2mref            ;mark as new pkt, don't check
006355 005674 2           lda [-32.]            ;give a pkt 32 tries
006356 050005 2           sta inch x            ;to be accepted
006357 044006 2           lda neth x
006360 010001 2           sta ireg              ;save neth for later
006361 072202 2           ldx ombl
006362 044106 2           lda maxoctet x        ;old format line?
006363 101040 2           snz
006364 003547 2            jmp i2mold           ;yes
006365 004001 2           lda ireg              ;restore neth in a
006366 141050 2           cal                   ;isolate channel number
006367 054123 2   i2mgn1: add i2mtab x         ;compute address of channel s
006370 010241 2           sta i2mslt
006371 004240 2           lda i2mref
006372 110241 2           sta i2mslt i          ;put buffer pointer in slot
006373 001001 2           .inh m2i
006374 044125 2 1 i2mchf: lda retrqi x          ;come here with new packet
006375 072240 2 1         ldx i2mref            ;OR with a packet to retransm
006376 000002 2 1         enq                   ;attach to retransmission que
006377 101000 2 1          nop                  ;return here if queue was emp
006400 000401 2 1         .enb i2m
006401 072202 2           ldx ombl
006402 021553 2           jst getoctet          ;choose an octet, set up i2mc
006403 010002 2           sta jreg              ;getoctet returns rsex bits i
006404 044106 2           lda maxoctet x        ;old format line?
006405 072240 2           ldx i2mref            ;fix x before branching
006406 101040 2           snz
006407 003417 2            jmp i2mtph           ;yes, skip over neth word stu
006410 004242 2           lda i2moct            ;get octet number for packet
006411 141340 2           ica                   ;move to left byte
006412 052006 2           era neth x            ;insert into neth
006413 006074 2           ana [octet]
006414 052006 2           era neth x
006415 066006 2           ima neth x            ;store neth
006416 056006 2           sub neth x            ;for checksum update
006417 010001 2   i2mtph: sta ireg              ;a=0 if jump from above (old
006420 004002 2           lda jreg              ;get rsex bits to put in pack
006421 052007 2           era typh x
006422 141050 2           cal
006423 052007 2           era typh x            ;new typh word in a
006424 066007 2           ima typh x
006425 056007 2           sub typh x            ;difference in typh words
006426 014001 2           add ireg              ;difference in neth words
006427 054010 2           add chkh x            ;update checksum
006430 050010 2           sta chkh x
006431 121544 2           jst stats.im1 i       ;**stats
006432 000010 2   i2mdn0: %rdclok
006433 050121 2           sta itst x            ;sent time
```

```
006434 044007 2          lda typh x
006435 040467 2          lgr 9. 0&ldrtag          ;see if packet tagged
006436 121675 2          jst [gtdtyp] i           ;only good if data packet
006437 101001 2          ssc                      ;did carry bit stay on?
006440 003473 2           jmp i2mdn1              ;no, either not data or not t
006441 064016 2          irs data x               ;yes, increase tag count
006442 044004 2          lda wrdc x               ;use real packet length
006443 141140 2          icl 0&pktsiz
006444 016025 2          sub [data+2-hedr]        ;minus the overhead
006445 062016 2          cas data x
006446 044016 2           lda data x              ;plenty of room
006447 100000 2           skp                     ;o.k., but at limit
006450 003470 2             jmp i2mdn2           ;too big, don't write tag
006451 015676 2          add [data]               ;pointer to word to put tag i
006452 014000 2          add 0
006453 010001 2          sta ireg
006454 044121 2          lda itst x               ;sent time
006455 056003 2          sub artm x               ;arrival time-xmit delay
006456 040472 2          lgr 6.                   ;shyft delay
006457 022007 2          cas [377]
006460 004007 2           lda [377]               ;clamp to 377
006461 101000 2           nop
006462 141240 2          icr                      ;delay in left half
006463 012106 2          era mine                 ;our imp no in right half
006464 126001 2          ima ireg i               ;put into packet
006465 116001 2          sub ireg i               ;adjust checksum
006466 054010 2          add chkh x
006467 003471 2          jmp i2mdn3               ;skip over i2mdn2
```

```
;Next three entry points from various places on previous page

006470 044010 2   i2mdn2: lda chkh x
006471 016052 2   i2mdn3: sub one                 ;compensate for irs of data
006472 050010 2           sta chkh x
006473 032001 2   i2mdn1: stx ireg                ;save packet's address for la
006474 044004 2           lda wrdc x
006475 141140 2           icl 0&pktsiz
006476 041474 2           lgl 4.                  ;compute in bits, please
006477 072244 2           ldx oiocb
006500 050003 2           sta iocb.size x         ;set up size
006501 004001 2           lda ireg                ;get packet's address
006502 014031 2           add [hedr]              ;offset into first word to ov
006503 050004 2           sta iocb.addr x
                                                  ;?? what about SOM and EOM?
006504 005631 2   i2mdun: lda ovrhed             ;fetch overhead count
006505 012011 2           era [ 5 ? 6 ]           ;change 5 to 6 or
006506 011631 2           sta ovrhed              ;...6 to 5 so that average ov

006507 072244 2           ldx oiocb               ;fix index (never know who en
006510 054003 2           add iocb.size x         ;compute size+overhead
006511 072202 2           ldx ombl
006512 054133 2           add i2mwt2 x
006513 050133 2           sta i2mwt2 x
006514 100001 2           src
006515 064132 2            irs i2mwtn x           ;double precision count
006516 101000 2             nop

;We are now ready to ENQ this IOCB to the GET queue for the modem. Since
;we're going to do output any moment, declare ourselves busy while we still
;have OMBL in X.

006517 004052 2           lda [1]
006520 050137 2           sta i2mbsy x
006521 000020 2           pcb
006522 044023 2           lda pcb.get x           ;get ptr to get queue
006523 072244 2           ldx oiocb               ;get ptr to IOCB
006524 066005 2           ima iocb.flags x        ;set IALWAYS
006525 140500 2           ssm 0&%ioflags.always
006526 066005 2           ima iocb.flags x
006527 000002 2           enq                     ;place it
006530 100000 2            skp
006531 000000 2             %crash                ;M2I: saw extra IOCB
006532 000020 2   i2mdb2: pcb                     ;here from I2MNIO
006533 001003 2           pdv                     ;poke device to see new frame

006534 005677 2           lda [ %30sec / 3 ]      ;set to wait for hardware com
006535 000203 2           tpr                     ;set timeout
006536 000103 2           spr
006537 100000 2            skp
006540 003210 2           jmp i2m0                ;all went well, process some

;Here from the NMFS debreak if the wakeup was via a NMFS timeout. Kill
;the current xfer. This transfer will look like it had an error because
;the NMFS microcode sets the error flags in an IOCB which was reset.

006541 004052 2           lda [ %cm1.reset ]
006542 000403 2           xdv
006543 003210 2           jmp i2m0                ;then handle like interrupt w
```

```
006544 006545 2   stats.im1:      jsti2m
006545 000000 2   jsti2m: 0
006546 103545 2           jmp jsti2m i

006547 004001 2   i2mold: lda ireg                ;get channel
006550 006074 2           ana [chn8]
006551 141340 2           ica
006552 003367 2           jmp i2mgn1              ;continue with i2mgnu

;getoctet looks in the octet table and gets the octet number
;of the next octet to send (pointed to by "sendoct") and sets
;i2moct to that value.  It updates the table by moving sendoct
;one position (wrapping if necessary) and it also turns off
;i2moct's bit in snull, to show that that octet is no longer in
;the table.  Then this octet of rsex is returned in the A register.
;It is not necessary to inhibit for this routine since task is
;the only other manipulator of the octet table and snull, and
;task can't interrupt i2m.
006553 000000 2   getoctet: 0
006554 044110 2           lda snull x             ;anything in octet table?
006555 101040 2           snz
006556 003611 2            jmp tempty             ;no, table empty
006557 044113 2           lda sendoct x           ;get pointer to octet number
006560 010001 2           sta ireg
006561 104001 2           lda ireg i              ;octet number in A
006562 010242 2           sta i2moct              ;return value for calling rou
006563 015700 2           add [bittab]            ;compute address of mask for
006564 010001 2           sta ireg
006565 104001 2           lda ireg i              ;mask for snull
006566 052110 2           era snull x             ;turn off bit in snull
006567 050110 2           sta snull x
006570 044113 2           lda sendoct x           ;bump pointer
006571 141206 2           aoa
006572 062112 2           cas octend x            ;wrap around?
006573 000000 2            %crash
006574 044111 2            lda octtab x           ;yes, reset pointer to top
006575 050113 2           sta sendoct x           ;store new value
006576 004242 2           lda i2moct              ;offset modem block pointer
006577 040477 2   getrsex:lgr 1                   ;by adding octet number div 2
006600 014202 2           add ombl
006601 010000 2           sta 0                   ;use this to access rsex word
006602 004242 2           lda i2moct
006603 100100 2           slz                     ;odd octet?
006604 003626 2            jmp getodd             ;yes, get odd byte of rsex
006605 044065 2           lda rsex x
006606 141050 2   getfin: cal                     ;return rsex bits in right ha
006607 072202 2           ldx ombl                ;restore modem block pointer

006610 103553 2           jmp getoctet i          ;return

;if octet table is empty, bump both sendoct and putoct
;back to the previous spot and send that octet number.
;the idea is that even if the octet table is empty,
;there may still be acks outstanding due to a dropped
;packet with our acks in it.  by sending octets that
;recently had acks, perhaps we can prevent a retransmission.
006611 044113 2   tempty: lda sendoct x           ;octet table is empty
006612 016052 2           sub one                 ;bump sendoct backwards
```

```
006613 062111 2          cas octtab x          ;wrap backwards?
```

```
006614 003620 2              jmp tempt1          ;no
006615 003620 2              jmp tempt1          ;no
006616 044112 2            lda octend x          ;yes, move to bottom of table
006617 016052 2            sub one               ;octend points one word past

006620 050113 2   tempt1:  sta sendoct x         ;table is empty so
006621 050114 2            sta putoct x          ;bump putoct too
006622 010001 2            sta ireg              ;get the octet number from th
006623 104001 2            lda ireg i
006624 010242 2            sta i2moct
006625 003577 2            jmp getrsex           ;continue with getoctet

006626 044065 2   getodd:  lda rsex x            ;switch bytes of rsex
006627 141340 2            ica                   ;so odd octet gets returned
006630 003606 2            jmp getfin            ;and finish up

                  .lev var
006631         V  ovrhed:  .block 1              ;toggles between 5 & 6 avg =
```

```
;Subroutine to do NMFS-related I2M initialization. This is a subroutine only
;for clarity (it has only one caller, near I2MINI). We set up the get and
;put queue pointers and headers, then create the IOCB (and put it on the
;PUT queue).

                .lev i2m
006632 000000 2 i2mnmf: 0
006633 000020 2         pcb                             ;get PCB pointer
006634 004000 2         lda 0                           ;get base
006635 015701 2         add [ i2mpcb.getqh ]            ;get ptr to queue header
006636 050023 2         sta pcb.get x
006637 021651 2         jst i2mnmq                      ;initialize the queue header,
006640 014054 2         add [ i2mpcb.putqh - i2mpcb.getqh ] ;to PUT queue, no
006641 050024 2         sta pcb.put x
006642 021651 2         jst i2mnmq                      ;initialize the queue header,
006643 014054 2         add [ i2mpcb.iocb - i2mpcb.putqh ] ;to IOCB
006644 026000 2         ima 0                           ;put IOCB ptr in X, PCB in A

006645 015702 2         add [ i2mpcb.putqh ]            ;compute address of put queue
006646 000002 2         enq                             ;equivelant to LDA PCB.PUT X,
006647 103632 2          jmp i2mnmf i                   ;return
006650 000000 2           %crash                        ;I2MNMF: found extra IOCB

;Subroutine (called only from I2MNMF) to init a queue header pointed to by A.
;Preserves A, X.

006651 000000 2 i2mnmq: 0
006652 032001 2         stx ireg                        ;save caller's Index
006653 010002 2         sta jreg                        ;  ... and AC
006654 010000 2         sta 0                           ;put address of queue in X
006655 050000 2         sta q.forw x                    ;point it at self
006656 050001 2         sta q.back x
006657 050002 2         sta q.hedr x
006660 140040 2         cra                             ;length = 0, too
006661 050003 2         sta q.size x
006662 072001 2         ldx ireg                        ;fix index
006663 004002 2         lda jreg                        ;and AC
006664 103651 2         jmp i2mnmq i                    ;return
```

```
                    .section pg0
                    .lev var
000240          V   i2mref: .block 1        ;retransmit flag: checksum if
000241          V   i2mslt: .block 1        ;ptr to channel slot
000242          V   i2moct: .block 1        ;octet number for output
```

```
;Come here from I2MHIP when I2MRAR(modem) is non-zero. Figure out which of the
;many special actions needs attention..

;I2MREM(modem) non-zero means we need to reset the modem.

                    .section pg7
                    .lev i2m
007000 140040 2   i2mrrr: cra                     ;reset modem?
007001 066141 2           ima i2mrem x
007002 101040 2           snz
007003 003006 2            jmp i2mpcc             ;no
007004 004012 2           lda [%cm1.unloop]       ;yes, get the unloop code
007005 000403 2           xdv                     ;poof

007006 140040 2   i2mpcc: cra                     ;clear/check packet core comm
007007 066134 2           ima i2mpkc x            ;..flag
007010 100040 2           sze                     ;was last output to a loader?
007011 003072 2            jmp i2m04              ;yes, see if it should be re
                                                  ;note: AC guarenteed to be ze
007012 066135 2           ima i2mrup x            ;rup output complete ?
007013 101040 2           snz                     ;
007014 003024 2            jmp i2mspc             ;no ,check for pkc done

;If the RUP has been output, clear it's bit and possibly flush it.

007015 010001 2           sta ireg
007016 072243 2           ldx ochn
007017 044052 2           lda bittab x            ;get bit for this modem
007020 072001 2           ldx ireg                ;x gets buffer address
007021 001001 2           .inh m2i
007022 121537 2 1         jst [rupfls] i          ;dequeue and flush if last us
007023 000401 2 1         .enb i2m

                          ;fall into I2MSPC
```

```
                        ;from previous page
;Check for more "rare" conditions...

007024 072202 2   i2mspc: ldx ombl
007025 105540 2           lda [pkcmdn] i        ;time to send pck core?
007026 012000 2           era 0                 ;our modem ?
007027 100040 2           sze                   ;
007030 003034 2            jmp i2mzpk           ;no ,check for z-packet

007031 105541 2           lda [pkcbff] i        ;buffer busy flag
007032 100040 2           sze                   ;anything to send ?
007033 003123 2            jmp i2mpcr           ;yes

007034 140040 2   i2mzpk: cra                   ;time to send line up/down
007035 066160 2           ima zsend x           ;..protocol packet ?
007036 100040 2           sze                   ;no ,do send rup check
007037 003244 2            jmp i2mzlt           ;yes
007040 044204 2           lda sndrup x          ;time to send rup ?
007041 100040 2           sze                   ;no ,do demand core check
007042 003145 2            jmp i2msru           ;yes

007043 044115 2   i2mdck: lda slt x             ;do demand core ?
007044 100040 2           sze                   ;no check retransmission
007045 103542 2            jmp [i2mdmc] i       ;yes

007046 044117 2           lda line x            ;check line state
007047 101400 2           smi                   ;down or reset?
007050 100100 2            slz
007051 103543 2             jmp [i2midl] i      ;yes exit w/o checking for mo
                                                 ;(we just checked them all an
007052 044202 2           lda retflg x          ;time to transmit packet ?
007053 101040 2           snz
007054 003066 2            jmp i2mclr           ;no do low pri. checks
007055 010001 2           sta ireg              ;address of buffer to retrans
007056 001001 2           .inh m2i
007057 044125 2 1         lda retrqi x          ;see if it has already been a
007060 010000 2 1         sta 0                 ;by checking if it is still t
007061 044000 2 1         lda q.forw x          ;first buffer on retrq
007062 012001 2 1         era ireg              ;same pointer?
007063 072202 2 1         ldx ombl              ;fix X before branching
007064 101040 2 1         snz                   ;skip if pointers not equal
007065 003421 2 1           jmp i2mret          ;yes ,go retrans.
007066 140040 2 1 i2mclr: cra                   ;clear retflg
007067 050202 2 1         sta retflg x
007070 050140 2 1         sta i2mrar x          ;clear high pri flg
007071 103544 2 1         jmp [i2mlip] i        ;now check for non-rare thing
```

```
                        .lev i2m
007072 173545 2   i2m04:  ldx [pkcbfr] i          ;ptr to pc buffer
007073 064005 2           irs inch x             ;should it be retrans.
007074 003104 2            jmp i2mpc1            ;yes, don't flush yet

007075 001001 2           .inh all
007076 120115 2 0         jst flushi i           ;no, free buffer
007077 024224 2 0         irs nres               ;give back to reassembly
007100 000401 2 0         .enb i2m

007101 140040 2           cra
007102 111541 2           sta [pkcbff] i         ;clear buffer busy flag
007103 111546 2           sta [pkcbf2] i

007104 072202 2   i2mpc1: ldx ombl
007105 064137 2           irs i2mbsy x           ;declare ourselves busy
007106 000020 2           pcb
007107 044024 2           lda pcb.put x          ;put iocb back on put queue
007110 072244 2           ldx oiocb
007111 000002 2           enq
007112 100000 2            skp
007113 000000 2             %crash               ;I2MPC1 saw extra IOCB
007114 000020 2           pcb
007115 004056 2           lda [ %25.6ms ]        ;?? what is this wait here fo
007116 000203 2           tpr                    ;probably to prevent hello's
007117 000103 2           spr                    ;being sent out this line, wh
007120 100000 2            skp                   ;for bck (fhppkc) to run agai
007121 000000 2             %crash               ;I2M: got poked while waiting
007122 103547 2           jmp [i2m0] i           ;now we're done

;come here if pkcbff is non-zero, meaning that a PKC output,
;presumably to a neighbor's loader, is scheduled.
007123 105546 2   i2mpcr: lda [pkcbf2] i
007124 101040 2           snz                    ;retransmitting ?
007125 003140 2            jmp i2mpk1            ;no, first time
007126 173545 2           ldx [pkcbfr] i         ;yes ,time to retransmit?
007127 000010 2           %rdclok
007130 056121 2           sub itst x
007131 100400 2           spl
007132 140407 2            tca
007133 017550 2           sub [pkcrtt]           ;retransmission interval
007134 072202 2           ldx ombl
007135 100400 2           spl
007136 003034 2            jmp i2mzpk            ;not time to retrans.
007137 003141 2           jmp i2mpk2             ;go retransmit

007140 125546 2   i2mpk1: irs [pkcbf2] i         ;make retr. flag non-zero
007141 072202 2   i2mpk2: ldx ombl
007142 064134 2           irs i2mpkc x
007143 173545 2           ldx [pkcbfr] i
007144 103551 2           jmp [i2mdn0] i
```

```
007145 072243 2   i2msru: ldx ochn
007146 044052 2           lda bittab x          ;mod bit mask
007147 010241 2           sta i2mslt

007150 001001 2           .inh m2i
007151 121552 2 1         jst [rupdq] i         ;get rup buffr
007152 003173 2 1          jmp i2mal5           ;nothing on q for us
007153 000401 2 1         .enb i2m

007154 020002 2           jst chksum            ;check checksum of packet
007155 103553 2            jmp [i2mrc1] i       ;bad length
007156 100040 2           sze                   ;
007157 103554 2            jmp [i2mrce] i       ;bad checksum
007160 121536 2           jst stats.anrumi i    ;**stats
007161 004000 2           lda 0                 ;
007162 072202 2           ldx ombl              ;
007163 050135 2           sta i2mrup x          ;set rup output complete
007164 014031 2           add [hedr]            ;..flg
007165 072244 2           ldx oiccb             ;get ptr to this IOCB
007166 050004 2           sta iocb.addr x       ;set up ptr to output
007167 072202 2           ldx ombl              ;fix index
007170 044135 2           lda i2mrup x
007171 010000 2           sta 0
007172 103555 2           jmp [i2mdn1] i

                  .lck m2i
007173 072202 2 1 i2mal5: ldx ombl
007174 050204 2 1         sta sndrup x          ;note: a=0 from rupdq
007175 000401 2 1         .enb i2m
007176 003043 2           jmp i2mdck

007177 001001 2   i2mrc1: .inh all
                          ;defhlt(/75. i2m - routing update with bad length)
007200 021203 2 0         jst i2mrch

                  i2mrce:
007201 001001 2 0         .inh all
                          ;defhlt(/76. i2m - routing update with intra-imp xsum
007202 021203 2 0         jst i2mrch

007203 000000 2 0 i2mrch: 0
007204 120126 2 0         jst hltncc i
007205 004241 2 0         lda i2mslt
007206 121537 2 0         jst [rupfls] i        ;flush bad update
007207 000401 2 0         .enb i2m
007210 003145 2           jmp i2msru            ;go look for another update
```

```
;Come here from I2MLIP+few when SNULL(modem) is non-zero implying the need
;to send a NULL to the neighbor. We build the NULL here. This code is also
;used by the send-packet-with-discard-bit-set.

                    stats.nulpsi:
007211 040000 2             stats.dummy

007212 072243 2     i2mnul: ldx ochn
007213 165211 2             irs stats.nulpsi ix     ;** stats
007214 101000 2              nop
007215 072202 2             ldx ombl                ;restore modem block pointer

007216 121556 2             jst [getoctet] i        ;return with rsex bits in A
007217 013557 2             era [swttyp+swtcpt+nulbit] ;null pkt type
007220 010001 2             sta ireg                ;set up typh of null or disca
007221 044144 2             lda nulptr x            ;pointer to neth word of nul
007222 010000 2             sta 0
007223 004001 2             lda ireg                ;typh word contents
007224 050001 2             sta typh-hedr x
007225 004242 2             lda i2moct              ;set up by getoctet
007226 141340 2             ica                     ;will be ignored by old forma
007227 050000 2             sta neth-hedr x
007230 021306 2             jst i2mnlb              ;set up transfer from NULL ar
007231 004153 2     i2mn14: lda sync                ;put sync time into nulls,z-p
007232 050004 2             sta seqh-hedr x
007233 004106 2             lda mine
007234 050003 2             sta srch-hedr x
007235 004012 2     i2mn16: lda [seqh-hedr+1]       ;build checksum
007236 056000 2             sub neth-hedr x
007237 056001 2             sub typh-hedr x
007240 056003 2             sub srch-hedr x
007241 056004 2             sub seqh-hedr x
007242 050002 2             sta chkh-hedr x         ;and store it
007243 103560 2             jmp [i2mdun] i          ;up for output (okay to have

;generate a z-protocol packet: hello or ihy. Much like I2MNUL (build packet
;in the NULL area), but different packet contents (of course).

007244 072243 2     i2mzlt: ldx ochn
007245 165305 2             irs stats.ppksi ix      ;**stats
007246 101000 2              nop
007247 072202 2             ldx ombl
007250 001001 2             .inh m2i                ;in case m2i kills line while
007251 044117 2 1           lda line x              ;..we're building a zpacket
007252 100040 2 1           sze                     ;line z-up?
007253 004054 2 1            lda [zhloup]           ;no, change bit
007254 013561 2 1           era [swttyp+compat+zhloup+zhltyp]
007255 010001 2 1           sta ireg                ;prepare typh word
007256 044175 2 1           lda zmastr x
007257 100040 2 1           sze                     ;master?
007260 004052 2 1            lda [zhlhlo]           ;get hello bit
007261 012001 2 1           era ireg
007262 010001 2 1           sta ireg                ;typh all set up
007263 044117 2 1           lda line x              ;if line silent or down,
007264 100400 2 1           spl                     ;..send cfoctet
007265 003303 2 1            jmp i2mzld             ;silent
007266 100100 2 1           slz
007267 003303 2 1            jmp i2mzld             ;down
007270 044106 2 1           lda maxoctet x          ;otherwise, send maxoctet
```

```
007271 010002 2 1 i2mzl1: sta jreg                ;must get pointer in x first
```

```
007272 000401 2 1          .enb i2m
007273 044144 2            lda nulptr x
007274 010000 2            sta 0                    ;pointer to neth word of null
007275 004002 2            lda jreg
007276 050000 2            sta neth-hedr x          ;number of octets on line
007277 004001 2            lda ireg                 ;now store typh
007300 050001 2            sta typh-hedr x
007301 021306 2            jst i2mn1b               ;build up transfer params in
007302 003231 2            jmp i2mn14               ;finish packet and output it


007303 044107 2   i2mz1d: lda cfcctet x            ;send config value while sile
007304 003271 2            jmp i2mz11

                  stats.ppksi:
007305 040000 2            stats.dummy
```

```
;This subroutine sets up IOCB parameters (SIZE and ADDR) for transfers
;from the NULL area (also used by line up/down as well).
007306 000000 2   i2m.11: 0
007307 032001 2           stx ireg                  ;save nulptr
007310 004000 2           lda 0                     ;get nulptr in a
007311 072244 2           ldx oiocb                 ;get ptr to IOCB
007312 050004 2           sta iocb.addr x           ;set up addr
007313 005562 2           lda [16.*(seqh-hedr+1)] ;set up length (in bits)
007314 050003 2           sta iocb.size x
007315 072001 2           ldx ireg
007316 103306 2           jmp i2mnlb i
```

```
;send demand core

007317 016052 2   i2mdmc: sub one
007320 050115 2           sta slt x
007321 005563 2           lda [dmndcr]
007322 072244 2           ldx oiocb
007323 050004 2           sta iocb.addr x
007324 004060 2           lda [16.*(seqh-hedr)]
007325 050003 2           sta iocb.size x
007326 073563 2           ldx [dmndcr]
007327 003235 2           jmp i2mn16              ;get checksums and send it

                          ;defhlt(/23. i2m bad length packet for adder)
00733C 021332 2   i2mchl: jst i2mch1              ;come here inhibited from i2m

                  .lev i2m
                  .lck all
                          ;defhlt(/24. modem cut detected intra-imp checksum er
007331 021332 2 0 i2mche: jst i2mch1              ;come here inhibited from i2m
```

```
007332 000000 2 0 i2mch1: 0
007333 120126 2 0         jst hltncc i          ;report it
007334 021406 2 0         jst diagpt            ;put on diag queue
007335 044004 2 0         lda wrdc x            ;check if pkt being traced
007336 141240 2 0         icr
007337 101400 2 0         smi 0&trcpkt
007340 003343 2 0          jmp i2mch2           ;no
007341 004243 2 0         lda ochn
007342 120145 2 0         jst trace.m2idone i   ;** trace pkt
007343 044006 2 0 i2mch2: lda neth x            ;get channel number
007344 010002 2 0         sta jreg
007345 072202 2 0         ldx ombl
007346 044106 2 0         lda maxoctet x
007347 101040 2 0         snz                   ;old format line?
007350 003402 2 0          jmp i2mch8           ;yes, extract channel number

007351 004002 2 0         lda jreg
007352 141050 2 0         cal
007353 010002 2 0 i2mch3: sta jreg              ;channel number from either
007354 054123 2 0         add i2mtab x
007355 010001 2 0         sta ireg
007356 140040 2 0         cra
007357 110001 2 0         sta ireg i            ;zero channel slot
007360 044105 2 0         lda chcounter x
007361 016052 2 0         sub one
007362 050105 2 0         sta chcounter x       ;one less channel in use
007363 100040 2 0         sze                   ;was this the only buffer on
007364 024223 2 0          irs nsfs             ;no, return to store and forw
007365 004002 2 0         lda jreg              ;compute addr of bit mask for
007366 006072 2 0         ana [17]              ;channel number mod 16
007367 015564 2 0         add [bittab]
007370 010001 2 0         sta ireg              ;get channel bit
007371 004002 2 0         lda jreg              ;offset modem block pointer
007372 040474 2 0         lgr 4                 ;to get correct chfree word
007373 014202 2 0         add ombl
007374 010000 2 0         sta 0
007375 104001 2 0         lda ireg i
007376 052075 2 0         era chfree x          ;turn off channel-busy bit
007377 050075 2 0         sta chfree x
007400 000401 2 0         .enb i2m
007401 103547 2           jmp [i2m0] i          ;look for new things to do

007402 004002 2   i2mch8: lda jreg
007403 006074 2           ana [chn8]            ;get channel number from old
007404 141340 2           ica
007405 003353 2           jmp i2mch3

                  .lev (m2i,i2m)
007406 000000 1   diagpt: 0
007407 004055 1           lda ten               ;room for diag pkt?
007410 120117 1           jst maxchi i
007411 003417 1            jmp diagp1           ;no
007412 024220 1           irs nrea              ;res, count in reassembly
007413 004000 1           lda 0                 ;and put on diag queue
007414 026200 1           ima diagq
007415 050000 1           sta 0 x
007416 103406 1           jmp diagpt i

007417 120115 1   diagp1: jst flushi i
```

```
007420 103406 1          jmp diagpt i
```

```
                        .lev i2m
                        .lck m2i
007421 066202 2 1 i2mret: ima retflg x          ;A=0
007422 010000 2 1         sta 0                 ;buffer address
007423 064005 2 1         irs inch x            ;count another retransmit
007424 003463 2 1          jmp i2mrtr           ;not over the limit
007425 044006 2 1         lda neth x
007426 006067 2 1         ana [dscpkt]
007427 101040 2 1         snz                   ;discarded packet?
007430 003442 2 1          jmp i2mrrc           ;/ no, try to reroute
007431 072243 2 1         ldx ochn              ;yes, give up
                          ;defhlt(/25. too many retransmissions of discarded pa
007432 021433 2 1         jst i2mhlt
007433 000000 2 1 i2mhlt: 0
007434 120126 2 1         jst hltncc i          ;trap to ncc
007435 072202 2 1         ldx ombl
007436 120123 2 1         jst killii i          ;kill the line
007437 000401 2 1         .enb i2m
007440 103543 2           jmp [i2midl] i        ;and dismiss

                        stats.ret32i:
007441 006545 2           jsti2m

                          .lck m2i
007442 032001 2 1 i2mrrc: stx ireg              ;save x
007443 121441 2 1         jst stats.ret32i i    ;** stats
007444 044014 2 1         lda dsth x
007445 010000 2 1         sta 0                 ;x<-dest
007446 004243 2 1         lda ochn              ;calculate added delay
                          ;defhlt(/78. i2m: 32 retransmissions)
007447 120120 2 1         jst hltjst i          ;trap if 32 retransmissions
007450 072001 2 1         ldx ireg              ;restore x
007451 004202 2 1         lda ombl              ;delay requires A=modem block
007452 121565 2 1         jst [delay] i
007453 044006 2 1 i2mdsc: lda neth x            ;discard the packet
007454 012067 2 1         era [dscpkt]
007455 066006 2 1         ima neth x            ;turn on discard bit
007456 056006 2 1         sub neth x            ;adjust checksum
007457 054010 2 1         add chkh x
007460 050010 2 1         sta chkh x
007461 005566 2 1         lda [-32.]            ;reset discard retrans. limit
007462 050005 2 1         sta inch x
007463 044007 2 1 i2mrtr: lda typh x            ;is packet tagged?
007464 040467 2 1         lgr 9. 0&ldrtag
007465 121567 2 1         jst [gtdtyp] i
007466 101001 2 1         ssc
007467 003475 2 1          jmp i2mrt1           ;no
007470 044016 2 1         lda data x            ;yes, adjust tagged count
007471 016052 2 1         sub [1]
007472 050016 2 1         sta data x
007473 064010 2 1         irs chkh x            ;and now adjust checksum
007474 101000 2 1          nop
007475 004000 2 1 i2mrt1: lda 0
007476 072202 2 1         ldx ombl
007477 050142 2 1         sta i2mnxt x
007500 010240 2 1         sta i2mref            ;buffer pointer in i2mref
007501 044125 2 1         lda retrqi x          ;remove buffer from retrq
007502 000022 2 1         deq                   ;will be enq'ed by i2mchf
007503 101000 2 1          nop                  ;if it passes checksum test
```

```
007504 020002 2 1          jst chksum          ;do checksum
007505 103570 2 1           jmp [i2mchl] i     ;bad length
007506 100040 2 1          sze
007507 103571 2 1           jmp [i2mche] i     ;checksum error!
007510 072202 2 1          ldx ombl            ;enq, retransmit, etc
007511 103572 2 1          jmp [i2mcnf] i      ;still inhibited
```

```
007512 000000 2 1 gtdtyp: 0                     ;determine if packet is data
007513 044007 2 1         lda typh x
007514 006037 2 1         ana [pactyp]
007515 100040 2 1         sze 0&dattyp          ;is it type 0?
007516 140200 2 1          rcb                  ;result returned in carry bit
007517 044015 2 1         lda midh x            ;check for raw packet
007520 006072 2 1         ana [subtyp]
007521 012011 2 1         era [subunc]
007522 101040 2 1         snz
007523 103512 2 1          jmp gtdtyp i         ;raw, flags present
007524 044013 2 1         lda pkth x            ;checkout spf leader flags
007525 006072 2 1         ana [pktcod]          ;ok if pktcod=0 (data)
007526 022052 2 1         cas [1]               ;or if pktcod=1 & mltpkt=0
007527 140200 2 1          rcb                  ;n.g., no tagging
007530 100000 2 1          skp                  ;must check if mltpkt=0
007531 103512 2 1           jmp gtdtyp i        ;pktcod=0
007532 044013 2 1         lda pkth x
007533 100400 2 1         spl 0&mltpkt          ;is mltpkt=0?
007534 140200 2 1          rcb                  ;no, no tagging
007535 103512 2 1         jmp gtdtyp i

                    .lev con
                    stats.anrumi:
007536 006545 C             jsti2m

                    .section pg0
                    .lev var
000243         V    ochn:   .block 1
000244         V    oiocb:  .block 1
```

```
                .stitl HOST TO IMP (H2I)
                .INCLUDE h2i.m4
;NMFS notes:



;H2I's PCB:

        000026  h2ipcb.getqh = pcb.ff           ;get queue header
        000032  h2ipcb.putqh = h2ipcb.getqh + qhedr.size ;put queue header
        000036  h2ipcb.iocb = h2ipcb.putqh + qhedr.size ;IOCB
        000044  h2ipcb.hacmem = h2ipcb.iocb + biocb.size ; HACMEM word for ho
        000045  h2ipcb.haccom = h2ipcb.hacmem + 1 ; HACCOM word for host
        000046  h2ipcb.swpdv = h2ipcb.haccom + 1 ;sw host pdv PCB addr
        000047  h2ipcb.swxdv = h2ipcb.swpdv + 1 ;sw host xdv subr addr
        000050  h2ipcb.size = h2ipcb.swxdv + 1  ;size
```

```
                .section pg10
                .lev h2i

;Come here after INI runs. We just build the NMFS stuff and fall
;into HIR (H2I Reset).

010000 121645 4   h2iini: jst [h2inmf] i          ;set up NMFS stuff

;The only difference between HIR and HIR1 is whether the padding word is
;cleared.  On subtler resets such as host blocked, etc., we don't want to
;clear the host's previous padding.

010001 121646 4   hir:    jst [hiwspr] i          ;return from any debreak
010002 003006 4           jmp hir1a

010003 121646 4   hir1:   jst [hiwspr] i
010004 140040 4           cra
010005 150253 4           sta hilti ix

;Tell device driver to flush current transfer and reset. Then give associated
;buffer back (iff there is one).

010006 121647 4   hir1a:  jst [hixfls] i          ;flush the current IOCB
010007 140040 4           cra
010010 166252 4           ima hispi ix            ;get ptr to buffer (0=none, 1
010011 010000 4           sta 0                   ;also into X in case we go to
010012 006015 4           ana [ 177776 ]          ;see if anything but 0 or 1
010013 101040 4           snz
010014 003021 4            jmp hir2               ;0=nothing, 1=leader, skip th
010015 001001 4           .inh all
010016 120115 4 0         jst flushi i            ;flush it
010017 000401 4 0         .enb h2i
010020 024224 4           irs nres                ;back to reassembly

;Now wait for HOST to be declared UP. Note that X is trash when we fall
;into here from above.  In addition, we'll accept data from a tardy host
;as well as one which is up.  This is because we know that I2H declares
;the host down only when ts ready line is down.  Therefore, a tardy host
;is okay to accept messages from.  Note that its NOT OKAY to accept messages
;from a host whose state is HSTIGD (IMP going down) for these hosts are
;being nice-stopped.

010021 121646 4   hir2:   jst [hiwspr] i          ;for SPR, below
010022 144276 4           lda hosti ix            ;is HOST up?
010023 012052 4           era [hstup]
010024 101040 4           snz
010025 003035 4            jmp hir3               ;yes, check for fake host, fl
010026 012011 4           era [hstup?hsttdy]      ;or tardy?
010027 101040 4           snz
010030 003035 4            jmp hir3               ;okay to accept data from tar
010031 121650 4           jst [hi640] i           ;wait 640 ms and check again

010032 000103 4           spr
010033 003021 4            jmp hir2               ;check again
010034 000000 4           %crash                 ;HIR2: got poked

010035 150251 4   hir3:   sta hirsi ix           ;and clear reset flag because
010036 121651 4           jst [hifakt] i          ;is this host fake?
010037 003064 4            jmp hilead             ;yes, expect a leader next
```

```
010040 003053 4          jmp hichew                ;else, go chew until EOM (no
```

```
;Note that we use the leader area to flush until EOM. Pre-4300 IMPs used
;to use buffers.  Enter at HICHFL to discard the current buffer before chewing

;up the message.

010041 072245 4   hichfl: ldx hip                ;fix X
010042 144252 4           lda hispi ix           ;get packet pointer
010043 010000 4           sta 0                  ;in case we go to FLUSH
010044 006015 4           ana [ 177776 ]         ;look at all but LSB
010045 101040 4           snz                    ;is HISP=0,1?
010046 003053 4            jmp hichew            ;yes, skip this
010047 001001 4           .inh all
010050 120115 4 0         jst flushi i           ;yes, flush the current buffe
010051 000401 4 0         .enb h2i               ;(no need to restore X becaus
010052 024224 4           irs nres               ;count it

010053 121646 4   hichew: jst [hiwspr] i         ;return from debreak (see bel
010054 121652 4           jst [hieom] i          ;already EOM?
010055 100000 4            skp
010056 003064 4             jmp hilead           ;yes, done chewing, expect le
010057 121653 4           jst [hilenq] i         ;put the leader area up for i
010060 000103 4           spr                    ;wait for it to come in
010061 000000 4            %crash               ;HICHEW: timeout waiting for E
010062 003053 4           jmp hichew             ;test for EOM now
```

```
;Here when we expect a leader to be input. Put up the leader area for
;input. When it arrives, set HIWT to the arrival time so we can time
;out inputs which take too long to finish up. Although we wait forever
;for a new leader, we periodically (every 640ms) check for HIRS being
;non-zero. Recall that HIRS is set to non-zero when I2H wants us to
;reset the host.

010063 040000 4   stats.cnt2i: stats.dummy

010064 140040 4   hilead: cra                    ;cleanup: clear HIBL so RESET
010065 150260 4           sta hibli ix
010066 121653 4           jst [hilenq] i        ;enq the leader
010067 121654 4   hilchk: jst [hiwsp2] i        ;set up X, don't clear repok
010070 121655 4           jst [hirchk] i        ;check if I2H wants reset, go
010071 121650 4           jst [hi640] i         ; ... evey 640 ms
010072 000103 4           spr                   ;us to reset
010073 003067 4            jmp hilchk           ;loop if got here from timeou

010074 121646 4           jst [hiwspr] i        ;return from debreak
010075 004014 4           lda [ -1 ]            ;set first packet flag after
010076 150270 4           sta hifpi ix
010077 000010 4           %rdclok               ;set up for trace
010100 150257 4           sta hiiti ix          ;save input time
010101 004151 4           lda time              ;get time
010102 150254 4           sta hiwti ix          ;set up HIWT to time out thi
010103 165063 4           irs stats.cnt2i ix    ;**stats host message count

;Now process the leader. First check the leader type (new, old) to
;be sure it's a 96-bit (new) leader. If it's old, make it into an
;illegal type to force out an "error in leader message" shortly.

010104 104246 4           lda hilp i            ;get ptr to first word of lea
010105 024246 4           irs hilp              ;and step to the second leade
010106 006074 4           ana [ldrnlf]          ;is it 96-bit leader?
010107 012074 4           era [ldrnlf]
010110 100040 4           sze
010111 003167 4            jmp hilold           ;if old, set HILMTY to bogus
010112 104246 4           lda hilp i            ;set it to message type
010113 141050 4           cal 0&ldrmty
010114 010002 4   hilsto: sta jreg              ;(enter from HILOLD, too)

;Now process more of the second leader word (HILP still points there). JREG
;has message type.

010115 104246 4           lda hilp i            ;fetch it
010116 006074 4           ana [ldrtfl+ldrlfl]   ;get trace and leader flags
010117 150263 4           sta hihti ix          ;save in TYPH word
010120 024246 4           irs hilp              ;step to word 3

010121 104246 4           lda hilp i
010122 150265 4           sta hihhi ix          ;save handling type and host

010123 024246 4           irs hilp              ;step to word 4

010124 104246 4           lda hilp i
010125 150266 4           sta hihii ix          ;save dest IMP number
010126 024246 4           irs hilp              ; ... to word 5

010127 104246 4           lda hilp i
```

```
010130 150267 4          sta hihli ix          ;save mess id and subtype
```

```
;fall into next page
```

```
                        ;from previous page

;Now check to be sure the IOCB is not in error. If it is, send off an
;"error in the first 32 bits" message. JREG still has message type.

010131 072247 4          ldx hiiocb             ;get ptr to IOCB (still on PU
010132 044005 4          lda iocb.flags x       ;get flags word
010133 006072 4          ana [%ioflags.cc]      ;look at completion codes
010134 072245 4          ldx hip                ;fix index..
010135 100040 4          sze                    ;must be zero for good input

010136 003643 4           jmp hier32            ;give "error in first 32 bits

;Now check the length of the transfer (IOCB is pointed to by HIIOCB)

010137 072247 4          ldx hiiocb             ;get ptr to IOCB
010140 044003 4          lda iocb.size x        ;pick out size (in bits)
010141 017656 4          sub [ 80. ]            ;make sure it's .GE. 80. bits
010142 072245 4          ldx hip
010143 100400 4          spl
010144 003642 4           jmp hiersh            ;if less than that, give "lea

;Now dispatch on the type which has been stored into JREG. Note that
;if the leader was "old", JREG will have been smashed (by HILOLD) to
;contain an illegal type.

010145 004002 4          lda jreg               ;retrieve computed type
010146 022055 4          cas [8.]               ;is it greater than 8?
010147 003641 4           jmp hierbc            ;yes, bogus
010150 101000 4           nop
010151 015657 4          add [hidspt]
010152 010001 4          sta ireg
010153 104001 4          lda ireg i
010154 010001 4          sta ireg
010155 102001 4          jmp ireg i

010156 010275 4  hidspt: hi0 0&1creg            ;0 reg
010157 010053 4          hichew 0&1cerwo        ;1 imp format err
01016C 010200 4          hidown 0&1chgdn        ;2 host going down
010161 010641 4          hierbc                 ;3 bad code
010162 010171 4          hinop 0&1cnop          ;4 nop
010163 010641 4          hierbc                 ;5
010164 010641 4          hierbc                 ;6
010165 010641 4          hierbc                 ;7
010166 010053 4          hichew 0&1cerwi        ;8 err with id

;Here from HILUP when we detect an old leader. Force HILMTY to be "17"
;which is guarenteed to be determined "bad".

010167 004072 4  hilold: lda [17]               ;get a bogus tyle
01017C 003114 4          jmp hilsto             ;and STOre it into HILMTY, th
```

```
;Here from dispatch mechanism on NOPs. Copy the Subtype into the padding word
;(HILT).

010171 144267 4   hinop:  lda hihli ix          ;get subtype
010172 006072 4           ana [ldrsty]
010173 022024 4           cas [9.]              ;padding must be 0-9.
010174 003053 4            jmp hichew           ;.ge. 9, ignore it
010175 101000 4            nop
010176 150253 4           sta hilti ix          ;.le. 9, copy into padding wo
010177 003053 4           jmp hichew            ;then ignore till EOM

;Here from dispatch mechanism on HOST-GOING-DOWN messages. Validate
;the remaining information and store new down data. Then chew up message.

010200 144267 4   hidown: lda hihli ix          ;get going-down info
010201 006072 4           ana [hstwhy]
010202 022054 4           cas [4.]              ;is reason legitimate?
010203 023660 4            cas [14.]            ; 5. to 13. legal
010204 101000 4            nop
010205 003211 4            jmp hidwnx           ;no
010206 144267 4           lda hihli ix          ;get info back again
010207 150307 4   hidwnz: sta hdowni ix         ;store it
010210 003053 4           jmp hichew            ;and chew up message

010211 152267 4   hidwnx: era hihli ix          ;turn off all the reason bits
010212 012072 4           era [hstwhy]          ;and insert code 15. (unknwon
010213 003207 4           jmp hidwnz            ;then set it up and leave
```

```
;Here from below (only) when this is a RAW packet. Check the host's permission
;to use RAW packets, etc.

010214 000020 4   hiraw:  pcb                     ;point to PCB
010215 044044 4           lda h2ipcb.hacmem x     ;look at raw permission bit
010216 072245 4           ldx hip                 ;fix X
010217 101100 4           sln                     ;permitted?
010220 003053 4            jmp hichew             ;no, flush it
010221 012052 4           era one                 ;turn off for hac check
010222 150262 4           sta hihpi ix            ;save hacmem
010223 121652 4           jst [hieom] i           ;too short?
010224 100000 4            skp                    ;no
010225 003053 4             jmp hichew            ;yes, flush it
010226 004015 4           lda minus2              ;raw packet flag
010227 003315 4           jmp hiCsto              ;do pkt input, type=-2

;Here from below when we've input the raw message. Compute where the next
;available word is in the buffer.   X=HIP coming here, for hcnf.

010230 001001 4   hi1raw: .inh all                ;see if X25 data length
010231 121661 4 0         jst [hcnf] i            ;returns host config block ir
010232 000401 4 0         .enb h2i
010233 044000 4           lda cf.htype x          ; type 4=X25
010234 012054 4           era [4]
010235 100040 4           sze
010236 004017 4            lda [ (odata1-1) - (data1-1) ] ; not X25, use odata1
010237 015662 4           add [data1-1]
010240 010001 4           sta ireg                ;save data length in IREG
010241 004247 4           lda hiiocb              ;get ptr to IOCB
010242 010000 4           sta 0                   ;into X
010243 044003 4           lda iocb.size x         ;get size
010244 040474 4           lgr 4.                  ;convert to words, not bits
010245 054004 4           add iocb.addr x         ;plus base address = first av
010246 010002 4           sta jreg                ;save in a temp
010247 004001 4           lda ireg                ;finish size computation
010250 015660 4           add [ (data-1) + 1 ]    ;set up size plus one
010251 054004 4           add iocb.addr x         ;compute address of last word
010252 072245 4           ldx hip                 ;fix index
010253 022002 4           cas jreg                ;compare with first available
010254 121652 4            jst [hieom] i          ;last + 1 > next, check for e
010255 003041 4            jmp hichfl             ;not eom, or last+1=next, bad
010256 144266 4           lda hihii ix
010257 121663 4           jst [htpmt] i           ;count throughput
010260 000020 4           pcb                     ;point to PCB
010261 044045 4           lda h2ipcb.haccom x     ;get haccom
010262 072245 4           ldx hip                 ;and fix index
010263 110002 4           sta jreg i              ;put haccom in next word in b
010264 024002 4           irs jreg                ;bump fill ptr
010265 121664 4           jst [hihstp] i          ;get dest/source host pair
010266 150264 4           sta hihsi ix            ;save in seqh word
010267 121665 4           jst [hipkt] i           ;form raw packet header
010270 144266 4           lda hihii ix
010271 121666 4           jst [htppt] i           ;count a packet of throughput
010272 144252 4           lda hispi ix            ;restore pkt ptr
010273 010000 4           sta 0
010274 003544 4           jmp histat              ;runs STATs, checksum it and

;Here from dispatch mechanism on REG (type 0) messages. Continue filling
```

;in fields for packets.

```
010275 144265 4   hi0:    lda hihhi ix          ;get priority bit from handli
010276 006071 4           ana sign 0&ldrhpr
010277 040475 4           lgr 3 0&prirty        ;move it into right place for
010300 152263 4           era hihti ix
010301 150263 4           sta hihti ix          ;update HIHT (TYPH)

010302 144267 4           lda hihli ix          ;get subtype
010303 006072 4           ana [ldrsty]
010304 022011 4           cas three             ;what subtype is it?
010305 003641 4            jmp hierbc           ;.GT. 3 is illegal
010306 003214 4            jmp hiraw            ;.EQ. 3 is RAW
010307 100040 4           sze 0&lcnorm          ;is it 0,1, or 2?
010310 003641 4            jmp hierbc           ;type 1,2 illegal

;Here when its a subtype 0 message. Make sure there's more than just
;a leader. Then force the HITY word to zero indicating non-raw packet.

010311 121652 4           jst [hieom] i
010312 100000 4            skp                  ;if not EOM, fine
010313 003641 4             jmp hierbc          ;if EOM, subtype 0 leader w/

010314 140040 4           cra                   ;fall into HIOSTO to put 0 in
010315 150261 4   hi0sto: sta hityi ix          ;set HITY to AC (entered from
010316 144266 4           lda hihii ix          ;get destination IMP number
010317 121667 4           jst [impchk] i        ;check it out
010320 003327 4            jmp hi0err           ;bad IMP number, handle the e
010321 144266 4           lda hihii ix          ;get destination IMP
010322 010000 4           sta 0                 ;into X
010323 144131 4           lda spfrti ix         ;get routing entry
010324 072245 4           ldx hip               ;fix index...
010325 101400 4           smi 0&spfded          ;is it reachable?
010326 003335 4            jmp hirch            ;yes, skip Dest Dead

;The destination is unreachable (or the destination consists of a bad
;IMP number -see HIOSTO+few). If the type is
;RAW (HITY=-2), then we go off to HICHFL (chew), otherwise, we go to
;HIERDD.

010327 144261 4   hi0err: lda hityi ix          ;get this packet's type
010330 100400 4           spl                   ;raw?
010331 003053 4            jmp hichew           ;yes - just chew it up
010332 103670 4           jmp [hierdd] i        ;no  - issue destination dead

;We have now copied all the relevant info from the leader into the host
;variables (HIHI, etc.). Call TRACE if necessary then go get the first packet.


010333 000000 4   jsth2i: 0
010334 103333 4           jmp jsth2i i

010335 120146 4   hirch:  jst trace.hitrch i    ;** trace
010336 004014 4           lda [ -1 ]            ;set HIFP non-zero for first
010337 150270 4           sta hifpi ix

                          ;fall into next page
```

```
                        ;from previous page

;Here when we expect the first packet next. First we get a buffer to
;read the packet into. Note that we are willing to wait forever.

010340 003343 4           jmp hipakt              ;first time, skip over loop
010341 000401 4  hi1bfw:  .enb h2i                ;for .inh below
010342 121671 4           jst [hiwait] i
010343 004053 4  hipakt:  lda two                 ;try to get a buffer, p=2
010344 001001 4           .inh fre
010345 120116 4 0         jst getfri i
010346 003341 4 0          jmp hi1bfw             ;need to wait for it (we are
010347 000401 4 0         .enb h2i                ;have the packet

010350 004000 4           lda 0                   ;get pointer to packet into A
010351 072245 4           ldx hip                 ;fix index
010352 150252 4           sta hispi ix            ;set pointer to it
                          ;defhlt(55. HI1BFW: long wait for buffer)
010353 121672 4           jst [hiwtrp] i          ;check for a long enough wait

;Now offset the input based on the value of HILT (0 to -9) for padding if
;this is the first packet, else don't.

010354 121673 4           jst [hideq] i           ;get at the IOCB
010355 072245 4           ldx hip                 ;see if X25 data size
010356 001001 4           .inh all
010357 121661 4 0         jst [hcnf] i            ;returns host config block in
010360 000401 4 0         .enb h2i
010361 044000 4           lda cf.htype x          ; type 4=X25
010362 012054 4           era [4]
010363 100040 4           sze
010364 004023 4            lda [ (odata1 * 16.) - (data1 * 16.) ] ; not X25, us
010365 015674 4           add [data1 * 16.]
010366 010001 4           sta ireg                ;save bit count for below
010367 072245 4           ldx hip                 ;fix X
010370 144270 4           lda hifpi ix            ;is this the first packet?
010371 101040 4           snz
010372 003401 4            jmp hipak2             ;no, then no padding
010373 144253 4           lda hilti ix            ;yes, get # padding words
010374 041474 4           lgl 4.                  ;convert to bits
010375 014001 4           add ireg
010376 010001 4           sta ireg                ;add to computed bit count
010377 144253 4           lda hilti ix            ;also, recede starting addres
010400 140407 4           tca                     ;by number of padding words
010401 154252 4  hipak2:  add hispi ix            ;from start of packet
010402 015660 4           add [data]              ;compute displacement into IN
010403 072247 4           ldx hiiocb              ;get ptr to IOCB
010404 050004 4           sta iocb.addr x         ;set up address
010405 004001 4           lda ireg                ;set up size (bits)
010406 050003 4           sta iocb.size x         ; (HIENQ sets IOFLAGS.ALWAYS
010407 121675 4           jst [hienq] i           ;enq and PDV the device

;Then compute the time to read the input, TPR and debreak waiting for it.
;Index is still trash.

010410 005676 4  hirpkt:  lda [ %15sec ]
010411 000020 4           pcb
010412 000203 4           tpr
010413 000103 4           spr                     ;debreak
```

```
010414 103677 4          jmp [hito] i          ;if timed out, mark as inc or
010415 121646 4          jst [hiwspr] i        ;return from debreak
010416 121655 4          jst [hirchk] i        ;if HIRS becomes non-zero, re
010417 000010 4          %rdclok               ;save the input time
010420 150257 4          sta hiiti ix

;Now decipher RAW-ness and size of packet (single packet message, multi-
;packet message).

010421 144261 4          lda hityi ix          ;is this RAW?
010422 100400 4          spl
010423 003230 4           jmp hi1raw           ;if so, handle differently
010424 144270 4          lda hifpi ix          ;is this a middle or last pac
010425 101040 4          snz
010426 003465 4           jmp himdls           ;yes, skip getting messno, et

010427 121652 4          jst [hieom] i         ;must be first packet - singl
010430 103700 4           jmp [himult] i       ;multi-packet message - get A

                         ;fall into HISING on next page
```

```
                        ;from previous page only (HISING for documentation on

010431 164261 4   hising: irs hityi ix              ;single set HITY=1 (was 0) fo
010432 072247 4           ldx hiiocb                ;make sure padding completed

010433 044003 4           lda iocb.size x
010434 040474 4           lgr 4.                    ;convert to words
010435 072245 4           ldx hip                   ;fix index
010436 156253 4           sub hilti ix              ;take out padding from count

010437 100400 4           spl                       ;must .GE. than padding
010440 003642 4            jmp hiersh               ;too short otherwise

;Given a well-formed single packet message, try to allocate a message
;number. If one is not available, wait.

010441 003444 4           jmp hi1gmz                ;skip over wait
010442 121671 4   hi1gmw: jst [hiwait] i
                        ;defhlt(30. HI1GMW: host blocked waiting messno for s
010443 121701 4           jst [hiwblk] i
010444 121702 4   hi1gmz: jst [hiblgt] i            ;get a message number
010445 003442 4            jmp hi1gmw               ;can't, wait for one
010446 012052 4           era one 0&reqcod&mltpkt&lstpkt ; REQ1 in pkt type
010447 150262 4           sta hihpi ix
                        ;defhlt(57. HI1GMW: long wait messno for single packe
010450 121672 4           jst [hiwtrp] i
010451 121665 4           jst [hipkt] i             ;set up other packet fields

;Now get a TSB to hold it, up the use count of the buffer (because it
;will also be pointed to by the TSB).

010452 003454 4           jmp hitsbz
010453 121671 4   hitsbw: jst [hiwait] i
010454 004052 4   hitsbz: lda one                   ;leave 1 TSB when done
010455 121703 4           jst [tsbput] i
010456 003453 4            jmp hitsbw               ;can't do that - wait for it

010457 150255 4           sta hitbi ix              ;save pointer to TSB in HITB(
                        ;defhlt(60. HITSBW: long wait for REQ1 TSB)
010460 121672 4           jst [hiwtrp] i
010461 144252 4           lda hispi ix              ;get pointer to buffer
010462 010000 4           sta 0                     ;(into X)
010463 064004 4           irs wrdc x 0&usecnt
010464 072245 4           ldx hip                   ;count message of throughput

;Count this message of throughput and pass it off to TASK. It is also
;at this point we check for an error in the input. The reason the check
;is simple: the host code isn't particularly well-suited to handle
;error conditions. Instead, it treats erroneous packets as normal
;packets except that it changes the subtype in the TSB to be INCCOD
;to cause I2H to tell the host.

;HIMESS and HIMDLS (MiDdle/LaSt) are where all types of packets come together
;(single packet messages, first of multi go to HIMESS, middle and last of
;multi packet messages go to HIMDLS).

010465 121665 4   himdls: jst [hipkt] i             ;fill in the fields of the pa
```

```
010466 003471 4             jmp hims2                   ;and skip over message counti

010467 144266 4   himess: lda hihii ix                  ;(enter from HI8 code below,
010470 121663 4           jst [htpmt] i
```

```
010471 072247 4   hims2:  ldx hiiocb              ;get ptr to IOCB
010472 044005 4           lda iocb.flags x        ;get the flags
010473 006072 4           ana [ %ioflags.cc ]     ;look at the completion code

010474 101040 4           snz                     ;non-zero (error)?
010475 003522 4            jmp hinoer             ;noerr if zero

;When transfer had an error, change the subtype of the TSB we just queued
;and mark an error as having occured so we discard the rest of the message
;after queuing this piece to TASK, below.

010476 004054 4           lda four 0&cerror       ;fall into HISUBC with AC=er
010477 072245 4   hisubc: ldx hip                 ;(enter here to set up subtyp
010500 010001 4           sta ireg                ;save caller's error code
010501 144255 4           lda hitbi ix            ;get pointer to TSB
010502 026001 4           ima ireg                ;into IREG, get error code ba
010503 112001 4           era ireg i              ;OR in new part of sub-code
010504 006072 4           ana [subtyp]
010505 112001 4           era ireg i
010506 110001 4           sta ireg i              ;(first word of TSB)
010507 004014 4           lda minus1              ;set ptr to TSB to -1 indicat
010510 150255 4           sta hitbi ix
010511 144252 4           lda hispi ix            ;get ptr to packet
010512 010000 4           sta 0
010513 044013 4           lda pkth x              ;mark it as incomplete
010514 012011 4           era three 0&inccod&msgcod
010515 050013 4           sta pkth x
010516 121704 4           jst [hicksm] i          ;checksum it
010517 003516 4            jmp .-1                ;"r1 happens here only for er
010520 003552 4           jmp hi2tsk              ;give it to TASK
```

```
                    stats.hs0i:
010521 010333 4             jsth2i

;Here when the transfer did not have an error.

010522 072245 4   hinoer: ldx hip
010523 144266 4           lda hihii ix          ;count it as throughput
010524 121666 4           jst [htppt] i
010525 121652 4           jst [hieom] i         ;EOM?
010526 100000 4            skp                  ;no
010527 003537 4              jmp hilast         ;yes, set LSTPKT in packet
010530 144262 4           lda hihpi ix          ;get packet number
010531 007705 4           ana [160] 0&pktnum
010532 013705 4           era [160] 0&pktnum
010533 100040 4           sze                   ;is it number 7?
010534 003546 4            jmp hitskc           ;no, checksum it and pass it
010535 004052 4           lda one 0&clong
010536 003477 4           jmp hisubc            ;give it an error code

;Here to set LSTPKT when the transfer had EOM set, checksub buffer and
;pass it to TASK.

010537 144252 4   hilast: lda hispi ix          ;get ptr to buffer
010540 010000 4           sta 0
010541 044013 4           lda pkth x            ;turn on LSTPKT
010542 012070 4           era [lstpkt]
010543 050013 4           sta pkth x

;Here to call STATs, checksum buffer and pass it to TASK

010544 121521 4   histat: jst stats.hs0i i      ;**cumstats
010545 072245 4           ldx hip               ;fix index

;Here to checksum buffer, then pass it to TASK.

010546 144252 4   hitskc: lda hispi ix          ;get ptr to it
010547 010000 4           sta 0
010550 121704 4           jst [hicksm] i
010551 003634 4            jmp hierdm           ;if wrong length, give host C

                          ;fall into HI2TSK
```

```
                        ;from previous page

;Finally, here to queue it to TASK

010552 072245 4   hi2tsk: ldx hip
010553 144252 4           lda hispi ix
010554 010000 4           sta 0                    ;get X=pkt pointer
010555 000010 4           %rdclok                  ;set arrival time
010556 050003 4           sta artm x
010557 001001 4           .inh m2i
01056C 105706 4 1         lda [etq] i
010561 010001 4 1         sta ireg
010562 132001 4 1         stx ireg i
010563 133706 4 1         stx [etq] i              ;pass it to TASK
010564 072245 4 1         ldx hip                  ;fix index

;In pre-4300 IMPs, TASK used to reach back and incremented HILO (the host
;code left-off address) for those hosts which were "blocked" waiting for
;TASK to take a packet. Since there is no HILO and it is very ugly to have
;TASK reach out of its context to cause someone else to skip, we have
;eliminated that mechanism. Instead, we watch TSKFLG(host) which TASK
;maintains. TASK GPR's us when it changes TSKFLG. TSKFLG=1 means TASK
;refuses the packet, TSKFLG=2 means it accepted it.

010565 140040 4 1         cra                      ;start TSKFLG at zero
010566 150306 4 1         sta tskfli ix
010567 000401 4 1         .enb h2i                 ;now enable
01057C 072136 4           ldx tskpci
010571 000043 4           gpr                      ;poke TASK
010572 000103 4           spr                      ;and wait for GPR from TASK
010573 000000 4            %crash                  ;HI2TSK: NMFS timeout
010574 121646 4           jst [hiwspr] i           ;return from debreak

;Now examine TSKFLG to see what TASK had to say about the packet.

010575 144306 4           lda tskfli ix
010576 101040 4           snz
010577 000000 4            %crash                  ;HI2TSK: awoke but TSKFLG stil
01060C 101100 4           sln                      ;check low bit (TSKFLG is 1 o
010601 003612 4            jmp hitook              ;TASK took it

;Here when TASK refuses the packet. If it's been 15 seconds AND the packet
;is RAW, discard it, else keep trying.

010602 144254 4           lda hiwti ix             ;get time
010603 015707 4           add [ 600. ]             ;plus 15 seconds
010604 016151 4           sub time
010605 146261 4           ana hityi ix             ;HITY is a small neg number w
010606 100400 4           spl                      ; HIWT+15 - TIME negative if
010607 003041 4            jmp hichfl              ;RAW and older than 15. secon
01061C 121671 4   hi2ts2: jst [hiwait] i           ;sleep for a fast tick
010611 003552 4           jmp hi2tsk               ;try again

;TASK has accepted the packet. Free up re-assembly if REQ8 or RAW
;(low bit of HITY is zero). Then check to see if the transfer had
;an error (TSB ptr in HITB is -1). If so, discard until EOM.

                  hitook:
                        ;defhlt(65. HI2TSK/HITOOK: long wait for TASK to take
```

```
010612 121672 4           jst [hiwtrp] i
010613 144261 4           lda hityi ix              ;get packet type
010614 101100 4           sln                       ;RAW or part of multi-packet
010615 024224 4            irs nres                 ;yes, give back reassembly bu
010616 100400 4           spl                       ;RAW?
010617 003624 4            jmp hituk2               ;yes, skip check for TSB erro
010620 144255 4           lda hitbi ix              ;get ptr to TSB
010621 012014 4           era minus1                ;is it -1?
010622 101040 4           snz
010623 003053 4            jmp hichew               ;yes, chew up the rest of the

;Here after a packet has been sent to TASK.  Figure out whether there's more
;to this message or not, jumping off to HIPAKT or HILEAD as appropriate.

010624 140040 4   hituk2: cra                       ;clear HIFP because it will b
010625 150270 4           sta hifpi ix              ; ... HILEAD
010626 004056 4           lda [pktnm1]              ;update the packet counter in
010627 154262 4           add hihpi ix              ; ... HIPAKT
010630 150262 4           sta hihpi ix
010631 121652 4           jst [hieom] i             ;eom?
010632 003343 4            jmp hipakt               ;no - get another packet
010633 003064 4           jmp hilead                ;yes - get another message (
```

```
;Here from above to give host CDMCER message unless it's a RAW message

010634 144261 4   hierdm: lda hityi ix
010635 100400 4          spl
010636 003041 4           jmp hichfl            ;if RAW, just eat it up
010637 004012 4          lda [cdmcer]           ;the old "DMC" error
010640 003477 4          jmp hisubc

;here from all over to issue various error messages to the host.

010641 164267 4   hierbc: irs hihli ix 0&cillgl   ;bad code
010642 164267 4   hiersh: irs hihli ix 0&cshort   ;too short
010643 004062 4   hier32: lda [cerrld]
010644 103710 4          jmp [hilput] i
```

```
                .section pg32
                .lev h2i

;Here when the first packet we just read in DOES NOT have EOM set, indicating
;it's the first of a multi-packet message. First, allocate a message number.

032000 003003 4   himult: jmp hi8gmz
032001 121317 4   hi8gmw: jst [hiwait] i
                          ;defhlt(31. HI8GMW: host blocked waiting for messno f
032002 121320 4           jst [hiwblk] i
032003 121321 4   hi8gmz: jst [hiblgt] i          ;get the message number
032004 003001 4            jmp hi8gmw             ;can't - wait and try again
032005 012071 4           era [100000] 0&mltpkt&lstpkt&msgcod ;pick up mlt-pkt
032006 150262 4           sta hihpi ix           ;set up type
                          ;defhlt(59. HI8GMW: long wait for messno for REQ8/MES
032007 121322 4           jst [hiwtrp] i

;Now see if there's an ALLOC already available for this MSG8.

032010 144260 4           lda hibli ix           ;get block pointer
032011 010000 4           sta 0                  ;(into X)
032012 121323 4           jst [getall] i         ;look for an ALL8
032013 003026 4            jmp hinoal            ;no ALL8 avail, turn this MSG
032014 072245 4   hiplta: ldx hip                ;fix index (here from bottom
032015 121324 4           jst [hipkt] i          ;fill in packet fields

;Now get a TSB to hold MSG8 and queue it.

032016 003020 4           jmp hi8ggz
032017 121317 4   hi8ggw: jst [hiwait] i
032020 004053 4   hi8ggz: lda [2]                ;get a TSB, p=2
032021 121325 4           jst [tsbput] i
032022 003017 4            jmp hi8ggw            ;can't, wait and try again
032023 150255 4           sta hitbi ix           ;remember pointer to the trar
                          ;defhlt(60. HI8GGW: long wait for TSB for MESS8)
032024 121322 4           jst [hiwtrp] i
032025 103326 4           jmp [himess] i         ;then merge with main-line ac
```

```
;Here when there was no ALL8 available. Get a TSB and queue a REQ8.

032026 072245 4   hinoal: ldx hip                    ;fix index
032027 164262 4          irs hihpi ix 0&reqcod&msgcod ;mark as REQ8
032030 121324 4          jst [hipkt] i               ;copy header
032031 003033 4          jmp hi8gqz
032032 121317 4   hi8gqw: jst [hiwait] i
032033 004053 4   hi8gqz: lda [2]                    ; p=2
032034 121325 4          jst [tsbput] i
032035 003032 4           jmp hi8gqw                 ;can't, wait for one
                         ;defhlt(61. HI8GQW: long wait for TSB to queue a REQ8
032036 121322 4          jst [hiwtrp] i

;Now we turn the buffer into a REQ8 and give it to TASK.  Note that this
;smashes the word count, but that's okay because we'll re-compute it from
;the IOCB later (in HIPKT).

032037 144252 4          lda hispi ix                ;get ptr to buffer (redundant
032040 010000 4          sta 0
032041 044004 4          lda wrdc x                  ;get size + use count
032042 141050 4          cal                         ;clear up left half
032043 015327 4          add [ (8.<<8.) + 1 ]        ;set length = 8., up use cou
032044 050004 4          sta wrdc x
032045 121330 4          jst [hicksm] i              ;checksum it
032046 003045 4           jmp .-1                    ;"r1" can't happen twice, oka
032047 003051 4          jmp hi8tsk                  ;submit packet to task; skip

032050 121317 4   hi8ts2: jst [hiwait] i             ;wait after each task refusal
032051 072245 4   hi8tsk: ldx hip                    ;get host number
032052 144252 4          lda hispi ix                ;get packet pointer
032053 010000 4          sta 0                       ;this code needed when TASK r
032054 000010 4          %rdclok
032055 050003 4          sta artm x                  ;set up the arrival time
032056 001001 4          .inh m2i                    ;pass it to task
032057 105331 4 1        lda [etq] i
032060 010001 4 1        sta ireg
032061 132001 4 1        stx ireg i
032062 133331 4 1        stx [etq] i
032063 072245 4 1        ldx hip                     ;fix index
032064 140040 4 1        cra                         ;set TSKFLG to zero
032065 150306 4 1        sta tskfli ix
032066 000401 4 1        .enb h2i
032067 072136 4          ldx tskpci                  ;go an poke TASK
032070 000043 4          gpr
032071 000103 4          spr                         ;then debreak, awaiting a pok
032072 000000 4           %crash                     ;HI8TSK: NMFS timeout
032073 121332 4          jst [hiwspr] i              ;return from debreak

;Having given it to task, check TSKFLG to see how it liked it.

032074 144306 4          lda tskfli ix               ;get TASK's answer
032075 101040 4          snz
032076 000000 4           %crash                     ;HI8TSK: Awoke but TSKFLG=0
032077 100100 4          slz                         ;check for 1 or 2
032100 003050 4           jmp hi8ts2                 ;if 1, TASK rejected - try a
                         ;defhlt(62. HI8TSK: long wait for TASK to take REQ8)

032101 121322 4          jst [hiwtrp] i
```

```
;TASK accepted it, look for the ALL8 again.
```

```
032102 003105 4            jmp hi8g8z
032103 121317 4   hi8g8w:  jst [hiwait] i
                           ;defhlt(47. HI8G8W: host block waiting for REQ8)
032104 121320 4            jst [hiwblk] i
032105 144260 4   hi8g8z:  lda hibli ix              ;get block pointer
032106 010000 4            sta 0
032107 121323 4            jst [getall] i           ;try to find the ALL8
032110 003103 4             jmp hi8g8w              ;if can't, wait and try again

                           ;fall into next page
```

```
                        ;from previous page

;Having gotten the ALL8, we do something else perverse to the packet (??).
;This code seems to be concerned with waiting for the packet to be acked
;or something. Beats me.

032111 100000 4           skp
032112 121317 4  hi8tkw: jst [hiwait] i
032113 072245 4          ldx hip                ;restore index
032114 144252 4          lda hispi ix
032115 010000 4          sta 0
032116 044004 4          lda wrdc x
032117 101100 4          sln 0&usecnt
032120 003112 4           jmp hi8tkw            ;pkt still on line, so wait
032121 072245 4          ldx hip                ;fix index again
                         ;defhlt(63. HI8TKW: long wait for ALL8)
032122 121322 4          jst [hiwtrp] i         ;check for long wait

;now get a message number for the MSG8 and save it in the header, then
;change the messgae type back to MSG8 (from REQ8)?

032123 003126 4          jmp hi8g2z
032124 121317 4  hi8g2w: jst [hiwait] i
                         ;defhlt(48. HI8G2W: host blocked waiting for messno f
032125 121320 4          jst [hiwblk] i
032126 144260 4  hi8g2z: lda hibli ix           ;get block number
032127 010000 4          sta 0
032130 121333 4          jst [mesget] i         ;get the message number
032131 003124 4           jmp hi8g2w            ;if in use, wait and try late
032132 072245 4          ldx hip                ;fix index
032133 150264 4          sta hihsi ix           ;save msg number in header
                         ;defhlt(64. HI8G2W: long wait for messno for MESS8)
032134 121322 4          jst [hiwtrp] i
032135 144262 4          lda hihpi ix 0&pktcod  ;fix type
032136 012052 4          era [1] 0&msgcod&reqcod
032137 150262 4          sta hihpi ix
032140 003014 4          jmp hiplta             ;and start again
```

```
;Here when a first (HIFP=1)  packet or middle/last (HIFP=0) packet times out/
;If it's raw, just ignore it.  If it's a first packet, send back a CSLOWS
;incomplete error.  If it's part of a multi-packet message, set the incomplete

;flag in the packet header so the destination will know this is a broken
;message.

032141 121332 4   hito:   jst [hiwspr] i          ;return from SPR we came from
032142 121334 4           jst [hixfls] i          ;use XDV to flush interface
032143 144270 4           lda hifpi ix            ;is this a first packet?
032144 101040 4           snz
032145 003163 4            jmp hi8to              ;no, handle differently

032146 001001 4   hi1to: .inh all
                                ;defhlt(29. HI1TO: first packet timed out)
032147 120120 4 0         jst hltjst i
032150 000401 4 0         .enb h2i
032151 144261 4           lda hityi ix            ;get message type
032152 100400 4           spl
032153 103335 4            jmp [hichfl] i         ;if raw, just chew it up
032154 004053 4           lda [cslows]            ;else, set up sub-type of CIN
032155 152267 4           era hihli ix
032156 006072 4           ana [ldrsty]
032157 152267 4           era hihli ix
032160 150267 4           sta hihli ix
032161 005336 4           lda [cinctr]            ;get error message code
032162 103337 4           jmp [hilput] i          ;and queue it for the host

032163 001001 4   hi8to:  .inh all
                                ;defhlt(32. HI8TO: middle packet timed out)
032164 120120 4 0         jst hltjst i
032165 000401 4 0         .enb h2i
032166 004053 4           lda [cslows]
032167 103340 4           jmp [hisubc] i          ;set packet incomplete and co
```

```
;These two subroutines are concerned with watching out for long waits
;and host blocked conditions.  A long wait is simply a trap that is sent
;off to the NCC if H2I has been trying to do a given something for longer
;than 3 seconds.  A host blocked is much worse - it happens when H2I
;hasn't made progress in 15 seconds.  In addition, a host can become
;blocked if H2I (here) detects the "HIBG" (for Block Gone) flag set.
;HIBG is set by RESETT when the connection is reset while H2I is in the
;process of working on a message for that connection.

032170 000000 4   hiwtrp: 0                       ;enter here to maybe cause a
032171 004151 4           lda time                ;get current time
032172 166254 4           ima hiwti ix            ;has this interval been long?
032173 156254 4           sub hiwti ix
032174 015341 4           add [ 120. ]            ;(over 3 seconds, fast tics)

032175 101400 4           smi
032176 103170 4            jmp hiwtrp i
032177 005170 4           lda hiwtrp
032200 021202 4           jst hitrap              ;long time, give trap
032201 103170 4           jmp hiwtrp i

032202 000000 4   hitrap: 0
032203 011211 4           sta hiwtpc
032204 144265 4           lda hihhi ix            ;dest host + dest imp in A
032205 141240 4           icr 0&ldrhst
032206 152266 4           era hihii ix
032207 001001 4           .inh all
032210 100000 4 0         skp
032211 000000 4 0 hiwtpc:  0                      ;gets arg from AC
032212 120126 4 0         jst hltncc i
032213 000401 4 0         .enb h2i
032214 103202 4           jmp hitrap i            ;return to caller

032215 000000 4   hiwblk: 0                       ;enter here to maybe cause ho
032216 121342 4           jst [hirchk] i          ;if ready line goes away, pun
032217 005343 4           lda [ 600. ]            ;15 seconds
032220 154254 4           add hiwti ix            ;plus when this all started
032221 016151 4           sub time                ;  ... better still be in the
032222 100400 4           spl
032223 003236 4            jmp hierbl             ;if not, host blocked for sur

;although the host still has more time left before being blocked, we make
;sure that the message block hasn't been reset in the process.  This is
;done by looking at HIBG (block gone), which is set by timeout in RESETT
;when the message block is deleted while the host code is working on it.
;In addition, we check to be sure the destination is still reachable.

032224 144266 4           lda hihii ix            ;get destination IMP
032225 010000 4           sta 0                   ;into X
032226 144131 4           lda spfrti ix           ;get routing entry
032227 072245 4           ldx hip                 ;fix index...
032230 100400 4           spl 0&spfded
032231 103344 4            jmp [hierdd] i         ;unreachable, (dd=destination
032232 140040 4           cra
032233 166271 4           ima hibgi ix            ;did block go away?
032234 101040 4           snz
032235 103215 4            jmp hiwblk i           ;no, return to caller

                          ;fall into HIERBL
```

```
                        ;from previous page, too

032236 005215 4   hierbl: lda hiwblk              ;get caller's PC for HITRAP
032237 021202 4           jst hitrap
032240 004054 4           lda [4] 0&cblock
032241 152267 4           era hihli ix
032242 006072 4           ana [ldrsty]
032243 152267 4           era hihli ix
032244 150267 4           sta hihli ix
032245 005336 4           lda [cinctr]
032246 103337 4           jmp [hilput] i

;called to do PDVs
;come here and return with x=pcb
                  .lev h2i
032247 000000 4   h2ipdv: 0
032250 044022 4           lda pcb.iocb x          ;check if i/o already going
032251 100040 4           sze
032252 103247 4            jmp h2ipdv i           ;if so, just return
032253 044005 4           lda pcb.type x          ;else, check if sw host
032254 100040 4           sze
032255 003275 4            jmp h2ipd2             ;if real host, do real PDV
032256 044023 4           lda pcb.get x           ;else, simulate PDV ucode
032257 000022 4           deq                     ;first, take an iocb off the
032260 000000 4            hlt
032261 032001 4           stx ireg
032262 000020 4           pcb
032263 004001 4           lda ireg
032264 050022 4           sta pcb.iocb x          ;and place it in pcb.iocb
032265 044046 4           lda h2ipcb.swpdv x
032266 000020 4           pcb                     ;set x=pcb in case we return

032267 101040 4           snz
032270 103247 4            jmp h2ipdv i
032271 010000 4           sta 0
032272 000043 4           gpr                     ;then poke the sw host PCB
032273 000020 4           pcb                     ;restore x=pcb
032274 103247 4           jmp h2ipdv i            ;and return

032275 001003 4   h2ipd2: pdv                     ;real host, do real PDV
032276 103247 4           jmp h2ipdv i            ;and return

;called to perform XDVs
;come here and return with x=pcb
032277 000000 4   h2ixdv: 0
032300 010001 4           sta ireg                ;save XDV code
032301 044005 4           lda pcb.type x          ;check if sw host
032302 100040 4           sze
032303 003314 4            jmp h2ixd2             ;real host, do real xdv
032304 044047 4           lda h2ipcb.swxdv x      ;sw host
032305 101040 4           snz                     ;swxdv routine exist?
032306 103277 4            jmp h2ixdv i           ;no, just return
032307 010002 4           sta jreg                ;yes, jst to it
032310 004001 4           lda ireg                ;with a=XDV code
032311 120002 4           jst jreg i
032312 000020 4           pcb                     ;restore x=pcb
032313 103277 4           jmp h2ixdv i

032314 004001 4   h2ixd2: lda ireg                ;regular host, do real XDV
```

```
032316 103277 4          jmp h2ixdv i
```

```
                .section pg11
                .lev h2i
;Subroutine to do NMFS-related H2I initialization. This is a subroutine only
;for clarity (it has only one caller, near H2IINI). We set up the get and
;put queue pointers and headers, then create the IOCB (and put it on the
;PUT queue).

011000 000000 4   h2inmf: 0
011001 000020 4           pcb                          ;get PCB pointer
011002 004000 4           lda 0                        ;get base
011003 015744 4           add [ h2ipcb.getqh ]         ;get ptr to queue header
011004 050023 4           sta pcb.get x
011005 021017 4           jst h2inmq                   ;initialize the queue header,
011006 014054 4           add [ h2ipcb.putqh - h2ipcb.getqh ] ;to PUT queue, no
011007 050024 4           sta pcb.put x
011010 021017 4           jst h2inmq                   ;initialize the queue header,
011011 014054 4           add [ h2ipcb.iocb - h2ipcb.putqh ] ;to IOCB
011012 026000 4           ima 0                        ;put IOCB ptr in X, PCB in A

011013 015745 4           add [ h2ipcb.putqh ]         ;compute address of put queue
011014 000002 4           enq                          ;equivelant to LDA PCB.PUT X,
011015 103000 4            jmp h2inmf i                ;return
011016 000000 4             %crash                     ;H2INMF: found extra IOCB

;Subroutine (called only from H2INMF) to init a queue header pointed to by A.
;Preserves A, X.

011017 000000 4   h2inmq: 0
011020 032001 4           stx ireg                     ;save caller's Index
011021 010002 4           sta jreg                     ;  ... and AC
011022 010000 4           sta 0                        ;put address of queue in X
011023 050000 4           sta q.forw x                 ;point it at self
011024 050001 4           sta q.back x
011025 050002 4           sta q.hedr x
011026 140040 4           cra                          ;length = 0, too
011027 050003 4           sta q.size x
011030 072001 4           ldx ireg                     ;fix index
011031 004002 4           lda jreg                     ;and AC
011032 103017 4           jmp h2inmq i                 ;return
```

```
;The code here is used to debreak for a short time (one fast tick) and
;then try again for a resource.

011033 000000 4   hiwait: 0
011034 005033 4           lda hiwait              ;pick up entry point
011035 010001 4           sta ireg                ;save it
011036 072245 4           ldx hip                 ;fix X for setting HIRW flag
;&&     sta hirwi ix             ;set HIRW to non-zero
011037 000020 4           pcb
01104C 004056 4           lda [ %25.6ms ]         ;get one fast tick
011041 000203 4           tpr
011042 000103 4           spr
011043 101000 4            nop
011044 021055 4           jst hiwspr
011045 021133 4           jst hirchk              ;check to see if someone want
011046 102001 4           jmp ireg i              ;return to caller

;HIXFLS does an XDV of %C18.RESET on the host.  Called from HI1TO and
;HIR.

011047 000000 4   hixfls: 0
01105C 000020 4           pcb
011051 004006 4           lda [ %c18.reset ]
011052 121746 4           jst [h2ixdv] i          ;do it
011053 072245 4           ldx hip                 ;restore X
011054 103047 4           jmp hixfls i            ;return
```

```
;Call HIWSPR after de-breaking to NMFS via the SPR instruction. We
;re-compute HIP (which is left in X for caller) and HILP here. Also
;set HIIOCB to the address of the IOCB at the top of the PUT queue,
;or to 0 if none.  For non-i/o completion wakeups, use the similar
;routine hiwsp2 whcih follows hiwspr.

011055 000000 4   hiwspr: 0
011056 005055 4           lda hiwspr            ;save entry point
011057 000020 4           pcb                   ;get ptr to ourselves
011060 000043 4           gpr                   ;clear repoked bit by setting
011061 000103 4           spr                   ;and clearing it
011062 000000 4            %crash               ;HIWSPR: timed out
011063 011055 4           sta hiwspr            ;restore entry point
011064 044025 4           lda pcb.num x         ;get host number
011065 010245 4           sta hip               ;set up HIP
011066 044024 4           lda pcb.put x         ;get ptr to PUT queue header

011067 010000 4           sta 0
011070 044000 4           lda q.forw x          ;get ptr to top of PUT queue

011071 012000 4           era 0                 ;empty? (i.e. q.forw=q.hedr)?
011072 100040 4           sze                   ;yes - leave at zero
011073 012000 4            era 0                ;no, fix it
011074 010247 4           sta hiiocb
011075 021114 4           jst hiihlp            ;compute HILP, set X=HIP too

011076 103055 4           jmp hiwspr i          ;return

011077 000000 4   hiwsp2: 0
011100 000020 4           pcb
011101 044025 4           lda pcb.num x         ;get host number
011102 010245 4           sta hip               ;set up HIP
011103 044024 4           lda pcb.put x         ;get ptr to PUT queue header

011104 010000 4           sta 0
011105 044000 4           lda q.forw x          ;get ptr to top of PUT queue

011106 012000 4           era 0                 ;empty? (i.e. q.forw=q.hedr)?
011107 100040 4           sze                   ;yes - leave at zero
011110 012000 4            era 0                ;no, fix it
011111 010247 4           sta hiiocb
011112 021114 4           jst hiihlp            ;compute HILP, set X=HIP too

011113 103077 4           jmp hiwsp2 i          ;return

011114 000000 4   hiihlp: 0                     ;subr to compute HILP
011115 004245 4           lda hip
011116 041476 4           lgl 2.                ;compute HIP*4
011117 014245 4           add hip               ;*5
011120 014245 4           add hip               ;*6 = size of a leader
011121 015747 4           add [hildra]          ;plus base of leader area
011122 010246 4           sta hilp              ;set up leader pointer
011123 072245 4           ldx hip               ;return with HIP in X
011124 103114 4           jmp hiihlp i          ;return to caller

;HI640 sets the TPR debreak time to 640 ms

011125 000000 4   hi640:  0
011126 005750 4           lda [ %640ms ]
```

```
011127 000020 4              pcb
011130 054025 4              add pcb.num x
011131 000203 4              tpr
011132 103125 4              jmp hi640 i

;HIRCHK will return iff HIRS is zero. If HIRS<>0, it clears it and goes
```

```
;to HIR.

011133 000000 4   hirchk: 0
011134 140040 4           cra
011135 166251 4           ima hirsi ix          ;is HIRS zero?
011136 101040 4           snz
011137 103133 4            jmp hirchk i         ;yes - return
01114C 103751 4           jmp [hir] i           ;no, start resetting

;HIFAKT will skip if IHP is a real host, non-skip means fake.
;Preserves everything.

011141 000000 4   hifakt: 0
011142 026245 4           ima hip               ;get host number, save old AC
011143 016056 4           sub [nh]              ;compare with lowest numbered
011144 100400 4           spl
011145 025141 4            irs hifakt           ;if HIP .LT. lowest fake, mus
011146 014056 4           add [nh]              ;fix AC
011147 026245 4           ima hip               ;and restore IHP, getting old
01115C 103141 4           jmp hifakt i          ;return (maybe skip)

;HIEOM will skip if the current IOCB has %C18.EOM set.

011151 000000 4   hieom:  0
011152 004247 4           lda hiiocb            ;make sure caller is not con
011153 101040 4           snz
011154 000000 4            %crash              ;HIEOM: called without an IOCB
011155 010000 4           sta 0
011156 044005 4           lda iocb.flags x      ;get the flags
011157 006061 4           ana [ %c18.eom ]      ;check the EOM bit
01116C 100040 4           sze                   ;off?
011161 025151 4            irs hieom            ;no, cause a skip
011162 072245 4           ldx hip               ;fix X
011163 103151 4           jmp hieom i

;HILENQ enq's the leader area in an IOCB for input. Also sets HISP to 1
;indicating leader IO.

011164 000000 4   hilenq: 0
011165 021200 4           jst hideq             ;get IOCB for use, ptr in X a
011166 005752 4           lda [ 16.*6. ]        ;size of a leader
011167 050003 4           sta iocb.size x
01117C 021114 4           jst hiihlp            ;compute HILP into A
011171 072247 4           ldx hiiocb            ;and store that as IOCB.ADDR

011172 050004 4           sta iocb.addr x
011173 021207 4           jst hienq             ;enq this IOCB
011174 072245 4           ldx hip
011175 004052 4           lda [1]
011176 150252 4           sta hispi ix          ;set flag saying we're sendin
011177 103164 4           jmp hilenq i          ;(mostly for HIR) and return


;Subr to get the IOCB back from NMFS's PUT queue. Will crash the IMP if
;it's not there. IREG okay across this routine.

01120C 000000 4   hideq:  0
011201 000020 4           pcb
011202 044024 4           lda pcb.put x         ;get ptr to put queue
```

```
011203 000022 4                 deq                     ;try to get IOCB
011204 000000 4                  %crash                 ;HIDEQ: cant DEQ IOCB
```

```
011205 032247 4           stx hiiocb              ;set ptr to IOCB
011206 103200 4           jmp hideq i             ;and return to caller

;Subr to ENQ the IOCB (in HIIOCB) onto the NMFS GET queue and poke the
;device.

011207 000000 4   hienq:  0
011210 000020 4           pcb
011211 044023 4           lda pcb.get x           ;get ptr to GET qeue
011212 072247 4           ldx hiiocb              ;and to IOCB
011213 066005 4           ima iocb.flags x        ;get the flags word, save the
011214 140500 4           ssm 0&%ioflags.always   ;interrupt when done
011215 066005 4           ima iocb.flags x        ;restore AC
011216 000002 4           enq                     ;give it to microcode
011217 100000 4            skp
011220 000000 4             %crash                ;HIENQ: extra IOCB?
011221 000020 4           pcb                     ;now poke the device
011222 121753 4           jst [h2ipdv] i
011223 103207 4           jmp hienq i             ;and return
```

```
                .lev (h2i,bck)
;transaction block put (insert) subroutine
;expects host # in x (th if back5 calling), 0,1, or 2 in a
;preserves x, skip returns with pointer to lid word in a if
;a free block is found, with info copied into it
011224 000000 4   tsbput: 0
011225 015754 4           add [-ntsb]             ;number of entries-reserve
011226 011321 4           sta tsbpt2
011227 005755 4           lda [tsbtab+1ms]        ;first entry sequence word
011230 011320 4           sta tsbpt1
011231 105320 4   tsbptl: lda tsbpt1 i            ;free?
011232 101040 4           snz
011233 003240 4            jmp tsbptf             ;yes
011234 025320 4           irs tsbpt1
011235 025321 4           irs tsbpt2
011236 003231 4            jmp tsbptl
011237 103224 4           jmp tsbput i            ;none found, no-skip return

011240 033317 4   tsbptf: stx tsbptx             ;save host number
011241 144260 4           lda hibli ix            ;get block ptr
011242 111320 4           sta tsbpt1 i 0&lms      ;save in lms word
011243 144252 4           lda hispi ix            ;get buf ptr
011244 011321 4           sta tsbpt2
011245 014025 4           add [seqh]              ;get ptr to seqh word
011246 027320 4           ima tsbpt1              ;save, get tsb ptr
011247 017756 4           sub [lms]
011250 010000 4           sta 0                   ;into x
011251 000010 4           %rdclok
011252 050000 4           sta lch x               ;**stat time
011253 044044 4           lda lms x               ;get local block no
011254 017757 4           sub [tmblkt]
011255 113320 4           era tsbpt1 i            ;+mess no from seqh of buf
011256 141050 4           cal
011257 113320 4           era tsbpt1 i
011260 066044 4           ima lms x               ;save mess/block ns, get bloc
011261 032001 4           stx ireg                ;save tsb ptr
011262 010000 4           sta 0                   ;block pointer in x
011263 044070 4           lda mbhst x             ;get host pair
011264 072001 4           ldx ireg                ;restore tsb ptr
011265 141140 4           icl 0&mbfhst            ;foreign host
011266 050110 4           sta lcd x 0&desths&tsbcod ;save in tsb
011267 025320 4           irs tsbpt1              ;ptr to pkth of buf
011270 105320 4           lda tsbpt1 i            ;form tsb code
011271 041677 4           alr 1 0&mltpkt
011272 006013 4           ana seven 0&pktcod&mltpkt
011273 141240 4           icr 0&tsbcod
011274 052110 4           era lcd x               ;add to host info
011275 050110 4           sta lcd x
```

```
011276 012063 4            era [tsbrq1]            ;a req1?
011277 006010 4            ana [tsbcod]
011300 100040 4            sze
011301 005321 4             lda tsbpt2             ;no,
011302 013321 4            era tsbpt2              ;yes, get buf ptr
011303 050264 4            sta lpk x               ;save ptr for req1, zero for
011304 025320 4            irs tsbpt1              ;ptr to dsth of buf
011305 105320 4            lda tsbpt1 i
011306 050154 4            sta ldi x               ;save dest imp
011307 025320 4            irs tsbpt1              ;ptr to midh of buf
011310 105320 4            lda tsbpt1 i
011311 050220 4            sta lid x               ;save messid/subtype
011312 005317 4            lda tsbptx              ;restore host no
011313 026000 4            ima 0                   ;in x
011314 015760 4            add [lid]               ;and link ptr in a
011315 025224 4            irs tsbput              ;skip return
011316 103224 4            jmp tsbput i

                    .lev var
011317          V   tsbptx: .block 1
011320          V   tsbpt1: .block 1
011321          V   tsbpt2: .block 1

                    .lev (h2i,tsk,bck)
;transaction block get (find) subroutine
;expects messno/blockno in a, skip returns with
;tsb ptr in x if found. a is preserved
011322 000000 4   tsbget: 0
011323 073754 4            ldx [-ntsb]             ;number of entries
011324 101040 4            snz                     ;an hi reply thing?
011325 073761 4             ldx [-ntsb+8.]         ;yes
011326 032001 4            stx ireg
011327 073762 4            ldx [tsbtab]            ;first entry
011330 062044 4   tsbgtl: cas lms x                ;equal?
011331 100000 4             skp                    ;no
011332 003337 4               jmp tsbgtf           ;yes
011333 024000 4            irs 0                   ;no
011334 024001 4            irs ireg                ;loop
011335 003330 4             jmp tsbgtl
011336 103322 4            jmp tsbget i

011337 025322 4   tsbgtf: irs tsbget               ;skip return
011340 103322 4            jmp tsbget i
```

```
;hi get local block, called with hip in x (preserved),
;returns:hidisc if got-no-block (task leaves data)
                        ;1 - wait, no block yet
                        ;2 - ok, usenum in a, hihs x with mesnum/blknum,
                        ;hibl x with block pointer
                    .lev h2i
011341 000000 4   hiblgt: 0
011342 105763 4           lda [blkslc] i
011343 012071 4           era [100000]              ;search code locked?
011344 100040 4           sze
011345 103341 4            jmp hiblgt i             ;yes, r1
011346 021512 4           jst hihstp                ;get host pair
011347 111763 4           sta [blkslc] i
011350 144265 4           lda hihhi ix              ;get handling type
011351 141044 4           car 0&ldrhnd
011352 100400 4           spl 0&ldrhpr
011353 013764 4            era [104000]
011354 041474 4           lgl 4 C&ldrhln
011355 152266 4           era hihii ix              ;and dest imp
011356 073765 4           ldx [tmblkt+1]            ;first usable transmit block

011357 121766 4           jst [blksrc] i
                    .lck all
011360 003412 4 0          jmp blkgx1               ;no free blocks, wait
011361 003366 4 0          jmp blkggd               ;got a match
011362 005767 4 0         lda [mbinit+mesbts] 0&mbrst&mbinct
011363 050340 4 0         sta mbmes x               ;got a fresh block, init
011364 133770 4 0         stx [back5i] i            ;point bh5 immediately at thi
011365 003412 4 0         jmp blkgx1                ;and take r1

011366 044340 4 0 blkggd: lda mbmes x               ;get init,rst,and stp bits
011367 040463 4 0         lgr 13. 0&mbinit&mbrst&mbstp
011370 022031 4 0         cas [6]                   ;any one on?
011371 003420 4 0          jmp blkgdm               ;all on ==> dead
011372 003415 4 0          jmp blkgdd               ;init, rst on ==> dead, turn
011373 100040 4 0         sze                       ;any on?
011374 003412 4 0          jmp blkgx1               ;yes, not yet ready ==> r1
011375 021524 4 0         jst blkage                ;none on, reset age bits
011376 044160 4 0         lda mbfor x
011377 006074 4 0         ana [usenum]              ;get foreign use number
011400 011507 4 0         sta blkgus
011401 021537 4 0         jst mesget                ;and get message number
011402 003411 4 0          jmp blkgx0               ;can't get mess no yet
011403 033510 4 0         stx blkgsp                ;save block pointer
011404 072245 4 0         ldx hip
011405 150264 4 0         sta hihsi ix              ;save mess/block
011406 005510 4 0         lda blkgsp                ;get blk ptr
011407 150260 4 0         sta hibli ix              ;save in host table
011410 025341 4 0         irs hiblgt                ;take r2
011411 005507 4 0 blkgx0: lda blkgus                ;restore usenum
011412 000401 4 0 blkgx1: .enb h2i
011413 072245 4           ldx hip                   ;restore x
011414 103341 4           jmp hiblgt i              ;and return


                    .lev h2i
                    .lck all
011415 044340 4 0 blkgdd: lda mbmes x               ;turn on stop bit
011416 012067 4 0         era [mbstp]
011417 050340 4 0         sta mbmes x
```

```
                    blkgdm:
011420 000401 4 0           .enb h2i
011421 044160 4             lda mbfor x             ;get right half
011422 052250 4             era mbtim x
011423 141050 4             cal 0&mesnum&mbfrnb
011424 052250 4             era mbtim x             ;and left half
011425 011507 4             sta blkgus              ;of host status
011426 044340 4             lda mbmes x             ;get subtype
011427 141140 4             icl 0&mbinct
011430 006011 4             ana three 0&subtyp
011431 072245 4             ldx hip
011432 152267 4             era hihli ix
011433 006072 4             ana [ldrsty]
011434 152267 4             era hihli ix
011435 150267 4             sta hihli ix            ;save subtype
011436 005507 4             lda blkgus
011437 150262 4             sta hihpi ix            ;save host status
011440 004041 4   hierdd:   lda [cdestd]            ;send dest dead
011441 150263 4   hilput:   sta hihti ix
011442 140040 4   hilpt1:   cra                     ;get free tsb block
011443 021322 4             jst tsbget
011444 003505 4              jmp hilpt2             ;none found, wait
011445 033511 4             stx hiltsb              ;save tsb ptr
011446 050000 4             sta lch x               ;clear chain ptr
011447 072245 4             ldx hip
011450 144263 4             lda hihti ix            ;get msg type
011451 073511 4             ldx hiltsb
011452 050044 4             sta lms x
011453 072245 4             ldx hip
011454 144267 4             lda hihli ix            ;get link/subtype
011455 073511 4             ldx hiltsb
011456 050220 4             sta lid x
011457 072245 4             ldx hip
011460 144262 4             lda hihpi ix            ;get host status (for dest de
011461 073511 4             ldx hiltsb
011462 050264 4             sta lpk x
011463 072245 4             ldx hip
011464 144265 4             lda hihhi ix            ;get handling type/dest host

011465 073511 4             ldx hiltsb
011466 050110 4             sta lcd x
011467 072245 4             ldx hip
011470 144266 4             lda hihii ix            ;get dest imp
011471 073511 4             ldx hiltsb
011472 050154 4             sta ldi x
011473 005511 4             lda hiltsb              ;get tsb ptr
011474 072245 4             ldx hip
011475 001001 4             .inh i2h
011476 167771 4 3           ima [ehrq] ix           ;put on host reply queue
011477 010001 4 3           sta ireg
011500 005511 4 3           lda hiltsb
011501 110001 4 3           sta ireg i
011502 000401 4 3           .enb h2i
                            ;defhlt(56. HILPUT: long wait for TSB for host reply)
011503 121772 4             jst [hiwtrp] i
011504 103773 4             jmp [hichfl] i
```

```
011505 121774 4   hilpt2: jst [hiwait] i
                         ;defplc(/hi - waiting for tsb slot for host reply)
011506 003442 4          jmp hilpt1

                  .lev var
011507         V  blkgus: .block 1                ;usenum
011510         V  blkgsp: .block 1                ;saved block pointer
011511         V  hiltsb: .block 1                ;tsb block ptr

;hi get host pair
;called with local host in x,
;returns logical receive/transmit host pair in a
                  .lev h2i
011512 000000 4   hihstp: 0
011513 004000 4          lda 0
011514 022056 4          cas [nh]                  ;a fake host?
011515 101000 4           nop                      ;yes
011516 015775 4           add [400-fh-nh]
011517 010001 4          sta ireg
011520 144265 4          lda hihhi ix
011521 141240 4          icr 0&ldrhst              ;receive logical host in l.h.
011522 012001 4          era ireg
011523 103512 4          jmp hihstp i

;reset block age to 4 if >= 4
;blk ptr in x is preserved
                  .lev (bck,tsk,h2i)
                  .lck i2h
011524 000000 4 3 blkage: 0
011525 044250 4 3         lda mbtim x
011526 006072 4 3         ana [mbage]             ;get age
011527 022054 4 3         cas [4]
011530 101000 4 3          nop                    ;>4
011531 004054 4 3          lda [4]                ;or =4 ==> make it be 4
011532 052250 4 3         era mbtim x             ;<4  ==> leave it alone
011533 006072 4 3         ana [mbage]
011534 052250 4 3         era mbtim x
011535 050250 4 3         sta mbtim x
011536 103524 4 3         jmp blkage i
```

```
                .lev (h2i,bck)
                .lck all
;get messno, block pointer in x, preserves x
;skip returns if successful, with mesnum/blknum in a
011537 000000 4 0 mesget: 0
011540 044340 4 0         lda mbmes x
011541 006052 4 0         ana [1]                ;*modify to make window small
011542 101040 4 0         snz                    ;mess no in use?
011543 103537 4 0          jmp mesget i          ;yes
011544 044340 4 0         lda mbmes x
011545 006007 4 0         ana [mesbts]
011546 040477 4 0         lgr 1                  ;shyft bits
011547 052340 4 0         era mbmes x
011550 006007 4 0         ana [mesbts]
011551 052340 4 0         era mbmes x
011552 050340 4 0         sta mbmes x
011553 044250 4 0         lda mbtim x
011554 014062 4 0         add [mesnm1] 0&mesnum
011555 066250 4 0         ima mbtim x            ;inc mess no
011556 052160 4 0         era mbfor x
011557 141044 4 0         car 0&blknum
011560 052160 4 0         era mbfor x            ;get foreign block no
011561 025537 4 0         irs mesget             ;skip=success
011562 103537 4 0         jmp mesget i

                .lev (h2i,tsk,bck)
;get allocate, expects blk ptr in x, preserves x
011563 000000 4   getall: 0
011564 044340 4           lda mbmes x            ;get allocate count
011565 006070 4           ana [mbinit]           ;is init on?
011566 100040 4           sze
011567 004067 4            lda [mbstp]           ;yes, don't use stp bit
011570 012042 4           era [mbstp+mballc]     ;in allocate mask
011571 046340 4           ana mbmes x
011572 101040 4           snz                    ;any allocates?
011573 103563 4            jmp getall i          ;no, quit
011574 044340 4           lda mbmes x            ;yes
011575 016064 4           sub [mball1]
011576 050340 4           sta mbmes x            ;ok, save 1 less
011577 044070 4           lda mbhst x            ;get lgl trans host
011600 006007 4           ana [mblhst]
011601 033616 4           stx getalb             ;save blk ptr
011602 010000 4           sta 0
011603 144256 4           lda hiali ix
011604 141050 4           cal
011605 016052 4           sub [1]                ;decrement total count
011606 100400 4           spl                    ;any there?
011607 003613 4            jmp getal1            ;no
011610 100040 4           sze                    ;any left?
011611 015776 4            add [173000]          ;yes, new 10. tic timeout
011612 150256 4           sta hiali ix           ;reset timer
011613 073616 4   getal1: ldx getalb            ;restore blk ptr
011614 025563 4           irs getall             ;success=skip return
011615 103563 4           jmp getall i

                .lev var
011616         V  getalb: .block 1               ;temp block ptr
```