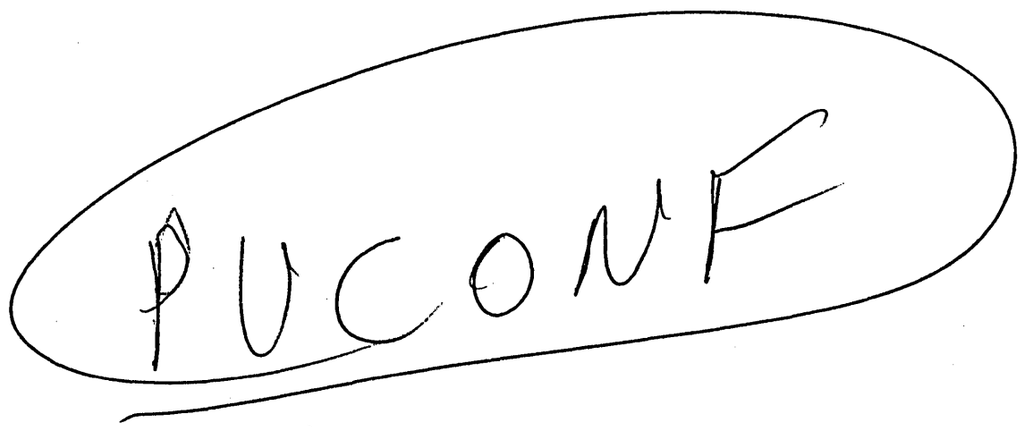
```
+submit :f1:gen3
+delete :f1:*.bak,:f1:*.obj,:f1:*.lst
 :F1:*.BAK, NO SUCH FILE
 :F1:PUCONF.OBJ, DELETED
 :F1:DLL.OBJ, DELETED
 :F1:GEN3.LST, CAN'T DELETE OPEN FILE
 :F1:PUCONF.LST, DELETED
 :F1:GEN4.LST, DELETED
 :F1:DLL.LST, DELETED
+run
ISIS-II RUN 8086, X100
>:f2:asm86 :f1:puconf.com debug
SERIES-III 8086/8087/8088 MACRO ASSEMBLER, V1.0

ASSEMBLY COMPLETE, NO ERRORS FOUND
>:F2:ASM86 :f1:dll.src debug
SERIES-III 8086/8087/8088 MACRO ASSEMBLER, V1.0

ASSEMBLY COMPLETE, NO ERRORS FOUND
>exit
+:F0:SUBMIT RESTORE :F1:GEN3.CS(:F3:GETHEL.CS,7,71)
+copy :f1:puconf.obj to :f3: b
COPIED :F1:PUCONF.OBJ TO :F3:PUCONF.OBJ
+copy :f1:dll.obj to :f3: b
COPIED :F1:DLL.OBJ TO :F3:DLL.OBJ
+consol ,:vo:
```



```
SERIES-III 8C86/8C87/8088 MACRO ASSEMBLER V1.0 ASSEMBLY OF MODULE PUCONF
OBJECT MODULE PLACED IN :F1:PUCONF.OBJ
INVOCATION LINE CONTRCLS:  DEBUG


LOC  OBJ                  LINE     SCURCE

                             1 +1  $title(Ethernet Controller Power-Up Confidence Test Version 1.2)
                             2 +1  $date(01/29/82)
                             3
                             4     ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                             5
                             6     ;           Ethernet Controller Power-Up Confidence Test
                             7     ;                        Version 1.2
                             8
                             9     ; Overview
                            10
                            11     ; the power-up confidence test is actually two tests -- a ram test to check
                            12     ; both DRAM and SRAM, and a functional test to verify basic Processor and
                            13     ; Serdes functioning. Each test is a "short" function procedure which returns
                            14     ; a byte value composed as follows:
                            15
                            16     ;        low byte -- confidence test passed/failed flag, with zero
                            17     ;                    being "passed" and any non-zero value being
                            18     ;                    "failed".
                            19     ;                    if test failed, value is a hexidecimal number
                            20     ;                    with value between 1 and 15H inclusive, a total
                            21     ;                    of 20 (decimal) possible failures. "cqramtest" returns
                            22     ;                    values of 2 or 4 and "cqdevicetest" returns the
                            23     ;                    the remaining values
                            24
                            25     ; Segments
                            26
                            27     ; Both the data segment and the stack segment have been constructed per PLM86
                            28     ;             guidelines. The data segment is called "data" and part of
                            29     ;             "dgroup". No stack segment need be explicitly declared.
                            30     ; One code segment exists for both ram and functional procedures.
                            31     ;             It also contains the offset of a jump table
                            32     ;             pointing to entry points of all 19 tests and some important
                            33     ;             utility routines. Segment is called "code".
                            34
                            35     ; Register usage:
                            36
                            37     ; Segment registers cs,ds,ss will not be manipulated. Registers sp and bp
                            38     ; are not either -- as per PLM86 conventions. All other register contents
                            39     ; upon entry via call of "cqdevicetest" are destroyed. A call of "cqramtest"
                            40     ; will trash all stack contents, however.
                            41
                            42     ; In each case, the return address is initially popped off the stack with
                            43     ; the old ip placed in di. Both procedures
                            44     ; use register si as a flag. If si = 0, then upon completion of a specific
                            45     ; test the following test is executed. That is, si = 0 only if this confidence
                            46     ; test was called by firmware to execute. if si = 1, then only 1 specific test
                            47     ; is to be executed and a far return is to be performed upon its (maybe un-
                            48     ; successful) completion. The remaining registers are used indiscriminately.
                            49     ;
                            50     ;  Call "cqramtest" to test the following:
```

```
LOC  OBJ                  LINE     SOURCE

                            51
                            52     ;                                  Error Code
                            53     ;            DRAM                       2
                            54     ;            SRAM                       4
                            55     ;
                            56     ;  Call "cqdevicetest" to test the following:
                            57
                            58     ;                                  Error Code
                            59     ;       DRAM Data Ripple                1
                            60     ;       SRAM Data Ripple                3
                            61     ;       Lower Prom Checksum             5
                            62     ;       Upper Prom Checksum             6
                            63     ;       8255 Write/Read                 7
                            64     ;       8237 Write/Read                 8
                            65     ;       8259 Write/Read                 9
                            66     ;       8253 Counter 0                  A
                            67     ;       8253 Counter 1                  B
                            68     ;       8253 Counter 2                  C
                            69     ;       Address Prom Read/ DMA
                            70     ;          Receive Channel 1            D
                            71     ;       DMA Receive Channel 2           E
                            72     ;       DMA Receive Channel 3           F
                            73     ;       Address Prom CRC               10
                            74     ;       Transmit/Receive Packet
                            75     ;         Broadcast Packet Recognize   11
                            76     ;       Receive Packet CRC
                            77     ;          Generation Circuitry        12
                            78     ;       Address Recognition
                            79     ;          Circuitry                   13
                            80     ;       Non Self-Addressed Packet
                            81     ;          Recognition                 14
                            82     ;       Transmit/Receive Path          15
                            83     ;
                            84
                            85     ;   Code entered in upper case was created by Jeff Pickering. All lower-case
                            86     ;   instructions were written by Mickey Gee
                            87
                            88     ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                            89 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                            90
                            91     ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                            92     ;
                            93     ; differences between version 1.0 and version 1.1
                            94     ;
                            95     ; 1. routine "bad_addr" was corrected to disallow recognition of certain
                            96     ;    Ethernet addresses as being multicast, thus causing the test to fail.
                            97     ;    the serdes will "mirror" all bytes of a data packet in on-board loopback
                            98     ;    mode (e.g., a 01100001 is seen as a 10000110). this is performed on
                            99     ;    Ethernet addresses also, and thus any comm board with the first byte
                           100     ;    of its Ethernet address being xxxxxxx1 would be seen as a multicast
                           101     ;    address.
                           102     ;    this is remedied by making all last bits of the first Ethernet byte = 0.
                           103     ;    this will alter the Ethernet address being looped back. However,
                           104     ;    "bad_addr" tests for improper Ethernet address recognition anyways.
                           105     ; 2. routine "verify" was changed to test for absence of Carrier Sense
                           106     ;    before performing any on-board loopbacks. all standard packet
                           107     ;    transmits are prevented from causing network collisions by comm
                           108     ;    hardware which will transmit if and only if Carrier Sense is inactive.
                           109     ;    however, this is not true for on-board loopbacks. consequently, Carrier
                           110     ;    Sense bit from 8255 must be low during this to prevent network collisions
                           111     ;    from occurring upon power-on of this board. for the carrier sense bit
                           112     ;    to watch out for Ethernet packets, the SerDes must have receive enabled
                           113     ;    and on-board verify disabled.
                           114     ; 3. once fix #2 was implemented, it was noticed that now tests "ver_sta_add"
                           115     ;    and "bad_addr" did not work properly while packets were traversing the
                           116     ;    ethernet. the SerDes would now only permit only 1 byte instead of the
                           117     ;    expected 14H bytes to be looped around. strangely enough, it was
                           118     ;    unexpectedly discovered that performing a read address before performing
                           119     ;    either routine would resolve this problem. a hardware bug in the address
                           120     ;    recognition circuitry leaving it in a weird state is suspected but not
                           121     ;    verified.
                           122     ; 4. once fix #3 was implemented, it was noticed that the stfs and detmon based
                           123     ;    tests failed tests 18 and 19. these tests are dependent upon the status of
                           124     ;    the serdes as seen by routine "verify". tests 18 and 19 would always
                           125     ;    obtain a later (and therefore incorrect) status byte. consequently,
                           126     ;    "verify" will always save the contents of 8255 port a instead of those
                           127     ;    externally resident tests.
                           128     ; 5. sixteen bit crc tests were discovered to perform crc on half of both
                           129     ;    Proms. once it was corrected, it added 600 milliseconds to the time it
                           130     ;    took perform power-up. at this point, it was agreed that this was too much
                           131     ;    time to perform power-up (total of 1.5 seconds) and a 24-bit checksum was
                           132     ;    agreed upon as an adequate and less time (but not Prom space) consuming
                           133     ;    alternative.
                           134     ;
                           135     ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                           136 +1  $eject
```

LOC OBJ LINE SOURCE

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
; differences between version 1.1 and version 1.2
;
; 1. inadvertently, instruction to initialize byte to be rippled was left out
;    of "sram_ripple". code to initialize cl register was reinserted.
; 2. 24-bit checksum routine incremented upper eight bits of checksum whenever
;    an overflow occurs. this does not work. instead, upper byte will be
;    incremented whenever a carry occurs.
; 3. code segment align-type made byte instead of paragraph to allow it to
;    be placed anywhere within firmware.
;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
$eject
```

```
LOC  OBJ                  LINE     SOURCE

                          151
                          152      ; make known to Controller firmware the title of confidence test.
                          153
                          154              name    puconf
                          155
                          156      ;           Comm board literals
                          157
  0000                    158        SET_TXSRT                       EQU 0
  0001                    159        RESET_TXSRT                     EQU 1
  0002                    160        CH1_AVAIL                       EQU 2
  0003                    161        ch2_avail                       equ 3
  0004                    162        ch3_avail                       equ 4
  0005                    163        RESET_ERRS                      EQU 5
  0006                    164        RESET_CH_CNTR                   EQU 6
                          165
  00E3                    166        PORT_8255_mode_ctl              EQU 0E3H
  00E0                    167        PORT_8255_A                     EQU 0E0H
  00E1                    168        PORT_8255_B                     EQU 0E1H
  00E2                    169        PORT_8255_C                     EQU 0E2H
                          170
  00CF                    171        PORT_8237_MASK                  EQU 0CFH
  00C0                    172        PORT_8237_CH0_ADDR              EQU 0C0H
  00C1                    173        PORT_8237_CH0_COUNT             EQU 0C1H
  00C2                    174        PORT_8237_CH1_ADDR              EQU 0C2H
  00C3                    175        PORT_8237_CH1_COUNT             EQU 0C3H
  00C4                    176        port_8237_ch2_addr              equ 0C4H
  00C5                    177        port_8237_ch2_count             equ 0C5H
  00C6                    178        port_8237_ch3_addr              equ 0C6H
  00C7                    179        port_8237_ch3_count             equ 0C7H
                          180
  00F1                    181        PORT_8259_MASK                  EQU 0F1H
  00F0                    182        PORT_8259_A0_LOW                EQU 0F0H
  00D0                    183        TIMER_0_COUNT                   EQU 0D0H
  00D1                    184        timer_1_count                   equ 0D1H
  00D2                    185        timer_2_count                   equ 0D2H
                          186
  0007                    187        LED_ON                          EQU 07H
  0005                    188        LED_OFF                         EQU 05H
  0080                    189        LOOP_BIT                        EQU 80H
  000C                    190        POLL_8259_CMD                   EQU 0CH
  0020                    191        EOI_CMD                         EQU 20H
                          192
  9FC2                    193        STATION_ADDRESS                 equ 9FC2H
  9FE0                    194        T_PACKET_DEST                   equ 9FE0H
                          195
                          196 +1   $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           197     cgroup          group   code
                           198     dgroup          group   data
                           199
----                       200     data    segment public 'data'
----                       201     data    ends
                           202
----                       203     code    SEGMENT byte public 'code'
                           204
                           205             PUBLIC  cqdevicetest
                           206             public  cqramtest
                           207             public  ctjumptable
                           208
                           209             extrn   cqhdwinit:      near    ;firmware routine initalizes chips
                           210
                           211             ASSUME CS:cgroup, ds:dgroup
                           212
                           213
                           214     ; entry point to DRAM/SRAM tests if calling code is firmware resident
                           215
C000                       216     ramtest         PROC near
                           217
C000                       218     cqramtest:
                           219
0000 E80000         E      220             call    cqhdwinit       ;initialize board and disable receive mode
                           221                                     ;  so that sram march test will work ok
0003 5F                    222             POP     DI              ;SAVE FIRMWARE'S CALL RET IP (short return)
0004 33F6                  223             xor     si,si           ;set stfs/firmware call
                           224                                     ;    switch = 0 -- firmware call
                           225
                           226 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           227     ; "dram_test" is only used during power up. it tests all of dram except for the
                           228     ; last eight bytes which are reserved for use by Ethernet controller firmware.
                           229     ; test sets error flag in ah reg = 2
                           230
                           231     ; register usage: ah -- error flag
                           232     ;                 al -- set to 80H if march test failed
                           233     ;                 bx -- first address (ds = f000, bx = 0, address = f0000H)
                           234     ;                 cx -- number of bytes to be tested
                           235     ;                 dx -- return address in "dram_test", jumped to by "mar_test"
                           236     ;                       routine
                           237
                           238     ; test recuires ds be set to F000H beforehand.
                           239
0006                       240     DRAM_TEST:
                           241
0006 B9F83F                242             MOV     CX,3FF8H        ;COUNT= 16K-8 bytes
0009 33DB                  243             xor     bx,bx           ;STARTING ADDRESS = 0
                           244
000B                       245     FILL_DRAM:
000B C607AA                246             MOV     byte ptr [BX],0AAH              ;DATA INTO MEM
000E 43                    247             INC     BX                              ;point to next dram byte
000F E2FA                  248             LOOP    FILL_DRAM
0011 B9F83F                249             MOV     CX,3FF8H                        ;16K - 8 bytes
0014 33DB                  250             xor     bx,bx
0016 BA1C009C        R     251             MOV     DX,OFFSET cgroup:DRAM_RET       ;JUMP RETURN ADDRESS
001A EB1A                  252             JMP     short test_march                ;TEST DYNAMIC RAM
                           253
001C                       254     DRAM_RET:
001C B402                  255             MOV     ah,2                    ;set error flag         <====
001E D0D0                  256             RCL     AL,1                    ;OVERFLOW IF FAILURE
0020 701B                  257             jo      ram_fail
                           258
                           259 +1  $eject
```

```
LOC  OBJ                  LINE     SCURCE

                           260     ; "sram_test" is only used during power up. test sets error flag in ah reg = 4
                           261
                           262     ; register usage: ah -- error flag
                           263     ;                 al -- set to 80H if march test failed
                           264     ;                 bx -- first address (ds = f000, bx = 8000, address = f8000H)
                           265     ;                 cx -- number of bytes to be tested (8k bytes)
                           266     ;                 dx -- return address in "sram_test", jumped to by "mar_test"
                           267     ;                       routine
                           268
                           269     ; test requires ds initialized as F000H beforehand.
                           270
0022                       271     sram_test:
                           272
0022 B520                  273             MOV     ch,20H          ;COUNT = 8K = 2000H (cl already = 0)
0024 BB0080                274             MOV     BX,8000H        ;STARTING ADDRESS
                           275
0027                       276     FILL_SRAM:
0027 C607AA                277             MOV     byte ptr [BX],0AAH
002A 43                    278             INC     BX
002B E2FA                  279             LOOP    FILL_SRAM
002D B520                  280             MOV     ch,20H                          ;count = 2000H
002F BB0080                281             MOV     BX,8000H
0032 BA39009C        R     282             MOV     DX,OFFSET cgroup:SRAM_RET       ;JUMP RETURN ADDRESS
                           283
0036                       284     test_march:
0036 E98702                285             JMP     MAR_TEST                        ;TEST STATIC RAM
                           286
0039                       287     SRAM_RET:
0039 B404                  288             MOV     ah,4                            ;ERROR CODE             <====
003B D0D0                  289             RCL     AL,1                            ;OVERFLOW IF FAIL
                           290
003D                       291     ram_fail:
003D 7072                  292             jc      fail_t8255
                           293
003F                       294     ccramexit:
003F E99501                295             jmp     pass            ;if firmware, then go to return-to-firmware
                           296                                     ;   routine
                           297
                           298     ramtest         endp
                           299 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          300      ; entry point for major portion of confidence test if calling code is firmware
                          301      ; resident
                          302
0042                      303      device_test     PROC near
                          304
0042                      305      cqdevicetest:
                          306
0042 E80000         E     307              call    cqhdwinit       ;initialize chips
                          308
0045 5F                   309              POP     DI              ;SAVE CALL RET IP
                          310
0046 33F6                 311              xor     si,si           ;set stfs/firmware call
                          312                                      ;  switch = 0 -- firmware call
                          313
                          314
                          315      ; "dram_ripple" ripples a 1 through dram byte at address F3F00. it starts with
                          316      ; pattern 1 and shifts left until finished with pattern = 80H.
                          317
                          318      ; register usage: cl -- present pattern
                          319      ;                 dl -- pattern read from F3F00
                          320      ;                 bx -- points to DRAM address used
                          321      ;                 ah -- error flag
                          322      ;                 ds -- data segment F0000
                          323
                          324      ; test requires only that ds be intialized before execution
                          325
0048                      326      dram_ripple:
                          327
0048 BB003F               328              mov     bx,3F00H        ;ripple at address F3F00H near end of DRAM
004B B101                 329              mov     cl,1            ;start with test pattern = 1
004D B401                 330              mov     ah,1            ;set error flag                          <===
                          331
004F                      332      dram_loop:
004F 880F                 333              mov     [bx],cl         ;write test pattern into DRAM
0051 8A17                 334              mov     dl,[bx]         ;retrieve pattern just written from cl
0053 3ACA                 335              cmp     cl,dl           ;compare read (reg dl) with written (reg cl)
0055 755A                 336              jnz     fail_t8255      ;if not same, then jump
0057 D0C1                 337              rol     cl,1            ;else, change pattern
0059 73F4                 338              jnc     dram_loop       ; and continue if not done
                          339
005B E88501               340      stfs_1: call    stfs_ret_check  ;test called by stfs or firmware?
                          341                                      ;continue if firmware
                          342 +1   $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           343     ; "sram_ripple" ripples a 1 through sram byte at address F9FFF. it starts with
                           344     ; pattern 1 and shifts left until finished with pattern = 80H.
                           345
                           346     ; register usage: cl -- present pattern, initialized by "dram_ripple" as
                           347     ;                      being set = 1
                           348     ;                  dl -- pattern read from F9FFF
                           349     ;                  bx -- points to SRAM address used
                           350     ;                  ah -- error flag
                           351
                           352     ; test requires ds initialized as F000H beforehand.
                           353
C05E                       354     sram_ripple:
                           355
C05E B101                  356             mov     cl,1            ;initialize data pattern to be rippled
0060 BBFF9F                357             mov     bx,9FFFH        ;ripple at address F9FFFH in middle of SRAM
C063 B403                  358             mov     ah,3            ;set error flag                          <====
                           359
0065                       360     sram_loop:
0065 880F                  361             mov     [bx],cl         ;write test pattern into SRAM
C067 8A17                  362             mov     dl,[bx]         ;retrieve pattern just written from cl
0069 3ACA                  363             cmp     cl,dl           ;compare read (reg dl) with written (reg cl)
C06B 7544                  364             jnz     fail_t8255      ;if not same, then jump
006D D0E1                  365             shl     cl,1            ;else, change pattern
006F 73F4                  366             jnc     sram_loop       ; and continue if not done
                           367
C071 E86F01                368     stfs_3: call    stfs_ret_check  ;test called by stfs or firmware?
                           369                                     ;continue if firmware
                           370 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           371     ; "low_rom_test" calculates a 24-bit checksum value on the first prom. test
                           372     ; expects a checksum value at address FFFFA for low prom.
                           373
                           374     ; register usage: ah -- error code = 5
                           375     ;                 bx -- calculated lower 16-bits of checksum
                           376     ;                 cx -- lower 16-bits of checksum value read from address FFFFA
                           377     ;                 dl -- high eight bits of 24-bit checksum read from prom
                           378     ;                 dh -- calculated high eight bits
                           379
                           380     ; test requires ds initialized to F000H beforehand.
                           381
0074                       382     low_ROM_TEST:
                           383
0074 B90008                384             MOV     CX,800H         ;perform 24-bit checksum on 2k words
0077 BB00E0                385             MOV     BX,0E000H       ;STARTING offset ADDRESS
007A E8B002                386             call    checksum        ;perform 24-bit checksum on addresses FE000-FEFFF
007D BBFAFF                387             mov     bx,0FFFAH       ;point bx to 6th to last byte on
                           388                                     ; second prom -- crc for first prom is here
0080 8A37                  389             mov     dh,[bx]         ;get high eight bits of stored checksum
0082 43                    390             inc     bx              ;point to and get
0083 8B0F                  391             mov     cx,[bx]         ;  lower sixteen bits of checksum in PROM
0085 8BD8                  392             mov     bx,ax           ;save that lower sixteen in reg bx
0087 B405                  393             MOV     ah,5            ;ERROR CODE                              <====
0089 3AF2                  394             cmp     dh,dl           ;upper eight bits = calculated eight bits?
008B 7524                  395             jne     fail_t8255      ;jump if bad
008D 3BCB                  396             CMP     cx,bx           ;CMP WITH CKSM IN MEMORY
008F 7520                  397             jne     fail_t8255      ;FAILURE IF NOT EQUAL
                           398
0091 E84F01                399     stfs_5: call    stfs_ret_check  ;test called by stfs or firmware?
                           400                                     ;continue if firmware
                           401 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           402     ; "upper_rom_test" calculates a 24-bit checksum value on the second prom. test
                           403     ; expects a checksum value at address FFFFD for upper prom.
                           404
                           405     ; register usage: ah -- error code = 6
                           406     ;                 bx -- calculated lower 16-bits of checksum
                           407     ;                 cx -- lower 16-bits of checksum value read from address FFFFD
                           408     ;                 dl -- high eight bits of 24-bit checksum read from prom
                           409     ;                 dh -- calculated high eight bits
                           410
                           411     ; test assumes ds initialized to F000H. stack is also assumed to exist
                           412     ; somewhere. note that checksums should never include the checksum value
                           413     ; itself.
                           414
0094                       415     upper_rom_test:
                           416
0094 B9FD07                417             MOV     CX,7FDH         ;perform crc on 2k minus 3 words (last 3 words
                           418                                     ; contains calculated checksums)
0097 BB00F0                419             MOV     BX,0F000H       ;STARTING ADDRESS
                           420
009A                       421     UPPER_ROM:
009A E89002                422             call    checksum        ;perform 24-bit checksum on addresses FF000-FFFF9
009D BBFDFF                423             mov     bx,0FFFDH       ;point bx to 3rd to last byte on
                           424                                     ; second prom -- crc for 2nd prom is here
00A0 8A37                  425             mov     dh,[bx]         ;get high eight bits of stored checksum
00A2 43                    426             inc     bx              ;point to and get
00A3 8B0F                  427             mov     cx,[bx]         ;  lower sixteen bits of checksum in PROM
00A5 8BD8                  428             mov     bx,ax           ;save that lower sixteen in reg bx
00A7 B406                  429             MOV     ah,6            ;ERROR CODE                                <====
00A9 3AF2                  430             cmp     dh,dl           ;upper eight bits = calculated eight bits?
00AB 7504                  431             jne     fail_t8255      ;jump if bad
00AD 3BCB                  432             CMP     cx,bx           ;CMP WITH CKSM IN MEMORY
00AF 7402                  433             je      stfs_6          ;FAILURE IF NOT EQUAL
                           434
00B1                       435     fail_t8255:
00B1 EB5D                  436             JMP     short FAIL_t8253
                           437
00B3 E82D01                438     stfs_6: call    stfs_ret_check  ;test called by stfs or firmware?
                           439                                     ;continue if firmware
                           440 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           441     ; "t8255" writes and reads back port b.
                           442
                           443     ; register usage: al -- input/output to port b go through this register
                           444     ;                 ah -- error code set to 7
                           445     ;                 bl -- data read from port (received)
                           446     ;                 bh -- expected data
                           447
                           448     ; this test does not require ds initialization. however, the 8255 is assumed
                           449     ; to be as follows:
                           450     ;                 port E3 = 98H   (mode control port)
                           451
00B6                       452     T8255:
00B6 B075                  453             MOV     AL,75H                  ;WRITE,READ PORT B (verify,rcv = 0)
00B8 E8D201                454             call    write_8255_port_b
00BB B407                  455             MOV     ah,7                    ;ERROR CODE                     <====
00BD 7551                  456             jne     fail_t8253
                           457
00BF                       458     SKIP2:
00BF B0AA                  459             MOV     AL,0AAH                 ;verify, receive = 1
00C1 E8C901                460             call    write_8255_port_b
00C4 754A                  461             jne     fail_t8253
                           462
00C6 E81A01                463     stfs_7: call    stfs_ret_check          ;test called by stfs or firmware?
                           464                                             ;continue if firmware
                           465 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           466     ; "t8237" only writes and reads a word value from an 8237 register. it is
                           467     ; executed only at power-up time.
                           468
                           469     ; register usage: ah -- error flag = 8
                           470     ;                 al -- input/output to 8237
                           471     ;                 bx -- word read from 8237
                           472     ;                 cx -- word expected to be read from 8237
                           473
                           474     ; expected initialization sequence: port c8 = a0
                           475     ;                                   port cb = 88
                           476     ;                                   port cb = 15
                           477     ;                                   port cb = 16
                           478     ;                                   port cb = 17
                           479
                           480     ; test does not need an initialized ds
                           481
00C9                       482     t8237:                                  ;WRITE,READ CH0 ADDR REG
00C9 E6CF                  483             OUT     PORT_8237_MASK,AL       ;MASK ALL CHANNELS
00CB B2C0                  484             mov     dl,PORT_8237_CH0_ADDR   ;set dx -- dh is already set = 0
                           485                                             ; this will save a byte in PROM
00CD B0AA                  486             MOV     AL,0AAH
00CF EE                    487             OUT     dx,AL                   ;CH0 LOW ADDR
00D0 B055                  488             MOV     AL,055H
00D2 EE                    489             OUT     dx,AL                   ;CH0 HIGH ADDR
00D3 EC                    490             IN      AL,dx                   ;LOW ADDR
00D4 8AE0                  491             MOV     AH,AL
00D6 EC                    492             IN      AL,dx                   ;HIGH ADDR
00D7 3D55AA                493             cmp     ax,0AA55H               ;received in ax = expected?
00DA B408                  494             MOV     ah,8                    ;error code                     <====
00DC 7532                  495             jnz     FAIL_t8253
                           496
                           497 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          498      ; "t8259" is used during power-up sequence.
                          499
                          500      ;  register usage: ah -- error code = 9
                          501      ;                  bl -- expected byte
                          502      ;                  bh -- byte read (maybe bad)
                          503
                          504      ; expected initialization sequence: port f0 = 13
                          505      ;                                   port f1 =  8
                          506      ;                                   port f1 =  3
                          507
                          508      ; test does not need an initialized ds. however, note that test assumes al
                          509      ; register is set to 55H from previous test "t8237".
                          510
00DE                      511      T8259:                              ;WRITE,READ MASK REG
00DE E88B01               512              call    write_8259_mask
00E1 B409                 513              MOV     ah,9                ;ERROR CODE                    <====
00E3 7528                 514              jnz     fail_t8253
                          515
00E5 B0AA                 516              MOV     AL,0AAH
00E7 E88201               517              call    write_8259_mask
00EA 7524                 518              jnz     fail_t8253
                          519
00EC B0DF                 520              MOV     AL,0DFH             ;CLEAR POSSIBLE TIMER 0 interrupt
00EE E6F1                 521              OUT     PORT_8259_MASK,AL
00F0 E80402               522              CALL    CLR_REQ
00F3 B00C                 523              MOV     AL,POLL_8259_CMD    ;VERIFY IS CLEARED
00F5 E6F0                 524              OUT     PORT_8259_A0_LOW,AL
00F7 E4F0                 525              IN      AL,PORT_8259_A0_LOW
00F9 D0D0                 526              RCL     AL,1                ;SETS CARRY IF STILL REQUEST
00FB 7213                 527              jb      FAIL_t8253
                          528
                          529 +1   $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          530      ; "t8253" counts down on all three 16-bit counters. it sends an interrupt to
                          531      ; the 8259 upon terminal count (this means the 8259 had better be working).
                          532      ; three entry points for stfs have been created -- one for each timer. this
                          533      ; tests assumes read/load both low/high bytes mode has been preset. due to
                          534      ; this, test has limited usefulness when trying to test low byte only or high
                          535      ; byte only for read/load.
                          536
                          537      ; expected initialization sequence: port d3 = 30
                          538      ;                                    port d3 = 70
                          539      ;                                    port d3 = b4
                          540
                          541      ; test does not need an intialized ds. however, 8259 interrupt mask must be
                          542      ; set to recognize the desired interrupt level individually.
                          543
00FD                      544      T8253_0:                                ;FORCE TIMER 0 TO ISSUE INT REQ
00FD B0DF                 545              mov     al,0DFH
00FF E6F1                 546              out     port_8259_mask,al       ;8259 unmasked to recognize interrupt 5
0101 B001                 547              MOV     AL,1
0103 E6D0                 548              OUT     TIMER_0_COUNT,AL        ;LSB
0105 E6D0                 549              OUT     TIMER_0_COUNT,AL        ;msb
0107 E8CE01               550              CALL    WAIT_FOR_INT_REQ        ;interrupt 5 sent to 8259?
010A 3C00                 551              CMP     AL,0
010C 7405                 552              JZ      stfs_a
010E B40A                 553              MOV     ah,0AH                  ;ERROR CODE                     <====
                          554
0110                      555      fail_t8253:
0110 E9C800               556              jmp     fail                    ;test failed
                          557
0113 E8CD00               558      stfs_a: call    stfs_ret_check          ;test called by stfs or firmware?
                          559                                              ;continue if firmware
                          560
0116                      561      T8253_1:                                ;FORCE TIMER 1 TO ISSUE INT REQ
0116 B0BF                 562              mov     al,0BFH
0118 E6F1                 563              out     port_8259_mask,al       ;8259 unmasked to recognize interrupt 6
011A B001                 564              MOV     AL,1
011C E6D1                 565              OUT     TIMER_1_COUNT,AL        ;LSB
011E E6D1                 566              OUT     TIMER_1_COUNT,AL        ;msb
0120 E8B501               567              CALL    WAIT_FOR_INT_REQ
0123 3C00                 568              CMP     AL,0
0125 B408                 569              MOV     ah,0BH                  ;ERROR CODE                     <====
0127 75E7                 570              jnz     fail_t8253
                          571
0129 E8B700               572      stfs_b: call    stfs_ret_check          ;test called by stfs or firmware?
                          573                                              ;continue if firmware
                          574
012C                      575      T8253_2:                                ;FORCE TIMER 2 TO ISSUE INT REQ
012C B07F                 576              mov     al,7FH
012E E6F1                 577              out     port_8259_mask,al       ;8259 unmasked to recognize interrupt 7
0130 B001                 578              MOV     AL,1
0132 E6D2                 579              OUT     TIMER_2_COUNT,AL        ;LSB
0134 E6D2                 580              OUT     TIMER_2_COUNT,AL        ;msb
0136 E89F01               581              CALL    WAIT_FOR_INT_REQ
0139 3C00                 582              CMP     AL,0
013B B40C                 583              MOV     ah,0CH                  ;ERROR CODE                     <====
013D 75D1                 584              jnz     fail_t8253
```

```
LOC  OBJ                  LINE      SOURCE

                           585
013F E8A100                586      stfs_c: call    stfs_ret_check          ;test called by stfs or firmware?
                           587                                              ;continue if firmware
                           588 +1   $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           589     ; the next three test perform dma receives via the three receive channels on
                           590     ; the 8237.  Each obtains the Ethernet address and the crc value from the
                           591     ; address prom on the Serdes. obtained are 14 bytes and 2 crc bytes in a string
                           592     ; starting at sram address F9FC2H (most significant digits are first).
                           593
                           594     ; test does requires ds = F000H and also assumes 8255 and 8237
                           595     ; to be initialized.
                           596
                           597     ; test "dma1" receives on dma channel 1. subsequent tests "dma2" and "dma3"
                           598     ; receive via dma channels 2 and 3, respectively.
                           599
0142                       600     dma1:
                           601
0142 E8A800                602             call    read_id                 ;perform dma read address via
                           603                                             ;  channel one
0145 B40D                  604             MOV     ah,0DH                  ;ERROR CODE                     <====
0147 74C7                  605             jz      FAIL_t8253              ;if interrupt returned, then ok
                           606
0149 E89700                607     stfs_d: call    stfs_ret_check          ;test called by stfs or firmware?
                           608                                             ;continue if firmware
                           609
014C                       610     dma2:
                           611
014C B00B                  612             MOV     AL,0BH                  ;UNMASK CHANNEL 2 on 8237
014E E6CF                  613             OUT     PORT_8237_MASK,AL
0150 E603                  614             OUT     CH2_AVAIL,AL            ;SET CHANNEL 2 AVAILABLE
0152 B0FD                  615             MOV     AL,0FDH                 ;UNMASK interrupt 1 -- channel 2
0154 E84301                616             call    mask_and_radd
0157 B40E                  617             MOV     ah,0EH                  ;ERROR CODE                     <====
0159 74B5                  618             jz      fail_t8253
                           619
015B E88500                620     stfs_e: call    stfs_ret_check          ;test called by stfs or firmware?
                           621                                             ;continue if firmware
                           622
015E                       623     dma3:
                           624
015E B007                  625             MOV     AL,7                    ;UNMASK CHANNEL 3 on 8237
0160 E6CF                  626             OUT     PORT_8237_MASK,AL
0162 E604                  627             OUT     CH3_AVAIL,AL            ;SET CHANNEL 3 AVAILABLE
0164 B0FB                  628             MOV     AL,0FBH                 ;UNMASK interrupt 2 -- channel 3
0166 E83101                629             call    mask_and_radd
0169 B40F                  630             MOV     ah,0FH                  ;ERROR CODE                     <====
016B 74A3                  631             jz      fail_t8253
                           632
016D E87300                633     stfs_f: call    stfs_ret_check          ;test called by stfs or firmware?
                           634                                             ;continue if firmware
                           635 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           636     ; "check_crc" performs 16-bit crc on Ethernet address and compares it with crc
                           637     ; read from Ethernet prom. test assumes Ethernet address in sram at address
                           638     ; F9FC2H, and crc value at F9FD0H.
                           639
                           640     ; register usage: ah -- error flag = 10H
                           641     ;                 bx -- calculated crc value
                           642     ;                 cx -- expected crc result of zero
                           643
                           644     ; test assumes ds initialized as F000H
                           645
0170                       646     CHECK_CRC:
0170 BBC29F                647             MOV     BX,9FC2H        ;DATA POINTER
0173 B91000                648             MOV     CX,10H          ;DATA COUNT -- 6 data bytes, 8 check bytes,
                           649                                     ;  and 2 16-bit crc bytes
0176 E88901                650             CALL    CRC_16          ;COMPUTE CRC ON DATA
                           651
                           652     ; note that crc is backwards initially
                           653
0179 33C9                  654             xor     cx,cx           ;expected crc value = 0
017B 8BD8                  655             mov     bx,ax           ;place calculated crc value in bx
017D 3BD9                  656             cmp     bx,cx           ;compare calculated versus read crc
017F B410                  657             MOV     ah,10H          ;ERROR CODE                              <===
0181 7558                  658             jnz     FAIL
                           659
0183 E85D00                660     stfs_10:call    stfs_ret_check          ;test called by stfs or firmware?
                           661                                             ;continue if firmware
                           662 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           663     ; "broadcast_rcv" tests reception of promiscuous packet with 8255 promiscuous
                           664     ; bit set to zero.
                           665
                           666     ; test assumes ds initialized as F000H, all LSI previously initialized.
                           667
0186                       668     broadcast_rcv:
                           669
0186 E82401                670             call    xmt_reset               ;reset channel 0 transmit
0189 E8B001                671             call    tstpkt_to_sram          ;move test packet to sram
018C B31A                  672             MOV     BL,1AH                  ;TO 8255 PORT B -- rcv,xmt,promisc/
018E E89200                673             CALL    VERIFY
0191 2430                  674             AND     AL,30H                  ;RX ERROR BITS -- len or crc err?
0193 B411                  675             MOV     ah,11H                  ;ERROR CODE                          <===
0195 7544                  676             jnz     fail
                           677
0197 E84900                678     stfs_11:call    stfs_ret_check          ;test called by stfs or firmware?
                           679                                             ;continue if firmware
                           680 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           681     ; "ver_crc" uses the standard test packet in verify mode to verify functioning
                           682     ; of FRC CRC circuitry to create a test packet with good data and a bad crc.
                           683
                           684     ; test assumes ds initialized as F000H, all LSI previously initialized.
                           685
C19A                       686     VER_CRC:
                           687
019A E81001                688             call    xmt_reset               ;reset channel 0 transmit
019D E89C01                689             call    tstpkt_to_sram          ;put test packet in place
01A0 B316                  690             MOV     BL,16H                  ;TO 8255 PORT B -- xmt,rcv,promisc,frc crc
01A2 E87E00                691             CALL    VERIFY
01A5 B412                  692             MOV     ah,12H                  ;ERROR CODE                     <===
01A7 2420                  693             AND     AL,20H                  ;check error status -- ignore len err
01A9 7430                  694             jz      FAIL                    ;if error, then jump
                           695
01AB E83500                696     stfs_12:call    stfs_ret_check          ;test called by stfs or firmware?
                           697                                             ;continue if firmware
                           698 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           699     ; "ver_sta_add" uses its own address as a test packet in verify mode to verify
                           700     ; functioning of address recognition circuitry. it sends a packet of its Ether-
                           701     ; net address and the crc appended to it.
                           702
                           703     ; test assumes: ds -- initialized to F000H
                           704     ;               LSI previously initialized
                           705     ;               SRAM already contains Ethernet address as placed in there by
                           706     ;                 procedure "radd" at address F9FC2H.
                           707     ;               8237 initialized with # bytes to be transmitted
                           708
                           709     ; note that a length error in verify mode is caused only by an address
                           710     ; miscompare (wrong Ethernet address)
                           711
01AE                       712     VER_STA_ADD:
                           713
01AE E83C00                714             call    read_id                 ;reset address recognition circuitry
01B1 E8F900                715             call    xmt_reset               ;reset channel 0 transmit
01B4 E8A300                716             call    address_verify
01B7 B413                  717             MOV     ah,13H                  ;ERROR CODE                          <===
01B9 7520                  718             JNZ     FAIL
                           719
01BB E82500                720     stfs_13:call    stfs_ret_check          ;test called by stfs or firmware?
                           721                                             ;continue if firmware
                           722 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           723     ; "bad_addr" uses its own address as a test packet in verify mode to verify
                           724     ; functioning of address recognition circuitry. it sends a packet of its Ether-
                           725     ; net address -- with one bit inverted -- and the correct crc appended to it.
                           726
                           727     ; test assumes: ds -- initialized to F000H
                           728     ;               LSI previously initialized
                           729     ;               SRAM already contains Ethernet address as placed in there by
                           730     ;                 procedure "radd" at address F9FC2H.
                           731     ;               8237 initialized with # bytes to be transmitted
                           732
C1BE                       733     bad_addr:
                           734
C1BE E82C00                735             call    read_id                 ;read in ethernet id and reset address
                           736                                             ;  recognition circuitry
C1C1 E8E900                737             call    xmt_reset               ;reset channel 0 transmit
C1C4 BBC39F                738             MOV     bx,STATION_ADDRESS+1    ;complement second byte of
C1C7 F617                  739             not     byte ptr [bx]           ;  ethernet address
C1C9 4B                    740             dec     bx                      ;point to first byte of ethernet
                           741                                             ;  address
C1CA 8027FE                742             and     byte ptr [bx],0FEH      ;set last bit = 0 always
C1CD E88A00                743             call    address_verify          ;perform on-board loopback
C1D0 B414                  744             MOV     ah,14H                  ;ERROR CODE                     <===
C1D2 7407                  745             JZ      FAIL                    ;LEN ERR IN VERIFY
                           746
C1D4 E80C00                747     stfs_14:call    stfs_ret_check          ;test called by stfs or firmware?
                           748                                             ;continue if firmware
                           749 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           750     ; "pass" routine executed only if firmware called up pu confidence test
                           751     ; entry via stfs goes to "stfs_pass"
                           752
01D7                       753     PASS:
01D7 32C0                  754             XOR     al,al           ;RESULT BYTE = 0 -- test passed!
01D9 EB06                  755             JMP     short leave_test
                           756
01DB 8AC4                  757     FAIL:   mov     al,ah           ;put error number in al reg -- test failed
01DD 85F6                  758             test    si,si           ;test called by stfs or firmware?
01DF 750B                  759             jnz     stfs_fail       ;if stfs, then jump
                           760
01E1                       761     leave_test:
                           762
01E1 57                    763             push    di              ;restore previous return ip
01E2 C3                    764             RET                     ;return to boot firmware control (near)
                           765
01E3                       766     stfs_ret_check proc near
                           767
01E3 85F6                  768             test    si,si           ;routine called by stfs or firmware?
01E5 7501                  769             jnz     stfs_pass       ;if stfs, then get out
01E7 C3                    770             ret                     ;otherwise, continue (short return)
                           771
                           772     stfs_ret_check endp
                           773
01E8                       774     stfs_return     proc    far
                           775
01E8                       776     stfs_pass:
01E8 44                    777             inc     sp              ;fix up stack so return to external (stfs)
01E9 44                    778             inc     sp              ; if leaving ram test, then don't touch stack
                           779                                     ;   (note that two INCs saves a byte over ADD).
                           780
01EA                       781     stfs_ram_pass:
01EA 32C0                  782             xor     al,al           ;test passed, set al = 0
                           783
01EC                       784     stfs_fail:
01EC CB                    785             ret                     ;long (intersegment) return to 86
                           786                                     ;   routine in stfs or other external code
                           787
                           788     stfs_return     endp
                           789
                           790     device_test     ENDP
                           791 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           792     ; "read_id" performs one read of the ethernet address prom via 8237 dma
                           793     ; channel one. this procedure is used to test out that channel and also
                           794     ; to reset the address recognition circuitry before performing a on-board
                           795     ; loopback for tests "ver_sta_add" and "bad_addr".
                           796
C1ED                       797     read_id         proc    near
                           798
01ED E80C00                799             call    set_up_ch1_2_3_rcv      ;initialize channel 1 for rcv
01F0 B00D                  800             MOV     AL,0DH                  ;UNMASK CHANNEL 1 on 8237
01F2 E6CF                  801             OUT     PORT_8237_MASK,AL
01F4 E602                  802             OUT     CH1_AVAIL,AL            ;SET CHANNEL 1 AVAILABLE
01F6 B0FE                  803             MOV     AL,0FEH                 ;UNMASK interrupt 0 -- channel 1
01F8 E89F00                804             call    mask_and_radd           ;perform read address
01FB C3                    805             ret
                           806
                           807     read_id         endp
                           808 +1  Seject
```

```
LOC  OBJ                  LINE     SOURCE

                           809     ;  "set_up_ch_1_2_3_rcv" initializes all 8237 dma receive channels to perform
                           810     ;  a Read Address. Each channel will read 12H bytes into address F9FC0 into
                           811     ;  sram -- first two bytes are garbage, next 6 bytes is the Ethernet address,
                           812     ;  the subsequent 8 bytes are check bytes, and the last two bytes are 16-bit
                           813     ;  crc bytes -- totalling up to 11H bytes. The extra 8237 count seems to be
                           814     ;  neccessary for some unknown reason. If not, then the last byte may or may
                           815     ;  not be read in.
                           816
01FC                       817     set_up_ch1_2_3_rcv      proc near
                           818
01FC B0FF                  819             MOV     AL,0FFH
01FE E6E1                  820             OUT     PORT_8255_b,AL          ;CLEAR SERDES
0200 B012                  821             MOV     AL,12H                  ;XFER COUNT, RADD TERMINATED BY 8237
0202 E6C3                  822             OUT     PORT_8237_CH1_COUNT,AL  ;count = 12 starting at sram address 1FC0
0204 E6C5                  823             out     port_8237_ch2_count,al  ;    for channels 1,2 and 3
0206 E6C7                  824             out     port_8237_ch3_count,al
0208 B000                  825             MOV     AL,0
020A E6C3                  826             OUT     PORT_8237_CH1_COUNT,AL  ;MSB
020C E6C5                  827             out     port_8237_ch2_count,al  ;    for channels 1,2 and 3
020E E6C7                  828             out     port_8237_ch3_count,al
0210 B0C0                  829             mov     al,0C0H                 ;data at address 1FC0H
0212 E6C2                  830             OUT     PORT_8237_CH1_ADDR,AL
0214 E6C4                  831             OUT     PORT_8237_CH2_ADDR,AL
0216 E6C6                  832             out     port_8237_ch3_ADDR,al   ;    for channels 1,2 and 3
0218 B01F                  833             mov     al,1FH
021A E6C2                  834             out     port_8237_ch1_addr,al
021C E6C4                  835             out     port_8237_ch2_addr,al
021E E6C6                  836             out     port_8237_ch3_addr,al
0220 E606                  837             out     reset_ch_cntr,al        ;reset channel counter to dma 1
0222 C3                    838             ret
                           839
                           840     set_up_ch1_2_3_rcv      endp
                           841 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           842
                           843     ;*************************************************************************
                           844
                           845     ; ROUTINE PERFORMS ON BOARD VERIFY.
                           846     ; REQUIRES BL=8255 PORT B VALUE TO BE USED.
                           847     ; RETURNS BL=0 if transmit complete interrupt OCCURED, BL=FF
                           848     ; if none occurred
                           849
                           850     ;*************************************************************************
                           851
0223                       852     verify  proc near
                           853
0223 B00E                  854             mov     al,0EH                  ;unnask 8237 channel 0 transmit
0225 E6CF                  855             out     port_8237_mask,al       ; and put it there
0227 B0F7                  856             MOV     AL,0F7H                 ;UNMASK TXC INT
0229 E6F1                  857             OUT     PORT_8259_MASK,AL
022B E8C900                858             CALL    CLR_REQ                 ; IN CASE ALREADY PRESENT
                           859
                           860     ; wait for Carrier Sense inactive (i.e., no packets on Ethernet) before
                           861     ; performing on-board loopback
                           862
022E B0B2                  863             mov     al,0B2H                 ;enable rcv and disable xmt,verify, and
                           864                                             ;  read address to see CS bit of serdes
                           865                                             ;set also into normal mode to reject
                           866                                             ;  probable CRC err
0230 E6E1                  867             out     port_8255_b,al          ;set serdes
0232 B90006                868             mov     cx,600H
0235                       869     test_cs:
0235 E4E0                  870             in      al,port_8255_a          ;read serdes status port
0237 2404                  871             and     al,4                    ;test Carrier Sense bit
0239 E0FA                  872             loopnz  test_cs                 ;if bit = 1, then loop
                           873
023B 8AC3                  874             MOV     AL,BL                   ;BL PASSED AS SerDes mode PARAMETER
023D E6E1                  875             OUT     PORT_8255_b,AL
023F E602                  876             out     ch1_avail,al            ;enable reception of crc/length errors
0241 E600                  877             OUT     SET_TXSRT,AL            ; START VERIFY
0243 E89200                878             CALL    WAIT_FOR_INT_REQ
0246 3C00                  879             CMP     AL,0                    ;CHECK FOR REQUEST
                           880
0248 B0FF                  881             mov     al,0FFH                 ;reset serdes -- make sure serdes is
024A E6E1                  882             out     port_8255_b,al          ; NOT left in receive mode. note that
                           883                                             ; this will not reset crc/len err.
                           884
024C 7506                  885             JNZ     NC_TXC
024E E4E0                  886             IN      AL,PORT_8255_A          ;GET STATUS of verify
0250 8AD8                  887             mov     bl,al                   ;save serdes status in reg bl
0252 EB05                  888             jmp     short ret_near          ;return
                           889
0254                       890     NC_TXC:
0254 5B                    891             POP     BX                      ;clean up stack by eliminating a
                           892                                             ; short offset left on stack
0255 B415                  893             MOV     ah,15H                  ;ERROR CODE                     <====
0257 EB82                  894             JMP     short FAIL
                           895
0259                       896     ret_near:
```

```
LOC  OBJ                  LINE      SOURCE

0259 C3                    897              ret                          ;short return
                           898
                           899      verify  endp
                           900 +1   $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           901     ; "address_verify" performs an on-board loopback of packet which contains the
                           902     ; ethernet address of this board. it then check serdes status for recognition
                           903     ; of this address -- 8255 port a contains length error if address is not
                           904     ; recognized as its self. notice that stack is popped before call to "verify"
                           905     ; and pushed afterwards to support "verify"s jump to "fail" if error 15H
                           906     ; occurs.
                           907
025A                       908     address_verify          proc near
                           909
025A 5A                    910             pop     dx                      ;store return offset here
025B B0C2                  911             MOV     AL,0C2H
025D E6C0                  912             OUT     PORT_8237_CH0_ADDR,AL   ;LSB
025F B01F                  913             mov     al,1FH
0261 E6C0                  914             OUT     PORT_8237_CH0_ADDR,AL   ;MSB -- address 2 (or F9FC2H)
0263 B31A                  915             MOV     BL,1AH                  ;8255 port b -- xmt,rcv,promisc/
0265 E8BBFF                916             CALL    VERIFY
0268 52                    917             push    dx                      ;restore return offset
0269 2410                  918             AND     AL,10H                  ;check xmt/rcv status for length error
026B C3                    919             ret
                           920
                           921     address_verify          endp
                           922 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           923
                           924     ; write value in al reg into 8259 mask register. mask register is then
                           925     ; read back and compared versus original
                           926
C26C                       927     write_8259_mask         proc near
                           928
C26C 8AD8                  929             mov     bl,al           ;put expected value into bl reg
C26E E6F1                  930             OUT     PORT_8259_MASK,AL
C270 E4F1                  931             IN      AL,PORT_8259_MASK
C272 8AF8                  932             mov     bh,al
C274 3AFB                  933             CMP     bh,bl
C276 C3                    934             ret
                           935
                           936     write_8259_mask         endp
                           937
                           938 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           939     ; read the ethernet address prom into sram via the current dma channel
                           940
0277                       941     radd    proc near
                           942
0277 E87D00                943             CALL    CLR_REQ
027A B0E2                  944             MOV     AL,0E2H                 ;put serdes into xmt/rcv mode BUT
                           945                                             ; wait awhile before setting read
                           946                                             ; address mode (allow hardware to
                           947                                             ; settle first)
027C E605                  948             OUT     RESET_ERRS,AL
027E E6E1                  949             OUT     PORT_8255_b,AL          ;START half of READ ADDRESS
                           950
0280 B90500                951             mov     cx,5
0283 E2FE                  952     again:  loop    again                   ;wait at least 9.6 uSEC
                           953
0285 B0E0                  954             mov     al,0E0H
0287 E6E1                  955             out     port_8255_b,al          ;start read address mode
0289 E84C00                956             CALL    WAIT_FOR_INT_REQ        ;wait for completion interrupt
028C C3                    957             ret
                           958
                           959     radd    endp
                           960 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           961     ; "write_8255_port_b" write the value in al into 8255 port b
                           962
028D                       963     write_8255_port_b       proc near
                           964
028D 8AF8                  965             mov     bh,al           ;stash expected value in bh
028F E6E1                  966             OUT     PORT_8255_b,AL
0291 E4E1                  967             IN      AL,PORT_8255_b
0293 3AF8                  968             CMP     bh,al           ;expected = received?
0295 B0FF                  969             mov     al,0FFH
0297 E6E1                  970             out     port_8255_b,al  ;reset serdes
0299 C3                    971             ret
                           972
                           973     write_8255_port_b       endp
                           974 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           975     ; "mask_and_radd" writes the value in al reg into 8259 mask register and then
                           976     ; performs a read address. Two attempts to read address are performed (i.e.,
                           977     ; it may not work the first time).
                           978
029A                       979     mask_and_radd   proc near
                           980
029A E6F1                  981             OUT     PORT_8259_MASK,AL
                           982
                           983     ; begin read address attempt
                           984
029C B90200                985             mov     cx,2
029F 51                    986     retry:  push    cx
02A0 E8D4FF                987             call    radd                    ;perform read address via dma channel
02A3 59                    988             pop     cx
02A4 3CFF                  989             CMP     AL,0FFH                 ;CHECK FOR NO REQUEST
02A6 E1F7                  990             loopz   retry                   ;if it didn't work, try again
                           991
02A8 B0FF                  992             mov     al,0FFH
02AA E6E1                  993             out     port_8255_b,al          ;reset serdes
02AC C3                    994             ret
                           995
                           996     mask_and_radd   endp
                           997
                           998 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           999     ; reinitialize 8237, serdes, and error latch in preparation for on-board
                          1000     ; packet verify mode. 13H + 1 bytes will be transmitted -- 10H data bytes
                          1001     ; and 4 crc bytes (32-bit crc)
                          1002
02AD                      1003     xmt_reset       proc near
                          1004
02AD B013                 1005             MOV     AL,13H                  ;14H - 1 for 8237
02AF E6C1                 1006             OUT     PORT_8237_CH0_COUNT,AL
02B1 32C0                 1007             XOR     AL,AL                   ;MSB
02B3 E6C1                 1008             OUT     PORT_8237_CH0_COUNT,AL
02B5 E601                 1009             OUT     RESET_TXSRT,AL
02B7 B0FF                 1010             MOV     AL,0FFH
02B9 E6E1                 1011             OUT     PORT_8255_b,AL          ;CLEAR SERDES
02BB E605                 1012             OUT     RESET_ERRS,AL
02BD E606                 1013             out     reset_ch_cntr,al        ;initialize to channel 1
02BF C3                   1014             ret
                          1015
                          1016     xmt_reset       endp
                          1017 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          1018     ;*************************************************************************
                          1019
                          1020     ; MARCH PATTERN MEMORY TEST.
                          1021     ; BX=START ADDRESS, CX=COUNT, ASSUMES MEM ALREADY FILLED WITH AA.
                          1022     ; RETURNS TO ADDRESS STORED IN DX.
                          1023     ; RETURN VALUES: AL= 0 (PASS) OR 80H (FAIL)
                          1024
                          1025     ;*************************************************************************
                          1026
02C0                      1027     MAR_TEST label near
                          1028
02C0 803FAA               1029     march:  CMP     byte ptr [bx],0AAH      ;expected = received pattern?
02C3 750F                 1030             JNZ     march_error             ;if error, then jump
02C5 B055                 1031             mov     al,55H                  ;next pattern = 55H
02C7 8807                 1032             MOV     [BX],al
02C9 3A07                 1033             CMP     AL,[bx]                 ;expected = received pattern?
02CB 7507                 1034             JNZ     march_error             ;if error, then jump
02CD 43                   1035             INC     BX                      ;point to next ram byte to be tested
02CE E2F0                 1036             LOOP    march                   ;continue testing while cx <> 0
02D0 32C0                 1037             XOR     AL,AL                   ;RESULT=PASS
02D2 EB02                 1038             jmp     short march_exit
                          1039
02D4                      1040     march_error:
02D4 B080                 1041             MOV     AL,80H                  ;RESULT=FAIL
                          1042
02D6                      1043     march_exit:
02D6 FFE2                 1044             JMP     DX                      ;return to calling routine
                          1045
                          1046 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          1047       ;*******************************************************
                          1048       ;
                          1049       ; POLLS 8259 TO DETERMINE IF INTERRUPT PENDING
                          1050       ; COUNTS DOWN WHILE WAITING. RETURNS ON IR OR TIMEOUT
                          1051       ; RETURNS IN AL; 0= REQ OCCURED, FF=TIMEOUT
                          1052       ;
                          1053       ;*******************************************************
                          1054
02D8                      1055       WAIT_FOR_INT_REQ proc near
                          1056
02D8 B9FF00               1057               MOV     CX,0FFH                 ;set TIMEOUT value
                          1058
02DB                      1059       INT_WAIT:
02DB B00C                 1060               MOV     AL,POLL_8259_CMD
02DD E6F0                 1061               OUT     PORT_8259_A0_LOW,AL     ;POLL CMD
02DF E4F0                 1062               IN      AL,PORT_8259_A0_LOW     ;POLL
02E1 2480                 1063               AND     AL,80H                  ;REQ PENDING?
02E3 E1F6                 1064               LOOPZ   INT_WAIT                ;LOOP IF REQ/ AND TIMEOUT/
02E5 740B                 1065               JZ      NO_REQ
02E7 B020                 1066               MOV     AL,EOI_CMD              ; DID OCCUR
02E9 E6F0                 1067               OUT     PORT_8259_A0_LOW,AL     ;ISSUE EOI CMD
02EB B0FF                 1068               MOV     AL,0FFH                 ;RESET INT MASK
02ED E6F1                 1069               OUT     PORT_8259_MASK,AL
02EF 32C0                 1070               XOR     AL,AL                   ;RET STATUS
02F1 C3                   1071               ret
                          1072
02F2                      1073       NO_REQ:
02F2 B0FF                 1074               MOV     AL,0FFH                 ;RET STATUS
02F4 E6F1                 1075               OUT     PORT_8259_MASK,AL       ;MASK ALL INT
02F6 C3                   1076               ret
                          1077
                          1078       wait_for_int_req endp
                          1079
                          1080 +1    $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          1081       ;*************************************************
                          1082       ;
                          1083       ; CLEARS A PENDING INTERUPT REQUEST
                          1084       ;
                          1085       ;*************************************************
                          1086
02F7                      1087     CLR_REQ proc near
                          1088
02F7 B00C                 1089             MOV     AL,POLL_8259_CMD
02F9 E6F0                 1090             OUT     PORT_8259_A0_LOW,AL
02FB E4F0                 1091             IN      AL,PORT_8259_A0_LOW
02FD B020                 1092             MOV     AL,EOI_CMD
02FF E6F0                 1093             OUT     PORT_8259_A0_LOW,AL
0301 C3                   1094             ret
                          1095
                          1096     clr_req endp
                          1097
                          1098 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          1099
                          1100       ;*****************************************************
                          1101       ; COMPUTES 16 BIT CRC from MSByte to LSByte
                          1102       ; AX=CRC, BX=DATA PTR, CX=COUNT
                          1103       ;*****************************************************
                          1104
0302                      1105     CRC_16  proc
                          1106
0302 B8FFFF               1107             MOV     AX,0FFFFH       ;CRC
                          1108
0305                      1109     BYTE_LOOP:
0305 8A37                 1110             MOV     DH,[BX]         ;GET NEXT DATA BYTE
0307 53                   1111             PUSH    BX
0308 51                   1112             PUSH    CX
0309 B90800               1113             MOV     CX,8            ;COMPUTES 1 BIT AT A TIME
030C                      1114       BIT_LOOP:
030C 8BD8                 1115             MOV     BX,AX
030E 33DA                 1116             XOR     BX,DX
0310 81E3008C             1117             AND     BX,8000H        ;CRC15 XOR DW15
0314 85DB                 1118             test    BX,bx
0316 7504                 1119             JNZ     SHIFT_AND_XOR
                          1120
0318                      1121     SHIFT_ONLY:
0318 D1E0                 1122             SHL     AX,1            ;CRC=SHL(CRC,1)
031A EB07                 1123             JMP     short END_BIT_LOOP
                          1124
031C                      1125     SHIFT_AND_XOR:
031C D1E0                 1126             SHL     AX,1            ;CRC=SHL(CRC,1)
031E 352010               1127             XOR     AX,1020H        ;CRC=CRC XOR 1020H
0321 0C01                 1128             OR      al,0001H        ;CRC=CRC OR 1
                          1129
0323                      1130     END_BIT_LOOP:
0323 D1E2                 1131             SHL     DX,1            ;NEXT DATA BIT
0325 E2E5                 1132             LOOP    BIT_LOOP
0327 59                   1133             POP     CX              ;BYTE COUNT
0328 5B                   1134             POP     BX              ;BYTE PTR
0329 43                   1135             INC     BX
032A E2D9                 1136             LOOP    BYTE_LOOP
032C C3                   1137             ret
                          1138
                          1139     crc_16  endp
                          1140
                          1141 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          1142     ;  "checksum" calculates a twenty-four bit checksum on the specified range
                          1143     ;  of addresses.
                          1144
                          1145     ;  register usage:  dl -- high eight bits of calculated checksum
                          1146     ;                   ax -- low sixteen bits of calculated checksum
                          1147     ;                   bx -- pointer to Prom word being used in checksum.
                          1148     ;                         should be initialized to point to first word to
                          1149     ;                         be checksummed
                          1150     ;                   cx -- the number of words to be checksummed
                          1151
                          1152     ;  routine assumes ds = F000H
                          1153
032D                      1154     checksum        proc
                          1155
032D 33D2                 1156             xor     dx,dx                   ;clear out high eight bit
032F 33C0                 1157             xor     ax,ax                   ;  and low sixteen bits of checksum
                          1158
0331                      1159     sum_loop:
0331 0307                 1160             add     ax,[bx]                 ;add Prom word to present checksum
0333 7302                 1161             jnc     continue_summing        ;if no carry, then jump
0335 FEC2                 1162             inc     dl                      ;otherwise, increment upper eight
                          1163                                             ;  bits of checksum
0337                      1164     continue_summing:
0337 43                   1165             inc     bx                      ;point to next word
0338 43                   1166             inc     bx
0339 E2F6                 1167             loop    sum_loop                ;continue checksumming
033B C3                   1168             ret
                          1169
                          1170     checksum        endp
                          1171
                          1172 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                          1173     ; move test packet from within code segment into SRAM at address F9FE0H, length
                          1174     ; of 20 bytes -- 16 bytes data, 4 bytes crc. note that any packet used in pu
                          1175     ; confidence test verify is of length 10H. also note that since dx cannot
                          1176     ; be used as a pointer, then si has its contents exchanged with dx so that
                          1177     ; si can be used for that purpose.
                          1178
033C                      1179     tstpkt_to_sram  proc near
                          1180
033C 87F2                 1181             xchg    si,dx                           ;save si contents
033E BB5C0390       R     1182             mov     bx,offset cgroup:test_packet    ;MOVE TEST PACKET TO SRAM
0342 BEE09F               1183             mov     si,t_packet_dest
0345 B91400               1184             MOV     CX,14H
                          1185
0348                      1186     MV_STRING:
0348 2E8A07               1187             MOV     AL,CS:[BX]
034B 8804                 1188             MOV     [si],AL
034D 43                   1189             INC     BX
034E 46                   1190             inc     si
034F E2F7                 1191             LOOP    MV_STRING
0351 B0E0                 1192             mov     al,0ECH                 ;SRAM address = 1FE0H (F9FE0H)
0353 E6C0                 1193             OUT     PORT_8237_CH0_ADDR,AL
0355 B01F                 1194             mov     al,1FH
0357 E6C0                 1195             OUT     PORT_8237_CH0_ADDR,AL
0359 87F2                 1196             xchg    si,dx                   ;restore contents of si
035B C3                   1197             ret
                          1198
                          1199     tstpkt_to_sram endp
                          1200
035C 8F                   1201     TEST_PACKET     DB      8FH,55H,0AAH,96H,69H,0F0H,0FH,87H,4BH
035D 55
035E AA
035F 96
0360 69
0361 F0
0362 0F
0363 87
0364 4B
0365 2D                   1202                     DB      2DH,1EH,0AAH,99H,0EDH,7DH,0ADH
0366 1E
0367 AA
0368 99
0369 ED
036A 7D
036B AD
036C 65                   1203     packet_crc      db      65H,0DAH,4FH,0EEH                       ;32-bit crc
036D DA
036E 4F
036F EE
                          1204
                          1205 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

0370                      1206     ctjumptable:
                          1207
0370 4800           R     1208           dw      cgroup:dram_ripple              ; stfs procedure number 30
0372 5E00           R     1209           dw      cgroup:sram_ripple              ; stfs procedure number 31
0374 7400           R     1210           dw      cgroup:low_rom_test             ; stfs procedure number 32
0376 9400           R     1211           dw      cgroup:upper_rom_test           ; stfs procedure number 33
0378 B600           R     1212           dw      cgroup:t8255                    ; stfs procedure number 34
037A FD00           R     1213           dw      cgroup:t8253_0                  ; stfs procedure number 35
037C 1601           R     1214           dw      cgroup:t8253_1                  ; stfs procedure number 36
037E 2C01           R     1215           dw      cgroup:t8253_2                  ; stfs procedure number 37
0380 4201           R     1216           dw      cgroup:dma1                     ; stfs procedure number 38
0382 4C01           R     1217           dw      cgroup:dma2                     ; stfs procedure number 39
0384 5E01           R     1218           dw      cgroup:dma3                     ; stfs procedure number 40
0386 7001           R     1219           dw      cgroup:check_crc                ; stfs procedure number 41
0388 8601           R     1220           dw      cgroup:broadcast_rcv            ; stfs procedure number 42
038A 9A01           R     1221           dw      cgroup:ver_crc                  ; stfs procedure number 43
038C AE01           R     1222           dw      cgroup:ver_sta_add              ; stfs procedure number 44
038E BE01           R     1223           dw      cgroup:bad_addr                 ; stfs procedure number 45
                          1224
                          1225     ; utility procedures (all declared near) used by PU confidence test
                          1226
0390 4F00           R     1227           dw      cgroup:dram_loop                ; stfs procedure number 46
0392 6500           R     1228           dw      cgroup:sram_loop                ; stfs procedure number 47
                          1229
----                      1230     code    ends
                          1231
                          1232     END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```