```
a ,h w
A ::                                                                    SEQ 000
1

ZQNA1           CZQNADO DEQNA FUNCTIONAL TEST                 14-Mar-1985 13:09:10    VAX-11 Bliss-16 V4.1-582              Page   1
                                                              14-Mar-1985 13:07:35    DISK$USER2:[MARSHALL.DEQNA]ZQNA1.BLI;4  (1)

;    0001  0     MODULE ZQNA1 (%TITLE 'CZQNADO DEQNA FUNCTIONAL TEST'
;    0002  0                   IDENT = 'V01.0',
;    0003  0                   ADDRESSING_MODE(ABSOLUTE),
;    0004  0                   LANGUAGE(BLISS16)) =
;    0005  0     %SBTTL 'GLOBAL DEFINITION MODULE'
;    0006  0
;    0007  1     BEGIN
;    0008  1
; C  0009  1     %(
; C  0010  1                               IDENTIFICATION
; C  0011  1                               --------------
; C  0012  1
; C  0013  1             PRODUCT CODE:     AC-T614D-MC
; C  0014  1
; C  0015  1             PRODUCT NAME:     CZQNADO DEQNA FUNCTIONAL TEST
; C  0016  1
; C  0017  1             PRODUCT DATE:     MARCH 14, 1985
; C  0018  1
; C  0019  1             MAINTAINER:       MSD DIAGNOSTIC ENGINEERING
; C  0020  1
; C  0021  1             AUTHOR:           S. MAZURCZYK
; C  0022  1
; C  0023  1
; C  0024  1             COPYRIGHT (C) 1984,1985
; C  0025  1
; C  0026  1             DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
; C  0027  1
; C  0028  1             THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
; C  0029  1             COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
; C  0030  1             ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
; C  0031  1             MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
; C  0032  1             EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
; C  0033  1             TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
; C  0034  1             REMAIN IN DEC.
; C  0035  1
; C  0036  1             THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
; C  0037  1             AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
; C  0038  1             CORPORATION.
; C  0039  1
; C  0040  1             DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
; C  0041  1             SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
; C  0042  1
; C  0043  1             THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:
; C  0044  1
; C  0045  1                     DIGITAL          PDP              UNIBUS           MASSBUS
; C  0046  1                     DEC              DECUS            DECTAPE
; C  0047  1
; C  0048  1
; C  0049  1
```

```
; C 0050  1
; C 0051  1
; C 0052  1
; C 0053  1
; C 0054  1
; C 0055  1
; C 0056  1                          TABLE OF CONTENTS
; C 0057  1                          ******************
; C 0058  1
; C 0059  1
; C 0060  1                  1.0     GENERAL INFORMATION
; C 0061  1                  1.1     PROGRAM ABSTRACT
; C 0062  1                  1.2     SYSTEM REQUIREMENTS
; C 0063  1                  1.3     RELATED DOCUMENTS AND STANDARDS
; C 0064  1                  1.4     ASSUMPTIONS
; C 0065  1
; C 0066  1                  2.0     OPERATING INSTRUCTIONS
; C 0067  1                  2.1     COMMANDS
; C 0068  1                  2.2     SWITCHES
; C 0069  1                  2.3     FLAGS
; C 0070  1                  2.4     HARDWARE QUESTIONS
; C 0071  1                  2.5     SOFTWARE QUESTIONS
; C 0072  1                  2.6     QUICK STARTUP PROCEDURE
; C 0073  1
; C 0074  1                  3.0     ERROR INFORMATION
; C 0075  1
; C 0076  1                  4.0     TEST SUMMARIES
; C 0077  1
; C 0078  1                  5.0     MAINTENANCE HISTORY
; C 0079  1
```

```
;  C 0080  1
;  C 0081  1               1.0  GENERAL INFORMATION
;  C 0082  1                    --------- ---------
;  C 0083  1
;  C 0084  1               1.1  PROGRAM ABSTRACT
;  C 0085  1                    ----------------
;  C 0086  1
;  C 0087  1               The DIGITAL ETHERNET Q-Bus Network Adapter (DEQNA) Field Functional
;  C 0088  1               Diagnostic Program (ZQNA) performs extensive functional testing of
;  C 0089  1               the DEQNA/M7504 module for Q18 or Q22-Bus based PDP-11 systems. ZQNA
;  C 0090  1               program attempts to isolate faults to the following Field Replacable
;  C 0091  1               Units (FRU's): DEQNA, bulkhead assembly, transceiver cable, circuit
;  C 0092  1               breaker ( fuse in bulkhead assembly ) and transceiver. This software
;  C 0093  1               also attempts to localize faults to the functional areas of the DEQNA
;  C 0094  1               module.
;  C 0095  1
;  C 0096  1               A test operator controls testing of the module from a console
;  C 0097  1               ( hard copy or CRT ).
;  C 0098  1
;  C 0099  1               This  diagnostic  has been  written for use  with the diagnostic
;  C 0100  1               runtime services  software (supervisor).  These services provide
;  C 0101  1               the  interface to  the operator and to the software environment.
;  C 0102  1               For a complete description of the runtime services, refer to the
;  C 0103  1               XXDP+ user's  manual.  There is a brief description of the runtime
;  C 0104  1               services in section 2 of this document.
;  C 0105  1
;  C 0106  1
;  C 0107  1               1.2  SYSTEM REQUIREMENTS
;  C 0108  1                    ------------------
;  C 0109  1
;  C 0110  1               The ZQNA software operates on a typical 'newer PDP-11 processor' system
;  C 0111  1               that has one or two DEQNA modules on the Q18 or Q22 system bus.
;  C 0112  1               The internal and internal/extended loopback mode tests do not
;  C 0113  1               require the transceiver or the loopback connector to be unplugged.
;  C 0114  1               The external loopback mode may be used with a terminated transceiver
;  C 0115  1               that has no network cable attached.
;  C 0116  1
;  C 0117  1               Testing DEQNA module and its interface to the Ethernet requires
;  C 0118  1               following hardware:
;  C 0119  1
;  C 0120  1               - Typical system ( PDP-11/23 Plus, ORION ) with Q-Bus,
;  C 0121  1               - DEQNA module,
;  C 0122  1               - Minimum of 28K words of memory ( supporting block or non-block mode ),
;  C 0123  1               - Console terminal,
;  C 0124  1               - Loopback connector ( male loopback connector, Part # 12 221 96-01 ),
;  C 0125  1               - Bulkhead assembly,
;  C 0126  1               - Transceiver cable,
;  C 0127  1               - and transceiver ( H4000 ).
;  C 0128  1
```

```
; C 0129  1          1.3  RELATED DOCUMENTS AND STANDARDS
; C 0130  1               -------------------------------
; C 0131  1
; C 0132  1          XXDP+ Supervisor/User's Manual - ( CHQUS ).
; C 0133  1
; C 0134  1
; C 0135  1          1.4  ASSUMPTIONS
; C 0136  1               ----- -- --
; C 0137  1
; C 0138  1          It is assumed that the system has been tested without DEQNA and found
; C 0139  1          working before this diagnostic is run, or that DEQNA DEC/X11
; C 0140  1          Exerciser has dropped DEQNA option module when running system test.
; C 0141  1
; C 0142  1
; C 0143  1          2.0  OPERATING INSTRUCTIONS
; C 0144  1               ----------------------
; C 0145  1
; C 0146  1          This  section contains a brief description of the runtime services.
; C 0147  1          for detailed information, refer to the XXDP+ User's Manual (CHQUS).
; C 0148  1
; C 0149  1
; C 0150  1          2.1  COMMANDS
; C 0151  1               --------
; C 0152  1
; C 0153  1          There  are eleven legal commands for the diagnostic runtime services
; C 0154  1          (supervisor).  This  section lists  the commands  and gives  a  very
; C 0155  1          brief description of them.  The XXDP+ User's Manual has more details.
; C 0156  1
; C 0157  1                  COMMAND         EFFECT
; C 0158  1                  -------         ------------------------------
; C 0159  1
; C 0160  1                  START           Start the diagnostic from an initial state
; C 0161  1                  RESTART         Start the diagnostic without initializing
; C 0162  1                  CONTINUE        Continue at test that was interrupted (after ^C)
; C 0163  1                  PROCEED         Continue from an error halt
; C 0164  1                  EXIT            Return to XXDP+ monitor (XXDP+ operation only!)
; C 0165  1                  ADD             Activate a unit for testing (all units are
; C 0166  1                                  considered to be active at start time
; C 0167  1                  DROP            Deactivate a unit
; C 0168  1                  PRINT           Print statistical information (if implemented
; C 0169  1                                  by the diagnostic - section 4.0)
; C 0170  1                  DISPLAY         Type a list of all device information
; C 0171  1                  FLAGS           Type the state of all flags (see section 2.3)
; C 0172  1                  ZFLAGS          Clear all flags (see section 2.3)
; C 0173  1
; C 0174  1          A command can be recognized by the first three characters.
; C 0175  1          So you may,  for example, type  "STA" instead  of "START".
```

```
; C 0176  1          2.2  SWITCHES
; C 0177  1               --------
; C 0178  1
; C 0179  1          There are  several  switches which are used to modify supervisor
; C 0180  1          operation. These  switches are  appended  to the legal  commands.
; C 0181  1          All  of  the  legal  switches  are  tabulated below with a brief
; C 0182  1          description of each. In the descriptions below, a decimal number
; C 0183  1          is designated by "DDDDD".
; C 0184  1
; C 0185  1                  SWITCH          EFFECT
; C 0186  1                  ------          ------------------------------
; C 0187  1
; C 0188  1                  /TESTS:LIST     Execute only those tests specified in
; C 0189  1                                  the list.  List is a string of test
; C 0190  1                                  numbers, for example - /TESTS:1:5:7-10.
; C 0191  1                                  This list will cause tests 1,5,7,8,9,10 to
; C 0192  1                                  be run.  All other tests will not be run.
; C 0193  1                  /PASS:DDDDD     Execute DDDDD passes (DDDDD = 1 to 64000)
; C 0194  1                  /FLAGS:FLGS     Set specified flags.  flags are described
; C 0195  1                                  in section 2.3.
; C 0196  1                  /EOP:DDDDD      Report end of pass message after every
; C 0197  1                                  DDDDD passes only.  (DDDDD = 1 to 64000)
; C 0198  1                  /UNITS:LIST     TEST/ADD/DROP only those units specified
; C 0199  1                                  in the list.  List example - /UNITS:0:5:10-12
; C 0200  1                                  use units 0,5,10,11,12 (unit numbers = 0-63)
; C 0201  1
; C 0202  1          Example of switch usage:
; C 0203  1
; C 0204  1                  START/TESTS:1-5/PASS:1000/EOP:100
; C 0205  1
; C 0206  1          The effect of this command will be:
; C 0207  1
; C 0208  1                  1. Tests 1 through 5 will be executed.
; C 0209  1                  2. All  units will be tested 1000 times.
; C 0210  1                  3. The  end of  pass messages will be
; C 0211  1                     printed after each 100 passes only.
; C 0212  1
; C 0213  1          A Switch can be recognized by the first three characters.  You
; C 0214  1          may, for  example, type  "/TES:1-5" instead  of "/TESTS:1-5".
; C 0215  1
```

```
;  C 0216  1          Below is a table that specifies which switches can be used by
;  C 0217  1          each command.
;  C 0218  1
;  C 0219  1                  TESTS   PASS    FLAGS   EOP     UNITS
;  C 0220  1                  ----- ------------------------------
;  C 0221  1          START     X       X       X       X       X
;  C 0222  1          RESTART   X       X       X       X       X
;  C 0223  1          CONTINUE          X       X       X
;  C 0224  1          PROCEED                   X
;  C 0225  1          DROP                                      X
;  C 0226  1          ADD                                       X
;  C 0227  1          PRINT
;  C 0228  1          DISPLAY                                   X
;  C 0229  1          FLAGS
;  C 0230  1          ZFLAGS
;  C 0231  1          EXIT
```

```
;C 0232  1          2.3  FLAGS
;C 0233  1               -----
;C 0234  1
;C 0235  1          Flags are used to set up certain operational parameters such as
;C 0236  1          looping  on error.  All flags are cleared at startup and remain
;C 0237  1          cleared  until explicitly  set  using the  flags switch.  Flags
;C 0238  1          are  also cleared  after  a start  command unless set using the
;C 0239  1          flag  switch.  The ZFLAGS  command  may also  be used  to clear
;C 0240  1          all flags.  with the exception of the START and ZFLAGS commands,
;C 0241  1          No  commands affect the state of the flags;  they remain set or
;C 0242  1          cleared as specified by the last flag switch.
;C 0243  1
;C 0244  1                  FLAG            EFFECT
;C 0245  1                  ----            ---------------------------------
;C 0246  1
;C 0247  1                  HOE             Halt on error - control is returned to
;C 0248  1                                  runtime services command mode
;C 0249  1                  LOE             Loop on error
;C 0250  1                  IER*            Inhibit all error reports
;C 0251  1                  IBR*            Inhibit  all error reports except first
;C 0252  1                                  level  (first level contains error type,
;C 0253  1                                  number, PC, test and unit)
;C 0254  1                  IXR*            Inhibit extended error reports (those
;C 0255  1                                  called by PRINTX macro's)
;C 0256  1                  PRI             Direct messages to line printer
;C 0257  1                  PNT             Print test number as test executes
;C 0258  1                  BOE             "BELL" on error
;C 0259  1                  UAM             Unattended mode (no manual intervention)
;C 0260  1                  ISR             Inhibit statistical reports (does not apply
;C 0261  1                                  to diagnostics which do not support statis-
;C 0262  1                                  tical reporting)
;C 0263  1                  IDR             Inhibit program dropping of units
;C 0264  1                  ADR             Execute autodrop code
;C 0265  1                  LOT             Loop on test
;C 0266  1                  EVL             Execute evaluation (on diagnostics which
;C 0267  1                                  have evaluation support)
;C 0268  1
;C 0269  1                         *error messages are described in section 3.0
;C 0270  1
;C 0271  1          See the XXDP+ User's Manual for more details on flags.  You may
;C 0272  1          specify  more  than one flag with the flag switch.  For example,
;C 0273  1          to  cause  the program to  loop on error, inhibit error reports
;C 0274  1          and  type a "BELL"  on error,  you may use the following string:
;C 0275  1
;C 0276  1                  /FLAGS:LOE:IER:BOE
```

```
; C 0277  1               2.4  HARDWARE QUESTIONS
; C 0278  1                    ------ -----------
; C 0279  1
; C 0280  1               When a diagnostic is started, the DRS prompts the user for hardware
; C 0281  1               information by displaying
; C 0282  1
; C 0283  1               "CHANGE HW (L) ?"
; C 0284  1
; C 0285  1               you must answer "Y" after a start command unless the hardware
; C 0286  1               information has been "preloaded" using the Setup Utility (see chapter
; C 0287  1               6 of the XXDP+ User's Manual). When you answer this question with a
; C 0288  1               "Y", the DRS asks for the number of units. You will then be asked the
; C 0289  1               following questions for each unit.
; C 0290  1
; C 0291  1               # OF DEVICES (D) ?
; C 0292  1
; C 0293  1               Answer with the number of units to be tested (no default). This answer
; C 0294  1               will determine how many times the following questions are asked.
; C 0295  1               One (1) device must be specified.
; C 0296  1
; C 0297  1               DEQNA I/O PAGE  ADR  (O) 174440 ?
; C 0298  1
; C 0299  1               Answer with the address of the I/O page register assigned for one
; C 0300  1               of the DEQNA devices. The I/O page addresses permited are: 174440
; C 0301  1               and 174460.
; C 0302  1
; C 0303  1               INTERRUPT VECTOR ADR (O) 700 ?
; C 0304  1
; C 0305  1               Answer with the interrupt vector address of the DEQNA module.
; C 0306  1               Interrupt vector address for device at I/O page address 174440 is 700
; C 0307  1               oct. and that for I/O page address of 174460 is 704 oct.
; C 0308  1
; C 0309  1
```

```
;C 0310  1          2.5  SOFTWARE QUESTIONS
;C 0311  1               ------- --- -----
;C 0312  1
;C 0313  1          After you have answered the hardware questions or after a RESTART or
;C 0314  1          CONTINUE command, the DRS asks for software parameters. These
;C 0315  1          parameters govern some diagnostic specific operation modes.  You will
;C 0316  1          be prompted by
;C 0317  1
;C 0318  1          CHANGE SW (L) ?
;C 0319  1
;C 0320  1          if you wish to change any parameters, answer by typing "Y". The
;C 0321  1          software questions and the default values are described in the next
;C 0322  1          paragraph(s).
;C 0323  1
;C 0324  1          DO YOU WANT TO TEST SANITY TIMER (L)?
;C 0325  1
;C 0326  1          If you wish to test the Sanity Timer logic, answer by typing "Y".
;C 0327  1          Whenever this question is answered with a "Y" following question
;C 0328  1          will follow:
;C 0329  1
;C 0330  1          SANITY TIMER TIMEOUT VALUE (D)?
;C 0331  1
;C 0332  1          Answer with the TIMEOUT VALUE being a decimal number between 0 and 7.
;C 0333  1          Use table below to select desired TIMEOUT VALUE.
;C 0334  1
;C 0335  1                  TIMEOUT VALUE     TIMEOUT PERIOD IN SEC.
;C 0336  1                  -------------     ----------------------
;C 0337  1
;C 0338  1                        0                    1/4
;C 0339  1                        1                     1
;C 0340  1                        2                     4
;C 0341  1                        3                    16
;C 0342  1                        4                    60
;C 0343  1                        5                   240
;C 0344  1                        6                   960
;C 0345  1                        7                  3840
;C 0346  1
;C 0347  1          EXTERNAL LOOPBACK MODE (L)?
;C 0348  1
;C 0349  1          Answer with "Y" if you want to execute include "TEST 7" in the test
;C 0350  1          sequence. "TEST 7" is the only test that uses external loopback mode.
;C 0351  1          "N" inhibits execution of "TEST 7".
;C 0352  1
;C 0353  1          SYSTEM HAS BLOCK-MODE MEMORY (L)?
;C 0354  1
;C 0355  1          Answer with "Y" if the system has block-mode memory and "N" if
;C 0356  1          it has non block-mode memory.
;C 0357  1
;C 0358  1          IS LOOPBACK CONNECTOR IN DEQNA (L)?
;C 0359  1
;C 0360  1          Answer with "Y" if loopback connector is in the back of the DEQNA
;C 0361  1          module.
```

```
; C 0362  1
; C 0363  1
; C 0364  1
; C 0365  1
; C 0366  1
; C 0367  1
; C 0368  1
; C 0369  1
; C 0370  1
; C 0371  1
; C 0372  1
; C 0373  1
; C 0374  1
; C 0375  1
; C 0376  1
; C 0377  1
; C 0378  1
; C 0379  1
; C 0380  1
; C 0381  1
; C 0382  1
; C 0383  1
; C 0384  1
; C 0385  1
; C 0386  1
; C 0387  1
; C 0388  1
```

2.6 QUICK START UP PROCEDURE (XXDP+)

To start-up this program:

- Boot XXDP+
- Give the date
- Type "R Name", where Name is the name of the BIN file for this program
- Type "START"
- Answer the "CHANGE HW" question with "Y"
- Answer all the hardware questions
- Answer the "CHANGE SW" question with "Y"
- Answer all the software questions

When you follow this procedure you will be using only the defaults for flags and software parameters. These defaults are described in the previous sections.

```
; C 0389  1          3.0  ERROR INFORMATION
; C 0390  1               ----  ---- --- --
; C 0391  1
; C 0392  1
; C 0393  1          TYPES OF ERROR MESSAGES
; C 0394  1          - -  ----  --------
; C 0395  1
; C 0396  1          There  are three  levels  of error  messages that may be issued by
; C 0397  1          a diagnostic: general, basic and extended.  General error messages
; C 0398  1          are always printed unless the IBE and/or IER flag is set. The general
; C 0399  1          error message is of the form:
; C 0400  1
; C 0401  1                  NAME  ER_TYPE  ER_NO  UNIT_NO  TEST_NO  PC_ADDR
; C 0402  1
; C 0403  1          ,where;
; C 0404  1                  NAME = Diagnostic name
; C 0405  1                  ER_TYPE = Error type ( all errors are HARD )
; C 0406  1                  ER_NO = Error number
; C 0407  1                  UNIT_NO = 0
; C 0408  1                  TEST_NO = Test and subtest where error occurred
; C 0409  1                  PC_ADDR = Program Counter contents
; C 0410  1
; C 0411  1          Basic error messages are messages that contain some additional
; C 0412  1          information about the error. These are always printed unless one or
; C 0413  1          more of the DRS error flag(s) ( IBE, IXE, IER ) is set. These
; C 0414  1          messages are printed before the associated general message.
; C 0415  1
; C 0416  1          Extended error messages contain supplementary error information such
; C 0417  1          as register contents or good/bad data. These are always printed unless
; C 0418  1          the IXE and/or IER flag is set. These messages are printed after the
; C 0419  1          associated general error message and any associated basic error
; C 0420  1          messages. A typical extended error message might have a following
; C 0421  1          format:
; C 0422  1
; C 0423  1
; C 0424  1                  TRANSMIT DESCRIPTOR LIST        RECEIVE DESCRIPTOR LIST
; C 0425  1
; C 0426  1                     Flag Word                       Flag Word
; C 0427  1                     Low  Order Addr Bits            Low  Order Addr Bits
; C 0428  1                     High Order Addr Bits            High Order Addr Bits
; C 0429  1                     Packet Length (byte)            Packet Length (byte)
; C 0430  1                     Status Word 1                   Status Word 1
; C 0431  1                     Status Word 2                   Status Word 2
; C 0432  1
```

```
;  C 0433  1          SPECIFIC ERROR MESSAGES
;  C 0434  1          ---------- -----------
;  C 0435  1
;  C 0436  1          The following are possible error messages.
;  C 0437  1
;  C 0438  1          DEQNA FATAL ERROR DETECTED
;  C 0439  1          ACTUAL DATA = octal number  EXPECTED DATA = octal number
;  C 0440  1          BAD CSR: ACT = octal number  EXP = octal number
;  C 0441  1          BAD TRANSMIT FLAG WORD: ACT = octal number  EXP = octal number
;  C 0442  1          BAD TRANSMIT STATUS WORD 1: ACT = octal number  EXP = octal number
;  C 0443  1          BAD RECEIVE FLAG WORD: ACT = octal number  EXP = octal number
;  C 0444  1          BAD RECEIVE STATUS WORD 1: ACT = octal number  EXP = octal number
;  C 0445  1          BAD RECEIVE BUFFER LENGTH: ACT = octal number  EXP = octal number
;  C 0446  1          BAD CSR = octal number
;  C 0447  1          LOOPBACK PACKET UNABLE TO SET CA BIT, CSR = octal number
;  C 0448  1          LOOPBACK PACKET UNABLE TO CLEAR CA BIT, CSR = octal number
;  C 0449  1          CA BIT OK, BUT RI BIT IS NOT ON, CSR = octal number
;  C 0450  1          CA BIT IN THE CSR WAS SET TOO EARLY, CSR = octal number
;  C 0451  1          BAD CSR, EXPECTED, XL AND RL ( BITS 4,5 ) TO BE RESET TO 0
;  C 0452  1          BAD CSR, EXPECTED, XL AND RL ( BITS 4,5 ) TO BE SET TO 1
;  C 0453  1          BAD CSR, EXPECTED, RI ( BIT 15 ) TO BE SET TO 1
;  C 0454  1          BAD CSR, EXPECTED, XI ( BIT 7 ) TO BE SET TO 1
;  C 0455  1          BAD CSR, EXPECTED, NI ( BIT 2 ) TO BE SET TO 1
;  C 0456  1          BAD CSR, EXPECTED, NI ( BIT 2 ) TO BE RESET TO 0
;  C 0457  1
;  C 0458  1          CSR ADR = octal number ACTUAL = octal number EXPECTED = octal number
;  C 0459  1          UNABLE TO RESET DEQNA: ADR: address  CSR = octal number
;  C 0460  1          WAIT ABOUT number SECOND(S)
;  C 0461  1          SANITY TIMER TIMED OUT AS EXPECTED
;  C 0462  1          NO SANITY TIMER INTERRUPT DETECTED
;  C 0463  1          DISCONNECT TRANSCEIVER CABLE FROM BULKHEAD ASSEMBLY AND CONNECT
;  C 0464  1             LOOPBACK CONNECTOR TO BULKHEAD ASSEMBLY, THEN RETEST
;  C 0465  1          DISCONNECT BULKHEAD ASSEMBLY FROM DEQNA AND CONNECT
;  C 0466  1          LOOPBACK CONNECTOR TO DEQNA, THEN RETEST
;  C 0467  1          CHECK FOR LOOSE WIRES IN A LOOPBACK CONNECTOR OR USE DIFFERENT
;  C 0468  1             LOOPBACK CONNECTOR, THEN RETEST
;  C 0469  1          REPLACE DEQNA, THEN RETEST
;  C 0470  1          REPLACE BULKHEAD CONNECTOR, THEN RETEST
;  C 0471  1          DISCONNECT TRANSCEIVER CABLE FROM TRANSCEIVER AND CONNECT IT TO
;  C 0472  1             LOOPBACK CONNECTOR AND BULKHEAD ASSEMBLY
;  C 0473  1          REPLACE TRANSCEIVER CABLE, THEN RETEST
;  C 0474  1          REPLACE TRANSCEIVER, THEN RETEST
;  C 0475  1          REPLACE THE FUSE IF BAD, THEN RETEST
;  C 0476  1          BAD RECEIVE DESCRIPTOR:
;  C 0477  1          BAD TRANSMIT DESCRIPTOR:
;  C 0478  1          BAD RECEIVE BUFFER:
;  C 0479  1          ACTUAL = octal number  EXPECTED = octal number  INDEX = decimal number
;  C 0480  1          DMA OPERATION TAKES TOO LONG
;  C 0481  1          TOO MANY DEVICES
;  C 0482  1          THERE WAS A POWER FAIL - WAITING
;  C 0483  1          WAIT ABOUT decimal number MINUTE(S)
;  C 0484  1          WAIT ABOUT decimal number HOUR
;  C 0485  1          IF NO RESET, TYPE ANY CHARACTER TO EXIT FROM TEST
```

```
;  C 0486  1          TDR VALUE = O%N' ),
;  C 0487  1          BAD CSR, BITS STUCK AT 0:
;  C 0488  1          BAD CSR, BITS STUCK AT 1:
;  C 0489  1          SOFTWARE RESET UNABLE TO CLEAR CSR STATIC BITS:
;  C 0490  1          BAD STATION ADDRESS CHECKSUM: ACT = octal number  EXP = octal number
;  C 0491  1          BAD STATION ADDRESS: station address
;  C 0492  1          BAD DEQNA I/O PAGE REGISTER: register address
;  C 0493  1          BAD CSR, EXPECTED RL ( BIT 5 ) TO BE SET TO 0
;  C 0494  1          BAD B/D PROM CHECKSUM: INDEX = octal number ACT = octal number  EXP = octal number
;  C 0495  1          B/D PROM CHECKSUM OFFSET = octal number  ACT = octal number  EXP = octal number
;  C 0496  1          BAD INTERRUPT: ADR = octal number  ACT LEV = octal number  EXP LEV = octal number
;  C 0497  1          REGISTER FAILED TO RESPOND AT ADDRESS:  register address
;  C 0498  1
```

```
;  C 0499  1               4.0  TEST SUMMARIES
;  C 0500  1                    --------------
;  C 0501  1
;  C 0502  1               Each test has its own test summary; therefore, test summaries are not
;  C 0503  1               included here.
;  C 0504  1
;  C 0505  1               5.0    MAINTENANCE HISTORY
;  C 0506  1                      ----- -------------
;  C 0507  1
;  C 0508  1               Rev. CZQNAC0 changed to CZQNADO in March, 1985 by Howard L. Marshall:
;  C 0509  1
;  C 0510  1               Modified DMA Timing Test, Test #14,  , allow the test to operate
;  C 0511  1               properly in the faster 18 MHz. KDJ11-B/BF. Changes are noted by "$$$"
;  C 0512  1               in the comment field of added of changed lines.
;  C 0513  1
;     0514  1      )%
;     0515  1
```

```
;    0516  1
;    0517  1     LIBRARY 'QNALIB'·
;    0518  1     REQUIRE 'BLSMAC.REQ';                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;    2008  1
;    2009  1     !++
;    2010  1     !       DEFINE THE NUMBER OF TESTS IN THIS DIAGNOSTIC
;    2011  1     !--
;    2012  1
;    2013  1     PSECT
;    2014  1         CODE = AA$CODE$;
;    2015  1
;    2016  1     LITERAL
;    2017  1         DS$NBR_OF_TESTS = 21;
;    2018  1
;    2019  1     EQUALS;
;    2020  1
;    2021  1     POINTER (ALL);
;    2022  1
;    2023  1     !++
;    2024  1     !        THE PROGRAM HEADER IS THE INTERFACE BETWEEN THE DIAGNOSTIC PROGRAM
;    2025  1     !        AND THE SUPERVISOR.
;    2026  1     !--
;    2027  1
;    2028  1     HEADER (%ASCII'CZQNA ',%ASCII'D',%ASCII'0', 120, 0, PRI00);
;    2029  1
;    2030  1
;    2031  1     !++
;    2032  1     !       NO POINTERS ARE OPTIONAL USING BLISS. MAKE SURE THE FOLLOWING
;    2033  1     !       SECTIONS OF CODE ARE IN PLACE (IN THE CORRECT SKELS),EVEN IF
;    2034  1     !       THE SECTIONS ARE BLANK.
;    2035  1     !
;    2036  1     !       ARGUMENT        FUNCTION
;    2037  1     !       --------        ------
;    2038  1     !       RPT             REPORT CODE
;    2039  1     !       SW              SOFTWARE TABLE
;    2040  1     !       SFT             SOFTWARE TABLE QUESTIONS
;    2041  1     !       AU              ADD CODE
;    2042  1     !       DU              DROP CODE
;    2043  1     !       TBL             ERROR TABLE
;    2044  1     !       SETUP           ASSEMBLED P-TABLES
;    2045  1     !
;    2046  1     !       CHANGE THE "HEADER" TO CONTAIN THE PROPER ARGUMENTS.
;    2047  1     !       ARGUMENTS ARE: NAME,REV,PATCH,LONGEST TEST TIME,TYPE
;    2048  1     !       WHERE "TYPE" = 0 FOR SEQUENTIAL DIAGNOSTIC AND =1
;    2049  1     !       FOR EXERCISER.  THERE IS ALSO AN OPTIONAL SIXTH ARGUMENT
;    2050  1     !       WHICH SPECIFIES THE PROCESSOR PRIORITY TO BE SET WHEN
;    2051  1     !       STARTING THE DIAGNOSTIC (DEFAULT IS 0).
;    2052  1     !--
;    2053  1
;    2054  1
```

```
;    2055  1     %SBTTL 'DISPATCH TABLE'
;    2056  1
;    2057  1     DISPATCH (DS$NBR_OF_TESTS);
;    2058  1
;    2059  1     !++
;    2060  1     !       THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
;    2061  1     !       IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
;    2062  1     !
;    2063  1     !        CHANGE THE LITERAL DECLARATION OF DS$NBR_OF_TESTS TO BE
;    2064  1     !        THE NUMBER OF HARDWARE TESTS IN YOUR PROGRAM.
;    2065  1     !
;    2066  1     !--
;    2067  1
;    2068  1     ERRTBL;
;    2069  1
;    2070  1     !++
;    2071  1     !       THE ERRTBL MACRO IS REQUIRED WHETHER OR NOT YOU REPORT ERRORS USING
;    2072  1     !       THE "ERROR" MACRO.  THE ERRTBL MACRO EXPANDS INTO FOUR WORDS THAT
;    2073  1     !       ARE USED BY THE RUNTIME SERVICES DURING AN ERROR CALL: ERROR TYPE,
;    2074  1     !       ERROR NUMBER, ADDRESS OF ERROR MESSAGE AND ADDRESS OF MESSAGE
;    2075  1     !       BLOCK.  THERE MUST BE ONLY ONE ERRTBL IN ANY PROGRAM.  THIS SECTION
;    2076  1     !       IS NOT OPTIONAL.
;    2077  1     !--
;    2078  1
```

```
;    2079  1     %SBTTL 'GLOBAL DATA SECTION'
;    2080  1
;    2081  1     PSECT
;    2082  1         PLIT   = $PLIT$,
;    2083  1         OWN    = $OWN$,
;    2084  1         GLOBAL = $GLOB$;
;    2085  1
;    2086  1     !++
;    2087  1     !       THE GLOBAL DATA DEFINED IN THIS SECTION IS USED BY MORE THAN ONE
;    2088  1     !       TEST.
;    2089  1     !--
;    2090  1
;    2091  1     GLOBAL
;    2092  1
;    2093  1     !++
;    2094  1     !       COMMUNICATION AREA DECLARATIONS
;    2095  1     !--
;    2096  1
;    2097  1         RCV_D_LIST      : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    2098  1         XMIT_D_LIST     : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    2099  1         RCV_BUFFER      : VECTOR [ B_SIZE, BYTE ],
;    2100  1         XMIT_BUFFER     : VECTOR [ B_SIZE, BYTE ],
;    2101  1         PHYS_ADR        : VECTOR [ 22, BYTE ],
;    2102  1         SETUP_BUFFER    : VECTOR [ SETUB_SIZE, WORD ],
;    2103  1         IOP_TABLE       : VECTOR [ 8, WORD ],
;    2104  1         ETH_STATION_ADR : VECTOR [ 6, WORD ],
;    2105  1         STATION_ADR     : VECTOR [ 4, WORD ],
;    2106  1         PTRN_TABLE      : VECTOR [ 8, BYTE ] INITIAL ( BYTE (
;    2107  1
;    2108  1                           %B'00000000', %B'11111111', %B'10101010', %B'01010101',
;    2109  1                           %B'11001100', %B'00110011', %B'11110000', %B'00001111')),
```

```
;   2110  1          TARGET_ADR      : VECTOR [ T_SIZE, BYTE ] INITIAL ( BYTE (
;   2111  1
;   2112  1          %X'00', %X'00', %X'00', %X'00', %X'00', %X'00',     !  1 - MEMORY PATTERN
;   2113  1          %X'55', %X'55', %X'55', %X'55', %X'55', %X'55',     !  2
;   2114  1          %X'AA', %X'AA', %X'AA', %X'AA', %X'AA', %X'AA',     !  3 - MEMORY PATTERN
;   2115  1          %X'55', %X'55', %X'55', %X'55', %X'55', %X'55',     !  4 - MEMORY PATTERN
;   2116  1          %X'FF', %X'FF', %X'FF', %X'FF', %X'FF', %X'FF',     !  5 - MEMORY PATTERN
;   2117  1          %X'00', %X'F4', %X'FA', %X'44', %X'44', %X'55',     !  6
;   2118  1          %X'AA', %X'00', %X'00', %X'00', %X'00', %X'00',     !  7 - MEMORY PATTERN
;   2119  1          %X'AA', %X'00', %X'02', %X'AA', %X'AA', %X'AA',     !  8
;   2120  1          %X'AA', %X'00', %X'05', %X'55', %X'55', %X'55',     !  9
;   2121  1          %X'AA', %X'00', %X'04', %X'FF', %X'FF', %X'FF',     ! 10
;   2122  1          %X'AA', %X'00', %X'04', %X'00', %X'00', %X'00',     ! 11 - LOW ETHERNET ADR
;   2123  1          %X'AA', %X'00', %X'04', %X'18', %X'81', %X'18',     ! 12 - HIGH ETHERNET ADR
;   2124  1          %X'01', %X'00', %X'00', %X'00', %X'00', %X'00',     ! 13 - ALL MULTICAST
;   2125  1          %X'AB', %X'AA', %X'AA', %X'AA', %X'AA', %X'AA',     ! 14 - ALL MULTICAST
;   2126  1          %X'FF', %X'00', %X'01', %X'02', %X'03', %X'04',     ! 15 - ALL MULTICAST
;   2127  1          %X'55', %X'05', %X'06', %X'07', %X'08', %X'09',     ! 16 - ALL MULTICAST
;   2128  1          %X'CD', %X'36', %X'26', %X'27', %X'27', %X'49',     ! 17
;   2129  1          %X'33', %X'A1', %X'67', %X'BB', %X'4C', %X'9F',     ! 18
;   2130  1          %X'E8', %X'BE', %X'C7', %X'8F', %X'33', %X'FF',     ! 19
;   2131  1          %X'FF', %X'FF', %X'FF', %X'FF', %X'FF', %X'FF' )),  ! 20 - STATION ADDR
```

```
;    2132  1
;    2133  1        BD_PROM_DESCR   : VECTOR [ BD_D_SIZE, WORD ] INITIAL ( WORD (
;    2134  1
;    2135  1                                NEWB,                   ! BUFFER NOT USED IF 1
;    2136  1                                V,                      ! VALID ADDRESS IF 1
;    2137  1                                RCV_BUFFER,             ! RCV BUFFER ADDRESS
;    2138  1                                BYTE_COUNT,             ! 1/4 THE BYTE COUNT
;    2139  1                                0,                      ! STATUS WORD 1
;    2140  1                                0,                      ! STATUS WORD 2
;    2141  1
;    2142  1                                NEWB,                   ! BUFFER NOT USED IF 1
;    2143  1                                V,                      ! VALID ADDRESS IF 1
;    2144  1                                XMIT_BUFFER,            ! XMIT BUFFER ADDRESS
;    2145  1                                BYTE_COUNT,             ! 1/4 THE BYTE COUNT
;    2146  1                                0,                      ! STATUS WORD 1
;    2147  1                                0,                      ! STATUS WORD 2
;    2148  1
;    2149  1                                NEWB,                   ! BUFFER NOT USED IF 1
;    2150  1                                E,                      ! VALID ADDRESS IF 1
;    2151  1                                0,                      ! 2 EXTRA WORDS
;    2152  1                                0 )),                   !
;    2153  1
;    2154  1
;    2155  1        TD16: VECTOR [ 44, WORD ] INITIAL ( WORD (
;    2156  1
;    2157  1              NEWB, VL , XMIT_BUFFER          , -1 , 0, 0,    ! 1 BYTE DESCRIPTOR
;    2158  1              NEWB, VHL, XMIT_BUFFER          , -2 , 0, 0,    ! 2 BYTE DESCRIPTOR
;    2159  1              NEWB, VH , XMIT_BUFFER + 2      , -1 , 0, 0,    ! 1 BYTE DESCRIPTOR
;    2160  1              NEWB, VE , XMIT_BUFFER + 4      ,  1 , 0, 0,    ! 2 BYTE DESCRIPTOR
;    2161  1              NEWB, E  , XMIT_D_LIST + 60     , -1 , 0, 0,    ! END OF DESCRIPTOR
;    2162  1              NEWB, V  , XMIT_D_LIST + 56     , -2 , 0, 0,    ! 4 BYTE DESCRIPTOR
;    2163  1              NEWB, VE , TARGET_ADR + 114     , -3 , 0, 0,    ! 6 BYTE DESCRIPTOR
;    2164  1              NEWB, E )),                                     ! END OF DESCRIPTOR
;    2165  1
;    2166  1        TD13: VECTOR [ 34, WORD ] INITIAL ( WORD (
;    2167  1
;    2168  1              NEWB, V  , XMIT_BUFFER          ,  -1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;    2169  1              NEWB, V  , XMIT_BUFFER + 2      ,-127 , 0, 0,   ! 378 BYTE DESCRIPTOR
;    2170  1              NEWB, V  , XMIT_BUFFER + 256    ,  -1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;    2171  1              NEWB, C  , XMIT_D_LIST + 48     ,  -1 , 0, 0,   ! CHAIN DESCRIPTOR
;    2172  1              NEWB, VE , XMIT_BUFFER + 258    , -63 , 0, 0,   ! 2 BYTE DESCRIPTOR
;    2173  1              NEWB, E )),                                     ! END OF DESCRIPTOR
;    2174  1
;    2175  1        RD13: VECTOR [ 64, WORD ] INITIAL ( WORD (
;    2176  1
;    2177  1              NEWB, V  , RCV_BUFFER           ,  -1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;    2178  1              NEWB, V  , RCV_BUFFER + 2       , -62 , 0, 0,   ! 124 BYTE DESCRIPTOR
;    2179  1              NEWB, V  , RCV_BUFFER + 126     ,   1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;    2180  1              NEWB, V  , RCV_BUFFER + 128     ,  -2 , 0, 0,   ! 4 BYTE DESCRIPTOR
;    2181  1              NEWB, V  , RCV_BUFFER + 132     , -60 , 0, 0,   ! 120 BYTE DESCRIPTOR
;    2182  1              NEWB, V  , RCV_BUFFER + 252     ,  -2 , 0, 0,   ! 4 BYTE DESCRIPTOR
;    2183  1              NEWB, VC , RCV_D_LIST + 84      ,   1 , 0, 0,   ! CHAIN DESCRIPTOR
;    2184  1              NEWB, V  , RCV_BUFFER + 256     ,  -3 , 0, 0,   ! 6 BYTE DESCRIPTOR
```

```
;   2185  1               NEWB, V  , RCV_BUFFER + 262    , -60 , 0, 0,   ! 120 BYTE DESCRIPTOR
;   2186  1               NEWB, V  , RCV_BUFFER + 382    ,  -1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;   2187  1               NEWB, E )),                                    ! END OF DESCRIPTOR
;   2188  1
```

```
;  2189  1     !++
;  2190  1     !       HARDWARE AND SOFTWARE P-TABLE STORAGE DECLARATIONS
;  2191  1     !--
;  2192  1
;  2193  1         HWP_TABLE    : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;  2194  1         SWP_TABLE    : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;  2195  1
;  2196  1         REG_ADR      : REF REG_STR FIELD ( IOP_FIELDS ),
;  2197  1         IOP_DATA     : REF REG_STR FIELD ( IOP_FIELDS ),
;  2198  1         GET_ADR      : REF ADR_STR FIELD ( IOP_FIELDS ),
;  2199  1
;  2200  1     !++
;  2201  1     !
;  2202  1     !       MISCELLANEOUS DATA DECLARATIONS
;  2203  1     !
;  2204  1     !--
;  2205  1
;  2206  1         XBUF_LENGTH          : WORD,                 ! XMIT BUFFER LENGTH IN WORDS
;  2207  1         RBUF_LENGTH          : WORD,                 ! RCV BUFFER LENGTH IN BYTES
;  2208  1         INTERRUPT_FLG        : WORD,                 ! 1 = INTERRUPT OCCURED
;  2209  1         DEQNA_NO             : WORD,                 ! DEQNA UNDER TEST THIS PASS
;  2210  1         COUNTER              : WORD,                 ! ITERATION COUNTER, INDEX
;  2211  1         UP_COUNTER           : WORD,                 ! ITERATION COUNTER, INDEX
;  2212  1         DOWN_COUNTER         : WORD,                 ! ITERATION COUNTER, INDEX
;  2213  1         CHECKSUM             : WORD,                 ! EXPECTED PROM CHECKSUM
;  2214  1         BUF_LENGTH           : WORD,                 ! XMIT BUFFER SIZE IN WORDS
;  2215  1         CSR_WORD             : WORD,                 !
;  2216  1         XC_FLAG              : WORD INITIAL (0),     !
;  2217  1         ERR_NUMBER           : WORD INITIAL (0),     !
;  2218  1         ERR_FLAG             : WORD INITIAL (0),     !
;  2219  1         ERR_COUNT            : WORD INITIAL (0),     !
;  2220  1
```

```
;   2221  1     !++
;   2222  1     !
;   2223  1     !   TEMPORARY STORAGE DATA DECLARATIONS
;   2224  1     !
;   2225  1     !--
;   2226  1
;   2227  1         TMP_IOP_ADR         : WORD,             ! I/O PAGE REGISTER ADDRESS
;   2228  1         TMP_REG_DATA        : WORD,             ! I/O PAGE REG CONTENTS
;   2229  1         TEMP1               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2230  1         TEMP2               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2231  1         TEMP3               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2232  1         TEMP4               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2233  1         TEMP5               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2234  1         TEMP6               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2235  1         TEMP7               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2236  1         TEMP8               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2237  1         TEMP9               : WORD,             ! TEMPORARY STORAGE LOCATION
;   2238  1         P1                  : WORD,             ! PARAMETER #1
;   2239  1         P2                  : WORD,             ! PARAMETER #2
;   2240  1         P3                  : WORD,             ! PARAMETER #3
;   2241  1         P4                  : WORD,             ! PARAMETER #4
;   2242  1         P5                  : WORD,             ! PARAMETER #5
;   2243  1         TBYTE1              : BYTE,             !
;   2244  1         TBYTE2              : BYTE,             !
;   2245  1         TBYTE3              : BYTE,             !
;   2246  1         TBYTE4              : BYTE,             !
;   2247  1         TADR1               : WORD,             !
;   2248  1         TADR2               : WORD;             !
;   2249  1
```

```
;  2250  1
;  2251  1     %SBTTL 'GLOBAL TEXT SECTION'
;  2252  1
;  2253  1     !++
;  2254  1     !       THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS, MESSAGES,
;  2255  1     !       AND ASCII INFORMATION THAT IS USED IN MORE THAN ONE TEST.
;  2256  1     !--
;  2257  1
;  2258  1     GLOBAL BIND
;  2259  1
;  2260  1         DESCR_LIST   = RCV_D_LIST,
;  2261  1         DATA_BUFFER  = RCV_BUFFER,
;  2262  1
;  2263  1     !++
;  2264  1     !       HARDWARE AND SOFTWARE QUESTIONS
;  2265  1     !--
;  2266  1
;  2267  1         QST01 = UPLIT (%ASCIZ'DEQNA I/O PAGE ADR     '),
;  2268  1         QST02 = UPLIT (%ASCIZ'INTERRUPT VECTOR ADR   '),
;  2269  1         QST03 = UPLIT (%ASCIZ'DO YOU WANT TO TEST SANITY TIMER '),
;  2270  1         QST04 = UPLIT (%ASCIZ'IS LOOPBACK CONNECTOR IN DEQNA   '),
;  2271  1         QST05 = UPLIT (%ASCIZ'SANITY TIMER TIME-OUT VALUE      '),
;  2272  1         QST06 = UPLIT (%ASCIZ'EXTERNAL LOOPBACK MODE           '),
;  2273  1         QST07 = UPLIT (%ASCIZ'SYSTEM HAS BLOCK-MODE MEMORY     '),
;  2274  1
;  2275  1
;  2276  1
;  2277  1     !++
;  2278  1     !       DEVICE ERROR MESSAGES
;  2279  1     !++
;  2280  1
;  2281  1         MSG00 = UPLIT (%ASCIZ' DEQNA FATAL ERROR DETECTED '),
;  2282  1         MSG01 = UPLIT (%ASCIZ'%N%N%A   DEQNA ADDRESS: %O6%A,  STATION ADDRESS: '),
;  2283  1         MSG02 = UPLIT (%ASCIZ'%A      ACTUAL DATA = %O6%A     EXPECTED DATA = %O6%N'),
;  2284  1         MSG03 = UPLIT (%ASCIZ'%A                              XMIT DESCRIPTOR    RCV DESCRIPTOR %N'),
;  2285  1         MSG04 = UPLIT (%ASCIZ'%A      FLAG WORD                   %O6%A              %O6%N'),
;  2286  1         MSG05 = UPLIT (%ASCIZ'%A      HIGH ORDER ADDR BITS        %O6%A              %O6%N'),
;  2287  1         MSG06 = UPLIT (%ASCIZ'%A      LOW  ORDER ADDR BITS        %O6%A              %O6%N'),
;  2288  1         MSG07 = UPLIT (%ASCIZ'%A      PACKET LENGTH ( WD )        %O6%A              %O6%N'),
;  2289  1         MSG08 = UPLIT (%ASCIZ'%A      STATUS WORD 1               %O6%A              %O6%N'),
;  2290  1         MSG09 = UPLIT (%ASCIZ'%A      STATUS WORD 2               %O6%A              %O6%N'),
;  2291  1         MSG10 = UPLIT (%ASCIZ'%A      DEQNA CSR REGISTER                     %O6%N'),
;  2292  1         MSG11 = UPLIT (%ASCIZ'%A      DEQNA I/O PAGE ADR                     %O6%N%N'),
;  2293  1         MSG12 = UPLIT (%ASCIZ'%A BAD CSR: ACT = %O6%A EXP = %O6%N'),
;  2294  1         MSG13 = UPLIT (%ASCIZ'%A BAD TRANSMIT FLAG WORD: ACT = %O6%A EXP = %O6%N'),
;  2295  1         MSG14 = UPLIT (%ASCIZ'%A BAD TRANSMIT STATUS WORD 1: ACT = %O6%A EXP = %O6%N'),
;  2296  1         MSG15 = UPLIT (%ASCIZ'%A BAD RECEIVE FLAG WORD: ACT = %O6%A EXP = %O6%N'),
;  2297  1         MSG16 = UPLIT (%ASCIZ'%A BAD RECEIVE STATUS WORD 1: ACT = %O6%A EXP = %O6%N'),
;  2298  1         MSG17 = UPLIT (%ASCIZ'%A BAD RECEIVE BUFFER LENGTH: ACT = %O6%A EXP = %O6%N'),
;  2299  1         MSG18 = UPLIT (%ASCIZ'%A BAD CSR = %O6%N'),
;  2300  1         MSG19 = UPLIT (%ASCIZ'%A LOOPBACK PACKET UNABLE TO SET CA BIT, CSR = %O6%N'),
;  2301  1         MSG20 = UPLIT (%ASCIZ'%A LOOPBACK PACKET UNABLE TO CLEAR CA BIT, CSR = %O6%N '),
;  2302  1         MSG21 = UPLIT (%ASCIZ'%A CA BIT OK, BUT RI BIT IS NOT ON, CSR = %O6%N '),
```

```
;  2303  1        MSG22 = UPLIT (%ASCIZ'%A CA BIT IN THE CSR WAS SET TOO EARLY, CSR = %O6%N'),
;  2304  1        MSG23 = UPLIT (%ASCIZ'%A XL AND RL ( BITS 4,5 ) TO BE RESET TO 0%N'),
;  2305  1        MSG24 = UPLIT (%ASCIZ'%A XL AND RL ( BITS 4,5 ) TO BE SET TO 1%N'),
;  2306  1        MSG25 = UPLIT (%ASCIZ'%A RI ( BIT 15 ) TO BE SET TO 1%N'),
;  2307  1        MSG26 = UPLIT (%ASCIZ'%A XI ( BIT 7 ) TO BE SET TO 1%N'),
;  2308  1        MSG27 = UPLIT (%ASCIZ'%A NI ( BIT 2 ) TO BE SET TO 1%N'),
;  2309  1        MSG28 = UPLIT (%ASCIZ'%A NI ( BIT 2 ) TO BE RESET TO 0%N'),
;  2310  1        MSG29 = UPLIT (%ASCIZ'%A BAD CSR, EXPECTED'),
;  2311  1        MSG30 = UPLIT (%ASCIZ'%A CSR ADR = %O6%A  ACTUAL = %O6%A  EXPECTED = %O6%N'),
;  2312  1        MSG31 = UPLIT (%ASCIZ'%N%A UNABLE TO RESET DEQNA: ADR: %O6%A  CSR = %O6%N'),
;  2313  1        MSG32 = UPLIT (%ASCIZ'%N%A WAIT ABOUT %D2%A SECOND(S) -'),
;  2314  1        MSG33 = UPLIT (%ASCIZ'%N%A SANITY TIMER TIMED OUT AS EXPECTED %N'),
;  2315  1        MSG34 = UPLIT (%ASCIZ'%N%A NO SANITY TIMER INTERRUPT DETECTED %N'),
;  2316  1        MSG35 = UPLIT (%ASCIZ'%N%A DISCONNECT TRANSCEIVER CABLE FROM BULKHEAD ASSEMBLY AND'),
;  2317  1        MSG36 = UPLIT (%ASCIZ'%N%A CONNECT LOOPBACK CONNECTOR TO BULKHEAD ASSEMBLY, THEN RETEST%N '),
;  2318  1        MSG37 = UPLIT (%ASCIZ'%N%A DISCONNECT BULKHEAD ASSEMBLY FROM DEQNA AND CONNECT'),
;  2319  1        MSG38 = UPLIT (%ASCIZ'%N%A LOOPBACK CONNECTOR TO DEQNA, THEN RETEST%N'),
;  2320  1        MSG39 = UPLIT (%ASCIZ'%N%A CHECK FOR LOOSE WIRES IN A LOOPBACK CONNECTOR'),
;  2321  1        MSG40 = UPLIT (%ASCIZ'%N%A OR USE DIFFERENT LOOPBACK CONNECTOR, THEN RETEST%N'),
;  2322  1        MSG41 = UPLIT (%ASCIZ'%N%A REPLACE DEQNA, THEN RETEST%N'),
;  2323  1        MSG42 = UPLIT (%ASCIZ'%N%A REPLACE BULKHEAD CONNECTOR, THEN RETEST%N'),
;  2324  1        MSG43 = UPLIT (%ASCIZ'%N%A DISCONNECT TRANSCEIVER CABLE FROM TRANSCEIVER'),
;  2325  1        MSG44 = UPLIT (%ASCIZ'%N%A AND CONNECT IT TO LOOPBACK CONNECTOR AND BULKHEAD ASSEMBLY%N '),
;  2326  1        MSG45 = UPLIT (%ASCIZ'%N%A REPLACE TRANSCEIVER CABLE, THEN RETEST%N'),
;  2327  1        MSG46 = UPLIT (%ASCIZ'%N%A REPLACE TRANSCEIVER, THEN RETEST%N'),
;  2328  1        MSG47 = UPLIT (%ASCIZ'%N%A REPLACE THE FUSE IF BAD, THEN RETEST%N'),
;  2329  1        MSG48 = UPLIT (%ASCIZ'%N%A BAD RECEIVE DESCRIPTOR:'),
;  2330  1        MSG49 = UPLIT (%ASCIZ'%N%A BAD TRANSMIT DESCRIPTOR:'),
;  2331  1        MSG50 = UPLIT (%ASCIZ'%A ACTUAL = %O6%A EXPECTED = %O6%A INDEX = %D4%N'),
;  2332  1        MSG51 = UPLIT (%ASCIZ'%N%A BAD RECEIVE BUFFER: ),
;  2333  1        MSG52 = UPLIT (%ASCIZ'%N%A DMA OPERATION TAKES TOO LONG%N'),
;  2334  1        MSG53 = UPLIT (%ASCIZ'%N%A TOO MANY DEVICES%N'),
;  2335  1        MSG54 = UPLIT (%ASCIZ'%N%A THERE WAS A POWER FAIL - WAITING%N'),
;  2336  1        MSG55 = UPLIT (%ASCIZ'%N%A WAIT ABOUT %D2%A MINUTE(S) -'),
;  2337  1        MSG56 = UPLIT (%ASCIZ'%N%A WAIT ABOUT %D2%A HOUR -'),
;  2338  1        MSG57 = UPLIT (%ASCIZ'%A IF NO RESET, TYPE ANY CHARACTER TO EXIT FROM TEST%N'),
;  2339  1        MSG58 = UPLIT (%ASCIZ'%N%A TDR VALUE IS EQUAL TO ZERO %N'),
;  2340  1        MSG59 = UPLIT (%ASCIZ'%N%N%A----------------------------------------------------------------------%N'),
;  2341  1        MSG60 = UPLIT (%ASCIZ'%N%A BAD CSR, BITS STUCK AT 0:%N'),
;  2342  1        MSG61 = UPLIT (%ASCIZ'%N%A BAD CSR, BITS STUCK AT 1:%N'),
;  2343  1        MSG62 = UPLIT (%ASCIZ'%N%A SOFTWARE RESET UNABLE TO CLEAR CSR STATIC BITS:%N'),
;  2344  1        MSG63 = UPLIT (%ASCIZ'%N%A BAD STATION ADDRESS CHECKSUM: ACT = %O6%A EXP = %O6%N'),
;  2345  1        MSG64 = UPLIT (%ASCIZ'%N%A BAD STATION ADDRESS: '),
;  2346  1        MSG65 = UPLIT (%ASCIZ'%N%A BAD DEQNA I/O PAGE REGISTER:%N'),
;  2347  1        MSG66 = UPLIT (%ASCIZ'%N%A BAD CSR, EXPECTED RL ( BIT 5 ) TO BE SET TO 0%N'),
;  2348  1        MSG67 = UPLIT (%ASCIZ'%N%A BAD B/D PROM CHECKSUM: INDEX = %O6%A ACT = %O6%A EXP = %O6%N'),
;  2349  1        MSG68 = UPLIT (%ASCIZ'%N%A B/D PROM CHECKSUM OFFSET = %O6%A ACT = %O6%A EXP = %O6%N '),
;  2350  1        MSG69 = UPLIT (%ASCIZ'%N%A BAD INTERRUPT: ADR = %O6%A ACT LEV = %O6%A EXP LEV = %O6%N'),
;  2351  1        MSG70 = UPLIT (%ASCIZ'%N%A REGISTER FAILED TO RESPOND AT ADDRESS:  %O6%N '),
;  2352  1        MSG71 = UPLIT (%ASCIZ'%N%A BAD TRANSMIT STATUS, TOO MANY COLLISIONS%N ');
;  2353  1
;  2354  1
;  2355  1
```

```
;    2356  1      %SBTTL 'DEFAULT HARDWARE P TABLE'
;    2357  1
;    2358  1      BGNHW ( HP_TABLE );
;    2359  1
;    2360  1      !++
;    2361  1      !       THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF THE
;    2362  1      !       TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE IS IDENTICAL TO
;    2363  1      !       THE STRUCTURE OF THE HARDWARE P-TABLES, AND IS USED AS A "TEMPLATE"
;    2364  1      !       FOR BUILDING THE P-TABLES.
;    2365  1      !
;    2366  1      !
;    2367  1      !       PLACE YOUR DEFAULT HARDWARE P-TABLE HERE.  THE VALUES AND
;    2368  1      !       SIZE WILL BE USED AS A "TEMPLATE" FOR CREATING ACTUAL P-TABLE
;    2369  1      !       ENTRIES AND THE DEFAULT VALUES IN THE OPERATOR DIALOGUE.
;    2370  1      !       THE ACTUAL P-TABLE BUILT AT RUNTIME IS STORED IN SUPERVISOR
;    2371  1      !       SPACE.
;    2372  1      !--
;    2373  1
;    2374  1          GLOBAL
;    2375  1              DFSTBL  : BLOCK [ HWP_SIZE, WORD ] INITIAL ( %O'174440', %O'700' );
;    2376  1      ENDHW;
;    2377  1
;    2378  1
```

```
;    2379  1      %SBTTL 'SOFTWARE P-TABLE'
;    2380  1
;    2381  1      !++
;    2382  1      !       THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
;    2383  1      !       PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
;    2384  1      !       SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
;    2385  1      !       AT RUN TIME.
;    2386  1      !
;    2387  1      !
;    2388  1      !       PLACE YOUR SOFTWARE P-TABLE HERE,USING GLOBAL OR OWN DECLARATIONS
;    2389  1      !       THIS TABLE IS NOT OPTIONAL.  THIS TABLE, UNLIKE THE HARDWARE TABLE,
;    2390  1      !       WILL CONTAIN THE ACTUAL VALUES ENTERED BY THE OPERATOR.
;    2391  1      !--
;    2392  1
;    2393  1      BGNSW ( SP_TABLE );
;    2394  1
;    2395  1         GLOBAL
;    2396  1              SWP_TIMER       : WORD INITIAL ( NO ),  ! NO SANITY TIMER TEST
;    2397  1              SWP_LBC         : WORD INITIAL ( NO ),  ! NO LOOPBACK IN DEQNA
;    2398  1              SWP_TOUT_VAL    : WORD INITIAL ( 3 ),   ! TIMEOUT VALUE = 16 SEC.
;    2399  1              SWP_ILOOP       : WORD INITIAL ( NO ),  ! EXTERNAL LOOPBACK MODE
;    2400  1              SWP_BLOCK_MEM   : WORD INITIAL ( YES ); ! BLOCK-MODE MEMORY PRESENT
;    2401  1
;    2402  1      ENDSW;
;    2403  1
;    2404  1
```

```
;    2405  1      %SBTTL 'PROTECTION TABLE'
;    2406  1
;    2407  1      !++
;    2408  1      !       THIS TABLE IS USED BY THE RUNTIME SERVICES TO PROTECT THE LOAD MEDIA.
;    2409  1      !
;    2410  1      !       1ST ARG =       OFFSET INTO P-TABLE FOR CSR ADDRESS
;    2411  1      !       2ND ARG =       OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;    2412  1      !       3RD ARG =       OFFSET INTO P-TABLE FOR DRIVE NUMBER
;    2413  1      !
;    2414  1      !       INSERT BYTE OFFSET FOR DATA NOTED IN COMMENTS ABOVE.  (OFFSET
;    2415  1      !       REFERS TO THE NUMBER OF BYTES FROM THE BEGINNING OF A PTABLE
;    2416  1      !       ENTRY TO THE ITEM IN QUESTION.)  IF THE PARTICULAR
;    2417  1      !       ITEM DOES NOT APPLY, LEAVE ENTRY AS -1.  WHEN THE RUNTIME
;    2418  1      !       SERVICES EXECUTES A GPHARD, IT USES THESE OFFSETS (IF NOT
;    2419  1      !       SET TO -1) TO GET THE ITEMS AND COMPARE WITH THOSE SAVED
;    2420  1      !       IN THE XXDP+ MONITOR.  IF THE UNIT BEING REQUESTED MATCHES THE
;    2421  1      !       LOAD DEVICE, THE RUNTIME SERVICES RETURN AN INCOMPLETE FLAG ON
;    2422  1      !       THE GPHARD.
;    2423  1      !--
;    2424  1
;    2425  1      BGNPROT (-1, -1, -1);
;    2426  1
;    2427  1      ENDPROT;
;    2428  1
;    2429  1
;    2430  1
;    2431  1      END
;    2432  0      ELUDOM


                                        .TITLE  ZQNA1 CZQNADO DEQNA FUNCTIONAL TEST
                                        .IDENT  /V01.0/
                                        .ENABL  AMA


000000                                  .PSECT  $CODE$,  RO
000000      103     132     121     L$NAME::.ASCII  /CZQ/
000003      116     101     040             .ASCII  /NA /
000006      000                             .BYTE   0
000007      000                             .BYTE   0
000010                              L$REV::
000010      104                             .ASCII  /D/
000011      060                             .ASCII  /0/
000012  000000G                     L$UNIT::.WORD   T$PTHV
000014  000170                      L$TIML::.WORD   170
000016  000000G                     L$HPCP::.WORD   L$HARD
000020  000000G                     L$SPCP::.WORD   L$SOFT
000022  000210'                     L$HPTP::.WORD   L$HW
000024  000220'                     L$SPTP::.WORD   L$SW
000026  000000G                     L$LADP::.WORD   L$LAST
000030  000000                      L$STA:: .WORD   0
000032  000000                      L$CO::  .WORD   0
000034  000000                      L$DTYP::.WORD   0
```

```
000036  000000                          L$APT:: .WORD   0
000040  000124'                         L$DTP:: .WORD   L$DISPATCH
000042  000000                          L$PRIO::.WORD   0
000044  000000                          L$ENVI::.WORD   0
000046  000000                          L$EXP1::.WORD   0
000050                                  L$MREV::
000050     003                                  .BYTE   3
000051     003                                  .BYTE   3
000052  000000                          L$EF::  .WORD   0
000054  000000                                  .WORD   0
000056  000000                          L$SPC:: .WORD   0
000060  000000G                         L$DEVP::.WORD   L$DVTYP
000062  000000G                         L$REPP::.WORD   L$RPT
000064  000000                          L$EXP4::.WORD   0
000066  000000                          L$EXP5::.WORD   0
000070  000000G                         L$AUT:: .WORD   L$AU
000072  000000G                         L$DUT:: .WORD   L$DU
000074  000000                          L$LUN:: .WORD   0
000076  000000G                         L$DESP::.WORD   L$DESC
000100  104035                          L$LOAD::.WORD   -73743
000102  000176'                         L$ETP:: .WORD   L$ERRTBL
000104  000000G                         L$ICP:: .WORD   L$INIT
000106  000000G                         L$CCP:: .WORD   L$CLEAN
000110  000000G                         L$ACP:: .WORD   L$AUTO
000112  000234'                         L$PRT:: .WORD   L$PROT
000114  000000                          L$TEST::.WORD   0
000116  000000                          L$DLY:: .WORD   0
000120  000000                          L$HlME::.WORD   0
000122  000025                          D$PCNT::.WORD   25
000124  000000G                         L$DISPATCH::
                                                .WORD   T1
000126  000000G                                 .WORD   T2
000130  000000G                                 .WORD   T3
000132  000000G                                 .WORD   T4
000134  000000G                                 .WORD   T5
000136  000000G                                 .WORD   T6
000140  000000G                                 .WORD   T7
000142  000000G                                 .WORD   T8
000144  000000G                                 .WORD   T9
000146  000000G                                 .WORD   T10
000150  000000G                                 .WORD   T11
000152  000000G                                 .WORD   T12
000154  000000G                                 .WORD   T13
000156  000000G                                 .WORD   T14
000160  000000G                                 .WORD   T15
000162  000000G                                 .WORD   T16
000164  000000G                                 .WORD   T17
000166  000000G                                 .WORD   T18
000170  000000G                                 .WORD   T19
000172  000000G                                 .WORD   T20
000174  000000G                                 .WORD   T21
000176                                  ERRTYP::.BLKW   1
000200                                  ERRNBR::.BLKW   1
```

```
000202                                  ERRMSG::.BLKW   1
000204                                  ERRBLK::.BLKW   1
000206  000000C                         L$HWLEN::
                                                .WORD   <<L$NDHW-L$HWLEN>/2>
000210  174440                          DFSTBL::.WORD   -3340
000212  000700                                  .WORD   700
000214                                  L$NDHW::.BLKW   1
000216  000000C                         L$SWLEN::
                                                .WORD   <<L$NDSW-L$SWLEN>/2>
000220  000000                          SWP.TIMER::
                                                .WORD   0
000222  000000                          SWP.LBC::
                                                .WORD   0
000224  000003                          SWP.TOUT.VAL::
                                                .WORD   3
000226  000000                          SWP.ILOOP::
                                                .WORD   0
000230  000001                          SWP.BLOCK.MEM::
                                                .WORD   1
000232                                  L$NDSW::.BLKW   1
000234  177777                          L$PROT::.WORD   -1
000236  177777                                  .WORD   -1
000240  177777                                  .WORD   -1



000000                                          .PSECT  $PLIT$,  RO ,  D
000000      104     105     121         P.AAA:  .ASCII  /DEQ/
000003      116     101     040                 .ASCII  /NA /
000006      111     057     117                 .ASCII  /I/<57>/O/
000011      040     120     101                 .ASCII  / PA/
000014      107     105     040                 .ASCII  /GE /
000017      101     104     122                 .ASCII  /ADR/
000022      040     040     040                 .ASCII  /   /
000025      040     000     000                 .ASCII  / /<00><00>
000030      111     116     124         P.AAB:  .ASCII  /INT/
000033      105     122     122                 .ASCII  /ERR/
000036      125     120     124                 .ASCII  /UPT/
000041      040     126     105                 .ASCII  / VE/
000044      103     124     117                 .ASCII  /CTO/
000047      122     040     101                 .ASCII  /R A/
000052      104     122     040                 .ASCII  /DR /
000055      040     000     000                 .ASCII  / /<00><00>
000060      104     117     040         P.AAC:  .ASCII  /DO /
000063      131     117     125                 .ASCII  /YOU/
000066      040     127     101                 .ASCII  / WA/
000071      116     124     040                 .ASCII  /NT /
000074      124     117     040                 .ASCII  /TO /
000077      124     105     123                 .ASCII  /TES/
000102      124     040     123                 .ASCII  /T S/
000105      101     116     111                 .ASCII  /ANI/
000110      124     131     040                 .ASCII  /TY /
000113      124     111     115                 .ASCII  /TIM/
```

```
000116  105  122  040              .ASCII  /ER /
000121  000                       .ASCII  <00>
000122  111  123  040  P.AAD:     .ASCII  /IS /
000125  114  117  117              .ASCII  /LOO/
000130  120  102  101              .ASCII  /PBA/
000133  103  113  040              .ASCII  /CK /
000136  103  117  116              .ASCII  /CON/
000141  116  105  103              .ASCII  /NEC/
000144  124  117  122              .ASCII  /TOR/
000147  040  111  116              .ASCII  / IN/
000152  040  104  105              .ASCII  / DE/
000155  121  116  101              .ASCII  /QNA/
000160  040  040  040              .ASCII  /   /
000163  000                       .ASCII  <00>
000164  123  101  116  P.AAE:     .ASCII  /SAN/
000167  111  124  131              .ASCII  /ITY/
000172  040  124  111              .ASCII  / TI/
000175  115  105  122              .ASCII  /MER/
000200  040  124  111              .ASCII  / TI/
000203  115  105  055              .ASCII  /ME-/
000206  117  125  124              .ASCII  /OUT/
000211  040  126  101              .ASCII  / VA/
000214  114  125  105              .ASCII  /LUE/
000217  040  040  040              .ASCII  /   /
000222  040  040  040              .ASCII  /   /
000225  000                       .ASCII  <00>
000226  105  130  124  P.AAF:     .ASCII  /EXT/
000231  105  122  116              .ASCII  /ERN/
000234  101  114  040              .ASCII  /AL /
000237  114  117  117              .ASCII  /LOO/
000242  120  102  101              .ASCII  /PBA/
000245  103  113  040              .ASCII  /CK /
000250  115  117  104              .ASCII  /MOD/
000253  105  040  040              .ASCII  /E  /
000256  040  040  040              .ASCII  /   /
000261  040  040  040              .ASCII  /   /
000264  040  040  040              .ASCII  /   /
000267  000                       .ASCII  <00>
000270  123  131  123  P.AAG:     .ASCII  /SYS/
000273  124  105  115              .ASCII  /TEM/
000276  040  110  101              .ASCII  / HA/
000301  123  040  102              .ASCII  /S B/
000304  114  117  103              .ASCII  /LOC/
000307  113  055  115              .ASCII  /K-M/
000312  117  104  105              .ASCII  /ODE/
000315  040  115  105              .ASCII  / ME/
000320  115  117  122              .ASCII  /MOR/
000323  131  040  040              .ASCII  /Y  /
000326  040  040  040              .ASCII  /   /
000331  000                       .ASCII  <00>
000332  040  104  105  P.AAH:     .ASCII  / DE/
000335  121  116  101              .ASCII  /QNA/
000340  040  106  101              .ASCII  / FA/
```

```
000343  124  101  114              .ASCII  /TAL/
000346  040  105  122              .ASCII  / ER/
000351  122  117  122              .ASCII  /ROR/
000354  040  104  105              .ASCII  / DE/
000357  124  105  103              .ASCII  /TEC/
000362  124  105  104              .ASCII  /TED/
000365  040  000  000              .ASCII  / /<00><00>
000370  045  116  045    P.AAI:    .ASCII  /%N%/
000373  116  045  101              .ASCII  /N%A/
000376  040  040  040              .ASCII  /   /
000401  104  105  121              .ASCII  /DEQ/
000404  116  101  040              .ASCII  /NA /
000407  101  104  104              .ASCII  /ADD/
000412  122  105  123              .ASCII  /RES/
000415  123  072  040              .ASCII  /S: /
000420  045  117  066              .ASCII  /%O6/
000423  045  101  054              .ASCII  /%A,/
000426  040  040  123              .ASCII  /  S/
000431  124  101  124              .ASCII  /TAT/
000434  111  117  116              .ASCII  /ION/
000437  040  101  104              .ASCII  / AD/
000442  104  122  105              .ASCII  /DRE/
000445  123  123  072              .ASCII  /SS:/
000450  040  000                   .ASCII  / /<00>
000452  045  101  040    P.AAJ:    .ASCII  /%A /
000455  040  040  040              .ASCII  /   /
000460  040  040  101              .ASCII  /  A/
000463  103  124  125              .ASCII  /CTU/
000466  101  114  040              .ASCII  /AL /
000471  104  101  124              .ASCII  /DAT/
000474  101  040  075              .ASCII  /A =/
000477  040  045  117              .ASCII  / %O/
000502  066  045  101              .ASCII  /6%A/
000505  040  040  040              .ASCII  /   /
000510  040  040  105              .ASCII  /  E/
000513  130  120  105              .ASCII  /XPE/
000516  103  124  105              .ASCII  /CTE/
000521  104  040  104              .ASCII  /D D/
000524  101  124  101              .ASCII  /ATA/
000527  040  075  040              .ASCII  / = /
000532  045  117  066              .ASCII  /%O6/
000535  045  116  000              .ASCII  /%N/<00>
000540  045  101  040    P.AAK:    .ASCII  /%A /
000543  040  040  040              .ASCII  /   /
000546  040  040  040              .ASCII  /   /
000551  040  040  040              .ASCII  /   /
000554  040  040  040              .ASCII  /   /
000557  040  040  040              .ASCII  /   /
000562  040  040  040              .ASCII  /   /
000565  040  040  040              .ASCII  /   /
000570  040  040  040              .ASCII  /   /
000573  040  040  040              .ASCII  /   /
000576  130  115  111              .ASCII  /XMI/
```

```
000601  124  040  104            .ASCII  /T D/
000604  105  123  103            .ASCII  /ESC/
000607  122  111  120            .ASCII  /RIP/
000612  124  117  122            .ASCII  /TOR/
000615  040  040  040            .ASCII  /   /
000620  040  122  103            .ASCII  / RC/
000623  126  040  104            .ASCII  /V D/
000626  105  123  103            .ASCII  /ESC/
000631  122  111  120            .ASCII  /RIP/
000634  124  117  122            .ASCII  /TOR/
000637  040  045  116            .ASCII  / %N/
000642  000  000                 .ASCII  <00><00>
000644  045  101  040  P.AAL:    .ASCII  /%A /
000647  040  040  040            .ASCII  /   /
000652  040  C40  106            .ASCII  /  F/
000655  114  101  107            .ASCII  /LAG/
000660  040  127  117            .ASCII  / WO/
000663  122  104  040            .ASCII  /RD /
000666  040  040  040            .ASCII  /   /
000671  040  040  040            .ASCII  /   /
000674  040  040  040            .ASCII  /   /
000677  040  040  040            .ASCII  /   /
000702  040  040  040            .ASCII  /   /
000705  040  040  045            .ASCII  /  %/
000710  117  066  045            .ASCII  /06%/
000713  101  040  040            .ASCII  /A  /
000716  040  040  040            .ASCII  /   /
000721  040  040  040            .ASCII  /   /
000724  040  040  040            .ASCII  /   /
000727  040  045  117            .ASCII  / %O/
000732  066  045  116            .ASCII  /6%N/
000735  000                      .ASCII  <00>
000736  045  101  040  P.AAM:    .ASCII  /%A /
000741  040  040  040            .ASCII  /   /
000744  040  040  110            .ASCII  /  H/
000747  111  107  110            .ASCII  /IGH/
000752  040  117  122            .ASCII  / OR/
000755  104  105  122            .ASCII  /DER/
000760  040  101  104            .ASCII  / AD/
000763  104  122  040            .ASCII  /DR /
000766  102  111  124            .ASCII  /BIT/
000771  123  040  040            .ASCII  /S  /
000774  040  040  040            .ASCII  /   /
000777  040  040  045            .ASCII  /  %/
001002  117  066  045            .ASCII  /06%/
001005  101  040  040            .ASCII  /A  /
001010  040  040  040            .ASCII  /   /
001013  040  040  040            .ASCII  /   /
001016  040  040  040            .ASCII  /   /
001021  040  045  117            .ASCII  / %O/
001024  066  045  116            .ASCII  /6%N/
001027  000                      .ASCII  <00>
001030  045  101  040  P.AAN:    .ASCII  /%A /
```

```
001033  040  040  040              .ASCII  /   /
001036  040  040  114              .ASCII  /  L/
001041  117  127  040              .ASCII  /OW /
001044  040  117  122              .ASCII  / OR/
001047  104  105  122              .ASCII  /DER/
001052  040  101  104              .ASCII  / AD/
001055  104  122  040              .ASCII  /DR /
001060  102  111  124              .ASCII  /BIT/
001063  123  040  040              .ASCII  /S  /
001066  040  040  040              .ASCII  /   /
001071  040  040  045              .ASCII  /  %/
001074  117  066  045              .ASCII  /O6%/
001077  101  040  040              .ASCII  /A  /
001102  040  040  040              .ASCII  /   /
001105  040  040  040              .ASCII  /   /
001110  040  040  040              .ASCII  /   /
001113  040  045  117              .ASCII  / %O/
001116  066  045  116              .ASCII  /6%N/
001121  000                       .ASCII  <00>
001122  045  101  040  P.AAO:      .ASCII  /%A /
001125  040  040  040              .ASCII  /   /
001130  040  040  120              .ASCII  /  P/
001133  101  103  113              .ASCII  /ACK/
001136  105  124  040              .ASCII  /ET /
001141  114  105  116              .ASCII  /LEN/
001144  107  124  110              .ASCII  /GTH/
001147  040  050  040              .ASCII  / ( /
001152  127  104  040              .ASCII  /WD /
001155  051  040  040              .ASCII  /)  /
001160  040  040  040              .ASCII  /   /
001163  040  040  045              .ASCII  /  %/
001166  117  066  045              .ASCII  /O6%/
001171  101  040  040              .ASCII  /A  /
001174  040  040  040              .ASCII  /   /
001177  040  040  040              .ASCII  /   /
001202  040  040  040              .ASCII  /   /
001205  040  045  117              .ASCII  / %O/
001210  066  045  116              .ASCII  /6%N/
001213  000                       .ASCII  <00>
001214  045  101  040  P.AAP:      .ASCII  /%A /
001217  040  040  040              .ASCII  /   /
001222  040  040  123              .ASCII  /  S/
001225  124  101  124              .ASCII  /TAT/
001230  125  123  040              .ASCII  /US /
001233  127  117  122              .ASCII  /WOR/
001236  104  040  061              .ASCII  /D 1/
001241  040  040  040              .ASCII  /   /
001244  040  040  040              .ASCII  /   /
001247  040  040  040              .ASCII  /   /
001252  040  040  040              .ASCII  /   /
001255  040  040  045              .ASCII  /  %/
001260  117  066  045              .ASCII  /O6%/
001263  101  040  040              .ASCII  /A  /
```

```
001266  040     040     040             .ASCII  /   /
001271  040     040     040             .ASCII  /   /
001274  040     040     040             .ASCII  /   /
001277  040     045     117             .ASCII  / %O/
001302  066     045     116             .ASCII  /6%N/
001305  000                             .ASCII  <00>
001306  045     101     040     P.AAQ:  .ASCII  /%A /
001311  040     040     040             .ASCII  /   /
001314  040     040     123             .ASCII  /  S/
001317  124     101     124             .ASCII  /TAT/
001322  125     123     040             .ASCII  /US /
001325  127     117     122             .ASCII  /WOR/
001330  104     040     062             .ASCII  /D 2/
001333  040     040     040             .ASCII  /   /
001336  040     040     040             .ASCII  /   /
001341  040     040     040             .ASCII  /   /
001344  040     040     040             .ASCII  /   /
001347  040     040     045             .ASCII  /  %/
001352  117     066     045             .ASCII  /O6%/
001355  101     040     040             .ASCII  /A  /
001360  040     040     040             .ASCII  /   /
001363  040     040     040             .ASCII  /   /
001366  040     040     040             .ASCII  /   /
001371  040     045     117             .ASCII  / %O/
001374  066     045     116             .ASCII  /6%N/
001377  000                             .ASCII  <00>
001400  045     101     040     P.AAR:  .ASCII  /%A /
001403  040     040     040             .ASCII  /   /
001406  040     040     104             .ASCII  /  D/
001411  105     121     116             .ASCII  /EQN/
001414  101     040     103             .ASCII  /A C/
001417  123     122     040             .ASCII  /SR /
001422  122     105     107             .ASCII  /REG/
001425  111     123     124             .ASCII  /IST/
001430  105     122     040             .ASCII  /ER /
001433  040     040     040             .ASCII  /   /
001436  040     040     040             .ASCII  /   /
001441  040     040     040             .ASCII  /   /
001444  040     040     040             .ASCII  /   /
001447  040     040     040             .ASCII  /   '
001452  040     040     045             .ASCII  /  %/
001455  117     066     045             .ASCII  /O6%/
001460  116     000                     .ASCII  /N/<00>
001462  045     101     040     P.AAS:  .ASCII  /%A /
001465  040     040     040             .ASCII  /   /
001470  040     040     104             .ASCII  /  D/
001473  105     121     116             .ASCII  /EQN/
001476  101     040     111             .ASCII  /A I/
001501  057     117     040             .ASCII  <57>/O /
001504  120     101     107             .ASCII  /PAG/
001507  105     040     101             .ASCII  /E A/
001512  104     122     040             .ASCII  /DR /
001515  040     040     040             .ASCII  /   /
```

```
001520  040     040     040             .ASCII  /   /
001523  040     040     040             .ASCII  /   /
001526  040     040     040             .ASCII  /   /
001531  040     040     040             .ASCII  /   /
001534  040     040     045             .ASCII  /  %/
001537  117     066     045             .ASCII  /O6%/
001542  116     045     116             .ASCII  /N%N/
001545  000                             .ASCII  <00>
001546  045     101     040     P.AAT:  .ASCII  /%A /
001551  102     101     104             .ASCII  /BAD/
001554  040     103     123             .ASCII  / CS/
001557  122     072     040             .ASCII  /R: /
001562  101     103     124             .ASCII  /ACT/
001565  040     075     040             .ASCII  / = /
001570  045     117     066             .ASCII  /%O6/
001573  045     101     040             .ASCII  /%A /
001576  105     130     120             .ASCII  /EXP/
001601  040     075     040             .ASCII  / = /
001604  045     117     066             .ASCII  /%O6/
001607  045     116     000             .ASCII  /%N/<00>
001612  045     101     040     P.AAU:  .ASCII  /%A /
001615  102     101     104             .ASCII  /BAD/
001620  040     124     122             .ASCII  / TR/
001623  101     116     123             .ASCII  /ANS/
001626  115     111     124             .ASCII  /MIT/
001631  040     106     114             .ASCII  / FL/
001634  101     107     040             .ASCII  /AG /
001637  127     117     122             .ASCII  /WOR/
001642  104     072     040             .ASCII  /D: /
001645  101     103     124             .ASCII  /ACT/
001650  040     075     040             .ASCII  / = /
001653  045     117     066             .ASCII  /%O6/
001656  045     101     040             .ASCII  /%A /
001661  105     130     120             .ASCII  /EXP/
001664  040     075     040             .ASCII  / = /
001667  045     117     066             .ASCII  /%O6/
001672  045     116     000             .ASCII  /%N/<00>
001675  000                             .ASCII  <00>
001676  045     101     040     P.AAV:  .ASCII  /%A /
001701  102     101     104             .ASCII  /BAD/
001704  040     124     122             .ASCII  / TR/
001707  101     116     123             .ASCII  /ANS/
001712  115     111     124             .ASCII  /MIT/
001715  040     123     124             .ASCII  / ST/
001720  101     124     125             .ASCII  /ATU/
001723  123     040     127             .ASCII  /S W/
001726  117     122     104             .ASCII  /ORD/
001731  040     061     072             .ASCII  / 1:/
001734  040     101     103             .ASCII  / AC/
001737  124     040     075             .ASCII  /T =/
001742  040     045     117             .ASCII  / %O/
001745  066     045     101             .ASCII  /6%A/
001750  040     105     130             .ASCII  / EX/
```

```
001753  120  040  075              .ASCII  /P =/
001756  040  045  117              .ASCII  / %O/
001761  066  045  116              .ASCII  /6%N/
001764  000  000                   .ASCII  <00><00>
001766  045  101  040  P.AAW:      .ASCII  /%A /
001771  102  101  104              .ASCII  /BAD/
001774  040  122  105              .ASCII  / RE/
001777  103  105  111              .ASCII  /CEI/
002002  126  105  040              .ASCII  /VE /
002005  106  114  101              .ASCII  /FLA/
002010  107  040  127              .ASCII  /G W/
002013  117  122  104              .ASCII  /ORD/
002016  072  040  101              .ASCII  /: A/
002021  103  124  040              .ASCII  /CT /
002024  075  040  045              .ASCII  /= %/
002027  117  066  045              .ASCII  /O6%/
002032  101  040  105              .ASCII  /A E/
002035  130  120  040              .ASCII  /XP /
002040  075  040  045              .ASCII  /= %/
002043  117  066  045              .ASCII  /O6%/
002046  116  000                   .ASCII  /N/<00>
002050  045  101  040  P.AAX:      .ASCII  /%A /
002053  102  101  104              .ASCII  /BAD/
002056  040  122  105              .ASCII  / RE/
002061  103  105  111              .ASCII  /CEI/
002064  126  105  040              .ASCII  /VE /
002067  123  124  101              .ASCII  /STA/
002072  124  125  123              .ASCII  /TUS/
002075  040  127  117              .ASCII  / WO/
002100  122  104  040              .ASCII  /RD /
002103  061  072  040              .ASCII  /1: /
002106  101  103  124              .ASCII  /ACT/
002111  040  075  040              .ASCII  / = /
002114  045  117  066              .ASCII  /%O6/
002117  045  101  040              .ASCII  /%A /
002122  105  130  120              .ASCII  /EXP/
002125  040  075  040              .ASCII  / = /
002130  045  117  066              .ASCII  /%O6/
002133  045  116  000              .ASCII  /%N/<00>
002136  045  101  040  P.AAY:      .ASCII  /%A /
002141  102  101  104              .ASCII  /BAD/
002144  040  122  105              .ASCII  / RE/
002147  103  105  111              .ASCII  /CEI/
002152  126  105  040              .ASCII  /VE /
002155  102  125  106              .ASCII  /BUF/
002160  106  105  122              .ASCII  /FER/
002163  040  114  105              .ASCII  / LE/
002166  116  107  124              .ASCII  /NGT/
002171  110  072  040              .ASCII  /H: /
002174  101  103  124              .ASCII  /ACT/
002177  040  075  040              .ASCII  / = /
002202  045  117  066              .ASCII  /%O6/
002205  045  101  040              .ASCII  /%A /
```

```
002210  105  130  120                .ASCII  /EXP/
002213  040  075  040                .ASCII  / = /
002216  045  117  066                .ASCII  /%O6/
002221  045  116  000                .ASCII  /%N/<00>
002224  045  101  040  P.AAZ:        .ASCII  /%A /
002227  102  101  104                .ASCII  /BAD/
002232  040  103  123                .ASCII  / CS/
002235  122  040  075                .ASCII  /R =/
002240  040  045  117                .ASCII  / %O/
002243  066  045  116                .ASCII  /6%N/
002246  000  000                     .ASCII  <00><00>
002250  045  101  040  P.ABA:        .ASCII  /%A /
002253  114  117  117                .ASCII  /LOO/
002256  120  102  101                .ASCII  /PBA/
002261  103  113  040                .ASCII  /CK /
002264  120  101  103                .ASCII  /PAC/
002267  113  105  124                .ASCII  /KET/
002272  040  125  116                .ASCII  / UN/
002275  101  102  114                .ASCII  /ABL/
002300  105  040  124                .ASCII  /E T/
002303  117  040  123                .ASCII  /O S/
002306  105  124  040                .ASCII  /ET /
002311  103  101  040                .ASCII  /CA /
002314  102  111  124                .ASCII  /BIT/
002317  054  040  103                .ASCII  /, C/
002322  123  122  040                .ASCII  /SR /
002325  075  040  045                .ASCII  /= %/
002330  117  066  045                .ASCII  /O6%/
002333  116  000  000                .ASCII  /N/<00><00>
002336  045  101  040  P.ABB:        .ASCII  /%A /
002341  114  117  117                .ASCII  /LOO/
002344  120  102  101                .ASCII  /PBA/
002347  103  113  040                .ASCII  /CK /
002352  120  101  103                .ASCII  /PAC/
002355  113  105  124                .ASCII  /KET/
002360  040  125  116                .ASCII  / UN/
002363  101  102  114                .ASCII  /ABL/
002366  105  040  124                .ASCII  /E T/
002371  117  040  103                .ASCII  /O C/
002374  114  105  101                .ASCII  /LEA/
002377  122  040  103                .ASCII  /R C/
002402  101  040  102                .ASCII  /A B/
002405  111  124  054                .ASCII  /IT,/
002410  040  103  123                .ASCII  / CS/
002413  122  040  075                .ASCII  /R =/
002416  040  045  117                .ASCII  / %O/
002421  066  045  116                .ASCII  /6%N/
002424  000  000                     .ASCII  <00><00>
002426  045  101  040  P.ABC:        .ASCII  /%A /
002431  103  101  040                .ASCII  /CA /
002434  102  111  124                .ASCII  /BIT/
002437  040  117  113                .ASCII  / OK/
002442  054  040  102                .ASCII  /, B/
```

```
002445  125  124  040            .ASCII  /UT /
002450  122  111  040            .ASCII  /RI /
002453  102  111  124            .ASCII  /BIT/
002456  040  111  123            .ASCII  / IS/
002461  040  116  117            .ASCII  / NO/
002464  124  040  117            .ASCII  /T O/
002467  116  054  040            .ASCII  /N, /
002472  103  123  122            .ASCII  /CSR/
002475  040  075  040            .ASCII  / = /
002500  045  117  066            .ASCII  /%O6/
002503  045  116  000            .ASCII  /%N/<00>
002506  045  101  040  P.ABD:    .ASCII  /%A /
002511  103  101  040            .ASCII  /CA /
002514  102  111  124            .ASCII  /BIT/
002517  040  111  116            .ASCII  / IN/
002522  040  124  110            .ASCII  / TH/
002525  105  040  103            .ASCII  /E C/
002530  123  122  040            .ASCII  /SR /
002533  127  101  123            .ASCII  /WAS/
002536  040  123  105            .ASCII  / SE/
002541  124  040  124            .ASCII  /T T/
002544  117  117  040            .ASCII  /OO /
002547  105  101  122            .ASCII  /EAR/
002552  114  131  054            .ASCII  /LY,/
002555  040  103  123            .ASCII  / CS/
002560  122  040  075            .ASCII  /R =/
002563  040  045  117            .ASCII  / %O/
002566  066  045  116            .ASCII  /6%N/
002571  000                      .ASCII  <00>
002572  045  101  040  P.ABE:    .ASCII  /%A /
002575  130  114  040            .ASCII  /XL /
002600  101  116  104            .ASCII  /AND/
002603  040  122  114            .ASCII  / RL/
002606  040  050  040            .ASCII  / ( /
002611  102  111  124            .ASCII  /BIT/
002614  123  040  064            .ASCII  /S 4/
002617  054  065  040            .ASCII  /,5 /
002622  051  040  124            .ASCII  /) T/
002625  117  040  102            .ASCII  /O B/
002630  105  040  122            .ASCII  /E R/
002633  105  123  105            .ASCII  /ESE/
002636  124  040  124            .ASCII  /T T/
002641  117  040  060            .ASCII  /O 0/
002644  045  116  000            .ASCII  /%N/<00>
002647  000                      .ASCII  <00>
002650  045  101  040  P.ABF:    .ASCII  /%A /
002653  130  114  040            .ASCII  /XL /
002656  101  116  104            .ASCII  /AND/
002661  040  122  114            .ASCII  / RL/
002664  040  050  040            .ASCII  / ( /
002667  102  111  124            .ASCII  /BIT/
002672  123  040  064            .ASCII  /S 4/
002675  054  065  040            .ASCII  /,5 /
```

```
002700  051  040  124              .ASCII  /) T/
002703  117  040  102              .ASCII  /O B/
002706  105  040  123              .ASCII  /E S/
002711  105  124  040              .ASCII  /ET /
002714  124  117  040              .ASCII  /TO /
002717  061  045  116              .ASCII  /1%N/
002722  000  000                   .ASCII  <00><00>
002724  045  101  040  P.ABG:      .ASCII  /%A /
002727  122  111  040              .ASCII  /RI /
002732  050  040  102              .ASCII  /( B/
002735  111  124  040              .ASCII  /IT /
002740  061  065  040              .ASCII  /15 /
002743  051  040  124              .ASCII  /) T/
002746  117  040  102              .ASCII  /O B/
002751  105  040  123              .ASCII  /E S/
002754  105  124  040              .ASCII  /ET /
002757  124  117  040              .ASCII  /TO /
002762  061  045  116              .ASCII  /1%N/
002765  000                       .ASCII  <00>
002766  045  101  040  P.ABH:      .ASCII  /%A /
002771  130  111  040              .ASCII  /XI /
002774  050  040  102              .ASCII  /( B/
002777  111  124  040              .ASCII  /IT /
003002  067  040  051              .ASCII  /7 )/
003005  040  124  117              .ASCII  / TO/
003010  040  102  105              .ASCII  / BE/
003013  040  123  105              .ASCII  / SE/
003016  124  040  124              .ASCII  /T T/
003021  117  040  061              .ASCII  /O 1/
003024  045  116  000              .ASCII  /%N/<00>
003027  000                       .ASCII  <00>
003030  045  101  040  P.ABI:      .ASCII  /%A /
003033  116  111  040              .ASCII  /NI /
003036  050  040  102              .ASCII  /( B/
003041  111  124  040              .ASCII  /IT /
003044  062  040  051              .ASCII  /2 )/
003047  040  124  117              .ASCII  / TO/
003052  040  102  105              .ASCII  / BE/
003055  040  123  105              .ASCII  / SE/
003060  124  040  124              .ASCII  /T T/
003063  117  040  061              .ASCII  /O 1/
003066  045  116  000              .ASCII  /%N/<00>
003071  000                       .ASCII  <00>
003072  045  101  040  P.ABJ:      .ASCII  /%A /
003075  116  111  040              .ASCII  /NI /
003100  050  040  102              .ASCII  /( B/
003103  111  124  040              .ASCII  /IT /
003106  062  040  051              .ASCII  /2 )/
003111  040  124  117              .ASCII  / TO/
003114  040  102  105              .ASCII  / BE/
003117  040  122  105              .ASCII  / RE/
003122  123  105  124              .ASCII  /SET/
003125  040  124  117              .ASCII  / TO/
```

```
003130  040  060  045            .ASCII  / 0%/
003133  116  000  000            .ASCII  /N/<00><00>
003136  045  101  040  P.ABK:    .ASCII  /%A /
003141  102  101  104            .ASCII  /BAD/
003144  040  103  123            .ASCII  / CS/
003147  122  054  040            .ASCII  /R, /
003152  105  130  120            .ASCII  /EXP/
003155  105  103  124            .ASCII  /ECT/
003160  105  104  000            .ASCII  /ED/<00>
003163  000                      .ASCII  <00>
003164  045  101  040  P.ABL:    .ASCII  /%A /
003167  103  123  122            .ASCII  /CSR/
003172  040  101  104            .ASCII  / AD/
003175  122  040  075            .ASCII  /R =/
003200  040  045  117            .ASCII  / %O/
003203  066  045  101            .ASCII  /6%A/
003206  040  040  101            .ASCII  /  A/
003211  103  124  125            .ASCII  /CTU/
003214  101  114  040            .ASCII  /AL /
003217  075  040  045            .ASCII  /= %/
003222  117  066  045            .ASCII  /O6%/
003225  101  040  040            .ASCII  /A  /
003230  105  130  120            .ASCII  /EXP/
003233  105  103  124            .ASCII  /ECT/
003236  105  104  040            .ASCII  /ED /
003241  075  040  045            .ASCII  /= %/
003244  117  066  045            .ASCII  /O6%/
003247  116  000  000            .ASCII  /N/<00><00>
003252  045  116  045  P.ABM:    .ASCII  /%N%/
003255  101  040  125            .ASCII  /A U/
003260  116  101  102            .ASCII  /NAB/
003263  114  105  040            .ASCII  /LE /
003266  124  117  040            .ASCII  /TO /
003271  122  105  123            .ASCII  /RES/
003274  105  124  040            .ASCII  /ET /
003277  104  105  121            .ASCII  /DEQ/
003302  116  101  072            .ASCII  /NA:/
003305  040  101  104            .ASCII  / AD/
003310  122  072  040            .ASCII  /R: /
003313  045  117  066            .ASCII  /%O6/
003316  045  101  040            .ASCII  /%A /
003321  040  103  123            .ASCII  / CS/
003324  122  040  075            .ASCII  /R =/
003327  040  045  117            .ASCII  / %O/
003332  066  045  116            .ASCII  /6%N/
003335  000                      .ASCII  <00>
003336  045  116  045  P.ABN:    .ASCII  /%N%/
003341  101  040  127            .ASCII  /A W/
003344  101  111  124            .ASCII  /AIT/
003347  040  101  102            .ASCII  / AB/
003352  117  125  124            .ASCII  /OUT/
003355  040  045  104            .ASCII  / %D/
003360  062  045  101            .ASCII  /2%A/
```

```
003363  040  123  105                .ASCII  / SE/
003366  103  117  116                .ASCII  /CON/
003371  104  050  123                .ASCII  /D(S/
003374  051  040  055                .ASCII  /) -/
003377  000                         .ASCII  <00>
003400  045  116  045       P.ABO:   .ASCII  /%N%/
003403  101  040  123                .ASCII  /A S/
003406  101  116  111                .ASCII  /ANI/
003411  124  131  040                .ASCII  /TY /
003414  124  111  115                .ASCII  /TIM/
003417  105  122  040                .ASCII  /ER /
003422  124  111  115                .ASCII  /TIM/
003425  105  104  040                .ASCII  /ED /
003430  117  125  124                .ASCII  /OUT/
003433  040  101  123                .ASCII  / AS/
003436  040  105  130                .ASCII  / EX/
003441  120  105  103                .ASCII  /PEC/
003444  124  105  104                .ASCII  /TED/
003447  040  045  116                .ASCII  / %N/
003452  000  000                    .ASCII  <00><00>
003454  045  116  045       P.ABP:   .ASCII  /%N%/
003457  101  040  116                .ASCII  /A N/
003462  117  040  123                .ASCII  /O S/
003465  101  116  111                .ASCII  /ANI/
003470  124  131  040                .ASCII  /TY /
003473  124  111  115                .ASCII  /TIM/
003476  105  122  040                .ASCII  /ER /
003501  111  116  124                .ASCII  /INT/
003504  105  122  122                .ASCII  /ERR/
003507  125  120  124                .ASCII  /UPT/
003512  040  104  105                .ASCII  / DE/
003515  124  105  103                .ASCII  /TEC/
003520  124  105  104                .ASCII  /TED/
003523  040  045  116                .ASCII  / %N/
003526  000  000                    .ASCII  <00><00>
003530  045  116  045       P.ABQ:   .ASCII  /%N%/
003533  101  040  104                .ASCII  /A D/
003536  111  123  103                .ASCII  /ISC/
003541  117  116  116                .ASCII  /ONN/
003544  105  103  124                .ASCII  /ECT/
003547  040  124  122                .ASCII  / TR/
003552  101  116  123                .ASCII  /ANS/
003555  103  105  111                .ASCII  /CEI/
003560  126  105  122                .ASCII  /VER/
003563  040  103  101                .ASCII  / CA/
003566  102  114  105                .ASCII  /BLE/
003571  040  106  122                .ASCII  / FR/
003574  117  115  040                .ASCII  /OM /
003577  102  125  114                .ASCII  /BUL/
003602  113  110  105                .ASCII  /KHE/
003605  101  104  040                .ASCII  /AD /
003610  101  123  123                .ASCII  /ASS/
003613  105  115  102                .ASCII  /EMB/
```

```
003616  114  131  040                .ASCII  /LY /
003621  101  116  104                .ASCII  /AND/
003624  000  000                     .ASCII  <00><00>
003626  045  116  045        P.ABR:   .ASCII  /%N%/
003631  101  040  103                .ASCII  /A C/
003634  117  116  116                .ASCII  /ONN/
003637  105  103  124                .ASCII  /ECT/
003642  040  114  117                .ASCII  / LO/
003645  117  120  102                .ASCII  /OPB/
003650  101  103  113                .ASCII  /ACK/
003653  040  103  117                .ASCII  / CO/
003656  116  116  105                .ASCII  /NNE/
003661  103  124  117                .ASCII  /CTO/
003664  122  040  124                .ASCII  /R T/
003667  117  040  102                .ASCII  /O B/
003672  125  114  113                .ASCII  /ULK/
003675  110  105  101                .ASCII  /HEA/
003700  104  040  101                .ASCII  /D A/
003703  123  123  105                .ASCII  /SSE/
003706  115  102  114                .ASCII  /MBL/
003711  131  054  040                .ASCII  /Y, /
003714  124  110  105                .ASCII  /THE/
003717  116  040  122                .ASCII  /N R/
003722  105  124  105                .ASCII  /ETE/
003725  123  124  045                .ASCII  /ST%/
003730  116  000                     .ASCII  /N/<00>
003732  045  116  045        P.ABS:   .ASCII  /%N%/
003735  101  040  104                .ASCII  /A D/
003740  111  123  103                .ASCII  /ISC/
003743  117  116  116                .ASCII  /ONN/
003746  105  103  124                .ASCII  /ECT/
003751  040  102  125                .ASCII  / BU/
003754  114  113  110                .ASCII  /LKH/
003757  105  101  104                .ASCII  /EAD/
003762  040  101  123                .ASCII  / AS/
003765  123  105  115                .ASCII  /SEM/
003770  102  114  131                .ASCII  /BLY/
003773  040  106  122                .ASCII  / FR/
003776  117  115  040                .ASCII  /OM /
004001  104  105  121                .ASCII  /DEQ/
004004  116  101  040                .ASCII  /NA /
004007  101  116  104                .ASCII  /AND/
004012  040  103  117                .ASCII  / CO/
004015  116  116  105                .ASCII  /NNE/
004020  103  124  000                .ASCII  /CT/<00>
004023  000                         .ASCII  <00>
004024  045  116  045        P.ABT:   .ASCII  /%N%/
004027  101  040  114                .ASCII  /A L/
004032  117  117  120                .ASCII  /OOP/
004035  102  101  103                .ASCII  /BAC/
004040  113  040  103                .ASCII  /K C/
004043  117  116  116                .ASCII  /ONN/
004046  105  103  124                .ASCII  /ECT/
```

```
004051  117     122     040             .ASCII  /OR /
004054  124     117     040             .ASCII  /TO /
004057  104     105     121             .ASCII  /DEQ/
004062  116     101     054             .ASCII  /NA,/
004065  040     124     110             .ASCII  / TH/
004070  105     116     040             .ASCII  /EN /
004073  122     105     124             .ASCII  /RET/
004076  105     123     124             .ASCII  /EST/
004101  045     116     000             .ASCII  /%N/<00>
004104  045     116     045     P.ABU:  .ASCII  /%N%/
004107  101     040     103             .ASCII  /A C/
004112  110     105     103             .ASCII  /HEC/
004115  113     040     106             .ASCII  /K F/
004120  117     122     040             .ASCII  /OR /
004123  114     117     117             .ASCII  /LOO/
004126  123     105     040             .ASCII  /SE /
004131  127     111     122             .ASCII  /WIR/
004134  105     123     040              ASCII  /ES /
004137  111     116     040             .ASCII  /IN /
004142  101     040     114             .ASCII  /A L/
004145  117     117     120             .ASCII  /OOP/
004150  102     101     103             .ASCII  /BAC/
004153  113     040     103             .ASCII  /K C/
004156  117     116     116             .ASCII  /ONN/
004161  105     103     124             .ASCII  /ECT/
004164  117     122     000             .ASCII  /OR/<00>
004167  000                             .ASCII  <00>
004170  045     116     045     P.ABV:  .ASCII  /%N%/
004173  101     040     117             .ASCII  /A O/
004176  122     040     125             .ASCII  /R U/
004201  123     105     040             .ASCII  /SE /
004204  104     111     106             .ASCII  /DIF/
004207  106     105     122             .ASCII  /FER/
004212  105     116     124             .ASCII  /ENT/
004215  040     114     117             .ASCII  / LO/
004220  117     120     102             .ASCII  /OPB/
004223  101     103     113             .ASCII  /ACK/
004226  040     103     117             .ASCII  / CO/
004231  116     116     105             .ASCII  /NNE/
004234  103     124     117             .ASCII  /CTO/
004237  122     054     040             .ASCII  /R, /
004242  124     110     105             .ASCII  /THE/
004245  116     040     122             .ASCII  /N R/
004250  105     124     105             .ASCII  /ETE/
004253  123     124     045             .ASCII  /ST%/
004256  116     000                     .ASCII  /N/<00>
004260  045     116     045     P.ABW:  .ASCII  /%N%/
004263  101     040     122             .ASCII  /A R/
004266  105     120     114             .ASCII  /EPL/
004271  101     103     105             .ASCII  /ACE/
004274  040     104     105             .ASCII  / DE/
004277  121     116     101             .ASCII  /QNA/
004302  054     040     124             .ASCII  /, T/
```

```
004305  110     105     116             .ASCII  /HEN/
004310  040     122     105             .ASCII  / RE/
004313  124     105     123             .ASCII  /IES/
004316  124     045     116             .ASCII  /T%N/
004321  000                             .ASCII  <00>
004322  045     116     045     P.ABX:  .ASCII  /%N%/
004325  101     040     122             .ASCII  /A R/
004330  105     120     114             .ASCII  /EPL/
004333  101     103     105             .ASCII  /ACE/
004336  040     102     125             .ASCII  / BU/
004341  114     113     110             .ASCII  /LKH/
004344  105     101     104             .ASCII  /EAD/
004347  040     103     117             .ASCII  / CO/
004352  116     116     105             .ASCII  /NNE/
004355  103     124     117             .ASCII  /CTO/
004360  122     054     040             .ASCII  /R, /
004363  124     110     105             .ASCII  /THE/
004366  116     040     122             .ASCII  /N R/
004371  105     124     105             .ASCII  /ETE/
004374  123     124     045             .ASCII  /ST%/
004377  116     000     000             .ASCII  /N/<00><00>
004402  045     116     045     P.ABY:  .ASCII  /%N%/
004405  101     040     104             .ASCII  /A D/
004410  111     123     103             .ASCII  /ISC/
004413  117     116     116             .ASCII  /ONN/
004416  105     103     124             .ASCII  /ECT/
004421  040     124     122             .ASCII  / TR/
004424  101     116     123             .ASCII  /ANS/
004427  103     105     111             .ASCII  /CEI/
004432  126     105     122             .ASCII  /VER/
004435  040     103     101             .ASCII  / CA/
004440  102     114     105             .ASCII  /BLE/
004443  040     106     122             .ASCII  / FR/
004446  117     115     040             .ASCII  /OM /
004451  124     122     101             .ASCII  /TRA/
004454  116     123     103             .ASCII  /NSC/
004457  105     111     126             .ASCII  /EIV/
004462  105     122     000             .ASCII  /ER/<00>
004465  000                             .ASCII  <00>
004466  045     116     045     P.ABZ:  .ASCII  /%N%/
004471  101     040     101             .ASCII  /A A/
004474  116     104     040             .ASCII  /ND /
004477  103     117     116             .ASCII  /CON/
004502  116     105     103             .ASCII  /NEC/
004505  124     040     111             .ASCII  /T I/
004510  124     040     124             .ASCII  /T T/
004513  117     040     114             .ASCII  /O L/
004516  117     117     120             .ASCII  /OOP/
004521  102     101     103             .ASCII  /BAC/
004524  113     040     103             .ASCII  /K C/
004527  117     116     116             .ASCII  /ONN/
004532  105     103     124             .ASCII  /ECT/
004535  117     122     040             .ASCII  /OR /
```

```
004540  101  116  104              .ASCII  /AND/
004543  040  102  125              .ASCII  / BU/
004546  114  113  110              .ASCII  /LKH/
004551  105  101  104              .ASCII  /EAD/
004554  040  101  123              .ASCII  / AS/
004557  123  105  115              .ASCII  /SEM/
004562  102  114  131              .ASCII  /BLY/
004565  045  116  000              .ASCII  /%N/<00>
004570  045  116  045    P.ACA:    .ASCII  /%N%/
004573  101  040  122              .ASCII  /A R/
004576  105  120  114              .ASCII  /EPL/
004601  101  103  105              .ASCII  /ACE/
004604  040  124  122              .ASCII  / TR/
004607  101  116  123              .ASCII  /ANS/
004612  103  105  111              .ASCII  /CEI/
004615  126  105  122              .ASCII  /VER/
004620  040  103  101              .ASCII  / CA/
004623  102  114  105              .ASCII  /BLE/
004626  054  040  124              .ASCII  /, T/
004631  110  105  116              .ASCII  /HEN/
004634  040  122  105              .ASCII  / RE/
004637  124  105  123              .ASCII  /TES/
004642  124  045  116              .ASCII  /T%N/
004645  000                       .ASCII  <00>
004646  045  116  045    P.ACB:    .ASCII  /%N%/
004651  101  040  122              .ASCII  /A R/
004654  105  120  114              .ASCII  /EPL/
004657  101  103  105              .ASCII  /ACE/
004662  040  124  122              .ASCII  / TR/
004665  101  116  123              .ASCII  /ANS/
004670  103  105  111              .ASCII  /CEI/
004673  126  105  122              .ASCII  /VER/
004676  054  040  124              .ASCII  /, T/
004701  110  105  116              .ASCII  /HEN/
004704  040  122  105              .ASCII  / RE/
004707  124  105  123              .ASCII  /TES/
004712  124  045  116              .ASCII  /T%N/
004715  000                       .ASCII  <00>
004716  045  116  045    P.ACC:    .ASCII  /%N%/
004721  101  040  122              .ASCII  /A R/
004724  105  120  114              .ASCII  /EPL/
004727  101  103  105              .ASCII  /ACE/
004732  040  124  110              .ASCII  / TH/
004735  105  040  106              .ASCII  /E F/
004740  125  123  105              .ASCII  /USE/
004743  040  111  106              .ASCII  / IF/
004746  040  102  101              .ASCII  / BA/
004751  104  054  040              .ASCII  /D, /
004754  124  110  105              .ASCII  /THE/
004757  116  040  122              .ASCII  /N R/
004762  105  124  105              .ASCII  /ETE/
004765  123  124  045              .ASCII  /ST%/
004770  116  000                   .ASCII  /N/<00>
```

```
004772  045  116  045        P.ACD:  .ASCII  /%N%/
004775  101  040  102                .ASCII  /A B/
005000  101  104  040                .ASCII  /AD /
005003  122  105  103                .ASCII  /REC/
005006  105  111  126                .ASCII  /EIV/
005011  105  040  104                .ASCII  /E D/
005014  105  123  103                .ASCII  /ESC/
005017  122  111  120                .ASCII  /RIP/
005022  124  117  122                .ASCII  /TOR/
005025  072  000  000                .ASCII  /:/<00><00>
005030  045  116  045        P.ACE:  .ASCII  /%N%/
005033  101  040  102                .ASCII  /A B/
005036  101  104  040                .ASCII  /AD /
005041  124  122  101                .ASCII  /TRA/
005044  116  123  115                .ASCII  /NSM/
005047  111  124  040                .ASCII  /IT /
005052  104  105  123                .ASCII  /DES/
005055  103  122  111                .ASCII  /CRI/
005060  120  124  117                .ASCII  /PTO/
005063  122  072  000                .ASCII  /R:/<00>
005066  045  101  040        P.ACF:  .ASCII  /%A /
005071  101  103  124                .ASCII  /ACT/
005074  125  101  114                .ASCII  /UAL/
005077  040  075  040                .ASCII  / = /
005102  045  117  066                .ASCII  /%06/
005105  045  101  040                .ASCII  /%A /
005110  105  130  120                .ASCII  /EXP/
005113  105  103  124                .ASCII  /ECT/
005116  105  104  040                .ASCII  /ED /
005121  075  040  045                .ASCII  /= %/
005124  117  066  045                .ASCII  /06%/
005127  101  040  111                .ASCII  /A I/
005132  116  104  105                .ASCII  /NDE/
005135  130  040  075                .ASCII  /X =/
005140  040  045  104                .ASCII  / %D/
005143  064  045  116                .ASCII  /4%N/
005146  000  000                     .ASCII  <00><00>
005150  045  116  045        P.ACG:  .ASCII  /%N%/
005153  101  040  102                .ASCII  /A B/
005156  101  104  040                .ASCII  /AD /
005161  122  105  103                .ASCII  /REC/
005164  105  111  126                .ASCII  /EIV/
005167  105  040  102                .ASCII  /E B/
005172  125  106  106                .ASCII  /UFF/
005175  105  122  072                .ASCII  /ER:/
005200  000  000                     .ASCII  <00><00>
005202  045  116  045        P.ACH:  .ASCII  /%N%/
005205  101  040  104                .ASCII  /A D/
005210  115  101  040                .ASCII  /MA /
005213  117  120  105                .ASCII  /OPE/
005216  122  101  124                .ASCII  /RAT/
005221  111  117  116                .ASCII  /ION/
005224  040  124  101                .ASCII  / TA/
```

```
005227  113  105  123                .ASCII  /KES/
005232  040  124  117                .ASCII  / TO/
005235  117  040  114                .ASCII  /O L/
005240  117  116  107                .ASCII  /ONG/
005243  045  116  000                .ASCII  /%N/<00>
005246  045  116  045  P.ACI:        .ASCII  /%N%/
005251  101  040  124                .ASCII  /A T/
005254  117  117  040                .ASCII  /OO /
005257  115  101  116                .ASCII  /MAN/
005262  131  040  104                .ASCII  /Y D/
005265  105  126  111                .ASCII  /EVI/
005270  103  105  123                .ASCII  /CES/
005273  045  116  000                .ASCII  /%N/<00>
005276  045  116  045  P.ACJ:        .ASCII  /%N%/
005301  101  040  124                .ASCII  /A T/
005304  110  105  122                .ASCII  /HER/
005307  105  040  127                .ASCII  /E W/
005312  101  123  040                .ASCII  /AS /
005315  101  040  120                .ASCII  /A P/
005320  117  127  105                .ASCII  /OWE/
005323  122  040  106                .ASCII  /R F/
005326  101  111  114                .ASCII  /AIL/
005331  040  055  040                .ASCII  / - /
005334  127  101  111                .ASCII  /WAI/
005337  124  111  116                .ASCII  /TIN/
005342  107  045  116                .ASCII  /G%N/
005345  000                         .ASCII  <00>
005346  045  116  045  P.ACK:        .ASCII  /%N%/
005351  101  040  127                .ASCII  /A W/
005354  101  111  124                .ASCII  /AIT/
005357  040  101  102                .ASCII  / AB/
005362  117  125  124                .ASCII  /OUT/
005365  040  045  104                .ASCII  / %D/
005370  062  045  101                .ASCII  /2%A/
005373  040  115  111                .ASCII  / MI/
005376  116  125  124                .ASCII  /NUT/
005401  105  050  123                .ASCII  /E(S/
005404  051  040  055                .ASCII  /) -/
005407  000                         .ASCII  <00>
005410  045  116  045  P.ACL:        .ASCII  /%N%/
005413  101  040  127                .ASCII  /A W/
005416  101  111  124                .ASCII  /AIT/
005421  040  101  102                .ASCII  / AB/
005424  117  125  124                .ASCII  /OUT/
005427  040  045  104                .ASCII  / %D/
005432  062  045  101                .ASCII  /2%A/
005435  040  110  117                .ASCII  / HO/
005440  125  122  040                .ASCII  /UR /
005443  055  000  000                .ASCII  /-/<00><00>
005446  045  101  040  P.ACM:        .ASCII  /%A /
005451  111  106  040                .ASCII  /IF /
005454  116  117  040                .ASCII  /NO /
005457  122  105  123                .ASCII  /RES/
```

```
005462  105  124  054                .ASCII  /ET,/
005465  040  124  131                .ASCII  / TY/
005470  120  105  040                .ASCII  /PE /
005473  101  116  131                .ASCII  /ANY/
005476  040  103  110                .ASCII  / CH/
005501  101  122  101                .ASCII  /ARA/
005504  103  124  105                .ASCII  /CTE/
005507  122  040  124                .ASCII  /R T/
005512  117  040  105                .ASCII  /O E/
005515  130  111  124                .ASCII  /XIT/
005520  040  106  122                .ASCII  / FR/
005523  117  115  040                .ASCII  /OM /
005526  124  105  123                .ASCII  /TES/
005531  124  045  116                .ASCII  /T%N/
005534  000  000                     .ASCII  <00><00>
005536  045  116  045          P.ACN: .ASCII  /%N%/
005541  101  040  124                .ASCII  /A T/
005544  104  122  040                .ASCII  /DR /
005547  126  101  114                .ASCII  /VAL/
005552  125  105  040                .ASCII  /UE /
005555  111  123  040                .ASCII  /IS /
005560  105  121  125                .ASCII  /EQU/
005563  101  114  040                .ASCII  /AL /
005566  124  117  040                .ASCII  /TO /
005571  132  105  122                .ASCII  /ZER/
005574  117  040  045                .ASCII  /O %/
005577  116  000  000                .ASCII  /N/<00><00>
005602  045  116  045          P.ACO: .ASCII  /%N%/
005605  116  045  101                .ASCII  /N%A/
005610  055  055  055                .ASCII  /---/
005613  055  055  055                .ASCII  /---/
005616  055  055  055                .ASCII  /---/
005621  055  055  055                .ASCII  /---/
005624  055  055  055                .ASCII  /---/
005627  055  055  055                .ASCII  /---/
005632  055  055  055                .ASCII  /---/
005635  055  055  055                .ASCII  /---/
005640  055  055  055                .ASCII  /---/
005643  055  055  055                .ASCII  /---/
005646  055  055  055                .ASCII  /---/
005651  055  055  055                .ASCII  /---/
005654  055  055  055                .ASCII  /---/
005657  055  055  055                .ASCII  /---/
005662  055  055  055                .ASCII  /---/
005665  055  055  055                .ASCII  /---/
005670  055  055  055                .ASCII  /---/
005673  055  055  055                .ASCII  /---/
005676  055  055  055                .ASCII  /---/
005701  055  055  055                .ASCII  /---/
005704  055  055  055                .ASCII  /---/
005707  055  055  045                .ASCII  /--%/
005712  116  000                     .ASCII  /N/<00>
005714  045  116  045          P.ACP: .ASCII  /%N%/
```

```
005717  101  040  102              .ASCII  /A B/
005722  101  104  040              .ASCII  /AD /
005725  103  123  122              .ASCII  /CSR/
005730  054  040  102              .ASCII  /, B/
005733  111  124  123              .ASCII  /ITS/
005736  040  123  124              .ASCII  / ST/
005741  125  103  113              .ASCII  /UCK/
005744  040  101  124              .ASCII  / AT/
005747  040  060  072              .ASCII  / 0:/
005752  045  116  000              .ASCII  /%N/<00>
005755  000                       .ASCII  <00>
005756  045  116  045  P.ACQ:      .ASCII  /%N%/
005761  101  040  102              .ASCII  /A B/
005764  101  104  040              .ASCII  /AD /
005767  103  123  122              .ASCII  /CSR/
005772  054  040  102              .ASCII  /, B/
005775  111  124  123              .ASCII  /ITS/
006000  040  123  124              .ASCII  / ST/
006003  125  103  113              .ASCII  /UCK/
006006  040  101  124              .ASCII  / AT/
006011  040  061  072              .ASCII  / 1:/
006014  045  116  000              .ASCII  /%N/<00>
006017  000                       .ASCII  <00>
006020  045  116  045  P.ACR:      .ASCII  /%N%/
006023  101  040  123              .ASCII  /A S/
006026  117  106  124              .ASCII  /OFT/
006031  127  101  122              .ASCII  /WAR/
006034  105  040  122              .ASCII  /E R/
006037  105  123  105              .ASCII  /ESE/
006042  124  040  125              .ASCII  /T U/
006045  116  101  102              .ASCII  /NAB/
006050  114  105  040              .ASCII  /LE /
006053  124  117  040              .ASCII  /TO /
006056  103  114  105              .ASCII  /CLE/
006061  101  122  040              .ASCII  /AR /
006064  103  123  122              .ASCII  /CSR/
006067  040  123  124              .ASCII  / ST/
006072  101  124  111              .ASCII  /ATI/
006075  103  040  102              .ASCII  /C B/
006100  111  124  123              .ASCII  /ITS/
006103  072  045  116              .ASCII  /:%N/
006106  000  000                  .ASCII  <00><00>
006110  045  116  045  P.ACS:      .ASCII  /%N%/
006113  101  040  102              .ASCII  /A B/
006116  101  104  040              .ASCII  /AD /
006121  123  124  101              .ASCII  /STA/
006124  124  111  117              .ASCII  /TIO/
006127  116  040  101              .ASCII  /N A/
006132  104  104  122              .ASCII  /DDR/
006135  105  123  123              .ASCII  /ESS/
006140  040  103  110              .ASCII  / CH/
006143  105  103  113              .ASCII  /ECK/
006146  123  125  115              .ASCII  /SUM/
```

```
006151  072  040  101              .ASCII  /: A/
006154  103  124  040              .ASCII  /CT /
006157  075  040  045              .ASCII  /= %/
006162  117  066  045              .ASCII  /O6%/
006165  101  040  105              .ASCII  /A E/
006170  130  120  040              .ASCII  /XP /
006173  075  040  045              .ASCII  /= %/
006176  117  066  045              .ASCII  /O6%/
006201  116  000  000              .ASCII  /N/<00><00>
006204  045  116  045  P.ACT:      .ASCII  /%N%/
006207  101  040  102              .ASCII  /A B/
006212  101  104  040              .ASCII  /AD /
006215  123  124  101              .ASCII  /STA/
006220  124  111  117              .ASCII  /TIO/
006223  116  040  101              .ASCII  /N A/
006226  104  104  122              .ASCII  /DDR/
006231  105  123  123              .ASCII  /ESS/
006234  072  040  000              .ASCII  /: /<00>
006237  000                       .ASCII  <00>
006240  045  116  045  P.ACU:      .ASCII  /%N%/
006243  101  040  102              .ASCII  /A B/
006246  101  104  040              .ASCII  /AD /
006251  104  105  121              .ASCII  /DEQ/
006254  116  101  040              .ASCII  /NA /
006257  111  057  117              .ASCII  /I/<57>/O/
006262  040  120  101              .ASCII  / PA/
006265  107  105  040              .ASCII  /GE /
006270  122  105  107              .ASCII  /REG/
006273  111  123  124              .ASCII  /IST/
006276  105  122  072              .ASCII  /ER:/
006301  045  116  000              .ASCII  /%N/<00>
006304  045  116  045  P.ACV:      .ASCII  /%N%/
006307  101  040  102              .ASCII  /A B/
006312  101  104  040              .ASCII  /AD /
006315  103  123  122              .ASCII  /CSR/
006320  054  040  105              .ASCII  /, E/
006323  130  120  105              .ASCII  /XPE/
006326  103  124  105              .ASCII  /CTE/
006331  104  040  122              .ASCII  /D R/
006334  114  040  050              .ASCII  /L (/
006337  040  102  111              .ASCII  / BI/
006342  124  040  065              .ASCII  /T 5/
006345  040  051  040              .ASCII  / ) /
006350  124  117  040              .ASCII  /TO /
006353  102  105  040              .ASCII  /BE /
006356  123  105  124              .ASCII  /SET/
006361  040  124  117              .ASCII  / TO/
006364  040  060  045              .ASCII  / 0%/
006367  116  000  000              .ASCII  /N/<00><00>
006372  045  116  045  P.ACW:      .ASCII  /%N%/
006375  101  040  102              .ASCII  /A B/
006400  101  104  040              .ASCII  /AD /
006403  102  057  104              .ASCII  /B/<57>/D/
```

```
006406  040  120  122                .ASCII  / PR/
006411  117  115  040                .ASCII  /OM /
006414  103  110  105                .ASCII  /CHE/
006417  103  113  123                .ASCII  /CKS/
006422  125  115  072                .ASCII  /UM:/
006425  040  111  116                .ASCII  / IN/
006430  104  105  130                .ASCII  /DEX/
006433  040  075  040                .ASCII  / = /
006436  045  117  066                .ASCII  /%O6/
006441  045  101  040                .ASCII  /%A /
006444  101  103  124                .ASCII  /ACT/
006447  040  075  040                .ASCII  / = /
006452  045  117  066                .ASCII  /%O6/
006455  045  101  040                .ASCII  /%A /
006460  105  130  120                .ASCII  /EXP/
006463  040  075  040                .ASCII  / = /
006466  045  117  066                .ASCII  /%O6/
006471  045  116  000                .ASCII  /%N/<00>
006474  045  116  045        P.ACX:  .ASCII  /%N%/
006477  101  040  102                .ASCII  /A B/
006502  057  104  040                .ASCII  <57>/D /
006505  120  122  117                .ASCII  /PRO/
006510  115  040  103                .ASCII  /M C/
006513  110  105  103                .ASCII  /HEC/
006516  113  123  125                .ASCII  /KSU/
006521  115  040  117                .ASCII  /M O/
006524  106  106  123                .ASCII  /FFS/
006527  105  124  040                .ASCII  /ET /
006532  075  040  045                .ASCII  /= %/
006535  117  066  045                .ASCII  /O6%/
006540  101  040  101                .ASCII  /A A/
006543  103  124  040                .ASCII  /CT /
006546  075  040  045                .ASCII  /= %/
006551  117  066  045                .ASCII  /O6%/
006554  101  040  105                .ASCII  /A E/
006557  130  120  040                .ASCII  /XP /
006562  075  040  045                .ASCII  /= %/
006565  117  066  045                .ASCII  /O6%/
006570  116  000                    .ASCII  /N/<00>
006572  045  116  045        P.ACY:  .ASCII  /%N%/
006575  101  040  102                .ASCII  /A B/
006600  101  104  040                .ASCII  /AD /
006603  111  116  124                .ASCII  /INT/
006606  105  122  122                .ASCII  /ERR/
006611  125  120  124                .ASCII  /UPT/
006614  072  040  101                .ASCII  /: A/
006617  104  122  040                .ASCII  /DR /
006622  075  040  045                .ASCII  /= %/
006625  117  066  045                .ASCII  /O6%/
006630  101  040  101                .ASCII  /A A/
006633  103  124  040                .ASCII  /CT /
006636  114  105  126                .ASCII  /LEV/
006641  040  075  040                .ASCII  / = /
```

```
006644  045  117  066                .ASCII  /%06/
006647  045  101  040                .ASCII  /%A /
006652  105  130  120                .ASCII  /EXP/
006655  040  114  105                .ASCII  / LE/
006660  126  040  075                .ASCII  /V =/
006663  040  045  117                .ASCII  / %O/
006666  066  045  116                .ASCII  /6%N/
006671  000                         .ASCII  <00>
006672  045  116  045       P.ACZ:   .ASCII  /%N%/
006675  101  040  122                .ASCII  /A R/
006700  105  107  111                .ASCII  /EGI/
006703  123  124  105                .ASCII  /STE/
006706  122  040  106                .ASCII  /R F/
006711  101  111  114                .ASCII  /AIL/
006714  105  104  040                .ASCII  /ED /
006717  124  117  040                .ASCII  /TO /
006722  122  105  123                .ASCII  /RES/
006725  120  117  116                .ASCII  /PON/
006730  104  040  101                .ASCII  /D A/
006733  124  040  101                .ASCII  /T A/
006736  104  104  122                .ASCII  /DDR/
006741  105  123  123                .ASCII  /ESS/
006744  072  040  040                .ASCII  /:  /
006747  045  117  066                .ASCII  /%06/
006752  045  116  000                .ASCII  /%N/<00>
006755  000                         .ASCII  <00>
006756  045  116  045       P.ADA:   .ASCII  /%N%/
006761  101  040  102                .ASCII  /A B/
006764  101  104  040                .ASCII  /AD /
006767  124  122  101                .ASCII  /TRA/
006772  116  123  115                .ASCII  /NSM/
006775  111  124  040                .ASCII  /IT /
007000  123  124  101                .ASCII  /STA/
007003  124  125  123                .ASCII  /TUS/
007006  054  040  124                .ASCII  /, T/
007011  117  117  040                .ASCII  /OO /
007014  115  101  116                .ASCII  /MAN/
007017  131  040  103                .ASCII  /Y C/
007022  117  114  114                .ASCII  /OLL/
007025  111  123  111                .ASCII  /ISI/
007030  117  116  123                .ASCII  /ONS/
007033  045  116  000                .ASCII  /%N/<00>


000000                              .PSECT  $GLOB$,  D
000000                    RCV.D.LIST::
                                    .BLKW   100
000200                    XMIT.D.LIST::
                                    .BLKW   100
000400                    RCV.BUFFER::
                                    .BLKW   2000
004400                    XMIT.BUFFER::
```

```
                                        .BLKW   2000
010400                          PHYS.ADR::
                                        .BLKW   13
010426                          SETUP.BUFFER::
                                        .BLKW   400
011426                          IOP.TABLE::
                                        .BLKW   10
011446                          ETH.STATION.ADR::
                                        .BLKW   6
011462                          STATION.ADR::
                                        .BLKW   4
011472                          PTRN.TABLE::
011472  000                             .BYTE   0
011473  377                             .BYTE   377
011474  252                             .BYTE   252
011475  125                             .BYTE   125
011476  314                             .BYTE   314
011477  063                             .BYTE   63
011500  360                             .BYTE   360
011501  017                             .BYTE   17
011502                          TARGET.ADR::
011502  000                             .BYTE   0
011503  000                             .BYTE   0
011504  000                             .BYTE   0
011505  000                             .BYTE   0
011506  000                             .BYTE   0
011507  000                             .BYTE   0
011510  125                             .BYTE   125
011511  125                             .BYTE   125
011512  125                             .BYTE   125
011513  125                             .BYTE   125
011514  125                             .BYTE   125
011515  125                             .BYTE   125
011516  252                             .BYTE   252
011517  252                             .BYTE   252
011520  252                             .BYTE   252
011521  252                             .BYTE   252
011522  252                             .BYTE   252
011523  252                             .BYTE   252
011524  125                             .BYTE   125
011525  125                             .BYTE   125
011526  125                             .BYTE   125
011527  125                             .BYTE   125
011530  125                             .BYTE   125
011531  125                             .BYTE   125
011532  377                             .BYTE   377
011533  377                             .BYTE   377
011534  377                             .BYTE   377
011535  377                             .BYTE   377
011536  377                             .BYTE   377
011537  377                             .BYTE   377
011540  000                             .BYTE   0
011541  364                             .BYTE   364
```

```
011542  372                             .BYTE   372
011543  104                             .BYTE   104
011544  104                             .BYTE   104
011545  125                             .BYTE   125
011546  252                             .BYTE   252
011547  000                             .BYTE   0
011550  000                             .BYTE   0
011551  000                             .BYTE   0
011552  000                             .BYTE   0
011553  000                             .BYTE   0
011554  252                             .BYTE   252
011555  000                             .BYTE   0
011556  002                             .BYTE   2
011557  252                             .BYTE   252
011560  252                             .BYTE   252
011561  252                             .BYTE   252
011562  252                             .BYTE   252
011563  000                             .BYTE   0
011564  005                             .BYTE   5
011565  125                             .BYTE   125
011566  125                             .BYTE   125
011567  125                             .BYTE   125
011570  252                             .BYTE   252
011571  000                             .BYTE   0
011572  004                             .BYTE   4
011573  377                             .BYTE   377
011574  377                             .BYTE   377
011575  377                             .BYTE   377
011576  252                             .BYTE   252
011577  000                             .BYTE   0
011600  004                             .BYTE   4
011601  000                             .BYTE   0
011602  000                             .BYTE   0
011603  000                             .BYTE   0
011604  252                             .BYTE   252
011605  000                             .BYTE   0
011606  004                             .BYTE   4
011607  030                             .BYTE   30
011610  201                             .BYTE   201
011611  030                             .BYTE   30
011612  001                             .BYTE   1
011613  000                             .BYTE   0
011614  000                             .BYTE   0
011615  000                             .BYTE   0
011616  000                             .BYTE   0
011617  000                             .BYTE   0
011620  253                             .BYTE   253
011621  252                             .BYTE   252
011622  252                             .BYTE   252
011623  252                             .BYTE   252
011624  252                             .BYTE   252
011625  252                             .BYTE   252
011626  377                             .BYTE   377
```

```
011627      000                                 .BYTE   0
011630      001                                 .BYTE   1
011631      002                                 .BYTE   2
011632      003                                 .BYTE   3
011633      004                                 .BYTE   4
011634      125                                 .BYTE   125
011635      005                                 .BYTE   5
011636      006                                 .BYTE   6
011637      007                                 .BYTE   7
011640      010                                 .BYTE   10
011641      011                                 .BYTE   11
011642      315                                 .BYTE   315
011643      066                                 .BYTE   66
011644      046                                 .BYTE   46
011645      047                                 .BYTE   47
011646      047                                 .BYTE   47
011647      111                                 .BYTE   111
011650      063                                 .BYTE   63
011651      241                                 .BYTE   241
011652      147                                 .BYTE   147
011653      273                                 .BYTE   273
011654      114                                 .BYTE   114
011655      237                                 .BYTE   237
011656      353                                 .BYTE   353
011657      276                                 .BYTE   276
011660      307                                 .BYTE   307
011661      217                                 .BYTE   217
011662      063                                 .BYTE   63
011663      377                                 .BYTE   377
011664      377                                 .BYTE   377
011665      377                                 .BYTE   377
011666      377                                 .BYTE   377
011667      377                                 .BYTE   377
011670      377                                 .BYTE   377
011671      377                                 .BYTE   377
011672                              BD.PROM.DESCR::
011672   100000                                 .WORD   -100000
011674   100000                                 .WORD   -100000
011676   000400'                                .WORD   RCV.BUFFER
011700   176000                                 .WORD   -2000
011702   000000                                 .WORD   0
011704   000000                                 .WORD   0
011706   100000                                 .WORD   -100000
011710   100000                                 .WORD   -100000
011712   004400'                                .WORD   XMIT.BUFFER
011714   176000                                 .WORD   -2000
011716   000000                                 .WORD   0
011720   000000                                 .WORD   0
011722   100000                                 .WORD   -100000
011724   020000                                 .WORD   20000
011726   000000                                 .WORD   0
011730   000000                                 .WORD   0
011732                              TD16::
```

```
011732  100000                          .WORD   -100000
011734  100200                          .WORD   -77600
011736  004400'                         .WORD   XMIT.BUFFER
011740  177777                          .WORD   -1
011742  000000                          .WORD   0
011744  000000                          .WORD   0
011746  100000                          .WORD   -100000
011750  100300                          .WORD   -77500
011752  004400'                         .WORD   XMIT.BUFFER
011754  177776                          .WORD   -2
011756  000000                          .WORD   0
011760  000000                          .WORD   0
011762  100000                          .WORD   -100000
011764  100100                          .WORD   -77700
011766  004402'                         .WORD   XMIT.BUFFER+2
011770  177777                          .WORD   -1
011772  000000                          .WORD   0
011774  000000                          .WORD   0
011776  100000                          .WORD   -100000
012000  120000                          .WORD   -60000
012002  004404'                         .WORD   XMIT.BUFFER+4
012004  177777                          .WORD   -1
012006  000000                          .WORD   0
012010  000000                          .WORD   0
012012  100000                          .WORD   -100000
012014  020000                          .WORD   20000
012016  000274'                         .WORD   XMIT.D.LIST+74
012020  177777                          .WORD   -1
012022  000000                          .WORD   0
012024  000000                          .WORD   0
012026  100000                          .WORD   -100000
012030  100000                          .WORD   -100000
012032  000270'                         .WORD   XMIT.D.LIST+70
012034  177776                          .WORD   -2
012036  000000                          .WORD   0
012040  000000                          .WORD   0
012042  100000                          .WORD   -100000
012044  120000                          .WORD   -60000
012046  011664'                         .WORD   TARGET.ADR+162
012050  177775                          .WORD   -3
012052  000000                          .WORD   0
012054  000000                          .WORD   0
012056  100000                          .WORD   -100000
012060  020000                          .WORD   20000
012062                          TD13::
012062  100000                          .WORD   -100000
012064  100000                          .WORD   -100000
012066  004400'                         .WORD   XMIT.BUFFER
012070  177777                          .WORD   -1
012072  000000                          .WORD   0
012074  000000                          .WORD   0
012076  100000                          .WORD   -100000
012100  100000                          .WORD   -100000
```

```
012102  004402'                         .WORD   XMIT.BUFFER+2
012104  177601                          .WORD   -177
012106  000000                          .WORD   0
012110  000000                          .WORD   0
012112  100000                          .WORD   -100000
012114  100000                          .WORD   -100000
012116  005000'                         .WORD   XMIT.BUFFER+400
012120  177777                          .WORD   -1
012122  000000                          .WORD   0
012124  000000                          .WORD   0
012126  100000                          .WORD   -100000
012130  040000                          .WORD   40000
012132  000260'                         .WORD   XMIT.D.LIST+60
012134  177777                          .WORD   -1
012136  000000                          .WORD   0
012140  000000                          .WORD   0
012142  100000                          .WORD   -100000
012144  120000                          .WORD   -60000
012146  005002'                         .WORD   XMIT.BUFFER+402
012150  177701                          .WORD   -77
012152  000000                          .WORD   0
012154  000000                          .WORD   0
012156  100000                          .WORD   -100000
012160  020000                          .WORD   20000
012162                                  .BLKB   4
012166                          RD13::
012166  100000                          .WORD   -100000
012170  100000                          .WORD   -100000
012172  000400'                         .WORD   RCV.BUFFER
012174  177777                          .WORD   -1
012176  000000                          .WORD   0
012200  000000                          .WORD   0
012202  100000                          .WORD   -100000
012204  100000                          .WORD   -100000
012206  000402'                         .WORD   RCV.BUFFER+2
012210  177702                          .WORD   -76
012212  000000                          .WORD   0
012214  000000                          .WORD   0
012216  100000                          .WORD   -100000
012220  100000                          .WORD   -100000
012222  000576'                         .WORD   RCV.BUFFER+176
012224  177777                          .WORD   -1
012226  000000                          .WORD   0
012230  000000                          .WORD   0
012232  100000                          .WORD   -100000
012234  100000                          .WORD   -100000
012236  000600'                         .WORD   RCV.BUFFER+200
012240  177776                          .WORD   -2
012242  000000                          .WORD   0
012244  000000                          .WORD   0
012246  100000                          .WORD   -100000
012250  100000                          .WORD   -100000
012252  000604'                         .WORD   RCV.BUFFER+204
```

```
012254  177704                          .WORD   -74
012256  000000                          .WORD   0
012260  000000                          .WORD   0
012262  100000                          .WORD   -100000
012264  100000                          .WORD   -100000
012266  000774'                         .WORD   RCV.BUFFER+374
012270  177776                          .WORD   -2
012272  000000                          .WORD   0
012274  000000                          .WORD   0
012276  100000                          .WORD   -100000
012300  140000                          .WORD   -40000
012302  000124'                         .WORD   RCV.D.LIST+124
012304  177777                          .WORD   -1
012306  000000                          .WORD   0
012310  000000                          .WORD   0
012312  100000                          .WORD   -100000
012314  100000                          .WORD   -100000
012316  001000'                         .WORD   RCV.BUFFER+400
012320  177775                          .WORD   -3
012322  000000                          .WORD   0
012324  000000                          .WORD   0
012326  100000                          .WORD   -100000
012330  100000                          .WORD   -100000
012332  001006'                         .WORD   RCV.BUFFER+406
012334  177704                          .WORD   -74
012336  000000                          .WORD   0
012340  000000                          .WORD   0
012342  100000                          .WORD   -100000
012344  100000                          .WORD   -100000
012346  001176'                         .WORD   RCV.BUFFER+576
012350  177777                          .WORD   -1
012352  000000                          .WORD   0
012354  000000                          .WORD   0
012356  100000                          .WORD   -100000
012360  020000                          .WORD   20000
012362                                  .BLKB   4
012366                          HWP.TABLE::
                                        .BLKW   1
012370                          SWP.TABLE::
                                        .BLKW   1
012372                          REG.ADR::
                                        .BLKW   1
012374                          IOP.DATA::
                                        .BLKW   1
012376                          GET.ADR::
                                        .BLKW   1
012400                          XBUF.LENGTH::
                                        .BLKW   1
012402                          RBUF.LENGTH::
                                        .BLKW   1
012404                          INTERRUPT.FLG::
                                        .BLKW   1
012406                          DEQNA.NO::
```

```
                                        .BLKW   1
012410                          COUNTER::
                                        .BLKW   1
012412                          UP.COUNTER::
                                        .BLKW   1
012414                          DOWN.COUNTER::
                                        .BLKW   1
012416                          CHECKSUM::
                                        .BLKW   1
012420                          BUF.LENGTH::
                                        .BLKW   1
012422                          CSR.WORD::
                                        .BLKW   1
012424  000000                  XC.FLAG::
                                        .WORD   0
012426  000000                  ERR.NUMBER::
                                        .WORD   0
012430  000000                  ERR.FLAG::
                                        .WORD   0
012432  000000                  ERR.COUNT::
                                        .WORD   0
012434                          TMP.IOP.ADR::
                                        .BLKW   1
012436                          TMP.REG.DATA::
                                        .BLKW   1
012440                          TEMP1::  .BLKW   1
012442                          TEMP2::  .BLKW   1
012444                          TEMP3::  .BLKW   1
012446                          TEMP4::  .BLKW   1
012450                          TEMP5::  .BLKW   1
012452                          TEMP6::  .BLKW   1
012454                          TEMP7::  .BLKW   1
012456                          TEMP8::  .BLKW   1
012460                          TEMP9::  .BLKW   1
012462                          P1::     .BLKW   1
012464                          P2::     .BLKW   1
012466                          P3::     .BLKW   1
012470                          P4::     .BLKW   1
012472                          P5::     .BLKW   1
012474                          TBYTE1::.BLKB   1
012475                          TBYTE2::.BLKB   1
012476                          TBYTE3::.BLKB   1
012477                          TBYTE4::.BLKB   1
012500                          TADR1::  .BLKW   1
012502                          TADR2::  .BLKW   1

                                .GLOBL  L$SOFT, T$PTHV, L$RPT, L$INIT
                                .GLOBL  L$CLEAN, L$LAST, L$HARD, L$DVTYP
                                .GLOBL  L$DESC, L$DU, L$AU, L$AUTO, T1
                                .GLOBL  T2, T3, T4, T5, T6, T7, T8, T9
                                .GLOBL  T10, T11, T12, T13, T14, T15, T16
                                .GLOBL  T17, T18, T19, T20, T21
```

100000	BIT15==	-100000
040000	BIT14==	40000
020000	BIT13==	20000
010000	BIT12==	10000
004000	BIT11==	4000
002000	BIT10==	2000
001000	BIT09==	1000
000400	BIT08==	400
000200	BIT07==	200
000100	BIT06==	100
000040	BIT05==	40
000020	BIT04==	20
000010	BIT03==	10
000004	BIT02==	4
000002	BIT01==	2
000001	BIT00==	1
001000	BIT9==	1000
000400	BIT8==	400
000200	BIT7==	200
000100	BIT6==	100
000040	BIT5==	40
000020	BIT4==	20
000010	BIT3==	10
000004	BIT2==	4
000002	BIT1==	2
000001	BIT0==	1
000040	EF.START==	40
000037	EF.RESTART==	37
000036	EF.CONTINUE==	36
000035	EF.NEW==	35
000034	EF.PWR==	34
000340	PRI07==	340
000300	PRI06==	300
000240	PRI05==	240
000200	PRI04==	200
000140	PRI03==	140
000100	PRI02==	100
000040	PRI01==	40
000000	PRI00==	0
000004	EVL==	4
000010	LOT==	10
000020	ADR==	20
000040	IDU==	40
000100	ISR==	100
000200	UAM==	200
000400	BOE==	400
001000	PNT==	1000
002000	PRI==	2000
004000	IXE==	4000
010000	IBE==	10000
020000	IER==	20000

```
040000                  LOE==           40000
100000                  HOE==           -100000
000176'                 L$ERRTBL==      ERRTYP
000220'                 L$SW==          L$SWLEN+2
000210'                 L$HW==          L$HWLEN+2
000011'                 L$DEPO==        L$REV+1
000000'                 DESCR.LIST==    RCV.D.LIST
000400'                 DATA.BUFFER==   RCV.BUFFER
000000'                 QST01==         P.AAA
000030'                 QST02==         P.AAB
000060'                 QST03==         P.AAC
000122'                 QST04==         P.AAD
000164'                 QST05==         P.AAE
000226'                 QST06==         P.AAF
000270'                 QST07==         P.AAG
000332'                 MSG00==         P.AAH
000370'                 MSG01==         P.AAI
000452'                 MSG02==         P.AAJ
000540'                 MSG03==         P.AAK
000644'                 MSG04==         P.AAL
000736'                 MSG05==         P.AAM
001030'                 MSG06==         P.AAN
001122'                 MSG07==         P.AAO
001214'                 MSG08==         P.AAP
001306'                 MSG09==         P.AAQ
001400'                 MSG10==         P.AAR
001462'                 MSG11==         P.AAS
001546'                 MSG12==         P.AAT
001612'                 MSG13==         P.AAU
001676'                 MSG14==         P.AAV
001766'                 MSG15==         P.AAW
002050'                 MSG16==         P.AAX
002136'                 MSG17==         P.AAY
002224'                 MSG18==         P.AAZ
002250'                 MSG19==         P.ABA
002336'                 MSG20==         P.ABB
002426'                 MSG21==         P.ABC
002506'                 MSG22==         P.ABD
002572'                 MSG23==         P.ABE
002650'                 MSG24==         P.ABF
002724'                 MSG25==         P.ABG
002766'                 MSG26==         P.ABH
003030'                 MSG27==         P.ABI
003072'                 MSG28==         P.ABJ
003136'                 MSG29==         P.ABK
003164'                 MSG30==         P.ABL
003252'                 MSG31==         P.ABM
003336'                 MSG32==         P.ABN
003400'                 MSG33==         P.ABO
003454'                 MSG34==         P.ABP
003530'                 MSG35==         P.ABQ
003626'                 MSG36==         P.ABR
003732'                 MSG37==         P.ABS
```

```
004024'                 MSG38**         P.ABT
004104'                 MSG39**         P.ABU
004170'                 MSG40**         P.ABV
004260'                 MSG41**         P.ABW
004322'                 MSG42**         P.ABX
004402'                 MSG43**         P.ABY
004466'                 MSG44**         P.ABZ
004570'                 MSG45**         P.ACA
004646'                 MSG46**         P.ACB
004716'                 MSG47**         P.ACC
004772'                 MSG48**         P.ACD
005030'                 MSG49**         P.ACE
005066'                 MSG50**         P.ACF
005150'                 MSG51**         P.ACG
005202'                 MSG52**         P.ACH
005246'                 MSG53**         P.ACI
005276'                 MSG54**         P.ACJ
005346'                 MSG55**         P.ACK
005410'                 MSG56**         P.ACL
005446'                 MSG57**         P.ACM
005536'                 MSG58**         P.ACN
005602'                 MSG59**         P.ACO
005714'                 MSG60**         P.ACP
005756'                 MSG61**         P.ACQ
006020'                 MSG62**         P.ACR
006110'                 MSG63**         P.ACS
006204'                 MSG64**         P.ACT
006240'                 MSG65**         P.ACU
006304'                 MSG66**         P.ACV
006372'                 MSG67**         P.ACW
006474'                 MSG68**         P.ACX
006572'                 MSG69**         P.ACY
006672'                 MSG70**         P.ACZ
006756'                 MSG71**         P.ADA
000210'                 HP.TABLE**      L$HWLEN*2
000220'                 SP.TABLE**      L$SWLEN*2


;                       PSECT SUMMARY
;
;       Psect Name      Words     Attributes
;        $CODE$           81       RO ,  I  , LCL, REL, CON
;        $GLOB$         2722       RW ,  D  , LCL, REL, CON
;        $PLIT$         1807       RO ,  D  , LCL, REL, CON


;                   Library Statistics
;
;                                 ---- --  Symbols  ----- --      Pages       Processing
;       File                      Total    Loaded   Percent      Mapped      Time
```

```
;
;  DISK$USER2:[MARSHALL.DEQNA]QNALIB.L16;15
;                                        223        88        39        14          00:00.1
```

```
;                                        COMMAND QUALIFIERS

;       BLISS/PDP11 ZQNA1.BLI/LIST=ZQNA1.LIS/OBJECT=ZQNA1.OBJ/SOURCE=PAGE:53

; Size:          0 code + 4610 data words
; Run Time:        00:23.8
; Elapsed Time:    01:12.3
; Lines/CPU Min:    6123
; Lexemes/CPU-Min: 38814
; Memory Used:  230 pages
; Compilation Complete
```

```
;    0001  0     MODULE ZQNA2 (%TITLE 'CZQNADO DEQNA FUNCTIONAL TEST'
;    0002  0                   IDENT = 'V01.0',
;    0003  0                   ADDRESSING_MODE(ABSOLUTE)
;    0004  0                   ) =
;    0005  0     %SBTTL 'PROGRAM INIT MODULE'
;    0006  0
;    0007  1     BEGIN
;    0008  1
;    0009  1     LIBRARY 'QNALIB';                          ! QNALIB LIBRARY
;    0010  1     REQUIRE 'BLSMAC.REQ';                      ! DIAGNOSTIC SUPERVISOR LIBRARY
;    1500  1
```

```
;    1501  1     %SBTTL 'EXTERNAL DECLARATIONS'
;    1502  1     !<BLF/FORMAT>
;    1503  1
;    1504  1     PSECT
;    1505  1         CODE = AA$CODE$;
;    1506  1
;    1507  1
;    1508  1     FORWARD ROUTINE
;    1509  1         NXM_INT             : L$ISR NOVALUE;
;    1510  1
;    1511  1     EXTERNAL ROUTINE
;    1512  1         RESET_DEQNA         : NOVALUE;
;    1513  1
```

```
;    1514  1      EXTERNAL
;    1515  1
;    1516  1      !++
;    1517  1      !       COMMUNICATION AREA DECLARATIONS
;    1518  1      !--
;    1519  1
;    1520  1          IOP_TABLE       : VECTOR [ 8, WORD ],
;    1521  1
;    1522  1
;    1523  1      !++
;    1524  1      !       HARDWARE AND SOFTWARE P-TABLE STORAGE DECLARATIONS
;    1525  1      !--
;    1526  1
;    1527  1          HWP_TABLE    : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;    1528  1          SWP_TABLE    : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    1529  1
;    1530  1          INTERRUPT_FLG       : WORD,                  ! 1 = INTERRUPT OCCURED
;    1531  1
;    1532  1          REG_ADR             : REF REG_STR FIELD ( IOP_FIELDS ),
;    1533  1          IOP_DATA            : REF REG_STR FIELD ( IOP_FIELDS ),
;    1534  1          GET_ADR             : REF ADR_STR FIELD ( IOP_FIELDS ),
;    1535  1
;    1536  1      !++
;    1537  1      !   TEMPORARY STORAGE DATA DECLARATIONS
;    1538  1      !--
;    1539  1
;    1540  1          TMP_IOP_ADR         : WORD,                  ! I/O PAGE REGISTER ADDRESS
;    1541  1          TMP_REG_DATA        : WORD,                  ! I/O PAGE REG CONTENTS
;    1542  1          TEMP1               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1543  1          TEMP2               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1544  1          TEMP3               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1545  1          TEMP4               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1546  1          TEMP5               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1547  1          TEMP6               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1548  1          TEMP7               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1549  1          TEMP8               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1550  1          TEMP9               : WORD,                  ! TEMPORARY STORAGE LOCATION
;    1551  1
;    1552  1
;    1553  1      !++
;    1554  1      !       QUESTIONS AND ERROR MESSAGEES DECLARED EXTERNALLY
;    1555  1      !--
;    1556  1
;    1557  1          QST01, QST02, QST03, QST04, QST05, QST06, QST07, MSG54;
;    1558  1
```

```
;   1559  1     %SBTTL 'TYPE AND DESCRIPTION'
;   1560  1
;   1561  1     !++
;   1562  1     !       NAMES OF DEVICES SUPPORTED BY PROGRAM
;   1563  1     !--
;   1564  1
;   1565  1     EQUALS;
;   1566  1     DEVTYP (%ASCIZ'DEQNA/M7504');
;   1567  1
;   1568  1     !++
;   1569  1     !       TEST DESCRIPTION
;   1570  1     !--
;   1571  1
;   1572  1     DESCRIPT (%ASCIZ'DEQNA FUNCTIONAL TEST');
;   1573  1
```

```
;    1574  1      %SBTTL 'HARDWARE PARAMETER CODING SECTION'
;    1575  1
;    1576  1      !++
;    1577  1      !       THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;    1578  1      !       THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    1579  1      !       MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    1580  1      !       INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    1581  1      !       MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    1582  1      !       WITH THE OPERATOR.
;    1583  1      !
;    1584  1      !       THIS CODE IS USED BY THE SUPERVISOR TO INTERROGATE THE OPERATOR
;    1585  1      !       FOR DEVICE INFORMATION TO PUT IN THE P-TABLE.  THIS CODE IS USED
;    1586  1      !       IN CONJUNCTION WITH THE DEFAULT P-TABLE TEMPLATE.  THE MACROS
;    1587  1      !       USED IN THIS SECTION ARE "GPRMD", "GPRMA".
;    1588  1      !--
;    1589  1
;    1590  1      BGNHRD;
;    1591  1      GPRMA (QST01, %O'0', 0, %O'174440', %O'174460', YES, 1);  ! I/O PAGE  ADDRESS ?
;    1592  1      GPRMA (QST02, %O'2', 0, %O'700',    %O'704',    YES, 1);  ! INTERRUPT VECTOR ADDR ?
;    1593  1      ENDHRD;
;    1594  1
;    1595  1
```

```
;    1596  1     %SBTTL 'SOFTWARE PARAMETER CODING SECTION'
;    1597  1
;    1598  1     !++
;    1599  1     !       THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;    1600  1     !       THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    1601  1     !       MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    1602  1     !       INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    1603  1     !       MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    1604  1     !       WITH THE OPERATOR.
;    1605  1     !--
;    1606  1
;    1607  1     BGNSFT;
;    1608  1
;    1609  1     GPRML ( QST03, %O'0', -1, YES, 1);      ! DO YOU WANT TO TEST SANITY TIMER ?
;    1610  1     XFERF(NOTIMER);
;    1611  1     GPRMD ( QST05, %O'4', D, -1, C, 7, YES, 1);
;    1612  1                                             ! SANITY TIMER TIME-OUT VALUE ?
;    1613  1     $L(NOTIMER);
;    1614  1
;    1615  1     GPRML ( QST06, %O'6',  -1, YES, 1);     ! EXTERNAL LOOPBACK MODE ?
;    1616  1     GPRML ( QST07, %O'10', -1, YES, 1);     ! SYSTEM HAS BLOCK-MODE MEMORY ?
;    1617  1     GPRML ( QST04, %O'2', -1, YES, 1);      ! LOOPBACK CONNECTOR IN DEQNA ?
;    1618  1
;    1619  1     ENDSFT;
;    1620  1
;    1621  1
```

```
;    1622  1     %SBTTL 'REPORT CODING SECTION'
;    1623  1
;    1624  1     !++
;    1625  1     !
;    1626  1     !       THE REPORT CODING SECTION CONTAINS THE
;    1627  1     !       "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;    1628  1     !
;    1629  1     !       THIS SECTION CONTAINS THE CODE FOR PRINTING
;    1630  1     !       STATISTICAL INFORMATION GATHERED BY THE DIAGNOSTIC.  IT IS
;    1631  1     !       EXECUTED BY THE OPERATOR COMMAND "PRINT" OR BY THE MACRO CALL
;    1632  1     !       "DORPT".  USE THE PRINTS MACRO TO PRINT THE INFORMATION.
;    1633  1     !       USE FORMAT STATEMENTS AS IN THE PRINTB/PRINTX MACROS.  IT IS
;    1634  1     !       THE PROGRAMMER'S RESPONSIBILTY TO DEVISE AND IMPLEMENT THE
;    1635  1     !       FORM AND CONTENT OF THE STATISTICS.
;    1636  1     !--
;    1637  1
;    1638  1
;    1639  2     BGNRPT;
;    1640  2
;    1641  2       TEMP1 = 1;
;    1642  2
;    1643  1     ENDRPT;

                                            .TITLE  ZQNA2 CZQNADO DEQNA FUNCTIONAL TEST
                                            .IDENT  /V01.0/
                                            .ENABL  AMA


000000                                      .PSECT  $CODE$,  RO
000000      104     105     121     L$DVTYP::
                                            .ASCII  /DEQ/
000003      116     101     057             .ASCII  /NA/<57>
000006      115     067     065             .ASCII  /M75/
000011      060     064     000             .ASCII  /04/<00>
000014                                      .BLKB   2
000016      104     105     121     L$DESC::.ASCII  /DEQ/
000021      116     101     040             .ASCII  /NA /
000024      106     125     116             .ASCII  /FUN/
000027      103     124     111             .ASCII  /CTI/
000032      117     116     101             .ASCII  /ONA/
000035      114     040     124             .ASCII  /L T/
000040      105     123     124             .ASCII  /EST/
000043      000                             .ASCII  <00>
000044                                      .BLKB   2
000046  000000C                   L$HRDLN::
                                            .WORD   <<<L$NDHRD-L$HRDLN>/2>-1>
000050  000031                    GP$1::  .WORD   31
000052  000000G                            .WORD   QST01
000054  174440                             .WORD   -3340
000056  174460                             .WORD   -3320
000060  001031                    GP$2::  .WORD   1031
000062  000000G                            .WORD   QST02
```

```
000064  000700                          .WORD   700
000066  000704                          .WORD   704
000070                          L$NDHRD::
                                        .BLKW   1
000072  000000C                 L$SFTLN::
                                        .WORD   <<<L$NDSFT L$SFTLN>/2>-1>
000074  000130                  GP$3::  .WORD   130
000076  000000G                         .WORD   QST03
000100  177777                          .WORD   -1
000102  000000C                 $NOTIMER:
                                        .WORD   <<<<$LNOTIMER $NOTIMER>*400>+4>+40>
000104  002052                  GP$4::  .WORD   2052
000106  000000G                         .WORD   QST05
000110  177777                          .WORD   -1
000112  000000                          .WORD   0
000114  000007                          .WORD   7
000116  001004                  $LNOTIMER:
                                        .WORD   1004
000120  003130                  GP$5::  .WORD   3130
000122  000000G                         .WORD   QST06
000124  177777                          .WORD   -1
000126  004130                  GP$6::  .WORD   4130
000130  000000G                         .WORD   QST07
000132  177777                          .WORD   -1
000134  001130                  GP$7::  .WORD   1130
000136  000000G                         .WORD   QST04
000140  177777                          .WORD   -1
000142                          L$NDSFT::
                                        .BLKW   1


                                        .GLOBL  RESET.DEQNA, IOP.TABLE, HWP.TABLE
                                        .GLOBL  SWP.TABLE, INTERRUPT.FLG, REG.ADR
                                        .GLOBL  IOP.DATA, GET.ADR, TMP.IOP.ADR
                                        .GLOBL  TMP.REG.DATA, TEMP1, TEMP2, TEMP3
                                        .GLOBL  TEMP4, TEMP5, TEMP6, TEMP7, TEMP8
                                        .GLOBL  TEMP9, QST01, QST02, QST03, QST04
                                        .GLOBL  QST05, QST06, QST07, MSG54


        100000                  BIT15== -100000
        040000                  BIT14== 40000
        020000                  BIT13== 20000
        010000                  BIT12== 10000
        004000                  BIT11== 4000
        002000                  BIT10== 2000
        001000                  BIT09== 1000
        000400                  BIT08== 400
        000200                  BIT07== 200
        000100                  BIT06== 100
        000040                  BIT05== 40
        000020                  BIT04== 20
        000010                  BIT03== 10
```

```
000004                          BIT02==         4
000002                          BIT01==         2
000001                          BIT00==         1
001000                          BIT9==          1000
000400                          BIT8==          400
000200                          BIT7==          200
000100                          BIT6==          100
000040                          BIT5==          40
000020                          BIT4==          20
000010                          BIT3==          10
000004                          BIT2==          4
000002                          BIT1==          2
000001                          BIT0==          1
000040                          EF.START==      40
000037                          EF.RESTART==    37
000036                          EF.CONTINUE==   36
000035                          EF.NEW==        35
000034                          EF.PWR==        34
000340                          PRI07==         340
000300                          PRI06==         300
000240                          PRI05==         240
000200                          PRI04==         200
000140                          PRI03==         140
000100                          PRI02==         100
000040                          PRI01==         40
000000                          PRI00==         0
000004                          EVL==           4
000010                          LOT==           10
000020                          ADR==           20
000040                          IDU==           40
000100                          ISR==           100
000200                          UAM==           200
000400                          BOE==           400
001000                          PNT==           1000
002000                          PRI==           2000
004000                          IXE==           4000
010000                          IBE==           10000
020000                          IER==           20000
040000                          LOE==           40000
100000                          HOE==           -100000
000050'                         L$HARD==        L$HRDLN+2
000074'                         L$SOFT==        L$SFTLN+2


                                        .SBTTL  LRPT REPORT CODING SECTION
000000                                  .PSECT  AA$CODE$,  RO

000000  012737  000001  000000G  LRPT:  MOV     #1,TEMP1                ;                1641
000006  000207                         RTS     PC                      ;                1619

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0000
; Maximum stack depth per invocation:  0 words
```

```
                                        .SBTTL  L$RPT REPORT CODING SECTION
000000  004737  000000'        L$RPT:: JSR     PC,LRPT                 ;                       1641
000004  104425                          TRAP    25
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0010
; Maximum stack depth per invocation:  2 words


;    1644  1
;    1645  1
;    1646  1
;    1647  1
```

```
;  1648  1     %SBTTL 'INITIALIZE SECTION'
;  1649  1
;  1650  1     !++
;  1651  1     !       THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
;  1652  1     !       AT THE BEGINNING OF EACH PASS.
;  1653  1     !
;  1654  1     !       THE INITIALIZE CODE IS EXECUTED UNDER FIVE CONDITIONS.  THERE
;  1655  1     !       ARE SUPERVISOR EVENT FLAGS THAT ARE USED TO LET THE
;  1656  1     !       DIAGNOSTIC KNOW UNDER WHICH CONDITION THE EXECUTION IS TAKING
;  1657  1     !       PLACE.  THE EVENT FLAGS ARE READ USING THE "READEF" MACRO.
;  1658  1     !       THE CONDITIONS UNDER WHICH THE INIT CODE IS EXECUTED AND THE
;  1659  1     !       CORRRESPONDING EVENT FLAGS ARE:
;  1660  1     !               START COMMAND           EF.START
;  1661  1     !               RESTART COMMAND         EF.RESTART
;  1662  1     !               CONTINUE COMMAND        EF.CONTINUE
;  1663  1     !               POWERDOWN/POWERUP       EF.PWR
;  1664  1     !               NEW PASS                EF.NEW
;  1665  1     !       EXAMPLE OF EVENT FLAG USE:
;  1666  1     !               IF READEF(EF.START) THEN
;  1667  1     !                   START_FLAG = 1;
;  1668  1     !       DURING THE INIT CODE, USE THE "GPHARD" MACRO TO OBTAIN P-TABLE
;  1669  1     !       INFORMATION FOR DEVICE TESTING.  GET ONE UNIT'S INFORMATION IF
;  1670  1     !       THIS IS A SEQUENTIAL DIAGNOSTIC. NUMBER OF UNITS AVAILABLE IS IN
;  1671  1     !       A HEADER LOCATION: "L$UNIT".
;  1672  1     !--
;  1673  1
;  1674  2     BGNINIT;
;  1675  2
;  1676  2     LOCAL
;  1677  2       START_FLAG,                           ! SET IF THIS PASS IS A START
;  1678  2       DELAY_MULT;                           ! CONTAINS DELAY FACTOR
;  1679  2
;  1680  2     SETPRI (PRI07);                         ! PRIORITY 7 - NO INTERRUPTS ALLOWED
;  1681  2     START_FLAG = CLEAR_FLG;                 ! CLEAR FLAG BEFORE TESTING IT
;  1682  2
;  1683  2     IF READEF (EF_PWR)                      ! ARE WE HERE BECAUSE OF POWER FAIL?
;  1684  2       THEN
;  1685  3         BEGIN
;  1686  3           PRINTF ( MSG54 );                 ! "THERE WAS POWER FAILURE - WAITING"
;  1687  3
;  1688  3           INCR COUNT FROM 0 TO 60 DO        ! WAIT APPROX. 60 SECONDS
;  1689  4             BEGIN
;  1690  4               DELAY_MULT = 10000;
;  1691  4               DELAY (.DELAY_MULT);
;  1692  4               BREAK;                        ! BREAK FOR APT
;  1693  3             END;
;  1694  2         END;
;  1695  2
;  1696  2     IF READEF (EF_START)                    ! IS THIS A START ?
;  1697  2       THEN
;  1698  3         BEGIN
;  1699  3           START_FLAG = TRUE;
;  1700  2         END;
```

```
;    1701  2
;    1702  2     !++
;    1703  2     !       CLEAR HARDWARE P-TABLE ON A START BEFORE DOING THE GPHARDS
;    1704  2     !--
;    1705  2
;    1706  2     IF .START_FLAG OR READEF (EF_NEW) OR READEF (EF_CONTINUE)
;    1707  2       THEN                                  ! IF THIS IS A START
;    1708  3         BEGIN
;    1709  3           LOCAL TABLE_POINTER;
;    1710  3
;    1711  3           INCR INDEX FROM 0 TO HWP_SIZE BY 2 DO     ! ZERO OUT THE TABLES
;    1712  3             (HWP_TABLE + .INDEX) = 0;
;    1713  3
;    1714  3           !++
;    1715  3           !   GET BASE ADDRESS OF HARDWARE P-TABLE AND DEQNA I/O PAGE
;    1716  3           !--
;    1717  3
;    1718  3           IF GPHARD ( 0, TABLE_POINTER ) NEQU 0             ! GET P-TABLE ADDRESS
;    1719  3             THEN
;    1720  4               BEGIN
;    1721  4                 IOP_DATA  = .HWP_TABLE [ ADDR ];
;    1722  4                 HWP_TABLE = .TABLE_POINTER;                 ! SAVE HW P-TABLE ADDRESS
;    1723  4                 REG_ADR   = .HWP_TABLE [ ADDR ];            ! SAVE I/O PAGE BASE ADDRESS
;    1724  4                 GET_ADR   = .HWP_TABLE [ ADDR ];            ! SAVE I/O PAGE BASE ADDRESS
;    1725  4                 TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;    1726  4                 INCR INDEX FROM 0 TO 7 DO
;    1727  5                   BEGIN
;    1728  5                     IOP_TABLE [ .INDEX ] = .TMP_IOP_ADR;
;    1729  5                     TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;    1730  4                   END;
;    1731  3               END;
;    1732  2         END;
;    1733  2     RETURN;
;    1734  1     ENDINIT;

                                        .GLOBL  L$DLY


                                        .SBTTL  LINIT INITIALIZE SECTION
000000  004137  000000G          LINIT: JSR     R1,$SAVE4                ;                    1643
000004  005746                         TST     -(SP)
000006  012700  000340                 MOV     #340,R0                  ;                    1680
000012  104441                         TRAP    41
000014  005004                         CLR     R4                       ; START.FLAG         1681
000016  012700  000034                 MOV     #34,R0                   ;                    1683
000022  104447                         TRAP    47
000024  103027                         BHIS    6$
000026  012746  000000G                MOV     #MSG54,-(SP)             ;                    1686
000032  012746  000001                 MOV     #1,-(SP)
000036  010600                         MOV     SP,R0                    ; SP,*
000040  104417                         TRAP    17
000042  012702  000075                 MOV     #75,R2                   ; *,COUNT            1688
```

```
                                                                                                                    SFQ 0076
ZQNA2           CZQNADO DEQNA FUNCTIONAL TEST                   14-Mar-1985 13:10:29    VAX-11 Bliss-16 V4.1 582        Page  13
V01.0           INITIALIZE SECTION                              14-Mar-1985 13:04:24    DISK$USER2:[MARSHALL.DEQNA]ZQNA2.BLI;4  (8)

000046  012703  023420                  1$:     MOV     #23420,R3               ; *,DELAY.MULT                          1690
000052  010301                                  MOV     R3,R1                   ; DELAY.MULT,$$TMP2                     1691
000054  001410                          2$:     BEQ     5$
000056  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000062  001403                                  BEQ     4$
000064  005066  000004                  3$:     CLR     4(SP)                   ; $$TMP
000070  077003                                  SOB     R0,3$                   ; $$TMP1,*
000072  005301                          4$:     DEC     R1                      ; $$TMP2
000074  000767                                  BR      2$
000076  104422                          5$:     TRAP    22
000100  077216                                  SOB     R2,1$                   ; COUNT,*                               1688
000102  022626                                  CMP     (SP)+,(SP)+             ;                                       1685
000104  012700  000040                  6$:     MOV     #40,R0                  ;                                       1696
000110  104447                                  TRAP    47
000112  103002                                  BHIS    7$
000114  012704  000001                          MOV     #1,R4                   ; *,START.FLAG                          1699
000120  006004                          7$:     ROR     R4                      ; START.FLAG                            1706
000122  103410                                  BLO     8$
000124  012700  000035                          MOV     #35,R0
000130  104447                                  TRAP    47
000132  103404                                  BCS     8$
000134  012700  000036                          MOV     #36,R0
000140  104447                                  TRAP    47
000142  103044                                  BHIS    11$
000144  005000                          8$:     CLR     R0                      ; INDEX                                 1711
000146  005060  000000G                 9$:     CLR     HWP.TABLE(R0)           ; *(INDEX)                              1712
000152  062700  000002                          ADD     #2,R0                   ; *,INDEX                               1711
000156  020027  000002                          CMP     R0,#2                   ; INDEX,*
000162  003771                                  BLE     9$
000164  005000                                  CLR     R0                      ;                                       1718
000166  104442                                  TRAP    42
000170  005700                                  TST     R0                      ; TABLE.POINTER
000172  001430                                  BEQ     11$
000174  017737  000000G 000000G                 MOV     @HWP.TABLE,IOP.DATA     ;                                       1721
000202  010037  000000G                         MOV     R0,HWP.TABLE            ; TABLE.POINTER,*                       1722
000206  011000                                  MOV     (R0),R0                 ; HWP.TABLE,*                           1723
000210  010037  000000G                         MOV     R0,REG.ADR
000214  010037  000000G                         MOV     R0,GET.ADR              ;                                       1724
000220  010037  000000G                         MOV     R0,TMP.IOP.ADR          ;                                       1725
000224  005000                                  CLR     R0                      ; INDEX                                 1726
000226  013760  000000G 000000G         10$:    MOV     TMP.IOP.ADR,IOP.TABLE(R0)       ; *,*(INDEX)                    1728
000234  062737  000002  000000G                 ADD     #2,TMP.IOP.ADR          ;                                       1729
000242  062700  000002                          ADD     #2,R0                   ; *,INDEX                               1726
000246  020027  000016                          CMP     R0,#16                  ; INDEX,*
000252  003765                                  BLE     10$
000254  005726                          11$:    TST     (SP)+                   ;                                       1643
000256  000207                                  RTS     PC

; Routine Size:  88 words,      Routine Base:  AA$CODE$ + 0020
; Maximum stack depth per invocation:  10 words
```

```
                                        .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  000020          L$INIT::JSR     PC,LINIT                        ;                       1733
000004  104411                          TRAP    11
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  AA$CODE$ + 0300
; Maximum stack depth per invocation:  2 words

;    1735  1
;    1736  1
;    1737  1
```

```
;    1738  1     %SBTTL 'AUTODROP SECTION
;    1739  1
;    1740  1     !++
;    1741  1     !
;    1742  1     !       THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;    1743  1     !       THE "ADR" FLAG WAS SET.  THE UNIT UNDER TEST IS CHECKED TO
;    1744  1     !       SEE IF IT WILL RESPOND.  IF IT DOESN'T IT IS IMMEDIATELY
;    1745  1     !       DROPPED FROM TESTING.
;    1746  1     !
;    1747  1     !--
;    1748  1
;    1749  2     BGNAUTO;
;    1750  2
;    1751  2       RETURN;
;    1752  2
;    1753  1     ENDAUTO;


                                                .SBTTL  LAUTO AUTODROP SECTION
000000  000207                            LAUTO:  RTS     PC                              ;                                1734

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0310
; Maximum stack depth per invocation:  0 words




                                                .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  000310'                   L$AUTO::JSR     PC,LAUTO                        ;                                1751
000004  104461                                   TRAP    61
000006  000207                                   RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0312
; Maximum stack depth per invocation:  2 words


;    1754  1
;    1755  1
```

```
;    1756  1     %SBTTL 'CLEANUP CODING SECTION'
;    1757  1
;    1758  1     !++
;    1759  1     !
;    1760  1     !       THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
;    1761  1     !       AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
;    1762  1     !
;    1763  1     !       INSERT YOUR CLEANUP CODING.  THIS CODING SHOULD
;    1764  1     !       RESTORE YOUR TEST-DEVICE TO A NEUTRAL STATE.
;    1765  1     !       THIS CODE WILL BE EXECUTED AFTER EACH PASS AND AFTER THE
;    1766  1     !       PROGRAM IS INTERRUPTED BY "^C".
;    1767  1     !--
;    1768  1
;    1769  2     BGNCLN;
;    1770  2
;    1771  2       RETURN;
;    1772  2
;    1773  1     ENDCLN;


                                        .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  000207                  LCLEAN: RTS     PC                              ;  1753

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0322
; Maximum stack depth per invocation:  0 words




                                        .SBTTL  L$CLEAN CLEANUP CODING SECTION
000000  004737  000322'         L$CLEAN::
                                        JSR     PC,LCLEAN                       ;  1771
000004  104412                          TRAP    12
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0324
; Maximum stack depth per invocation:  2 words


;    1774  1
;    1775  1
```

```
;    1776  1     %SBTTL 'DROP UNIT SECTION'
;    1777  1
;    1778  1     !++
;    1779  1     !
;    1780  1     !       THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;    1781  1     !       TO NO LONGER BE TESTED.
;    1782  1     !
;    1783  1     !       INSERT DROP CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
;    1784  1     !       A "DROP" COMMAND OR A "DODU" MACRO EXECUTION.  THE PURPOSE
;    1785  1     !       OR THIS CODE IS TO DO ANY NECESSARY HOUSEKEEPING AFTER A
;    1786  1     !       UNIT HAS BEEN DROPPED.
;    1787  1     !
;    1788  1     !--
;    1789  1
;    1790  2     BGNDU;
;    1791  2
;    1792  2       RETURN;
;    1793  2
;    1794  1     ENDDU;


                                            .SBTTL  LDU DROP UNIT SECTION
000000  000207                      LDU:    RTS     PC                      ;                       1773

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0334
; Maximum stack depth per invocation:  0 words




                                            .SBTTL  L$DU DROP UNIT SECTION
000000  004737  000334'             L$DU::  JSR     PC,LDU                  ;                       1792
000004  104453                              TRAP    53
000006  000207                              RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0336
; Maximum stack depth per invocation:  2 words


;    1795  1
;    1796  1
```

```
;    1797  1     %SBTTL 'ADD UNIT SECTION'
;    1798  1
;    1799  1     !++
;    1800  1     !
;    1801  1     !       THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;    1802  1     !       TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;    1803  1     !       TO THE TEST CYCLE.
;    1804  1     !
;    1805  1     !       INSERT ADD CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
;    1806  1     !       AN "ADD" COMMAND.  THE PURPOSE OF THIS CODE IS TO DO ANY
;    1807  1     !       HOUSEKEEPING THAT MAY BE NECESSARY AFTER A UNIT HAS BEEN ADDED.
;    1808  1     !
;    1809  1     !--
;    1810  1
;    1811  2     BGNAU;
;    1812  2
;    1813  2       RETURN;
;    1814  2
;    1815  1     ENDAU;


                                            .SBTTL  LAU ADD UNIT SECTION
000000  000207                      LAU:    RTS     PC                      ;                       1794

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0346
; Maximum stack depth per invocation:  0 words




                                            .SBTTL  L$AU ADD UNIT SECTION
000000  004737  000346'             L$AU::  JSR     PC,LAU                  ;                       1813
000004  104452                              TRAP    52
000006  000207                              RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0350
; Maximum stack depth per invocation:  2 words


;    1816  1
;    1817  1
```

```
ZQNA2           CZQNADO DEQNA FUNCTIONAL TEST                14-Mar-1985 13:10:29    VAX-11 Bliss-16 V4.1-582          SEQ 0082
V01.0           ADD UNIT SECTION                             14-Mar-1985 13:04:24    DISK$USER2:[MARSHALL.DEQNA]ZQNA2.BLI;4  Page  19
                                                                                                                             (13)

;    1818  1
;    1819  2     BGNSRV (NXM_INT);
;    1820  2
;    1821  2     !++
;    1822  2     !
;    1823  2     !       GLOBAL LOCATION "INTERRUPT_FLG" IS SET TO TRUE WHICH INDICATES
;    1824  2     !       THE INITIALIZATION SEQUENCE INTERRUPT OCCURED.
;    1825  2     !
;    1826  2     !--
;    1827  2
;    1828  2        INTERRUPT_FLG     = %O'177777';
;    1829  2
;    1830  1     ENDSRV;


                                                .SBTTL  NXM.INT ADD UNIT SECTION
000000  012737  177777  000000G          NXM.INT::
                                                MOV     #-1,INTERRUPT.FLG               ;                                1828
000006  000002                                 RTI                                     ;                                1819

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0360
; Maximum stack depth per invocation:  0 words


;    1831  1
;    1832  1     END
;    1833  0     ELUDOM


;                                       OTS external references
                                                .GLOBL  $SAVE4


;                                       PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;        $CODE$                           50       RO ,  I  ,  LCL,  REL,  CON
;        AA$CODE$                        124       RO ,  I  ,  LCL,  REL,  CON



;                                 Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[MARSHALL.DEQNA]QNALIB.L16;15
;                                           223        48        21        14          00:00.1
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZQNA2.BLI/LIST=ZQNA2.LIS/OBJECT=ZQNA2.OBJ/SOURCE=PAGE:53

; Size:          124 code + 50 data words
; Run Time:        00:12.9
; Elapsed Time:    00:43.3
; Lines/CPU Min:    8499
; Lexemes/CPU-Min: 58038
; Memory Used:  168 pages
; Compilation Complete
```

```
ZQNA3           CZQNADO DEQNA FUNCTIONAL TEST                 14-Mar-1985 13:11:16    VAX-11 Bliss-16 V4.1-582              SEQ 0084
                                                              14-Mar-1985 13:05:35    DISK$USER2:[MARSHALL.DEQNA]ZQNA3.BLI;4  Page   1
                                                                                                                              (1)

;    0001  0      MODULE ZQNA3 (%TITLE 'CZQNADO DEQNA FUNCTIONAL TEST'
;    0002  0                    IDENT = 'V01.0',
;    0003  0                    ADDRESSING_MODE(ABSOLUTE)
;    0004  0                    ) =
;    0005  0      %SBTTL 'DEQNA TEST DEFINITION MODULE'
;    0006  1      BEGIN
;    0007  1      !<BLF/FORMAT>
;    0008  1
;    0009  1      LIBRARY 'QNALIB';                             ! QNALIB LIBRARY
;    0010  1      REQUIRE 'BLSMAC.REQ';                         ! DIAGNOSTIC SUPERVISOR LIBRARY
;    1500  1
```

```
;    1501  1      PSECT
;    1502  1          CODE = AB$CODE$;
;    1503  1
;    1504  1      !++
;    1505  1      !       EXTERNAL DATA USED BY THIS MODULE
;    1506  1      !--
;    1507  1
;    1508  1      EXTERNAL ROUTINE
;    1509  1
;    1510  1         CHK_CSR_STATUS        : NOVALUE,
;    1511  1         CHK_RIXI_STATUS       : NOVALUE,
;    1512  1         CHK_RCV_STATUS        : NOVALUE,
;    1513  1         CHK_XMIT_STATUS       : NOVALUE,
;    1514  1         CLR_BUFFERS           : NOVALUE,
;    1515  1         CLR_DESCR             : NOVALUE,
;    1516  1         COMPARE_PACKETS       : NOVALUE,
;    1517  1         E1$REPORT             : NOVALUE,          ! PRINT EXTENDED ERROR MESSAGE
;    1518  1         ERROR$REPORT          : NOVALUE,          ! PRINT EXTENDED ERROR MESSAGE
;    1519  1         FORM_HEX_ADR          : NOVALUE,
;    1520  1         KBD_INT               : NOVALUE,
;    1521  1         NXM_INT               : L$ISR NOVALUE,    ! NXM INTERRUPT SERVICE ROUTINE
;    1522  1         PREP_FOR_SETUP        : NOVALUE,
;    1523  1         PWR_INT               : NOVALUE,
;    1524  1         RESET_DEQNA           : NOVALUE,
;    1525  1         SEND_ELOOP_PACKET     : NOVALUE,
;    1526  1         SEND_TEST_PACKET      : NOVALUE,
;    1527  1         SET_XDESCR_LIST       : NOVALUE,
;    1528  1         SET_RDESCR_LIST       : NOVALUE,
;    1529  1         TURN_OFF_LED          : NOVALUE,
;    1530  1         VER_DESCR_STATUS      : NOVALUE,
;    1531  1         WAIT_FOR_TIMEOUT      : NOVALUE,
;    1532  1         WALKING_BIT           : NOVALUE,
;    1533  1         WRT_STATION_ADR       : NOVALUE,
;    1534  1         XMIT_AND_RCV_PACKET   : NOVALUE,
;    1535  1         XMIT_ILOOP_PACKET     : NOVALUE,
;    1536  1         XMIT_SETUP_PACKET     : NOVALUE;
;    1537  1
;    1538  1
```

```
;    1539  1
;    1540  1
;    1541  1     EXTERNAL
;    1542  1
;    1543  1     !++
;    1544  1     !       COMMUNICATION AREA DECLARATIONS
;    1545  1     !--
;    1546  1
;    1547  1         RCV_D_LIST      : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1548  1         XMIT_D_LIST     : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1549  1         DESCR_LIST      : BLOCK  [ DESCR_SIZE, WORD] FIELD ( DL_FIELDS ),
;    1550  1         RCV_BUFFER      : VECTOR [ B_SIZE, BYTE ],
;    1551  1         XMIT_BUFFER     : VECTOR [ B_SIZE, BYTE ],
;    1552  1         DATA_BUFFER     : VECTOR [ BUF_SIZE, BYTE ],
;    1553  1         TARGET_ADR      : VECTOR [ T_SIZE, BYTE ],
;    1554  1         PHYS_ADR        : VECTOR [ 22, BYTE ],
;    1555  1         IOP_TABLE       : VECTOR [ 8, WORD ],
;    1556  1         RD13            : VECTOR [ 64, WORD ],
;    1557  1         TD13            : VECTOR [ 28, WORD ],
;    1558  1         TD16            : VECTOR [ 44, WORD ],
;    1559  1         BD_PROM_DESCR   : VECTOR [ BD_D_SIZE, WORD ],
;    1560  1         STATION_ADR     : VECTOR [ 4, WORD ],
;    1561  1         PTRN_TABLE      : VECTOR [ 8, BYTE ],
;    1562  1
;    1563  1     !++
;    1564  1     !   HARDWARE AND SOFTWARE P-TABLE STORAGE DECLARATIONS
;    1565  1     !--
;    1566  1
;    1567  1         HWP_TABLE   : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;    1568  1         SWP_TABLE   : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    1569  1
;    1570  1         REG_ADR             : REF REG_STR FIELD ( IOP_FIELDS ),
;    1571  1         GET_ADR             : REF ADR_STR FIELD ( IOP_FIELDS ),
;    1572  1         IOP_DATA            : REF REG_STR FIELD ( IOP_FIELDS ),
;    1573  1
```

```
;    1574  1                                                  ! (0=NONE,-1=L-CLOCK,1=P_CLOCK)
;    1575  1      !++
;    1576  1      !       MISCELLANEOUS DATA DECLARATIONS
;    1577  1      !--
;    1578  1
;    1579  1          XBUF_LENGTH,        RBUF_LENGTH,    INTERRUPT_FLG,      COUNTER,
;    1580  1          SWP_BLOCK_MEM,      SWP_TOUT_VAL,   SWP_ILOOP,          SWP_TIMER,
;    1581  1          UP_COUNTER,         DOWN_COUNTER,   CHECKSUM,           ERR_NUMBER,
;    1582  1          XC_FLAG,            SWP_LBC,
;    1583  1          ERR_COUNT,          ERR_FLAG,       CSR_WORD,           PRI00,
;    1584  1          PRI01,              PRI02,          PRI03,              PRI04,
;    1585  1          PRI05,              PRI06,          PRI07,              DEQNA_NO   : WORD,
;    1586  1
;    1587  1      !++
;    1588  1      !   TEMPORARY STORAGE DATA DECLARATIONS
;    1589  1      !--
;    1590  1
;    1591  1          P1,                 P2,             P3,                 P4,
;    1592  1          TMP_IOP_ADR,        TMP_REG_DATA,   TEMP1,              TEMP2,
;    1593  1          TEMP3,              TEMP4,          TEMP5,              TEMP6,
;    1594  1          TEMP7,              TEMP8,          TEMP9,              TADR1,
;    1595  1          TADR2                                                           : WORD,
;    1596  1          TBYTE1,             TBYTE2,         TBYTE3,             TBYTE4     : BYTE,
;    1597  1
```

```
;    1598  1
;    1599  1     !++
;    1600  1     !  ERROR MESSAGES DEFINED EXTERNALLY
;    1601  1     !--
;    1602  1
;    1603  1
;    1604  1         MSG00, MSG71,
;    1605  1         MSG01, MSG02, MSG03, MSG04, MSG05, MSG06, MSG07, MSG08, MSG09, MSG10,
;    1606  1         MSG11, MSG12, MSG13, MSG14, MSG15, MSG16, MSG17, MSG18, MSG19, MSG20,
;    1607  1         MSG21, MSG22, MSG23, MSG24, MSG25, MSG26, MSG27, MSG28, MSG29, MSG30,
;    1608  1         MSG31, MSG32, MSG33, MSG34, MSG35, MSG36, MSG37, MSG38, MSG39, MSG40,
;    1609  1         MSG41, MSG42, MSG43, MSG44, MSG45, MSG46, MSG47, MSG48, MSG49, MSG50,
;    1610  1         MSG51, MSG52, MSG53, MSG54, MSG55, MSG56, MSG57, MSG58, MSG59, MSG60,
;    1611  1         MSG61, MSG62, MSG63, MSG64, MSG65, MSG66, MSG67, MSG68, MSG69, MSG70;
;    1612  1
;    1613  1
```

```
;    1614  1     %SBTTL 'TEST 1 - NON-EXISTANT I/O PAGE REGISTER TEST'
;    1615  1     !++
;    1616  1     !
;    1617  1     !  TEST 1:      NON-EXISTANT I/O PAGE REGISTER TEST
;    1618  1     !
;    1619  1     !  DESCRIPTION:
;    1620  1     !
;    1621  1     !       This test verifies that all the device registers residing in the
;    1622  1     !       I/O Page can be accessed without forcing a non-existant memory (NXM)
;    1623  1     !       interrupt. If the operator specifies loop on error, the program
;    1624  1     !       re-executes the code that detected the error until ^C is entered.
;    1625  1     !
;    1626  1     !       Hardware tested:         Q-Bus to DEQNA Slave Registers Interface
;    1627  1     !
;    1628  1     !       Processing:
;    1629  1     !
;    1630  1     !            BEGIN
;    1631  1     !               get ready for NXM interrupt
;    1632  1     !               REPEAT for every I/O page register
;    1633  1     !                  read I/O page register
;    1634  1     !                  IF NXM occured
;    1635  1     !                  THEN
;    1636  1     !                     print error message if not inhibited
;    1637  1     !                  ENDIF
;    1638  1     !               ENDREPEAT
;    1639  1     !
;    1640  1     !               write any data pattern into the first 2 I/O page
;    1641  1     !                 registers
;    1642  1     !               IF NXM occured
;    1643  1     !               THEN
;    1644  1     !                  print error message if not inhibited
;    1645  1     !               ENDIF
;    1646  1     !            END
;    1647  1     !-
```

```
;    1648  3     BGNTST;
;    1649  3
;    1650  3     SETVEC (4, NXM_INT, PRIO7);                  ! SET UP FOR AN NXM INTERRUPT
;    1651  3     DELAY (M5_DELAY);                            ! DELAY 50 x 100 us = 5 ms
;    1652  3     INTERRUPT_FLG = CLEAR_FLG;                   ! CLEAR OUT NEX FLAG
;    1653  3
;    1654  3     TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;    1655  3     INCR INDEX FROM 0 TO 7 DO
;    1656  4       BEGIN
;    1657  6         BGNSUB;
;    1658  6           TEMP1 = ..TMP_IOP_ADR;
;    1659  6           DELAY(7);
;    1660  6           IF .INTERRUPT_FLG EQLU WORD_LIMIT      ! SEE IF WE GOT A NXM INTRT
;    1661  6             THEN
;    1662  7               BEGIN                              ! ADDRESS NOT THERE
;    1663  7                 INTERRUPT_FLG = CLEAR_FLG;       ! CLEAR TRAP FLAG
;    1664  7                 PRINTB ( MSG59 );
;    1665  7                 PRINTB ( MSG70, .TMP_IOP_ADR );
;    1666  7                 ERRDF (0101, MSG00, E1$REPORT);  ! 'I/O PAGE REG. NOT PRESENT'
;    1667  7                 DODU ( DEQNA_NO );
;    1668  7                 DOCLN;
;    1669  6               END;
;    1670  4         ENDSUB;
;    1671  4         TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;    1672  3       END;
;    1673  3
;    1674  3     TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;    1675  3     INCR INDEX FROM 0 TO 1 DO
;    1676  4       BEGIN
;    1677  6         BGNSUB;
;    1678  6           .TMP_IOP_ADR = %X'7F';                 ! WRITE FIRST 2 LOCATIONS
;    1679  6           DELAY(7);
;    1680  6           IF .INTERRUPT_FLG EQLU WORD_LIMIT      ! SEE IF WE GOT A NXM INTRT
;    1681  6             THEN
;    1682  7               BEGIN                              ! ADDRESS NOT THERE
;    1683  7                 INTERRUPT_FLG = CLEAR_FLG;       ! CLEAR TRAP FLAG
;    1684  7                 PRINTB ( MSG59 );
;    1685  7                 PRINTB ( MSG70, .TMP_IOP_ADR );
;    1686  7                 ERRDF (0102, MSG00, E1$REPORT);  ! 'I/O PAGE REG. NOT PRESENT'
;    1687  7                 DODU ( DEQNA_NO );
;    1688  7                 DOCLN;
;    1689  6               END;
;    1690  4         ENDSUB;
;    1691  4         TMP IOP ADR = .TMP_IOP_ADR + 2;
;    1692  3       END;
;    1693  3
;    1694  3     CLRVEC (4);                                  ! CLEAR INTERRUPT VECTOR
;    1695  3
;    1696  1     ENDTST;

                                        .TITLE  ZQNA3 CZQNADO DEQNA FUNCTIONAL TEST
                                        .IDENT  /V01.0/
```

```
                                        .ENABL  AMA

                                        .GLOBL  CHK.CSR.STATUS, CHK.RIXI.STATUS
                                        .GLOBL  CHK.RCV.STATUS, CHK.XMIT.STATUS
                                        .GLOBL  CLR.BUFFERS, CLR.DESCR, COMPARE.PACKETS
                                        .GLOBL  E1$REPORT, ERROR$REPORT, FORM.HEX.ADR
                                        .GLOBL  KBD.INT, NXM.INT, PREP.FOR.SETUP
                                        .GLOBL  PWR.INT, RESET.DEQNA, SEND.ELOOP.PACKET
                                        .GLOBL  SEND.TEST.PACKET, SET.XDESCR.LIST
                                        .GLOBL  SET.RDESCR.LIST, TURN.OFF.LED
                                        .GLOBL  VER.DESCR.STATUS, WAIT.FOR.TIMEOUT
                                        .GLOBL  WALKING.BIT, WRT.STATION.ADR, XMIT.AND.RCV.PACKET
                                        .GLOBL  XMIT.ILOOP.PACKET, XMIT.SETUP.PACKET
                                        .GLOBL  RCV.D.LIST, XMIT.D.LIST, DESCR.LIST
                                        .GLOBL  RCV.BUFFER, XMIT.BUFFER, DATA.BUFFER
                                        .GLOBL  TARGET.ADR, PHYS.ADR, IOP.TABLE
                                        .GLOBL  RD13, TD13, TD16, BD.PROM.DESCR
                                        .GLOBL  STATION.ADR, PTRN.TABLE, HWP.TABLE
                                        .GLOBL  SWP.TABLE, REG.ADR, GET.ADR, IOP.DATA
                                        .GLOBL  XBUF.LENGTH, RBUF.LENGTH, INTERRUPT.FLG
                                        .GLOBL  COUNTER, SWP.BLOCK.MEM, SWP.TOUT.VAL
                                        .GLOBL  SWP.ILOOP, SWP.TIMER, UP.COUNTER
                                        .GLOBL  DOWN.COUNTER, CHECKSUM, ERR.NUMBER
                                        .GLOBL  XC.FLAG, SWP.LBC, ERR.COUNT, ERR.FLAG
                                        .GLOBL  CSR.WORD, PRI00, PRI01, PRI02
                                        .GLOBL  PRI03, PRI04, PRI05, PRI06, PRI07
                                        .GLOBL  DEQNA.NO, P1, P2, P3, P4, TMP.IOP.ADR
                                        .GLOBL  TMP.REG.DATA, TEMP1, TEMP2, TEMP3
                                        .GLOBL  TEMP4, TEMP5, TEMP6, TEMP7, TEMP8
                                        .GLOBL  TEMP9, TADR1, TADR2, TBYTE1, TBYTE2
                                        .GLOBL  TBYTE3, TBYTE4, MSG00, MSG71, MSG01
                                        .GLOBL  MSG02, MSG03, MSG04, MSG05, MSG06
                                        .GLOBL  MSG07, MSG08, MSG09, MSG10, MSG11
                                        .GLOBL  MSG12, MSG13, MSG14, MSG15, MSG16
                                        .GLOBL  MSG17, MSG18, MSG19, MSG20, MSG21
                                        .GLOBL  MSG22, MSG23, MSG24, MSG25, MSG26
                                        .GLOBL  MSG27, MSG28, MSG29, MSG30, MSG31
                                        .GLOBL  MSG32, MSG33, MSG34, MSG35, MSG36
                                        .GLOBL  MSG37, MSG38, MSG39, MSG40, MSG41
                                        .GLOBL  MSG42, MSG43, MSG44, MSG45, MSG46
                                        .GLOBL  MSG47, MSG48, MSG49, MSG50, MSG51
                                        .GLOBL  MSG52, MSG53, MSG54, MSG55, MSG56
                                        .GLOBL  MSG57, MSG58, MSG59, MSG60, MSG61
                                        .GLOBL  MSG62, MSG63, MSG64, MSG65, MSG66
                                        .GLOBL  MSG67, MSG68, MSG69, MSG70, L$DLY


                                        .SBTTL  $T1 TEST 1 - NON-EXISTANT I/O PAGE REGISTER TEST
000000                                  .PSECT  AB$CODE$,  RO

000000  004137  000000G          $T1:   JSR     R1,$SAVE2                       ;                1611
000004  005746                         TST     -(SP)
000006  012746  000000G                MOV     #PRI07,-(SP)                    ;                1650
```

```
000012  012746  000000G                        MOV     #NXM.INT,-(SP)
000016  012746  000004                         MOV     #4,-(SP)
000022  012746  000003                         MOV     #3,-(SP)
000026  104437                                 TRAP    37
000030  012701  000062                         MOV     #62,R1                  ; *,$$TMP2                      1651
000034  001410                          1$:    BEQ     4$
000036  013700  000000G                        MOV     L$DLY,R0                ; *,$$TMP1
000042  001403                                 BEQ     3$
000044  005066  000010                  2$:    CLR     10(SP)                  ; $$TMP
000050  077003                                 SOB     R0,2$                   ; $$TMP1,*
000052  005301                          3$:    DEC     R1                      ; $$TMP2
000054  000767                                 BR      1$
000056  005037  000000G                  4$:    CLR     INTERRUPT.FLG           ;                               1652
000062  017737  000000G 000000G                MOV     @HWP.TABLE,TMP.IOP.ADR  ;                               1654
000070  012702  000010                         MOV     #10,R2                  ; *,INDEX                       1655
000074  104402                          5$:    TRAP    2                       ;                               1656
000076  017737  000000G 000000G                MOV     @TMP.IOP.ADR,TEMP1      ;                               1658
000104  012701  000007                         MOV     #7,R1                   ; *,$$TMP2                      1659
000110  001410                          6$:    BEQ     9$
000112  013700  000000G                        MOV     L$DLY,R0                ; *,$$TMP1
000116  001403                                 BEQ     8$
000120  005066  000010                  7$:    CLR     10(SP)                  ; $$TMP
000124  077003                                 SOB     R0,7$                   ; $$TMP1,*
000126  005301                          8$:    DEC     R1                      ; $$TMP2
000130  000767                                 BR      6$
000132  023727  000000G 177777          9$:    CMP     INTERRUPT.FLG,#-1       ;                               1660
000140  001032                                 BNE     10$
000142  005037  000000G                        CLR     INTERRUPT.FLG           ;                               1663
000146  012716  000000G                        MOV     #MSG59,(SP)             ;                               1664
000152  012746  000001                         MOV     #1,-(SP)
000156  010600                                 MOV     SP,R0                   ; SP,*
000160  104414                                 TRAP    14
000162  013716  000000G                        MOV     TMP.IOP.ADR,(SP)        ;                               1665
000166  012746  000000G                        MOV     #MSG70,-(SP)
000172  012746  000002                         MOV     #2,-(SP)
000176  010600                                 MOV     SP,R0                   ; SP,*
000200  104414                                 TRAP    14
000202  104455                                 TRAP    55                      ;                               1666
000204  000145                                 .WORD   145
000206  000000G                                .WORD   MSG00
000210  000000G                                .WORD   E1$REPORT
000212  012700  000000G                        MOV     #DEQNA.NO,R0            ;                               1667
000216  104451                                 TRAP    51
000220  104444                                 TRAP    44
000222  062706  000006                         ADD     #6,SP                   ;                               1662
000226  104467                          10$:   TRAP    67                      ;                               1669
000230  006000                                 ROR     R0
000232  103720                                 BLO     5$
000234  062737  000002  000000G                ADD     #2,TMP.IOP.ADR          ;                               1671
000242  077264                                 SOB     R2,5$                   ; INDEX,*                       1655
000244  017737  000000G 000000G                MOV     @HWP.TABLE,TMP.IOP.ADR  ;                               1674
000252  012702  000002                         MOV     #2,R2                   ; *,INDEX                       1675
000256  104402                          11$:   TRAP    2                       ;                               1676
```

```
000260  012777  000177  000000G                    MOV     #177,@TMP.IOP.ADR                ;                                1678
000266  012701  000007                             MOV     #7,R1                            ; *,$$TMP2                       1679
000272  001410                            12$:     BEQ     15$
000274  013700  000000G                            MOV     L$DLY,R0                         ; *,$$TMP1
000300  001403                                     BEQ     14$
000302  005066  000010                    13$:     CLR     10(SP)                           ; $$TMP
000306  077003                                     SOB     R0,13$                           ; $$TMP1,*
000310  005301                            14$:     DEC     R1                               ; $$TMP2
000312  000767                                     BR      12$
000314  023727  000000G 177777            15$:     CMP     INTERRUPT.FLG,#-1                ;                                1680
000322  001032                                     BNE     16$
000324  005037  000000G                            CLR     INTERRUPT.FLG                    ;                                1683
000330  012716  000000G                            MOV     #MSG59,(SP)                      ;                                1684
000334  012746  000001                             MOV     #1,-(SP)
000340  010600                                     MOV     SP,R0                            ; SP,*
000342  104414                                     TRAP    14
000344  013716  000000G                            MOV     TMP.IOP.ADR,(SP)                 ;                                1685
000350  012746  000000G                            MOV     #MSG70,-(SP)
000354  012746  000002                             MOV     #2,-(SP)
000360  010600                                     MOV     SP,R0                            ; SP,*
000362  104414                                     TRAP    14
000364  104455                                     TRAP    55                               ;                                1686
000366  000146                                     .WORD   146
000370  000000G                                    .WORD   MSG00
000372  000000G                                    .WORD   E1$REPORT
000374  012700  000000G                            MOV     #DEQNA.NO,R0                     ;                                1687
000400  104451                                     TRAP    51
000402  104444                                     TRAP    44
000404  062706  000006                             ADD     #6,SP                            ;                                1682
000410  104467                            16$:     TRAP    67                               ;                                1689
000412  006000                                     ROR     R0
000414  103720                                     BLO     11$
000416  062737  000002  000000G                    ADD     #2,TMP.IOP.ADR                   ;                                1691
000424  077264                                     SOB     R2,11$                           ; INDEX,*                        1675
000426  012700  000004                             MOV     #4,R0                            ;                                1694
000432  104436                                     TRAP    36
000434  062706  000012                             ADD     #12,SP                           ;                                1611
000440  000207                                     RTS     PC

; Routine Size:  145 words,     Routine Base:  AB$CODE$ + 0000
; Maximum stack depth per invocation:  13 words




                                                   .SBTTL  T1 TEST 1 - NON-EXISTANT I/O PAGE REGISTER TEST
000000  004737  000000'                    T1::
000000                                     1$:     JSR     PC,$T1                           ;                                1694
000004  104466                                     TRAP    66
000006  006000                                     ROR     R0
000010  103773                                     BLO     1$
000012  000207                                     RTS     PC
```

```
; Routine Size:  6 words,      Routine Base:  AB$CODE$ + 0442
; Maximum stack depth per invocation:  2 words


;   1697  1
;   1698  1
```

```
;  1699  1     %SBTTL 'TEST 2 - CSR STATIC BIT TEST'
;  1700  1     !++
;  1701  1     !
;  1702  1     !  TEST 2:       CSR STATIC BIT TEST
;  1703  1     !
;  1704  1     !  DESCRIPTION:
;  1705  1     !
;  1706  1     !        This test verifies that the CSR register static bits can be set
;  1707  1     !       and cleared as specified.  The host writes data patterns to this
;  1708  1     !       register and reads them back verifying  no static
;  1709  1     !       (stuck at 1 / stuck at 0) faults occur. If the operator specifies
;  1710  1     !       loop on error, the program re-executes the code that detected the
;  1711  1     !       error until ^C is entered.
;  1712  1     !
;  1713  1     !       Hardware tested:                Q-Bus to DEQNA Slave Regs. Interface
;  1714  1     !
;  1715  1     !       Processing:
;  1716  1     !
;  1717  1     !            BEGIN
;  1718  1     !               check Software Reset ( SR ) bit in the CSR for stuck at 0
;  1719  1     !                  and 1
;  1720  1     !               IF error
;  1721  1     !               THEN
;  1722  1     !                  print error message if not inhibited
;  1723  1     !               ENDIF
;  1724  1     !               set static bits ( 0,3,8,9 ) and check for expected CSR status
;  1725  1     !               IF error
;  1726  1     !               THEN
;  1727  1     !                  print error message if not inhibited
;  1728  1     !               ENDIF
;  1729  1     !               clear static bits and check for expected CSR status
;  1730  1     !               IF error
;  1731  1     !               THEN
;  1732  1     !                  print error message if not inhibited
;  1733  1     !               ENDIF
;  1734  1     !               set static bits ( 0,3,8,9 ) and check for expected CSR status
;  1735  1     !               IF error
;  1736  1     !               THEN
;  1737  1     !                  print error message if not inhibited
;  1738  1     !               ENDIF
;  1739  1     !               reset DEQNA and check for expected CSR status
;  1740  1     !               IF error
;  1741  1     !               THEN
;  1742  1     !                  print error message if not inhibited
;  1743  1     !               ENDIF
;  1744  1     !            END
;  1745  1     !-
```

```
;  1746  3     BGNTST;
;  1747  3
;  1748  5     BGNSUB;
;  1749  5
;  1750  5       !++
;  1751  5       !  CHECK IF CSR STATIC BITS (BIT 0,3,8 AND 9) ARE NOT STUCK AT 0
;  1752  5       !--
;  1753  5
;  1754  5       RESET_DEQNA ( );
;  1755  5       PUT_BIT ( CSR, ALL_BITS, PATRN1 );
;  1756  5       DELAY ( TIME6_LIMIT );
;  1757  5       TEMP1 = GET_BIT [ CSR_ALL ] AND PATRN1;
;  1758  5       IF .TEMP1 NEQU PATRN1
;  1759  5         THEN
;  1760  6           BEGIN
;  1761  6             PRINTB ( MSG59 );
;  1762  6             PRINTB ( MSG60 );
;  1763  6             PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP1, PATRN1 );
;  1764  6             ERRDF ( 0201, MSG00, E1$REPORT );
;  1765  5           END;
;  1766  3     ENDSUB;
;  1767  3
;  1768  3     !++
;  1769  3     !  CHECK IF CSR STATIC BITS (BIT 0,3,8 AND 9) ARE NOT STUCK AT 1
;  1770  3     !--
;  1771  3
;  1772  5     BGNSUB;
;  1773  5       PUT_BIT ( CSR, ALL_BITS, ZERO );
;  1774  5       DELAY ( TIME6_LIMIT );
;  1775  5       TEMP2 = GET_BIT [ CSR_ALL ] AND PATRN1;
;  1776  5       IF .TEMP2 NEQU ZERO
;  1777  5         THEN
;  1778  6           BEGIN
;  1779  6             PRINTB ( MSG59 );
;  1780  6             PRINTB ( MSG61 );
;  1781  6             PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP2, ZERO );
;  1782  6             ERRDF ( 0202, MSG00, E1$REPORT );
;  1783  5           END;
;  1784  3     ENDSUB;
;  1785  3
;  1786  5     BGNSUB;
;  1787  5       PUT_BIT ( CSR, ALL_BITS, PATRN1 );
;  1788  5       RESET_DEQNA ( );
;  1789  5       TEMP3 = GET_BIT [ CSR_ALL ] AND PATRN1;
;  1790  5       IF .TEMP3 NEQU ZERO
;  1791  5         THEN
;  1792  6           BEGIN
;  1793  6             PRINTB ( MSG59 );
;  1794  6             PRINTB ( MSG62 );
;  1795  6             PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP4, ZERO );
;  1796  6             ERRDF ( 0203, MSG00, E1$REPORT );
;  1797  5           END;
;  1798  3     ENDSUB;
```

```
;   1799  3
;   1800  1     ENDTST;


                                            .SBTTL  $T2 TEST 2 - CSR STATIC BIT TEST
000000  004137  000000G                $T2:  JSR     R1,$SAVE2                    ;                          1696
000004  162706  000016                      SUB     #16,SP
000010  104402                         1$:  TRAP    2                            ;                          1746
000012  004737  000000G                      JSR     PC,RESET.DEQNA               ;                          1754
000016  013701  000000G                      MOV     REG.ADR,R1                   ;                          1755
000022  012761  001411  000016              MOV     #1411,16(R1)
000030  012702  000001                      MOV     #1,R2                        ; *,$$TMP2                 1756
000034  001410                         2$:  BEQ     5$
000036  013700  000000G                      MOV     L$DLY,R0                     ; *,$$TMP1
000042  001403                              BEQ     4$
000044  005066  000014                 3$:  CLR     14(SP)                       ; $$TMP
000050  077003                              SOB     R0,3$                        ; $$TMP1,*
000052  005302                         4$:  DEC     R2                           ; $$TMP2
000054  000767                              BR      2$
000056  016116  000016                 5$:  MOV     16(R1),(SP)                  ; *,TMP.LOCATION           1757
000062  011637  000000G                      MOV     (SP),TEMP1                   ; TMP.LOCATION,*
000066  042737  176366  000000G             BIC     #176366,TEMP1
000074  023727  000000G 001411              CMP     TEMP1,#1411                  ;                          1758
000102  001444                              BEQ     6$
000104  012746  000000G                      MOV     #MSG59,-(SP)                 ;                          1761
000110  012746  000001                      MOV     #1,-(SP)
000114  010600                              MOV     SP,R0                        ; SP,*
000116  104414                              TRAP    14
000120  012716  000000G                      MOV     #MSG60,(SP)                  ;                          1762
000124  012746  000001                      MOV     #1,-(SP)
000130  010600                              MOV     SP,R0                        ; SP,*
000132  104414                              TRAP    14
000134  012716  001411                      MOV     #1411,(SP)                   ;                          1763
000140  013746  000000G                      MOV     TEMP1,-(SP)
000144  013766  000000G 000012              MOV     GET.ADR,12(SP)               ; *,TMP.LOCATION
000152  062766  000016  000012              ADD     #16,12(SP)                   ; *,TMP.LOCATION
000160  016646  000012                      MOV     12(SP),-(SP)                 ; TMP.LOCATION,*
000164  012746  000000G                      MOV     #MSG30,-(SP)
000170  012746  000004                      MOV     #4,-(SP)
000174  010600                              MOV     SP,R0                        ; SP,*
000176  104414                              TRAP    14
000200  104455                              TRAP    55                           ;                          1764
000202  000311                              .WORD   311
000204  000000G                             .WORD   MSG00
000206  000000G                             .WORD   E1$REPORT
000210  062706  000016                      ADD     #16,SP                       ;                          1760
000214  104467                         6$:  TRAP    67                           ;                          1765
000216  006000                              ROR     R0
000220  103673                              BLO     1$
000222  104402                         7$:  TRAP    2                            ;                          1766
000224  013701  000000G                      MOV     REG.ADR,R1                   ;                          1773
000230  005061  000016                      CLR     16(R1)
000234  012702  000001                      MOV     #1,R2                        ; *,$$TMP2                 1774
```

```
000240  001410                          8$:     BEQ     11$
000242  013700  000000G                       MOV     L$DLY,R0                ; *,$$TMP1
000246  001403                                BEQ     10$
000250  005066  000014                  9$:     CLR     14(SP)                  ; $$TMP
000254  077003                                SOB     R0,9$                   ; $$TMP1,*
000256  005302                          10$:    DEC     R2                      ; $$TMP2
000260  000767                                BR      8$
000262  016166  000016  000004          11$:    MOV     16(R1),4(SP)            ; *,TMP.LOCATION                        1775
000270  016637  000004  000000G               MOV     4(SP),TEMP2             ; TMP.LOCATION,*
000276  042737  176366  000000G               BIC     #176366,TEMP2
000304  001443                                BEQ     12$                     ;                                       1776
000306  012746  000000G                       MOV     #MSG59,-(SP)            ;                                       1779
000312  012746  000001                        MOV     #1,-(SP)
000316  010600                                MOV     SP,R0                   ; SP,*
000320  104414                                TRAP    14
000322  012716  000000G                       MOV     #MSG61,(SP)             ;                                       1780
000326  012746  000001                        MOV     #1,-(SP)
000332  010600                                MOV     SP,R0                   ; SP,*
000334  104414                                TRAP    14
000336  005016                                CLR     (SP)                    ;                                       1781
000340  013746  000000G                       MOV     TEMP2,-(SP)
000344  013766  000000G 000016                MOV     GET.ADR,16(SP)          ; *,TMP.LOCATION
000352  062766  000016  000016                ADD     #16,16(SP)              ; *,TMP.LOCATION
000360  016646  000016                        MOV     16(SP),-(SP)            ; TMP.LOCATION,*
000364  012746  000000G                       MOV     #MSG30,-(SP)
000370  012746  000004                        MOV     #4,-(SP)
000374  010600                                MOV     SP,R0                   ; SP,*
000376  104414                                TRAP    14
000400  104455                                TRAP    55                      ;                                       1782
000402  000312                                .WORD   312
000404  000000G                               .WORD   MSG00
000406  000000G                               .WORD   E1$REPORT
000410  062706  000016                        ADD     #16,SP                  ;                                       1778
000414  104467                          12$:    TRAP    67                      ;                                       1783
000416  006000                                ROR     R0
000420  103700                                BLO     7$
000422  104402                          13$:    TRAP    2                       ;                                       1784
000424  013700  000000G                       MOV     REG.ADR,R0              ;                                       1787
000430  012760  001411  000016                MOV     #1411,16(R0)
000436  004737  000000G                       JSR     PC,RESET.DEQNA          ;                                       1788
000442  013700  000000G                       MOV     REG.ADR,R0              ;                                       1789
000446  016066  000016  000010                MOV     16(R0),10(SP)           ; *,TMP.LOCATION
000454  016637  000010  000000G               MOV     10(SP),TEMP3            ; TMP.LOCATION,*
000462  042737  176366  000000G               BIC     #176366,TEMP3
000470  001443                                BEQ     14$                     ;                                       1790
000472  012746  000000G                       MOV     #MSG59,-(SP)            ;                                       1793
000476  012746  000001                        MOV     #1,-(SP)
000502  010600                                MOV     SP,R0                   ; SP,*
000504  104414                                TRAP    14
000506  012716  000000G                       MOV     #MSG62,(SP)             ;                                       1794
000512  012746  000001                        MOV     #1,-(SP)
000516  010600                                MOV     SP,R0                   ; SP,*
000520  104414                                TRAP    14
```

```
000522  005016                                  CLR     (SP)                            ;                                       1795
000524  013746  000000G                         MOV     TEMP4,-(SP)
000530  013766  000000G 000022                  MOV     GET.ADR,22(SP)                  ; *,TMP.LOCATION
000536  062766  000016  000022                  ADD     #16,22(SP)                      ; *,TMP.LOCATION
000544  016646  000022                          MOV     22(SP),-(SP)                    ; TMP.LOCATION,*
000550  012746  000000G                         MOV     #MSG30,-(SP)
000554  012746  000004                          MOV     #4,-(SP)
000560  010600                                  MOV     SP,R0                           ; SP,*
000562  104414                                  TRAP    14
000564  104455                                  TRAP    55                              ;                                       1796
000566  000313                                  .WORD   313
000570  000000G                                 .WORD   MSG00
000572  000000G                                 .WORD   E1$REPORT
000574  062706  000016                          ADD     #16,SP                          ;                                       1792
000600  104467                          14$:    TRAP    67                              ;                                       1797
000602  006000                                  ROR     R0
000604  103706                                  BLO     13$
000606  062706  000016                          ADD     #16,SP                          ;                                       1696
000612  000207                                  RTS     PC

; Routine Size:  198 words,     Routine Base:  AB$CODE$ + 0456
; Maximum stack depth per invocation:  19 words



                                                .SBTTL  T2 TEST 2 - CSR STATIC BIT TEST
000000  004737  000456'                  T2::
000000                                  1$:     JSR     PC,$T2                          ;                                       1798
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 1272
; Maximum stack depth per invocation:  2 words


;   1801  1
;   1802  1
```

```
;    1803  1     %SBTTL 'TEST 3 - ETHERNET STATION ADDRESS VERIFY TEST'
;    1804  1     !++
;    1805  1     !
;    1806  1     !  TEST 3:      ETHERNET STATION ADDRESS VERIFY TEST
;    1807  1     !
;    1808  1     !  DESCRIPTION:
;    1809  1     !
;    1810  1     !       This test verifies that  the Ethernet Station Address PROM can be
;    1811  1     !       read and loaded to host memory correctly. Ethernet Station Address is
;    1812  1     !       verified and checksum is computed from PROM data read and this checksum
;    1813  1     !       is compared to the checksum stored in the Ethernet Station Address
;    1814  1     !       PROM. Ethernet Station Address is always printed out on the console in
;    1815  1     !       the Ethernet standard format. If the address is not proper, the error
;    1816  1     !       is recorded and an appropriate error message is printed out on the
;    1817  1     !       console. If the operator specifies loop on error, the program
;    1818  1     !       re-executes the code that detected the error until ^C is entered.
;    1819  1     !
;    1820  1     !       Hardware tested:        Station Address PROM
;    1821  1     !                               Q-Bus DMA Interface
;    1822  1     !       Processing:
;    1823  1     !
;    1824  1     !            BEGIN
;    1825  1     !
;    1826  1     !               read DEQNA Station Address PROM and checksum
;    1827  1     !               save copy of Station Address PROM in host memory
;    1828  1     !               print Station Address on the console in standard format
;    1829  1     !               compute Station Address ROM checksum
;    1830  1     !               IF checksum read not equal checksum computed
;    1831  1     !               THEN
;    1832  1     !                  print error message if not inhibited
;    1833  1     !               ENDIF
;    1834  1     !               IF Station Address
;    1835  1     !                     [all 0's]
;    1836  1     !                  OR [all 1's]:
;    1837  1     !                  OR [multicast bit set]:
;    1838  1     !               THEN
;    1839  1     !                  print error message if not inhibited
;    1840  1     !               ENDIF
;    1841  1     !
;    1842  1     !            END
;    1843  1     !--
```

```
;    1844  3     BGNTST;
;    1845  3
;    1846  5     BGNSUB;
;    1847  5       RESET_DEQNA ( );
;    1848  5       FORM_HEX_ADR ( PHA_INDEX );
;    1849  5
;    1850  5       !++
;    1851  5       !  COMPUTE EXPECTED CHECKSUM
;    1852  5       !--
;    1853  5
;    1854  5       CHECKSUM = 0;
;    1855  5
;    1856  5       INCR INDEX FROM 0 TO 5 BY 2 DO
;    1857  6         BEGIN
;    1858  6           IF ( .CHECKSUM AND %O'100000' ) NEQU ZERO
;    1859  6             THEN
;    1860  7               BEGIN
;    1861  7                 CHECKSUM = .CHECKSUM ^ 1;
;    1862  7                 CHECKSUM = .CHECKSUM + 1;
;    1863  7               END
;    1864  6             ELSE
;    1865  6               CHECKSUM = .CHECKSUM ^ 1;
;    1866  6
;    1867  6           CHECKSUM = .CHECKSUM + .STATION_ADR [ .COUNTER ];
;    1868  6
;    1869  6           IF .CHECKSUM GTRU WORD_LIMIT
;    1870  6             THEN
;    1871  6               CHECKSUM = .CHECKSUM + 1;
;    1872  6
;    1873  6           COUNTER = .COUNTER + 1;
;    1874  5         END;
;    1875  5
;    1876  5       !++
;    1877  5       !  PRINT PHYSICAL STATION ADDRESS
;    1878  5       !--
;    1879  5
;    1880  5       PRINTB ( MSG01, .HWP_TABLE [ ADDR ] );
;    1881  5       PRINTB ( PHYS_ADR );
;    1882  5
;    1883  5       !++
;    1884  5       !  READ ACTUAL CHECKSUM FROM DEQNA STATION ADDRESS PROM AND COMPARE IT TO
;    1885  5       !  THE EXPECTED CHECKSUM COMPUTED ABOVE.
;    1886  5       !--
;    1887  5
;    1888  5       PUT_BIT ( CSR, LB, EXT_LOOPBACK );
;    1889  5       DELAY ( 5 );
;    1890  5       TEMP1  = .REG_ADR [ 1, ALL_BITS ];
;    1891  5       TEMP1  = .TEMP1 ^ 8;
;    1892  5       TEMP2  = .REG_ADR [ 0, ALL_BITS ];
;    1893  5       STATION_ADR [ CHSUM ] = .TEMP1 OR ( .TEMP2 AND %O'000377' );
;    1894  5       PUT_BIT ( CSR, LB, ZERO );
;    1895  5       IF .CHECKSUM NEQU .STATION_ADR [ CHSUM ]
;    1896  5         THEN
```

```
;  1897  6          BEGIN
;  1898  6            PRINTB ( MSG59 );
;  1899  6            PRINTB ( MSG63, .STATION_ADR [ CHSUM ], .CHECKSUM );
;  1900  6            ERRDF  (0301, MSG00, E1$REPORT);
;  1901  5          END;
;  1902  3      ENDSUB;
;  1903  3
;  1904  3      TEMP3 = ZERO;
;  1905  3      TEMP4 = ZERO;
;  1906  3      INCR INDEX FROM 0 TO 2 DO
;  1907  4        BEGIN
;  1908  4          TEMP3 = .TEMP3 + .STATION_ADR [ .INDEX ];
;  1909  4          IF .STATION_ADR [ .INDEX ] EQLU %X'FFFF'
;  1910  4            THEN
;  1911  4              TEMP4 = .TEMP4 + 1;
;  1912  3        END;
;  1913  3
;  1914  4      IF     ( .TEMP3 EQLU ZERO )
;  1915  4         OR ( .TEMP4 GTRU ZERO )
;  1916  4         OR (( .STATION_ADR [ ZERO ] AND %X'0100' ) EQLU %X'0100' )
;  1917  3            THEN
;  1918  4              BEGIN
;  1919  4                PRINTB ( MSG59 );
;  1920  4                PRINTB ( MSG64 );
;  1921  4                PRINTB ( PHYS_ADR );
;  1922  4                ERRDF  ( 0302, MSG00, E1$REPORT);
;  1923  3           END;
;  1924  3
;  1925  1      ENDTST;


                                        .SBTTL  $T3 TEST 3 - ETHERNET STATION ADDRESS VERIFY TEST
000000  004137  000000G          $T3:  JSR     R1,$SAVE2                 ;                    1800
000004  162706  000006                SUB     #6,SP
000010  104402                  1$:   TRAP    2                         ;                    1844
000012  004737  000000G               JSR     PC,RESET.DEQNA            ;                    1847
000016  012746  000023                MOV     #23,-(SP)                 ;                    1848
000022  004737  000000G               JSR     PC,FORM.HEX.ADR
000026  005037  000000G               CLR     CHECKSUM                  ;                    1854
000032  005001                        CLR     R1                        ; INDEX              1856
000034  013700  000000G          2$:  MOV     CHECKSUM,R0               ;                    1861
000040  006300                        ASL     R0
000042  032737  100000  000000G       BIT     #-100000,CHECKSUM         ;                    1858
000050  001405                        BEQ     3$
000052  010037  000000G               MOV     R0,CHECKSUM               ;                    1861
000056  005237  000000G               INC     CHECKSUM                  ;                    1862
000062  000402                        BR      4$                        ;                    1858
000064  010037  000000G          3$:  MOV     R0,CHECKSUM               ;                    1865
000070  013700  000000G          4$:  MOV     COUNTER,R0                ;                    1867
000074  006300                        ASL     R0
000076  066037  000000G 000000G       ADD     STATION.ADR(R0),CHECKSUM
000104  005237  000000G               INC     COUNTER                   ;                    1873
000110  062701  000002                ADD     #2,R1                     ; *.INDEX            1856
```

```
000114  020127  000005                          CMP     R1,#5                   ; INDEX,*
000120  003745                                  BLE     2$
000122  017716  000000G                         MOV     @HWP.TABLE,(SP)         ;                       1880
000126  012746  000000G                         MOV     #MSG01,-(SP)
000132  012746  000002                          MOV     #2,-(SP)
000136  010600                                  MOV     SP,R0                   ; SP,*
000140  104414                                  TRAP    14
000142  012716  000000G                         MOV     #PHYS.ADR,(SP)          ;                       1881
000146  012746  000001                          MOV     #1,-(SP)
000152  010600                                  MOV     SP,R0                   ; SP,*
000154  104414                                  TRAP    14
000156  013701  000000G                         MOV     REG.ADR,R1              ;                       1888
000162  052761  001400  000016                  BIS     #1400,16(R1)
000170  012702  000005                          MOV     #5,R2                   ; *,$$TMP2              1889
000174  001410                          5$:     BEQ     8$
000176  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000202  001403                                  BEQ     7$
000204  005066  000014                  6$:     CLR     14(SP)                  ; $$TMP
000210  077003                                  SOB     R0,6$                   ; $$TMP1,*
000212  005302                          7$:     DEC     R2                      ; $$TMP2
000214  000767                                  BR      5$
000216  016166  000002  000010          8$:     MOV     2(R1),10(SP)            ; *,TMP.LOCATION        1890
000224  016600  000010                          MOV     10(SP),R0               ; TEMP1,*               1891
000230  072027  000010                          ASH     #10,R0
000234  010037  000000G                         MOV     R0,TEMP1
000240  011166  000012                          MOV     (R1),12(SP)             ; *,TMP.LOCATION        1892
000244  011137  000000G                         MOV     (R1),TEMP2              ; TMP.LOCATION,*
000250  005037  000006G                         CLR     STATION.ADR+6           ;                       1893
000254  111137  000006G                         MOVB    (R1),STATION.ADR+6      ; TEMP2,*
000260  050037  000006G                         BIS     R0,STATION.ADR+6        ; TEMP1,*
000264  042761  001400  000016                  BIC     #1400,16(R1)            ;                       1894
000272  023737  000000G 000006G                 CMP     CHECKSUM,STATION.ADR+6  ;                       1895
000300  001426                                  BEQ     9$
000302  012716  000000G                         MOV     #MSG59,(SP)             ;                       1898
000306  012746  000001                          MOV     #1,-(SP)
000312  010600                                  MOV     SP,R0                   ; SP,*
000314  104414                                  TRAP    14
000316  013716  000000G                         MOV     CHECKSUM,(SP)           ;                       1899
000322  013746  000006G                         MOV     STATION.ADR+6,-(SP)
000326  012746  000000G                         MOV     #MSG63,-(SP)
000332  012746  000003                          MOV     #3,-(SP)
000336  010600                                  MOV     SP,R0                   ; SP,*
000340  104414                                  TRAP    14
000342  104455                                  TRAP    55                      ;                       1900
000344  000455                                  .WORD   455
000346  000000G                                 .WORD   MSG00
000350  000000G                                 .WORD   E1$REPORT
000352  062706  000010                          ADD     #10,SP                  ;                       1897
000356  062706  000010                  9$:     ADD     #10,SP                  ;                       1844
000362  104467                                  TRAP    67                      ;                       1901
000364  006000                                  ROR     R0
000366  103610                                  BLO     1$
000370  005037  000000G                         CLR     TEMP3                   ;                       1904
```

```
000374  005037  000000G                         CLR     TEMP4                           ;                                       1905
000400  005000                                  CLR     R0                              ; INDEX                                 1906
000402  066037  000000G 000000G         10$:    ADD     STATION.ADR(R0),TEMP3           ; *(INDEX),*                            1908
000410  026027  000000G 177777                 CMP     STATION.ADR(R0),^ 1             ; *(INDEX),*                            1909
000416  001002                                  BNE     11$
000420  005237  000000G                         INC     TEMP4                           ;                                       1911
000424  062700  000002                 11$:    ADD     #2,R0                           ; *,INDEX                               1906
000430  020027  000004                         CMP     R0,#4                           ; INDEX,*
000434  003762                                  BLE     10$
000436  005737  000000G                         TST     TEMP3                           ;                                       1914
000442  001407                                  BEQ     12$
000444  005737  000000G                         TST     TEMP4                           ;                                       1915
000450  001004                                  BNE     12$
000452  032737  000400  000000G                 BIT     #400,STATION.ADR                ;                                       1916
000460  001430                                  BEQ     13$
000462  012746  000000G                 12$:    MOV     #MSG59,-(SP)                    ;                                       1919
000466  012746  000001                         MOV     #1,-(SP)
000472  010600                                  MOV     SP,R0                           ; SP,*
000474  104414                                  TRAP    14
000476  012716  000000G                         MOV     #MSG64,(SP)                     ;                                       1920
000502  012746  000001                         MOV     #1,-(SP)
000506  010600                                  MOV     SP,R0                           ; SP,*
000510  104414                                  TRAP    14
000512  012716  000000G                         MOV     #PHYS.ADR,(SP)                  ;                                       1921
000516  012746  000001                         MOV     #1,-(SP)
000522  010600                                  MOV     SP,R0                           ; SP,*
000524  104414                                  TRAP    14
000526  104455                                  TRAP    55                              ;                                       1922
000530  000456                                  .WORD   456
000532  000000G                                 .WORD   MSG00
000534  000000G                                 .WORD   E1$REPORT
000536  062706  000010                         ADD     #10,SP                          ;                                       1918
000542  062706  000006                 13$:    ADD     #6,SP                           ;                                       1800
000546  000207                                  RTS     PC

; Routine Size:  180 words,     Routine Base:  AB$CODE$ + 1306
; Maximum stack depth per invocation:  16 words



                                                .SBTTL  T3 TEST 3 - ETHERNET STATION ADDRESS VERIFY TEST
000000  004737  001306'                 T3::
000000                                  1$:     JSR     PC,$T3                          ;                                       1923
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 2056
; Maximum stack depth per invocation:  2 words
```

```
;    1926  1
;    1927  1
```

```
;   1928  1     %SBTTL 'TEST 4 - INTERRUPT VECTOR ADDRESS TEST'
;   1929  1     !++
;   1930  1     !
;   1931  1     !  TEST 4:      INTERRUPT VECTOR ADDRESS TEST
;   1932  1     !
;   1933  1     !  DESCRIPTION:
;   1934  1     !
;   1935  1     !       This test verifies that all bits of the vector address register
;   1936  1     !       can be set and cleared as specified.  The host writes data patterns
;   1937  1     !       to this register and reads them back verifying no static
;   1938  1     !       (stuck at 1 / stuck at 0) faults occur. If the operator specifies
;   1939  1     !       loop on error, the program re-executes the code that detected the
;   1940  1     !       error until ^C is entered.
;   1941  1     !
;   1942  1     !       NOTE:  Only bits 9:2 of the Interrupt Vector Address Register are
;   1943  1     !              valid, rest read as 0.
;   1944  1     !
;   1945  1     !       The following BINARY data patterns are used:
;   1946  1     !
;   1947  1     !                       00000000        11111111
;   1948  1     !                       10101010        01010101
;   1949  1     !                       11001100        00110011
;   1950  1     !                       11110000        00001111
;   1951  1     !                       walking 1's, 1 propagating thru Vector Address Reg.
;   1952  1     !                       walking 0's, 0 propagating thru Vector Address Reg.
;   1953  1     !
;   1954  1     !       Hardware tested:        Device Vector Address Register
;   1955  1     !                               Slave Interface Registers
;   1956  1     !
;   1957  1     !       Processing:
;   1958  1     !
;   1959  1     !            BEGIN
;   1960  1     !
;   1961  1     !               reset device
;   1962  1     !               REPEAT for each pattern
;   1963  1     !                  write pattern to Vector Address Register ( bits 9:2 )
;   1964  1     !                  read pattern from Vector Address Register ( bits 9:2 )
;   1965  1     !                  compare write pattern to read pattern (less noise bits)
;   1966  1     !                  IF not equal
;   1967  1     !                  THEN
;   1968  1     !                     print error message if not inhibited
;   1969  1     !                  ENDIF
;   1970  1     !
;   1971  1     !               ENDREPEAT
;   1972  1     !            END
;   1973  1     !--
```

```
;  1974  3     BGNTST;
;  1975  3
;  1976  3     RESET_DEQNA ( );
;  1977  3
;  1978  3     !++
;  1979  3     !  WRITE ALTERNATING 0'S AND 1'S TO INTERRUPT VECTOR ADDRESS REGISTER
;  1980  3     !  IN THE I/O PAGE, THEN READ AND COMPARE TO THE WRITE PATTERN
;  1981  3     !--
;  1982  3
;  1983  3     INCR INDEX FROM 0 TO 7 DO
;  1984  4       BEGIN
;  1985  4         TBYTE1 = .PTRN_TABLE [ .INDEX ];
;  1986  6         BGNSUB;
;  1987  6           PUT_BIT [ INT_VEC, VEC_ADR, .TBYTE1 ];
;  1988  6           IF GET_BIT [ INT_VEC, VEC_ADR ] NEQU .TBYTE1
;  1989  6             THEN
;  1990  7               BEGIN
;  1991  7                 PRINTB ( MSG59 );
;  1992  7                 PRINTB ( MSG65 );
;  1993  7                 PRINTB ( MSG30, .GET_ADR [ VEC_ALL ], .TBYTE1, GET_BIT [ INT_VEC, VEC_ADR ] );
;  1994  7                 ERRDF  ( 0401, MSG00, E1$REPORT );
;  1995  6               END;
;  1996  4         ENDSUB;
;  1997  3       END;
;  1998  3     !++
;  1999  3     !  WRITE WALKING 1 PATTERN INTO THE INTERRUPT VECTOR ADDRESS IN THE I/O PAGE
;  2000  3     !  REGISTER THEN READ AND COMPARE TO THE WRITE PATTERN
;  2001  3     !--
;  2002  3
;  2003  3     TEMP1 = %B'00000001';
;  2004  3
;  2005  3     INCR INDEX FROM 0 TO 7 DO
;  2006  4       BEGIN
;  2007  6         BGNSUB;
;  2008  6         PUT_BIT [ INT_VEC, VEC_ADR, .TEMP1 ];
;  2009  6         IF GET_BIT [ INT_VEC, VEC_ADR ] NEQU .TEMP1
;  2010  6           THEN
;  2011  7             BEGIN
;  2012  7                 PRINTB ( MSG59 );
;  2013  7                 PRINTB ( MSG65 );
;  2014  7                 PRINTB ( MSG30, .GET_ADR [ VEC_ALL ], .TEMP1, GET_BIT [ INT_VEC, VEC_ADR ] );
;  2015  7                 ERRDF  ( 0402, MSG00, E1$REPORT );
;  2016  6             END;
;  2017  6         TEMP1 = .TEMP1 ^ 1;
;  2018  4         ENDSUB;
;  2019  3       END;
;  2020  3
;  2021  3     !++
;  2022  3     !  WRITE WALKING 0 PATTERN INTO THE INTERRUPT VECTOR ADDRESS IN THE I/O PAGE
;  2023  3     !  REGISTER THEN READ AND COMPARE TO THE WRITE PATTERN
;  2024  3     !--
;  2025  3
;  2026  3     TEMP1 = %B'11111110';
```

```
;    2027  3
;    2028  3     INCR INDEX FROM 0 TO 7 DO
;    2029  4       BEGIN
;    2030  6         BGNSUB;
;    2031  6         PUT_BIT ( INT_VEC, VEC_ADR, .TEMP1 );
;    2032  6         IF GET_BIT ( INT_VEC, VEC_ADR ) NEQU .TEMP1
;    2033  6           THEN
;    2034  7             BEGIN
;    2035  7                 PRINTB ( MSG59 );
;    2036  7                 PRINTB ( MSG65 );
;    2037  7                 PRINTB ( MSG30, .GET_ADR [ VEC_ALL ], .TEMP1, GET_BIT [ INT_VEC, VEC_ADR ] );
;    2038  7                 ERRDF  ( 0403, MSG00, E1$REPORT );
;    2039  6             END;
;    2040  6
;    2041  6         TEMP1 = (( .TEMP1 + 1 ) + 1 ) AND %O'000377' ;
;    2042  4         ENDSUB;
;    2043  3       END;
;    2044  3
;    2045  1     ENDTST;


                                                .SBTTL  $T4 TEST 4 - INTERRUPT VECTOR ADDRESS TEST
000000  004137  000000G                 $T4:    JSR     R1,$SAVE2                       ;                               1925
000004  162706  000022                          SUB     #22,SP
000010  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                               1976
000014  005001                                  CLR     R1                              ; INDEX                         1983
000016  116137  000000G 000000G         1$:     MOVB    PTRN.TABLE(R1),TBYTE1           ; *(INDEX),*                    1985
000024  105037  000001G                         CLRB    TBYTE1+1
000030  104402                          2$:     TRAP    2
000032  013700  000000G                         MOV     REG.ADR,R0                      ;                               1987
000036  013702  000000G                         MOV     TBYTE1,R2
000042  006302                                  ASL     R2
000044  006302                                  ASL     R2
000046  042702  176003                         BIC     #176003,R2
000052  042760  001774  000014               BIC     #1774,14(R0)
000060  050260  000014                         BIS     R2,14(R0)
000064  016016  000014                         MOV     14(R0),(SP)                     ; *,TMP.LOCATION                1988
000070  013702  000000G                         MOV     TBYTE1,R2
000074  011600                                  MOV     (SP),R0                         ; TMP.LOCATION,*
000076  006200                                  ASR     R0
000100  006200                                  ASR     R0
000102  042700  177400                         BIC     #177400,R0
000106  020002                                  CMP     R0,R2
000110  001456                                  BEQ     3$
000112  012746  000000G                         MOV     #MSG59,-(SP)                    ;                               1991
000116  012746  000001                         MOV     #1,-(SP)
000122  010600                                  MOV     SP,R0                           ; SP,*
000124  104414                                  TRAP    14
000126  012716  000000G                         MOV     #MSG65,(SP)                     ;                               1992
000132  012746  000001                         MOV     #1,-(SP)
000136  010600                                  MOV     SP,R0                           ; SP,*
000140  104414                                  TRAP    14
000142  013700  000000G                         MOV     REG.ADR,R0                      ;                               1993
```

```
000146  016066  000014  000010              MOV     14(R0),10(SP)           ; *,TMP.LOCATION
000154  016600  000010                      MOV     10(SP),R0               ; TMP.LOCATION,*
000160  006200                              ASR     R0
000162  006200                              ASR     R0
000164  042700  177400                      BIC     #177400,R0
000170  010016                              MOV     R0,(SP)
000172  013746  000000G                     MOV     TBYTE1,-(SP)
000176  013766  000000G 000014              MOV     GET.ADR,14(SP)          ; *,TMP.LOCATION
000204  062766  000014  000014              ADD     #14,14(SP)              ; *,TMP.LOCATION
000212  016646  000014                      MOV     14(SP),-(SP)            ; TMP.LOCATION,*
000216  012746  000000G                     MOV     #MSG30,-(SP)
000222  012746  000004                      MOV     #4,-(SP)
000226  010600                              MOV     SP,R0                   ; SP,*
000230  104414                              TRAP    14
000232  104455                              TRAP    55                      ;                                       1994
000234  000621                              .WORD   621
000236  000000G                             .WORD   MSG00
000240  000000G                             .WORD   E1$REPORT
000242  062706  000016                      ADD     #16,SP                  ;                                       1990
000246  104467                      3$:     TRAP    67                      ;                                       1995
000250  006000                              ROR     R0
000252  103666                              BLO     2$
000254  005201                              INC     R1                      ; INDEX                                 1983
000256  020127  000007                      CMP     R1,#7                   ; INDEX,*
000262  003655                              BLE     1$
000264  012737  000001  000000G             MOV     #1,TEMP1                ;                                       2003
000272  012701  000010                      MOV     #10,R1                  ; *,INDEX                               2005
000276  104402                      4$:     TRAP    2                       ;                                       2006
000300  013700  000000G                     MOV     REG.ADR,R0              ;                                       2008
000304  013702  000000G                     MOV     TEMP1,R2
000310  006302                              ASL     R2
000312  006302                              ASL     R2
000314  042702  176003                      BIC     #176003,R2
000320  042760  001774  000014              BIC     #1774,14(R0)
000326  050260  000014                      BIS     R2,14(R0)
000332  016066  000014  000006              MOV     14(R0),6(SP)            ; *,TMP.LOCATION                        2009
000340  013702  000000G                     MOV     TEMP1,R2
000344  016600  000006                      MOV     6(SP),R0                ; TMP.LOCATION,*
000350  006200                              ASR     R0
000352  006200                              ASR     R0
000354  042700  177400                      BIC     #177400,R0
000360  020002                              CMP     R0,R2
000362  001456                              BEQ     5$
000364  012746  000000G                     MOV     #MSG59,-(SP)            ;                                       2012
000370  012746  000001                      MOV     #1,-(SP)
000374  010600                              MOV     SP,R0                   ; SP,*
000376  104414                              TRAP    14
000400  012716  000000G                     MOV     #MSG65,(SP)             ;                                       2013
000404  012746  000001                      MOV     #1,-(SP)
000410  010600                              MOV     SP,R0                   ; SP,*
000412  104414                              TRAP    14
000414  013700  000000G                     MOV     REG.ADR,R0              ;                                       2014
000420  016066  000014  000016              MOV     14(R0),16(SP)           ; *,TMP.LOCATION
```

```
000426  016600  000016                          MOV     16(SP),R0                       ; TMP.LOCATION,*
000432  006200                                  ASR     R0
000434  006200                                  ASR     R0
000436  042700  177400                          BIC     #177400,R0
000442  010016                                  MOV     R0,(SP)
000444  013746  000000G                         MOV     TEMP1,-(SP)
000450  013766  000000G 000022                  MOV     GET.ADR,22(SP)                  ; *,TMP.LOCATION
000456  062766  000014  000022                  ADD     #14,22(SP)                      ; *,TMP.LOCATION
000464  016646  000022                          MOV     22(SP),-(SP)                    ; TMP.LOCATION,*
000470  012746  000000G                         MOV     #MSG30,-(SP)
000474  012746  000004                          MOV     #4,-(SP)
000500  010600                                  MOV     SP,R0                           ; SP,*
000502  104414                                  TRAP    14
000504  104455                                  TRAP    55                              ;                                       2015
000506  000622                                  .WORD   622
000510  000000G                                 .WORD   MSG00
000512  000000G                                 .WORD   E1$REPORT
000514  062706  000016                          ADD     #16,SP                          ;                                       2011
000520  006337  000000G             5$:         ASL     TEMP1                           ;                                       2017
000524  104467                                  TRAP    67
000526  006000                                  ROR     R0
000530  103662                                  BLO     4$
000532  005301                                  DEC     R1                              ; INDEX                                 2005
000534  001260                                  BNE     4$
000536  012737  000376  000000G                 MOV     #376,TEMP1                      ;                                       2026
000544  012701  000010                          MOV     #10,R1                          ; *,INDEX                               2028
000550  104402                     6$:         TRAP    2                               ;                                       2029
000552  013700  000000G                         MOV     REG.ADR,R0                      ;                                       2031
000556  013702  000000G                         MOV     TEMP1,R2
000562  006302                                  ASL     R2
000564  006302                                  ASL     R2
000566  042702  176003                          BIC     #176003,R2
000572  042760  001774  000014                  BIC     #1774,14(R0)
000600  050260  000014                          BIS     R2,14(R0)
000604  016066  000014  000014                  MOV     14(R0),14(SP)                   ; *,TMP.LOCATION                        2032
000612  013702  000000G                         MOV     TEMP1,R2
000616  016600  000014                          MOV     14(SP),R0                       ; TMP.LOCATION,*
000622  006200                                  ASR     R0
000624  006200                                  ASR     R0
000626  042700  177400                          BIC     #177400,R0
000632  020002                                  CMP     R0,R2
000634  001456                                  BEQ     7$
000636  012746  000000G                         MOV     #MSG59,-(SP)                    ;                                       2035
000642  012746  000001                          MOV     #1,-(SP)
000646  010600                                  MOV     SP,R0                           ; SP,*
000650  104414                                  TRAP    14
000652  012716  000000G                         MOV     #MSG65,(SP)                     ;                                       2036
000656  012746  000001                          MOV     #1,-(SP)
000662  010600                                  MOV     SP,R0                           ; SP,*
000664  104414                                  TRAP    14
000666  013700  000000G                         MOV     REG.ADR,R0                      ;                                       2037
000672  016066  000014  000024                  MOV     14(R0),24(SP)                   ; *,TMP.LOCATION
000700  016600  000024                          MOV     24(SP),R0                       ; TMP.LOCATION,*
```

```
000704  006200                                  ASR     R0
000706  006200                                  ASR     R0
000710  042700  177400                          BIC     #177400,R0
000714  010016                                  MOV     R0,(SP)
000716  013746  000000G                         MOV     TEMP1,-(SP)
000722  013766  000000G 000030                  MOV     GET.ADR,30(SP)          ; *,TMP.LOCATION
000730  062766  000014  000030                  ADD     #14,30(SP)              ; *,TMP.LOCATION
000736  016646  000030                          MOV     30(SP),-(SP)            ; TMP.LOCATION,*
000742  012746  000000G                         MOV     #MSG30,-(SP)
000746  012746  000004                          MOV     #4,-(SP)
000752  010600                                  MOV     SP,R0                   ; SP,*
000754  104414                                  TRAP    14
000756  104455                                  TRAP    55                      ;                                       2038
000760  000623                                  .WORD   623
000762  000000G                                 .WORD   MSG00
000764  000000G                                 .WORD   E1$REPORT
000766  062706  000016                          ADD     #16,SP                  ;                                       2034
000772  013700  000000G             7$:         MOV     TEMP1,R0                ;                                       2041
000776  006300                                  ASL     R0
001000  005200                                  INC     R0
001002  005037  000000G                         CLR     TEMP1
001006  110037  000000G                         MOVB    R0,TEMP1
001012  104467                                  TRAP    67
001014  006000                                  ROR     R0
001016  103654                                  BLO     6$
001020  005301                                  DEC     R1                      ; INDEX                                 2028
001022  001252                                  BNE     6$
001024  062706  000022                          ADD     #22,SP                  ;                                       1925
001030  000207                                  RTS     PC

; Routine Size:  269 words,     Routine Base:  AB$CODE$ + 2072
; Maximum stack depth per invocation:  21 words



                                                .SBTTL  T4 TEST 4 - INTERRUPT VECTOR ADDRESS TEST
000000  004737  002072'              T4::
000000                               1$:        JSR     PC,$T4                  ;                                       2043
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 3124
; Maximum stack depth per invocation:  2 words


;   2046  1
```

```
;    2047  1     %SBTTL 'TEST 5 - BOOT/DIAGNOSTIC PROM CHECKSUM TEST'
;    2048  1     !++
;    2049  1     !
;    2050  1     !  TEST 5:      BOOT/DIAGNOSTIC PROM CHECKSUM TEST
;    2051  1     !
;    2052  1     !  DESCRIPTION:
;    2053  1     !
;    2054  1     !       This test verifies that the contents of the on-board ROM
;    2055  1     !       (Boot/Diagnostic ROM) can be loaded to the host memory correctly.
;    2056  1     !       Checksum is generated from the ROM data read and this checksum is
;    2057  1     !       compared to the checksum stored in the last word location of the
;    2058  1     !       on-board ROM. If the operator specifies loop on error, the program
;    2059  1     !       re-executes the code that detected the  error until ↑C is entered.
;    2060  1     !
;    2061  1     !
;    2062  1     !       Hardware tested:        Q-Bus to DMA interface
;    2063  1     !                               I8051 microprocessor
;    2064  1     !                               I8051 ROM
;    2065  1     !                               CSR register
;    2066  1     !                               Receive FIFO
;    2067  1     !       Processing:
;    2068  1     !
;    2069  1     !            BEGIN
;    2070  1     !               reset device
;    2071  1     !               setup Receive Descriptor List(s)
;    2072  1     !               set Boot/Diagnostic ROM and External loopback bits
;    2073  1     !                   This moves ROM boot code into receive FIFO
;    2074  1     !               wait 10 msec. or until RL ( bit 5 in CSR ) = 0
;    2075  1     !               check CSR status ( bit 5 ) and RCV Descriptor List status
;    2076  1     !               IF error
;    2077  1     !               THEN
;    2078  1     !                  print error message if not inhibited
;    2079  1     !               ENDIF
;    2080  1     !               clear Boot/Diagnostic ROM bit in CSR
;    2081  1     !                   This moves contents of FIFO to host memory
;    2082  1     !               wait 10 msec. or until RCV Descriptor status changed
;    2083  1     !               IF change in status
;    2084  1     !               THEN
;    2085  1     !                  print error message if not inhibited
;    2086  1     !               ENDIF
;    2087  1     !               compute ROM checksum and compare to checksum read from ROM
;    2088  1     !               IF not equal
;    2089  1     !               THEN
;    2090  1     !                  print error message if not inhibited
;    2091  1     !               ENDIF
;    2092  1     !            END
;    2093  1     !
```

```
;  2094  3     BGNTST;
;  2095  3
;  2096  3     RESET_DEQNA ( );
;  2097  3     CLR_BUFFERS ( 2 * K );
;  2098  3
;  2099  3     !++
;  2100  3     !   COPY BOOT/DIAGNOSTIC PROM DESCRIPTOR LIST INTO WORK AREA
;  2101  3     !--
;  2102  3
;  2103  3     INCR INDEX FROM 0 TO BD_D_SIZE - 1 DO
;  2104  3       DESCR_LIST [ .INDEX, W_LEN ] = .BD_PROM_DESCR [ .INDEX ];
;  2105  3
;  2106  3     .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;  2107  3     .IOP_TABLE [ RHI_ADR ] = 0;
;  2108  3
;  2109  3     PUT_BIT ( CSR, LB, EXT_LOOPBACK );
;  2110  3     PUT_BIT ( CSR, BD, SET_IT );
;  2111  3
;  2112  3     DELAY ( K );
;  2113  3     INCR INDEX FROM 0 TO TIME3_LIMIT DO
;  2114  3       IF GET_BIT [ CSR, RL ] EQLU ZERO
;  2115  3         THEN
;  2116  4           BEGIN
;  2117  4             TEMP1 = .INDEX;
;  2118  4             EXITLOOP;
;  2119  4           END
;  2120  3         ELSE
;  2121  3           IF .INDEX EQLU TIME3_LIMIT
;  2122  3             THEN
;  2123  4               BEGIN
;  2124  4                 PRINTB ( MSG59 );
;  2125  4                 PRINTB ( MSG66, GET_BIT [ CSR_ALL ] );
;  2126  4                 ERRDF ( 0501, MSG00, ERROR$REPORT );
;  2127  3               END;
;  2128  3
;  2129  3     VER_DESCR_STATUS ( );
;  2130  3
;  2131  3     !++
;  2132  3     !  FINISH BOOT/DIAGNOSTIC PROM UPLOAD
;  2133  3     !--
;  2134  3
;  2135  3     PUT_BIT ( CSR, BD, CLR_IT );
;  2136  3     DELAY ( K );
;  2137  3
;  2138  3     !++
;  2139  3     !  CHECK IF RECEIVE STATUS CHANGED
;  2140  3     !--
;  2141  3
;  2142  3     VER_DESCR_STATUS ( );
;  2143  3
;  2144  3     RESET_DEQNA ( );
;  2145  3
;  2146  3     TEMP3 = 0;
```

```
;    2147  3     TEMP3 = .DATA_BUFFER [ CHSUM_OFFSET + 1 ];
;    2148  3     TEMP3 = ( .TEMP3 ^ 8 ) AND %X'FF00';
;    2149  3     TEMP3 = .DATA_BUFFER [ CHSUM_OFFSET ] + .TEMP3;
;    2150  3
;    2151  3     TEMP2 = .DATA_BUFFER [ .TEMP3 + 1 ];
;    2152  3     TEMP2 = ( .TEMP2 ^ 8 ) AND %X'FF00';
;    2153  3     TEMP2 = .DATA_BUFFER [ .TEMP3 ] + .TEMP2;
;    2154  3
;    2155  3     COUNTER  = 0;
;    2156  3     CHECKSUM = 0;
;    2157  3
;    2158  3     INCR INDEX FROM 0 TO PROM_SIZE - 2 DO
;    2159  3       IF .COUNTER EQLU .TEMP3
;    2160  3         THEN
;    2161  3           COUNTER = .COUNTER + 2
;    2162  3         ELSE
;    2163  4           BEGIN
;    2164  4             CHECKSUM = .CHECKSUM + ( .DATA_BUFFER [ .COUNTER ] AND %X'FF' );
;    2165  4             COUNTER = .COUNTER + 1;
;    2166  3           END;
;    2167  3
;    2168  4     IF ( .TEMP2 EQLU ZERO ) OR ( .TEMP2 NEQU .CHECKSUM )
;    2169  3       THEN
;    2170  4         BEGIN
;    2171  4           CSR_WORD = GET_UIT ( CSR_ALL );
;    2172  4           PRINTB ( MSG59 );
;    2173  4           PRINTB ( MSG67, .TEMP3, .TEMP2, .CHECKSUM );
;    2174  4           ERRDF  ( 0502, MSG00, E1$REPORT);
;    2175  3         END;
;    2176  3
;    2177  1     ENDTST;


                                        .SBTTL  $T5 TEST 5 - BOOT/DIAGNOSTIC PROM CHECKSUM TEST
000000  004137  000000G           $T5:  JSR     R1,$SAVE3                       ;                                   2045
000004  162706  000010                  SUB     #10,SP
000010  004737  000000G                 JSR     PC,RESET.DEQNA                  ;                                   2096
000014  012746  004000                  MOV     #4000,-(SP)                     ;                                   2097
000020  004737  000000G                 JSR     PC,CLR.BUFFERS
000024  005000                          CLR     R0                              ; INDEX                             2103
000026  016060  000000G 000000G   1$:   MOV     BD.PROM.DESCR(R0),DESCR.LIST(R0); *(INDEX),*(INDEX)                 2104
000034  062700  000002                  ADD     #2,R0                           ; *,INDEX                           2103
000040  020027  000036                  CMP     R0,#36                          ; INDEX,*
000044  003770                          BLE     1$
000046  012777  000000G 000004G         MOV     #RCV.D.LIST,@IOP.TABLE+4        ;                                   2106
000054  005077  000006G                 CLR     @IOP.TABLE+6                    ;                                   2107
000060  013700  000000G                 MOV     REG.ADR,R0                      ;                                   2109
000064  052760  001410  000016         BIS     #1410,16(R0)                    ;                                   2110
000072  012701  002000                  MOV     #2000,R1                        ; *,$$TMP2                          2112
000076  001410                  2$:     BEQ     5$
000100  013700  000000G                 MOV     L$DLY,R0                        ; *,$$TMP1
000104  001403                          BEQ     4$
000106  005066  000010          3$:     CLR     10(SP)                          ; $$TMP
```

```
000112  077003                          SOB     R0,3$                   ; $$TMP1,*
000114  005301                  4$:     DEC     R1                      ; $$TMP2
000116  000767                          BR      2$
000120  005001                  5$:     CLR     R1                      ; INDEX                         2113
000122  013700  000000G         6$:     MOV     REG.ADR,R0              ;                               2114
000126  016066  000016  000002          MOV     16(R0),2(SP)            ; *,TMP.LOCATION
000134  032766  000040  000002          BIT     #40,2(SP)               ; *,TMP.LOCATION
000142  001003                          BNE     7$
000144  010137  000000G                 MOV     R1,TEMP1                ; INDEX,*                       2117
000150  000440                          BR      9$                      ;                               2116
000152  020127  002000          7$:     CMP     R1,#2000                ; INDEX,*                       2121
000156  001031                          BNE     8$
000160  012716  000000G                 MOV     #MSG59,(SP)             ;                               2124
000164  012746  000001                  MOV     #1,-(SP)
000170  010600                          MOV     SP,R0                   ; SP,*
000172  104414                          TRAP    14
000174  013700  000000G                 MOV     REG.ADR,R0              ;                               2125
000200  016066  000016  000006          MOV     16(R0),6(SP)            ; *,TMP.LOCATION
000206  016616  000006                  MOV     6(SP),(SP)              ; TMP.LOCATION,*
000212  012746  000000G                 MOV     #MSG66,-(SP)
000216  012746  000002                  MOV     #2,-(SP)
000222  010600                          MOV     SP,R0                   ; SP,*
000224  104414                          TRAP    14
000226  104455                          TRAP    55                      ;                               2126
000230  000765                          .WORD   765
000232  000000G                         .WORD   MSG00
000234  000000G                         .WORD   ERROR$REPORT
000236  062706  000006                  ADD     #6,SP                   ;                               2123
000242  005201                  8$:     INC     R1                      ; INDEX                         2113
000244  020127  002000                  CMP     R1,#2000                ; INDEX,*
000250  003724                          BLE     6$
000252  004737  000000G         9$:     JSR     PC,VER.DESCR.STATUS     ;                               2129
000256  013700  000000G                 MOV     REG.ADR,R0              ;                               2135
000262  142760  000010  000016          BICB    #10,16(R0)
000270  012701  002000                  MOV     #2000,R1                ; *,$$TMP2                      2136
000274  001410                  10$:    BEQ     13$
000276  013700  000000G                 MOV     L$DLY,R0                ; *,$$TMP1
000302  001403                          BEQ     12$
000304  005066  000010          11$:    CLR     10(SP)                  ; $$TMP
000310  077003                          SOB     R0,11$                  ; $$TMP1,*
000312  005301                  12$:    DEC     R1                      ; $$TMP2
000314  000767                          BR      10$
000316  004737  000000G         13$:    JSR     PC,VER.DESCR.STATUS     ;                               2142
000322  004737  000000G                 JSR     PC,RESET.DEQNA          ;                               2144
000326  005037  000000G                 CLR     TEMP3                   ;                               2147
000332  113737  000007G 000000G         MOVB    DATA.BUFFER+7,TEMP3
000340  013700  000000G                 MOV     TEMP3,R0                ;                               2148
000344  072027  000010                  ASH     #10,R0
000350  010037  000000G                 MOV     R0,TEMP3
000354  042737  000377  000000G         BIC     #377,TEMP3
000362  005000                          CLR     R0                      ;                               2149
000364  153700  000006G                 BISB    DATA.BUFFER+6,R0
000370  060037  000000G                 ADD     R0,TEMP3
```

```
ZQNA3           CZQNADO DEQNA FUNCTIONAL TEST                    14-Mar-1985 13:11:16    VAX-11 Bliss-16 V4.1-582              SEQ 0116
V01.0           TEST 5 - BOOT/DIAGNOSTIC PROM CHECKSUM TEST      14-Mar-1985 13:05:35    DISK$USER2:[MARSHALL.DEQNA]ZQNA3.BLI;4  Page  33
                                                                                                                                  (15)

000374  013701  000000G                         MOV     TEMP3,R1                ;                                               2151
000400  116137  000001G 000000G                 MOVB    DATA.BUFFER+1(R1),TEMP2
000406  105037  000001G                         CLRB    TEMP2+1
000412  013700  000000G                         MOV     TEMP2,R0                ;                                               2152
000416  072027  000010                          ASH     #10,R0
000422  010037  000000G                         MOV     R0,TEMP2
000426  042737  000377  000000G                 BIC     #377,TEMP2
000434  005000                                  CLR     R0                      ;                                               2153
000436  156100  000000G                         BISB    DATA.BUFFER(R1),R0
000442  060037  000000G                         ADD     R0,TEMP2
000446  005037  000000G                         CLR     COUNTER                 ;                                               2155
000452  005037  000000G                         CLR     CHECKSUM                ;                                               2156
000456  012702  007777                          MOV     #7777,R2                ; *,INDEX                                       2158
000462  013700  000000G             14$:        MOV     COUNTER,R0              ;                                               2159
000466  020001                                  CMP     R0,R1
000470  001004                                  BNE     15$
000472  062737  000002  000000G                 ADD     #2,COUNTER              ;                                               2161
000500  000407                                  BR      16$                     ;                                               2159
000502  005003                 15$:             CLR     R3                      ;                                               2164
000504  156003  000000G                         BISB    DATA.BUFFER(R0),R3
000510  060337  000000G                         ADD     R3,CHECKSUM
000514  005237  000000G                         INC     COUNTER                 ;                                               2165
000520  077220                 16$:             SOB     R2,14$                  ; INDEX,*                                       2158
000522  013700  000000G                         MOV     TEMP2,R0                ;                                               2168
000526  001403                                  BEQ     17$
000530  020037  000000G                         CMP     R0,CHECKSUM
000534  001440                                  BEQ     18$
000536  013700  000000G             17$:        MOV     REG.ADR,R0              ;                                               2171
000542  016066  000016  000006                  MOV     16(R0),6(SP)            ; *,TMP.LOCATION
000550  016637  000006  000000G                 MOV     6(SP),CSR.WORD          ; TMP.LOCATION,*
000556  012716  000000G                         MOV     #MSG59,(SP)             ;                                               2172
000562  012746  000001                          MOV     #1,-(SP)
000566  010600                                  MOV     SP,R0                   ; SP,*
000570  104414                                  TRAP    14
000572  013716  000000G                         MOV     CHECKSUM,(SP)           ;                                               2173
000576  013746  000000G                         MOV     TEMP2,-(SP)
000602  013746  000000G                         MOV     TEMP3,-(SP)
000606  012746  000000G                         MOV     #MSG67,-(SP)
000612  012746  000004                          MOV     #4,-(SP)
000616  010600                                  MOV     SP,R0                   ; SP,*
000620  104414                                  TRAP    14
000622  104455                                  TRAP    55                      ;                                               2174
000624  000766                                  .WORD   766
000626  000000G                                 .WORD   MSG00
000630  000000G                                 .WORD   E1$REPORT
000632  062706  000012                          ADD     #12,SP                  ;                                               2170
000636  062706  000012             18$:        ADD     #12,SP                  ;                                               2045
000642  000207                                  RTS     PC

; Routine Size:  210 words,     Routine Base:  AB$CODE$ + 3140
; Maximum stack depth per invocation:  16 words
```

```
                                        .SBTTL  T5 TEST 5 - BOOT/DIAGNOSTIC PROM CHECKSUM TEST

000000  004737  003140'                T5::
000000                                  1$:     JSR     PC,$T5                  ;                                       2175
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 4004
; Maximum stack depth per invocation:  2 words


;   2178  1
```

```
;   2179  1     %SBTTL 'TEST 6 - INTERRUPT SANITY TEST'
;   2180  1     !++
;   2181  1     !
;   2182  1     !  TEST 6:      INTERRUPT SANITY TEST
;   2183  1     !
;   2184  1     !  DESCRIPTION:
;   2185  1     !
;   2186  1     !       This test verifies that DEQNA interrupts the processor only at
;   2187  1     !       the expected level ( 4 ) and not any other level. If the operator
;   2188  1     !       specifies loop on error, the program re-executes the code that
;   2189  1     !       detected the error until ^C is entered.
;   2190  1     !
;   2191  1     !       Hardware tested:        Q-Bus to QTDC interface
;   2192  1     !                               CSR register
;   2193  1     !                               Q-Bus timeout logic
;   2194  1     !                               QTDC interrupt logic
;   2195  1     !       Processing:
;   2196  1     !
;   2197  1     !            BEGIN
;   2198  1     !               reset device
;   2199  1     !               set-up for NXM interrupt
;   2200  1     !               REPEAT for each processor priority level
;   2201  1     !                  enable device interrupt (set CSR bit 6)
;   2202  1     !                  force NXM interrupt
;   2203  1     !                  check for expected CSR status
;   2204  1     !                  IF error
;   2205  1     !                  THEN
;   2206  1     !                     print error message if not inhibited
;   2207  1     !                  ENDIF
;   2208  1     !               ENDREPEAT
;   2209  1     !            END
;   2210  1     !--
```

```
;    2211  1
;    2212  3     BGNTST;
;    2213  3
;    2214  3     RESET_DEQNA ( );
;    2215  3     SETVEC ( .HWP_TABLE [ VEC ], NXM_INT, PRI07 );  ! SET UP FOR AN NXM INTERRUPT
;    2216  3     .IOP_TABLE [ INT_VEC ] = .HWP_TABLE [ VEC ];
;    2217  3     TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;    2218  3     COUNTER = 0;
;    2219  3
;    2220  3     INCR PRIORITY FROM PRI00 TO PRI07 BY %0'40' DO
;    2221  4       BEGIN
;    2222  4         SETPRI ( .PRIORITY );                       ! SET PROCESSOR PRI LEVEL
;    2223  6         BGNSUB;
;    2224  6           PUT_BIT ( CSR, IE, SET_IT );              ! ENABLE INTERRUPTS
;    2225  6           DELAY ( 5 );                              !
;    2226  6           INTERRUPT_FLG = CLEAR_FLG;
;    2227  6
;    2228  6           .IOP_TABLE [ XLO_ADR ] = NXM_LO_ADR;      ! WRITE LOW ADDRESS
;    2229  6           .IOP_TABLE [ XHI_ADR ] = NXM_HI_ADR;      ! WRITE HIGH ADDRESS
;    2230  6
;    2231  6           DELAY ( 2 );
;    2232  6           GETPRI ( TEMP1 );
;    2233  6           TEMP1 = .TEMP1 ^ ( - 5 );
;    2234  6
;    2235  6           IF .INTERRUPT_FLG EQLU WORD_LIMIT
;    2236  6             THEN                                    ! INTERRUPT SHOULD NOT OCCUR
;    2237  6               IF .PRIORITY GTRU PRI03
;    2238  6                 THEN
;    2239  7                   BEGIN
;    2240  7                     PRINTB ( MSG59 );
;    2241  7                     PRINTB ( MSG69, .TMP_IOP_ADR, .TEMP1, .COUNTER );
;    2242  7                     ERRDF  ( 0601, MSG00, E1$REPORT );
;    2243  6                   END;
;    2244  6
;    2245  6           IF .INTERRUPT_FLG EQLU ZERO
;    2246  6             THEN                                    ! INTERRUPT SHOULD OCCUR
;    2247  6               IF .PRIORITY LEQU PRI03
;    2248  6                 THEN
;    2249  6                   IF ( .XMIT_D_LIST [ FLGWD ] AND XFLG_MASK ) NEQU XFLG_MASK
;    2250  6                     THEN
;    2251  7                       BEGIN
;    2252  7                         PRINTB ( MSG59 );
;    2253  7                         PRINTB ( MSG69, .TMP_IOP_ADR, .TEMP1, .COUNTER );
;    2254  7                         ERRDF  ( 0602, MSG00, E1$REPORT );
;    2255  6                       END;
;    2256  6             RESET_DEQNA ( );
;    2257  4         ENDSUB;
;    2258  4         COUNTER = .COUNTER + 1;
;    2259  3       END;
;    2260  3
;    2261  3     SETPRI ( PRI03 );                               ! SET PROCESSOR PRI LEVEL
;    2262  3
;    2263  1     ENDTST;
```

```
                                                .SBTTL  $T6 TEST 6 - INTERRUPT SANITY TEST
000000  004137  000000G                 $T6:    JSR     R1,$SAVE2                       ;                       2177
000004  005746                                  TST     -(SP)
000006  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                       2214
000012  012746  000000G                         MOV     @PRI07,-(SP)                    ;                       2215
000016  012746  000000G                         MOV     @NXM.INT,-(SP)
000022  013700  000000G                         MOV     HWP.TABLE,R0
000026  016046  000002                          MOV     2(R0),-(SP)
000032  012746  000003                          MOV     #3,-(SP)
000036  104437                                  TRAP    37
000040  013700  000000G                         MOV     HWP.TABLE,R0                    ;                       2216
000044  016077  000002  000014G                 MOV     2(R0),@IOP.TABLE+14
000052  017737  000000G 000000G                 MOV     @HWP.TABLE,TMP.IOP.ADR          ;                       2217
000060  005037  000000G                         CLR     COUNTER                         ;                       2218
000064  012702  000000G                         MOV     @PRI00,R2                       ; *,PRIORITY            2220
000070  000570                                  BR      13$
000072  010200                          1$:     MOV     R2,R0                           ; PRIORITY,*            2222
000074  104441                                  TRAP    41
000076  104402                          2$:     TRAP    2
000100  013700  000000G                         MOV     REG.ADR,R0                      ;                       2224
000104  152760  000100  000016                 BISB    #100,16(R0)
000112  012701  000005                          MOV     #5,R1                           ; *,$$TMP2              2225
000116  001410                          3$:     BEQ     6$
000120  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000124  001403                                  BEQ     5$
000126  005066  000010                  4$:     CLR     10(SP)                          ; $$TMP
000132  077003                                  SOB     R0,4$                           ; $$TMP1,*
000134  005301                          5$:     DEC     R1                              ; $$TMP2
000136  000767                                  BR      3$
000140  005037  000000G                 6$:     CLR     INTERRUPT.FLG                   ;                       2226
000144  012777  160000  000010G                 MOV     @-20000,@IOP.TABLE+10           ;                       2228
000152  012777  000077  000012G                 MOV     @77,@IOP.TABLE+12               ;                       2229
000160  012701  000002                          MOV     @2,R1                           ; *,$$TMP2              2231
000164  001410                          7$:     BEQ     10$
000166  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000172  001403                                  BEQ     9$
000174  005066  000010                  8$:     CLR     10(SP)                          ; $$TMP
000200  077003                                  SOB     R0,8$                           ; $$TMP1,*
000202  005301                          9$:     DEC     R1                              ; $$TMP2
000204  000767                                  BR      7$
000206  104440                          10$:    TRAP    40                              ;                       2232
000210  072027  177773                          ASH     @-5,R0                          ;                       2233
000214  010037  000000G                         MOV     R0,TEMP1
000220  023727  000000G 177777                  CMP     INTERRUPT.FLG,@-1               ;                       2235
000226  001033                                  BNE     11$
000230  020227  000000G                         CMP     R2,@PRI03                       ; PRIORITY,*            2237
000234  101430                                  BLOS    11$
000236  012716  000000G                         MOV     @MSG59,(SP)                     ;                       2240
000242  012746  000001                          MOV     @1,-(SP)
000246  010600                                  MOV     SP,R0                           ; SP,*
000250  104414                                  TRAP    14
```

```
000252  013716  000000G                        MOV     COUNTER,(SP)                    ;                               2241
000256  013746  000000G                        MOV     TEMP1,-(SP)
000262  013746  000000G                        MOV     TMP.IOP.ADR,-(SP)
000266  012746  000000G                        MOV     #MSG69,-(SP)
000272  012746  000004                         MOV     #4,-(SP)
000276  010600                                MOV     SP,R0                           ; SP,*
000300  104414                                TRAP    14
000302  104455                                TRAP    55                              ;                               2242
000304  001131                                .WORD   1131
000306  000000G                               .WORD   MSG00
000310  000000G                               .WORD   E1$REPORT
000312  062706  000012                        ADD     #12,SP                          ;                               2239
000316  005737  000000G               11$:     TST     INTERRUPT.FLG                   ;                               2245
000322  001042                                BNE     12$
000324  020227  000000G                        CMP     R2,#PRI03                       ; PRIORITY,*                    2247
000330  101037                                BHI     12$
000332  013700  000000G                        MOV     XMIT.D.LIST,R0                  ;                               2249
000336  042700  037777                        BIC     #37777,R0
000342  020027  140000                        CMP     R0,#-40000
000346  001430                                BEQ     12$
000350  012716  000000G                        MOV     #MSG59,(SP)                     ;                               2252
000354  012746  000001                        MOV     #1,-(SP)
000360  010600                                MOV     SP,R0                           ; SP,*
000362  104414                                TRAP    14
000364  013716  000000G                        MOV     COUNTER,(SP)                    ;                               2253
000370  013746  000000G                        MOV     TEMP1,-(SP)
000374  013746  000000G                        MOV     TMP.IOP.ADR,-(SP)
000400  012746  000000G                        MOV     #MSG69,-(SP)
000404  012746  000004                         MOV     #4,-(SP)
000410  010600                                MOV     SP,R0                           ; SP,*
000412  104414                                TRAP    14
000414  104455                                TRAP    55                              ;                               2254
000416  001132                                .WORD   1132
000420  000000G                               .WORD   MSG00
000422  000000G                               .WORD   E1$REPORT
000424  062706  000012                        ADD     #12,SP                          ;                               2251
000430  004737  000000G               12$:     JSR     PC,RESET.DEQNA                  ;                               2256
000434  104467                                TRAP    67
000436  006000                                ROR     R0
000440  103616                                BLO     2$
000442  005237  000000G                        INC     COUNTER                         ;                               2258
000446  062702  000040                        ADD     #40,R2                          ; *,PRIORITY                    2220
000452  020227  000000G               13$:     CMP     R2,#PRI07                       ; PRIORITY,*
000456  003605                                BLE     1$
000460  012700  000000G                        MOV     #PRI03,R0                       ;                               2261
000464  104441                                TRAP    41
000466  062706  000012                        ADD     #12,SP                          ;                               2177
000472  000207                                RTS     PC

; Routine Size:  158 words,     Routine Base:  AB$CODE$ + 4020
; Maximum stack depth per invocation:  15 words
```

```
                                        .SBTTL  T6 TEST 6 - INTERRUPT SANITY TEST
000000  004737  004020'            T6::
000000                              1$:     JSR     PC,$T6                          ;                       2261
000004  104466                              TRAP    66
000006  006000                              ROR     R0
000010  103773                              BLO     1$
000012  000207                              RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 4514
; Maximum stack depth per invocation:  2 words


;   2264  1
```

```
;  2265  1     %SBTTL 'TEST 7   ETHERNET CARRIER SENSE TEST'
;  2266  1     !++
;  2267  1     !
;  2268  1     !  TEST 7:      ETHERNET CARRIER SENSE TEST
;  2269  1     !
;  2270  1     !  DESCRIPTION:
;  2271  1     !
;  2272  1     !       This test verifies that the DEQNA can transmit external loopback
;  2273  1     !       packets and if not faulty FRU is can be found by executing this
;  2274  1     !       by implementing the instructions printed on the operator's console.
;  2275  1     !
;  2276  1     !       In order to run this test successfully the operator has to make
;  2277  1     !       sure that DEQNA is connected to the transceiver. If the operator
;  2278  1     !       specifies loop on error, the program re-executes the code that detected
;  2279  1     !       the error until ^C is entered.
;  2280  1     !
;  2281  1     !       Hardware tested:        Carrier Sense circuitry
;  2282  1     !                               Encode/Decode ( ED ) chip
;  2283  1     !
;  2284  1     !       Processing:
;  2285  1     !
;  2286  1     !            BEGIN
;  2287  1     !               reset device
;  2288  1     !               select external loopback mode
;  2289  1     !               check external hardware
;  2290  1     !               IF bad hardware
;  2291  1     !               THEN
;  2292  1     !                  print error message if not inhibited
;  2293  1     !               ENDIF
;  2294  1     !               read CSR
;  2295  1     !               IF Ethernet Carrier Sense bit ( bit 13 ) = 1
;  2296  1     !               THEN
;  2297  1     !                  print error message if not inhibited
;  2298  1     !               ENDIF
;  2299  1     !               transmit longest unchained loopback packet ( ETHERNET format )
;  2300  1     !               read CSR while transmitting loopback packet
;  2301  1     !               IF Ethernet Carrier Sense bit (bit 13) = 0
;  2302  1     !               THEN
;  2303  1     !                  print error message if not inhibited
;  2304  1     !               ELSE
;  2305  1     !                  wait until Carrer Sense bit goes to 0
;  2306  1     !               ENDIF
;  2307  1     !               read CSR
;  2308  1     !               IF Ethernet Carrier Sense bit (bit 13) = 1
;  2309  1     !               THEN
;  2310  1     !                  print error message if not inhibited
;  2311  1     !               ENDIF
;  2312  1     !            END
;  2313  1     !--
```

```
;  2314  3     BGNTST;
;  2315  3
;  2316  3     IF .SWP_ILOOP
;  2317  3       THEN
;  2318  4         BEGIN
;  2319  4           RESET_DEQNA ( );
;  2320  5           IF ( NOT GET_BIT [ CSR, XC ] ) AND ( .SWP_LBC EQLU ZERO )
;  2321  4             THEN
;  2322  5               BEGIN
;  2323  5                 CSR_WORD = GET_BIT [ CSR_ALL ];
;  2324  5                 SELECTONE .XC_FLAG OF
;  2325  5                   SET
;  2326  5                   [ 0 ]:
;  2327  6                           BEGIN
;  2328  6                             XC_FLAG - .XC_FLAG + 1;
;  2329  6                             PRINTB ( MSG59 );
;  2330  6                             PRINTB ( MSG47 );
;  2331  6                             ERRDF  ( 0704, MSG00, ERROR$REPORT );
;  2332  5                           END;
;  2333  5                   [ 1 ]:
;  2334  6                           BEGIN
;  2335  6                             XC_FLAG = ZERO;
;  2336  6                             PRINTB ( MSG59 );
;  2337  6                             PRINTB ( MSG42 );
;  2338  6                             ERRDF  ( 0705, MSG00, ERROR$REPORT );
;  2339  5                           END;
;  2340  5                   TES;
;  2341  5                 EXIT_TST;
;  2342  5               END
;  2343  4                 ELSE
;  2344  4                   XC_FLAG = ZERO;
;  2345  4
;  2346  4           !++
;  2347  4           !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM IF EXECUTING
;  2348  4           !  TESTS IN EXTERNAL LOOPBACK MODE.
;  2349  4           !-
;  2350  4
;  2351  4           RESET_DEQNA ( );
;  2352  4           PREP_FOR_SETUP ( );
;  2353  4           INCR INDEX1 FROM 1 TO 14 DO
;  2354  4             WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  2355  4
;  2356  6           BGNSUB;
;  2357  6             XMIT_SETUP_PACKET ( N_MODE );
;  2358  4           ENDSUB;
;  2359  4
;  2360  4           ERR_FLAG = ZERO;
;  2361  4           INCR INDEX2 FROM 0 TO 19 DO
;  2362  5             BEGIN
;  2363  5               SEND_TEST_PACKET ( );
;  2364  5               DELAY ( 100 );
;  2365  5               CSR_WORD = GET_BIT ( CSR_ALL );
;  2366  5               IF ( .CSR_WORD AND %O'100220' ) EQLU %O'100220'
```

```
;    2367  5                THEN
;    2368  6                  BEGIN
;    2369  6                    ERR_FLAG = ZERO;
;    2370  6                    EXITLOOP;
;    2371  6                  END
;    2372  5                ELSE
;    2373  5                  ERR_FLAG = ONE;
;    2374  4              END;
;    2375  4
;    2376  4            IF .ERR_FLAG
;    2377  4              THEN
;    2378  5                BEGIN
;    2379  5                  SELECTONE .ERR_COUNT OF
;    2380  5                    SET
;    2381  5                    [ 0 ]:
;    2382  6                          BEGIN
;    2383  6                            ERR_COUNT  = 1;
;    2384  6                            PRINTB ( MSG59 );
;    2385  6                            PRINTB ( MSG35 );
;    2386  6                            PRINTB ( MSG36 );
;    2387  6                            ERRDF  ( 0706, MSG00, ERROR$REPORT );
;    2388  5                          END;
;    2389  5                    [ 1 ]:
;    2390  6                          BEGIN
;    2391  6                            ERR_COUNT  = 2;
;    2392  6                            PRINTB ( MSG59 );
;    2393  6                            PRINTB ( MSG37 );
;    2394  6                            PRINTB ( MSG38 );
;    2395  6                            ERRDF ( 0707, MSG00, ERROR$REPORT );
;    2396  5                          END;
;    2397  5                    [ 2 ]:
;    2398  6                          BEGIN
;    2399  6                            ERR_COUNT  = 3;
;    2400  6                            PRINTB ( MSG59 );
;    2401  6                            PRINTB ( MSG39 );
;    2402  6                            PRINTB ( MSG40 );
;    2403  6                            ERRDF ( 0708, MSG00, ERROR$REPORT );
;    2404  5                          END;
;    2405  5                    [ 3 ]:
;    2406  6                          BEGIN
;    2407  6                            ERR_COUNT  = 0;
;    2408  6                            PRINTB ( MSG59 );
;    2409  6                            PRINTB ( MSG41 );
;    2410  6                            ERRDF ( 0709, MSG00, ERROR$REPORT );
;    2411  5                          END;
;    2412  5                    [ 4 ]:
;    2413  6                          BEGIN
;    2414  6                            ERR_COUNT  = 0;
;    2415  6                            PRINTB ( MSG59 );
;    2416  6                            PRINTB ( MSG45 );
;    2417  6                            ERRDF ( 0710, MSG00, ERROR$REPORT );
;    2418  5                          END;
;    2419  5                    TES;
```

```
;  2420  5                EXIT_TST;
;  2421  5              END
;  2422  4            ELSE
;  2423  4              IF .ERR_COUNT GTRU ZERO
;  2424  4                THEN
;  2425  5                  BEGIN
;  2426  5                    CSR_WORD = GET_BIT ( CSR_ALL );
;  2427  5                    SELECTONE .ERR_COUNT OF
;  2428  5                      SET
;  2429  5                      [ 1 ]:
;  2430  6                              BEGIN
;  2431  6                                ERR_COUNT  = 4;
;  2432  6                                PRINTB ( MSG59 );
;  2433  6                                PRINTB ( MSG43 );
;  2434  6                                PRINTB ( MSG44 );
;  2435  6                                ERRDF ( 0711, MSG00, ERROR$REPORT );
;  2436  5                              END;
;  2437  5                      [ 2,3 ]:
;  2438  6                              BEGIN
;  2439  6                                ERR_COUNT  = 0;
;  2440  6                                PRINTB ( MSG59 );
;  2441  6                                PRINTB ( MSG42 );
;  2442  6                                ERRDF ( 0712, MSG00, ERROR$REPORT );
;  2443  5                              END;
;  2444  5                       [ 4 ]:
;  2445  6                              BEGIN
;  2446  6                                ERR_COUNT  = 0;
;  2447  6                                PRINTB ( MSG59 );
;  2448  6                                PRINTB ( MSG46 );
;  2449  6                                ERRDF ( 0713, MSG00, ERROR$REPORT );
;  2450  5                              END;
;  2451  5                       TES:
;  2452  5                      EXIT_TST;
;  2453  4                    END;
;  2454  4
;  2455  4              XC_FLAG   = ZERO;
;  2456  4              ERR_COUNT = ZERO;
;  2457  4
;  2458  6            BGNSUB;
;  2459  6              INCR INDEX2 FROM 0 TO TIME1_LIMIT DO
;  2460  7                BEGIN
;  2461  7                  RESET_DEQNA ( );
;  2462  7                  TEMP5 = .INDEX2;
;  2463  7
;  2464  7                  !++
;  2465  7                  !  CHECK ETHERNET CARRIER SENSE BIT ( CA - BIT 13 ) IN THE CSR. CA SHOULD BE
;  2466  7                  !  SET TO '1' WHILE THE DEQNA IS TRANSMITTING. IF CA ISN'T SET TO '1' WITHIN
;  2467  7                  !  THE EXPECTED TIME LIMIT, ERROR MESSAGE IS PRINTED OUT.
;  2468  7                  !--
;  2469  7
;  2470  7                  SEND_TEST_PACKET ( );
;  2471  7
;  2472  7                  INCR INDEX FROM 0 TO TIME1_LIMIT DO
```

```
;  2473  7                    IF GET_BIT [ CSR, CA ] EQLU ONE
;  2474  7                      THEN
;  2475  8                        BEGIN
;  2476  8                          TEMP2 = GET_BIT [ CSR_ALL ];
;  2477  8                          EXITLOOP;
;  2478  8                        END
;  2479  7                      ELSE
;  2480  7                        IF .INDEX EQLU TIME1_LIMIT
;  2481  7                          THEN
;  2482  8                            BEGIN
;  2483  8                              PRINTB ( MSG59 );
;  2484  8                              PRINTB ( MSG19, GET_BIT [ CSR_ALL ] );
;  2485  8                              ERRDF ( 0701, MSG00, ERROR$REPORT );
;  2486  7                            END;
;  2487  7
;  2488  7                    !++
;  2489  7                    !  NOW CHECK IF THE CA BIT RESETS TO '0' WHEN THE DEQNA COMPLETES TRANSMITTING
;  2490  7                    !  LOOPBACK PACKET. PRINT ERROR MESSAGE IF LOOPBACK PACKET TRANSMISSION
;  2491  7                    !  EXCEEDS SELECTED TIME LIMIT.
;  2492  7                    !--
;  2493  7
;  2494  7                    INCR INDEX FROM 0 TO TIME2_LIMIT DO
;  2495  7                      IF GET_BIT [ CSR, CA ] EQLU ZERO
;  2496  7                        THEN
;  2497  8                          BEGIN
;  2498  8                            TEMP3 = GET_BIT [ CSR_ALL ];
;  2499  8                            EXITLOOP;
;  2500  8                          END
;  2501  7                        ELSE
;  2502  7                          IF .INDEX EQLU TIME2_LIMIT
;  2503  7                            THEN
;  2504  0                              BEGIN
;  2505  8                                PRINTB ( MSG59 );
;  2506  8                                PRINTB ( MSG20, GET_BIT [ CSR_ALL ] );
;  2507  8                                ERRDF ( 0702, MSG00, ERROR$REPORT );
;  2508  7                              END;
;  2509  7
;  2510  7                    !++
;  2511  7                    !  CHECK RECEIVE INTERRUPT REQUEST BIT ( RI - BIT 15 ) TO VERIFY THAT DEQNA
;  2512  7                    !  ACTUALLY TRANSMITTED LOOPBACK PACKET.
;  2513  7                    !--
;  2514  7
;  2515  7                    DELAY ( 50 );
;  2516  7
;  2517  7                    IF GET_BIT [ CSR, RI ] EQLU ONE
;  2518  7                      THEN
;  2519  8                        BEGIN
;  2520  8                          TEMP4 = GET_BIT [ CSR_ALL ];
;  2521  8                          EXITLOOP;
;  2522  7                        END;
;  2523  6                  END;
;  2524  6
;  2525  6                IF .TEMP5 EQLU TIME1_LIMIT
```

```
;    2526  6               THEN
;    2527  7                 BEGIN
;    2528  7                   PRINTB ( MSG59 );
;    2529  7                   PRINTB ( MSG21, GET_BIT [ CSR_ALL ] );
;    2530  7                   ERRDF  ( 0703, MSGOO, ERROR$REPORT );
;    2531  6                 END;
;    2532  6
;    2533  6
;    2534  7             IF ( .XMIT_D_LIST [ ERRSU ] EQLU 1 ) AND ( .XMIT_D_LIST [ ABORT ] EQLU 1 )
;    2535  6               THEN
;    2536  7                 BEGIN
;    2537  7                   PRINTB ( MSG59 );
;    2538  7                   PRINTB ( MSG71 );
;    2539  7                   ERRDF  ( 0714, MSGOO, ERROR$REPORT );
;    2540  6                 END;
;    2541  6
;    2542  6             !++
;    2543  6             !  COMPARE STATUS REGISTERS TO EXPECTED VALUES
;    2544  6             !--
;    2545  6
;    2546  6             CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK      );  ! O'100220', O'100220'
;    2547  6             XMIT_D_LIST [ STWD1 ] = .XMIT_D_LIST [ STWD1 ] AND %O'177377';
;    2548  6             CHK_XMIT_STATUS ( XFLG_STATUS, XWD11_STATUS );  ! O'140000', O'000000'
;    2549  6             CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS  );  ! O'140000', O'020000'
;    2550  6
;    2551  6             IF .XMIT_D_LIST [ TDR ] EQLU ZERO
;    2552  6               THEN
;    2553  7                 BEGIN
;    2554  7                   PRINTB ( MSG59 );
;    2555  7                   PRINTB ( MSG58 );
;    2556  7                   ERRDF  ( 0715, MSGOO, ERROR$REPORT );
;    2557  6                 END;
;    2558  6
;    2559  4           ENDSUB;
;    2560  3         END;
;    2561  1     ENDTST;


                                        .SBTTL  $T7 TEST 7 - ETHERNET CARRIER SENSE TEST
000000  004137  000000G          $T7:   JSR     R1,$SAVE2                       ;                       2263
000004  162706  000034                 SUB     #34,SP
000010  032737  000001  000000G        BIT     #1,SWP.ILOOP                    ;                       2316
000016  001476                         BEQ     4$
000020  004737  000000G                JSR     PC,RESET.DEQNA                  ;                       2319
000024  013700  000000G                MOV     REG.ADR,R0                      ;                       2320
000030  016016  000016                 MOV     16(R0),(SP)                     ; *,TMP.LOCATION
000034  032716  010000                 BIT     #10000,(SP)                     ; *,TMP.LOCATION
000040  001067                         BNE     5$
000042  005737  000000G                TST     SWP.LBC
000046  001064                         BNE     5$
000050  011666  000002                 MOV     (SP),2(SP)                      ; *,TMP.LOCATION        2323
000054  011637  000000G                MOV     (SP),CSR.WORD                   ; TMP.LOCATION,*
000060  013700  000000G                MOV     XC.FLAG,R0                      ;                       2324
```

```
000064  001023                          BNE     1$                      ;                               2326
000066  005237  000000G                 INC     XC.FLAG                 ;                               2328
000072  012746  000000G                 MOV     #MSG59,-(SP)            ;                               2329
000076  012746  000001                  MOV     #1,-(SP)
000102  010600                          MOV     SP,R0                   ; SP,*
000104  104414                          TRAP    14
000106  012716  000000G                 MOV     #MSG47,(SP)             ;                               2330
000112  012746  000001                  MOV     #1,-(SP)
000116  010600                          MOV     SP,R0                   ; SP,*
000120  104414                          TRAP    14
000122  104455                          TRAP    55                      ;                               2331
000124  001300                          .WORD   1300
000126  000000G                         .WORD   MSG00
000130  000000G                         .WORD   ERROR$REPORT
000132  000425                          BR      2$                      ;                               2327
000134  020027  000001          1$:     CMP     R0,#1                   ;                               2333
000140  001024                          BNE     3$
000142  005037  000000G                 CLR     XC.FLAG                 ;                               2335
000146  012746  000000G                 MOV     #MSG59,-(SP)            ;                               2336
000152  012746  000001                  MOV     #1,-(SP)
000156  010600                          MOV     SP,R0                   ; SP,*
000160  104414                          TRAP    14
000162  012716  000000G                 MOV     #MSG42,(SP)             ;                               2337
000166  012746  000001                  MOV     #1,-(SP)
000172  010600                          MOV     SP,R0                   ; SP,*
000174  104414                          TRAP    14
000176  104455                          TRAP    55                      ;                               2338
000200  001301                          .WORD   1301
000202  000000G                         .WORD   MSG00
000204  000000G                         .WORD   ERROR$REPORT
000206  062706  000006          2$:     ADD     #6,SP                   ;                               2334
000212  104463                  3$:     TRAP    63                      ;                               2340
000214  000137  007046'          4$:     JMP     49$                     ;                               2322
000220  005037  000000G         5$:     CLR     XC.FLAG                 ;                               2344
000224  004737  000000G                 JSR     PC,RESET.DEQNA          ;                               2351
000230  004737  000000G                 JSR     PC,PREP.FOR.SETUP       ;                               2352
000234  012701  000001                  MOV     #1,R1                   ; *,INDEX1                      2353
000240  010146                  6$:     MOV     R1,-(SP)                ; INDEX1,*                      2354
000242  012746  000023                  MOV     #23,-(SP)
000246  004737  000000G                 JSR     PC,WRT.STATION.ADR
000252  022626                          CMP     (SP)+,(SP)+
000254  005201                          INC     R1                      ; INDEX1                        2353
000256  020127  000016                  CMP     R1,#16                  ; INDEX1,*
000262  003766                          BLE     6$
000264  104402                  7$:     TRAP    2                       ;                               2354
000266  012746  000200                  MOV     #200,-(SP)              ;                               2357
000272  004737  000000G                 JSR     PC,XMIT.SETUP.PACKET
000276  005726                          TST     (SP)+                   ;                               2354
000300  104467                          TRAP    67                      ;                               2357
000302  006000                          ROR     R0
000304  103767                          BLO     7$
000306  005037  000000G                 CLR     ERR.FLAG                ;                               2360
000312  012702  000024                  MOV     #24,R2                  ; *,INDEX2                      2361
```

```
000316  004737  000000G                 8$:     JSR     PC,SEND.TEST.PACKET             ;                               2363
000322  012701  000144                           MOV     #144,R1                         ; *,$$TMP2                      2364
000326  001410                           9$:     BEQ     12$
000330  013700  000000G                          MOV     L$DLY,R0                        ; *,$$TMP1
000334  001403                                   BEQ     11$
000336  005066  000032                 10$:     CLR     32(SP)                          ; $$TMP
000342  077003                                   SOB     R0,10$                          ; $$TMP1,*
000344  005301                         11$:     DEC     R1                              ; $$TMP2
000346  000767                                   BR      9$
000350  013700  000000G                 12$:     MOV     REG.ADR,R0                      ;                               2365
000354  016066  000016  000004                   MOV     16(R0),4(SP)                    ; *,TMP.LOCATION
000362  016637  000004  000000G                 MOV     4(SP),CSR.WORD                  ; TMP.LOCATION,*
000370  016600  000004                           MOV     4(SP),R0                        ; CSR.WORD,*                    2366
000374  042700  077557                         BIC     #77557,R0
000400  020027  100220                         CMP     R0,#-77560
000404  001003                                   BNE     13$
000406  005037  000000G                         CLR     ERR.FLAG                        ;                               2369
000412  000404                                   BR      14$                             ;                               2368
000414  012737  000001  000000G       13$:     MOV     #1,ERR.FLAG                     ;                               2373
000422  077243                                   SOB     R2,8$                           ; INDEX2,*                      2361
000424  013701  000000G                 14$:     MOV     ERR.COUNT,R1                    ;                               2379
000430  032737  000001  000000G               BIT     #1,ERR.FLAG                     ;                               2376
000436  001002                                   BNE     15$
000440  000137  005612'                         JMP     23$
000444  005701                         15$:     TST     R1                              ;                               2381
000446  001032                                   BNE     16$
000450  012737  000001  000000G               MOV     #1,ERR.COUNT                    ;                               2383
000456  012746  000000G                         MOV     #MSG59,-(SP)                    ;                               2384
000462  012746  000001                         MOV     #1,-(SP)
000466  010600                                 MOV     SP,R0                           ; SP,*
000470  104414                                 TRAP    14
000472  012716  000000G                         MOV     #MSG35,(SP)                     ;                               2385
000476  012746  000001                         MOV     #1,-(SP)
000502  010600                                 MOV     SP,R0                           ; SP,*
000504  104414                                 TRAP    14
000506  012716  000000G                         MOV     #MSG36,(SP)                     ;                               2386
000512  012746  000001                         MOV     #1,-(SP)
000516  010600                                 MOV     SP,R0                           ; SP,*
000520  104414                                 TRAP    14
000522  104455                                 TRAP    55                              ;                               2387
000524  001302                                 .WORD   1302
000526  000000G                               .WORD   MSG00
000530  000000G                               .WORD   ERROR$REPORT
000532  000471                                 BR      18$                             ;                               2382
000534  020127  000001                 16$:     CMP     R1,#1                           ;                               2389
000540  001032                                   BNE     17$
000542  012737  000002  000000G               MOV     #2,ERR.COUNT                    ;                               2391
000550  012746  000000G                         MOV     #MSG59,-(SP)                    ;                               2392
000554  012746  000001                         MOV     #1,-(SP)
000560  010600                                 MOV     SP,R0                           ; SP,*
000562  104414                                 TRAP    14
000564  012716  000000G                         MOV     #MSG37,(SP)                     ;                               2393
000570  012746  000001                         MOV     #1,-(SP)
```

```
000574  010600                          MOV     SP,R0                   ; SP,*
000576  104414                          TRAP    14
000600  012716  000000G                 MOV     #MSG38,(SP)             ;                       2394
000604  012746  000001                  MOV     #1,-(SP.
000610  010600                          MOV     SP,R0                   ; SP,*
000612  104414                          TRAP    14
000614  104455                          TRAP    55                      ;                       2395
000616  001303                          .WORD   1303
000620  000000G                         .WORD   MSG00
000622  000000G                         .WORD   ERROR$REPORT
000624  000434                          BR      18$                     ;                       2390
000626  020127  000002          17$:    CMP     R1,#2                   ;                       2397
000632  001034                          BNE     19$
000634  012737  000003  000000G         MOV     #3,ERR.COUNT            ;                       2399
000642  012746  000000G                 MOV     #MSG59,-(SP)            ;                       2400
000646  012746  000001                  MOV     #1,-(SP)
000652  010600                          MOV     SP,R0                   ; SP,*
000654  104414                          TRAP    14
000656  012716  000000G                 MOV     #MSG39,(SP)             ;                       2401
000662  012746  000001                  MOV     #1,-(SP)
000666  010600                          MOV     SP,R0                   ; SP,*
000670  104414                          TRAP    14
000672  012716  000000G                 MOV     #MSG40,(SP)             ;                       2402
000676  012746  000001                  MOV     #1,-(SP)
000702  010600                          MOV     SP,R0                   ; SP,*
000704  104414                          TRAP    14
000706  104455                          TRAP    55                      ;                       2403
000710  001304                          .WORD   1304
000712  000000G                         .WORD   MSG00
000714  000000G                         .WORD   ERROR$REPORT
000716  062706  000010          18$:    ADD     #10,SP                  ;                       2398
000722  000455                          BR      22$                     ;                       2379
000724  020127  000003          19$:    CMP     R1,#3                   ;                       2405
000730  001023                          BNE     20$
000732  005037  000000G                 CLR     ERR.COUNT               ;                       2407
000736  012746  000000G                 MOV     #MSG59,-(SP)            ;                       2408
000742  012746  000001                  MOV     #1,-(SP)
000746  010600                          MOV     SP,R0                   ; SP,*
000750  104414                          TRAP    14
000752  012716  000000G                 MOV     #MSG41,(SP)             ;                       2409
000756  012746  000001                  MOV     #1,-(SP)
000762  010600                          MOV     SP,R0                   ; SP,*
000764  104414                          TRAP    14
000766  104455                          TRAP    55                      ;                       2410
000770  001305                          .WORD   1305
000772  000000G                         .WORD   MSG00
000774  000000G                         .WORD   ERROR$REPORT
000776  000425                          BR      21$                     ;                       2406
001000  020127  000004          20$:    CMP     R1,#4                   ;                       2412
001004  001024                          BNE     22$
001006  005037  000000G                 CLR     ERR.COUNT               ;                       2414
001012  012746  000000G                 MOV     #MSG59,-(SP)            ;                       2415
001016  012746  000001                  MOV     #1,-(SP)
```

```
001022  010600                                  MOV     SP,R0                   ; SP,*
001024  104414                                  TRAP    14
001026  012716  000000G                         MOV     #MSG45,(SP)             ;                               2416
001032  012746  000001                          MOV     #1,-(SP)
001036  010600                                  MOV     SP,R0                   ; SP,*
001040  104414                                  TRAP    14
001042  104455                                  TRAP    55                      ;                               2417
001044  001306                                  .WORD   1306
001046  000000G                                 .WORD   MSG00
001050  000000G                                 .WORD   ERROR$REPORT
001052  062706  000006                  21$:    ADD     #6,SP                   ;                               2413
001056  104463                          22$:    TRAP    63                      ;                               2419
001060  000532                                  BR      28$                     ;                               2378
001062  005701                          23$:    TST     R1                      ;                               2423
001064  001532                                  BEQ     29$
001066  013700  000000G                         MOV     REG.ADR,R0              ;                               2426
001072  016066  000016  000006                  MOV     16(R0),6(SP)            ; *,TMP.LOCATION
001100  016637  000006  000000G                 MOV     6(SP),CSR.WORD          ; TMP.LOCATION,*
001106  020127  000001                          CMP     R1,#1                   ;                               2429
001112  001034                                  BNE     24$
001114  012737  000004  000000G                 MOV     #4,ERR.COUNT            ;                               2431
001122  012746  000000G                         MOV     #MSG59,-(SP)            ;                               2432
001126  012746  000001                          MOV     #1,-(SP)
001132  010600                                  MOV     SP,R0                   ; SP,*
001134  104414                                  TRAP    14
001136  012716  000000G                         MOV     #MSG43,(SP)             ;                               2433
001142  012746  000001                          MOV     #1,-(SP)
001146  010600                                  MOV     SP,R0                   ; SP,*
001150  104414                                  TRAP    14
001152  012716  000000G                         MOV     #MSG44,(SP)             ;                               2434
001156  012746  000001                          MOV     #1,-(SP)
001162  010600                                  MOV     SP,R0                   ; SP,*
001164  104414                                  TRAP    14
001166  104455                                  TRAP    55                      ;                               2435
001170  001307                                  .WORD   1307
001172  000000G                                 .WORD   MSG00
001174  000000G                                 .WORD   ERROR$REPORT
001176  062706  000010                          ADD     #10,SP                  ;                               2430
001202  000450                                  BR      27$                     ;                               2427
001204  020127  000002                  24$:    CMP     R1,#2                   ;                               2437
001210  002426                                  BLT     25$
001212  020127  000003                          CMP     R1,#3
001216  003023                                  BGT     25$
001220  005037  000000G                         CLR     ERR.COUNT               ;                               2439
001224  012746  000000G                         MOV     #MSG59,-(SP)            ;                               2440
001230  012746  000001                          MOV     #1,-(SP)
001234  010600                                  MOV     SP,R0                   ; SP,*
001236  104414                                  TRAP    14
001240  012716  000000G                         MOV     #MSG42,(SP)             ;                               2441
001244  012746  000001                          MOV     #1,-(SP)
001250  010600                                  MOV     SP,R0                   ; SP,*
001252  104414                                  TRAP    14
001254  104455                                  TRAP    55                      ;                               2442
```

```
001256  001310                                  .WORD   1310
001260  000000G                                 .WORD   MSG00
001262  000000G                                 .WORD   ERROR$REPORT
001264  000425                                  BR      26$                     ;                               2438
001266  020127  000004              25$:        CMP     R1,#4                   ;                               2444
001272  001024                                  BNE     27$
001274  005037  000000G                         CLR     ERR.COUNT               ;                               2446
001300  012746  000000G                         MOV     #MSG59,-(SP)            ;                               2447
001304  012746  000001                          MOV     #1,-(SP)
001310  010600                                  MOV     SP,R0                   ; SP,*
001312  104414                                  TRAP    14
001314  012716  000000G                         MOV     #MSG46,(SP)             ;                               2448
001320  012746  000001                          MOV     #1,-(SP)
001324  010600                                  MOV     SP,R0                   ; SP,*
001326  104414                                  TRAP    14
001330  104455                                  TRAP    55                      ;                               2449
001332  001311                                  .WORD   1311
001334  000000G                                 .WORD   MSG00
001336  000000G                                 .WORD   ERROR$REPORT
001340  062706  000006              26$:        ADD     #6,SP                   ;                               2445
001344  104463                      27$:        TRAP    63                      ;                               2451
001346  000137  007046'              28$:        JMP     49$                     ;                               2425
001352  005037  000000G             29$:        CLR     XC.FLAG                 ;                               2455
001356  005037  000000G                         CLR     ERR.COUNT               ;                               2456
001362  104402                      30$:        TRAP    2
001364  005002                                  CLR     R2                      ; INDEX2                        2459
001366  004737  000000G             31$:        JSR     PC,RESET.DEQNA          ;                               2461
001372  010237  000000G                         MOV     R2,TEMP5                ; INDEX2,*                      2462
001376  004737  000000G                         JSR     PC,SEND.TEST.PACKET     ;                               2470
001402  005001                                  CLR     R1                      ; INDEX                         2472
001404  013700  000000G             32$:        MOV     REG.ADR,R0              ;                               2473
001410  016066  000016  000010                 MOV     16(R0),10(SP)           ; *,TMP.LOCATION
001416  032766  020000  000010                 BIT     #20000,10(SP)           ; *,TMP.LOCATION
001424  001407                                  BEQ     33$
001426  016666  000010  000012                 MOV     10(SP),12(SP)           ; *,TMP.LOCATION                2476
001434  016637  000012  000000G               MOV     12(SP),TEMP2            ; TMP.LOCATION,*
001442  000440                                  BR      35$                     ;                               2475
001444  020127  000200              33$:        CMP     R1,#200                 ; INDEX,*                       2480
001450  001031                                  BNE     34$
001452  012746  000000G                         MOV     #MSG59,-(SP)            ;                               2483
001456  012746  000001                          MOV     #1,-(SP)
001462  010600                                  MOV     SP,R0                   ; SP,*
001464  104414                                  TRAP    14
001466  013700  000000G                         MOV     REG.ADR,R0              ;                               2484
001472  016066  000016  000020                 MOV     16(R0),20(SP)           ; *,TMP.LOCATION
001500  016616  000020                          MOV     20(SP),(SP)             ; TMP.LOCATION,*
001504  012746  000000G                         MOV     #MSG19,-(SP)
001510  012746  000002                          MOV     #2,-(SP)
001514  010600                                  MOV     SP,R0                   ; SP,*
001516  104414                                  TRAP    14
001520  104455                                  TRAP    55                      ;                               2485
001522  001275                                  .WORD   1275
001524  000000G                                 .WORD   MSG00
```

```
001526  000000G                                 .WORD   ERROR$REPORT
001530  062706  000010                          ADD     #10,SP                  ;                               2482
001534  005201                          34$:    INC     R1                      ; INDEX                         2472
001536  020127  000200                          CMP     R1,#200                 ; INDEX,*
001542  003720                                  BLE     32$
001544  005001                          35$:    CLR     R1                      ; INDEX                         2494
001546  013700  000000G                 36$:    MOV     REG.ADR,R0              ;                               2495
001552  016066  000016  000016                  MOV     16(R0),16(SP)           ; *,TMP.LOCATION
001560  032766  020000  000016                  BIT     #20000,16(SP)           ; *,TMP.LOCATION
001566  001007                                  BNE     37$
001570  016666  000016  000020                  MOV     16(SP),20(SP)           ; *,TMP.LOCATION                2498
001576  016637  000020  000000G                 MOV     20(SP),TEMP3            ; TMP.LOCATION,*
001604  000440                                  BR      39$                     ;                               2497
001606  020127  002000                  37$:    CMP     R1,#2000                ; INDEX,*                       2502
001612  001031                                  BNE     38$
001614  012746  000000G                         MOV     #MSG59,-(SP)            ;                               2505
001620  012746  000001                          MOV     #1,-(SP)
001624  010600                                  MOV     SP,R0                   ; SP,*
001626  104414                                  TRAP    14
001630  013700  000000G                         MOV     REG.ADR,R0              ;                               2506
001634  016066  000016  000026                  MOV     16(R0),26(SP)           ; *,TMP.LOCATION
001642  016616  000026                          MOV     26(SP),(SP)             ; TMP.LOCATION,*
001646  012746  000000G                         MOV     #MSG20,-(SP)
001652  012746  000002                          MOV     #2,-(SP)
001656  010600                                  MOV     SP,R0                   ; SP,*
001660  104414                                  TRAP    14
001662  104455                                  TRAP    55                      ;                               2507
001664  001276                                  .WORD   1276
001666  000000G                                 .WORD   MSG00
001670  000000G                                 .WORD   ERROR$REPORT
001672  062706  000010                          ADD     #10,SP                  ;                               2504
001676  005201                          38$:    INC     R1                      ; INDEX                         2494
001700  020127  002000                          CMP     R1,#2000                ; INDEX,*
001704  003720                                  BLE     36$
001706  012701  000062                  39$:    MOV     #62,R1                  ; *,$$TMP2                      2515
001712  001410                          40$:    BEQ     43$
001714  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
001720  001403                                  BEQ     42$
001722  005066  000032                  41$:    CLR     32(SP)                  ; $$TMP
001726  077003                                  SOB     R0,41$                  ; $$TMP1,*
001730  005301                          42$:    DEC     R1                      ; $$TMP2
001732  000767                                  BR      40$
001734  013700  000000G                 43$:    MOV     REG.ADR,R0              ;                               2517
001740  016066  000016  000024                  MOV     16(R0),24(SP)           ; *,TMP.LOCATION
001746  100007                                  BPL     44$
001750  016666  000024  000026                  MOV     24(SP),26(SP)           ; *,TMP.LOCATION                2520
001756  016637  000026  000000G                 MOV     26(SP),TEMP4            ; TMP.LOCATION,*
001764  000406                                  BR      45$                     ;                               2519
001766  005202                          44$:    INC     R2                      ; INDEX2                        2459
001770  020227  000200                          CMP     R2,#200                 ; INDEX2,*
001774  003002                                  BGT     45$
001776  000137  006116'                         JMP     31$
002002  023727  000000G 000200          45$:    CMP     TEMP5,#200              ;                               2525
```

```
002010  001031                                  BNE     46$
002012  012746  000000G                         MOV     #MSG59,-(SP)            ;                               2528
002016  012746  000001                          MOV     #1,-(SP)
002022  010600                                  MOV     SP,R0                   ; SP,*
002024  104414                                  TRAP    14
002026  013700  000000G                         MOV     REG.ADR,R0              ;                               2529
002032  016066  000016  000034                  MOV     16(R0),34(SP)           ; *,TMP.LOCATION
002040  016616  000034                          MOV     34(SP),(SP)             ; TMP.LOCATION,*
002044  012746  000000G                         MOV     #MSG21,-(SP)
002050  012746  000002                          MOV     #2,-(SP)
002054  010600                                  MOV     SP,R0                   ; SP,*
002056  104414                                  TRAP    14
002060  104455                                  TRAP    55                      ;                               2530
002062  001277                                  .WORD   1277
002064  000000G                                 .WORD   MSG00
002066  000000G                                 .WORD   ERROR$REPORT
002070  062706  000010                          ADD     #10,SP                  ;                               2527
002074  032737  040000  000010G         46$:    BIT     #40000,XMIT.D.LIST+10   ;                               2534
002102  001426                                  BEQ     47$
002104  032737  001000  000010G                 BIT     #1000,XMIT.D.LIST+10
002112  001422                                  BEQ     47$
002114  012746  000000G                         MOV     #MSG59,-(SP)            ;                               2537
002120  012746  000001                          MOV     #1,-(SP)
002124  010600                                  MOV     SP,R0                   ; SP,*
002126  104414                                  TRAP    14
002130  012716  000000G                         MOV     #MSG71,(SP)             ;                               2538
002134  012746  000001                          MOV     #1,-(SP)
002140  010600                                  MOV     SP,R0                   ; SP,*
002142  104414                                  TRAP    14
002144  104455                                  TRAP    55                      ;                               2539
002146  001312                                  .WORD   1312
002150  000000G                                 .WORD   MSG00
002152  000000G                                 .WORD   ERROR$REPORT
002154  062706  000006                          ADD     #6,SP                   ;                               2536
002160  012746  100220                  47$:    MOV     #-77560,-(SP)           ;                               2546
002164  011646                                  MOV     (SP),-(SP)
002166  004737  000000G                         JSR     PC,CHK.CSR.STATUS
002172  042737  000400  000010G                 BIC     #400,XMIT.D.LIST+10     ;                               2547
002200  012716  140000                          MOV     #-40000,(SP)            ;                               2548
002204  005046                                  CLR     -(SP)
002206  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
002212  012716  140000                          MOV     #-40000,(SP)            ;                               2549
002216  012746  020000                          MOV     #20000,-(SP)
002222  004737  000000G                         JSR     PC,CHK.RCV.STATUS
002226  032737  037777  000012G                 BIT     #37777,XMIT.D.LIST+12   ;                               2551
002234  001021                                  BNE     48$
002236  012716  000000G                         MOV     #MSG59,(SP)             ;                               2554
002242  012746  000001                          MOV     #1,-(SP)
002246  010600                                  MOV     SP,R0                   ; SP,*
002250  104414                                  TRAP    14
002252  012716  000000G                         MOV     #MSG58,(SP)             ;                               2555
002256  012746  000001                          MOV     #1,-(SP)
002262  010600                                  MOV     SP,R0                   ; SP,*
```

```
002264  104414                                  TRAP    14
002266  104455                                  TRAP    55                      ;                       2556
002270  001313                                  .WORD   1313
002272  000000G                                 .WORD   MSG00
002274  000000G                                 .WORD   ERROR$REPORT
002276  022626                                  CMP     (SP)+,(SP)+             ;                       2553
002300  062706  000010                  48$:    ADD     #10,SP                  ;                       2456
002304  104467                                  TRAP    67                      ;                       2557
002306  006000                                  ROR     R0
002310  103002                                  BHIS    49$
002312  000137  006112'                          JMP     30$
002316  062706  000034                  49$:    ADD     #34,SP                  ;                       2263
002322  000207                                  RTS     PC

; Routine Size:  618 words,     Routine Base:  AB$CODE$ + 4530
; Maximum stack depth per invocation:  25 words




                                        .SBTTL  T7 TEST 7 - ETHERNET CARRIER SENSE TEST
000000  004737  004530'                  T7::
000000                                  1$:     JSR     PC,$T7                  ;                       2560
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 7054
; Maximum stack depth per invocation:  2 words


;   2562  1
;   2563  1
```

```
;    2564  1     %SBTTL 'TEST 8 - STATION ADDRESS RAM TEST'
;    2565  1     !++
;    2566  1     !
;    2567  1     !  TEST 8:      STATION ADDRESS RAM TEST
;    2568  1     !
;    2569  1     !  DESCRIPTION:
;    2570  1     !
;    2571  1     !       This test verifies that Station Address RAM has no static faults.
;    2572  1     !       The host writes and then reads data patterns to all of the
;    2573  1     !       addressable RAM ( 128 decimal bytes ). The data is checked to see
;    2574  1     !       that the data pattern received is the same as the data pattern
;    2575  1     !       transmited. This test continues until all the data patterns are
;    2576  1     !       exhausted. If the operator specifies loop on error, the program
;    2577  1     !       re-executes the code that detected the error until ↑C is entered.
;    2578  1     !
;    2579  1     !       The following BINARY patterns are used:
;    2580  1     !
;    2581  1     !                       11111111        00000000
;    2582  1     !                       10101010        01010101
;    2583  1     !                       11001100        00110011
;    2584  1     !                       11110000        00001111
;    2585  1     !                       marching 1's, propagating 1's through the RAM
;    2586  1     !                       marching 0's, propagating 0's through the RAM
;    2587  1     !
;    2588  1     !       Hardware tested.        Station Address RAM
;    2589  1     !                               Q-Bus to QTDC interface
;    2590  1     !                               CSR register - Receiver Enable (bit 0)
;    2591  1     !                               Portion of Receive and Transmit FIFO
;    2592  1     !       Processing:
;    2593  1     !
;    2594  1     !            BEGIN
;    2595  1     !               reset device
;    2596  1     !               select Setup mode
;    2597  1     !               REPEAT for each pattern
;    2598  1     !                  load transmit packet with data pattern
;    2599  1     !                  transmit loopback packet (fill all of the RAM)
;    2600  1     !                  receive packet
;    2601  1     !                  check for expected loopback status
;    2602  1     !                  IF error
;    2603  1     !                  THEN
;    2604  1     !                     print error message if not inhibited
;    2605  1     !                  ENDIF
;    2606  1     !                  call compare_packets
;    2607  1     !               ENDREPEAT
;    2608  1     !            END
;    2609  1     !--
```

```
;   2610  3     BGNTST;
;   2611  3
;   2612  3           RESET_DEQNA ( );
;   2613  3
;   2614  3           DECR INDEX1 FROM 7 TO 0 DO
;   2615  4             BEGIN
;   2616  4               INCR INDEX2 FROM 0 TO 127 DO
;   2617  4                 XMIT_BUFFER [ .INDEX2 ] = .PTRN_TABLE [ .INDEX1 ];
;   2618  4
;   2619  6               BGNSUB;
;   2620  6               XMIT_SETUP_PACKET ( N_MODE );
;   2621  4               ENDSUB;
;   2622  3             END;
;   2623  3
;   2624  3     !       TEMP3 = ( N_MODE * 8 ) - 1;
;   2625  3     !       INCR INDEX1 FROM 0 TO .TEMP3 DO
;   2626  3     !         BEGIN
;   2627  3     !           P1 = ZERO;
;   2628  3     !           P2 = .INDEX1;
;   2629  3     !           WALKING_BIT ( );
;   2630  3     !           P1 = N_MODE;
;   2631  3     !           XMIT_SETUP_PACKET ( );
;   2632  3     !
;   2633  3     !           INCR INDEX FROM 0 TO .P3 DO
;   2634  3     !             XMIT_BUFFER [ .INDEX ] = ( - .XMIT_BUFFER [ .INDEX ] ) - 1;
;   2635  3     !           P1 = N_MODE;
;   2636  3     !           XMIT_SETUP_PACKET ( );
;   2637  3     !         END;
;   2638  3
;   2639  3           INCR INDEX1 FROM 0 TO N_MODE - 1 DO
;   2640  4             BEGIN
;   2641  4               INCR INDEX FROM 0 TO N_MODE - 1 DO
;   2642  4                 XMIT_BUFFER [ .INDEX ] = ZERO;
;   2643  4               XMIT_BUFFER [  .INDEX1 ] = %X'FF';
;   2644  4
;   2645  6               BGNSUB;
;   2646  6               XMIT_SETUP_PACKET ( N_MODE );
;   2647  4               ENDSUB;
;   2648  4
;   2649  4               INCR INDEX FROM 0 TO .P3 DO
;   2650  4                 XMIT_BUFFER [  INDEX ] = ( - .XMIT_BUFFER [ .INDEX ] ) - 1;
;   2651  4
;   2652  6               BGNSUB;
;   2653  6               XMIT_SETUP_PACKET ( N_MODE );
;   2654  4               ENDSUB;
;   2655  4
;   2656  3             END;
;   2657  1     ENDTST;


                                        .SBTTL  $T8 TEST 8 - STATION ADDRESS RAM TEST
000000  004137  000000G          $T8:   JSR     R1,$SAVE3                    ;                2561
000004  004737  000000G                 JSR     PC,RESET.DEQNA               ;                2612
```

```
000010  012701  000007                          MOV     #7,R1                           ; *,INDEX1                      2614
000014  005000                          1$:     CLR     R0                              ; INDEX2                        2616
000016  116160  000000G 000000G        2$:     MOVB    PTRN.TABLE(R1),XMIT.BUFFER(R0)  ; *(INDEX1),*(INDEX2)           2617
000024  005200                                  INC     R0                              ; INDEX2                        2616
000026  020027  000177                          CMP     R0,#177                         ; INDEX2,*
000032  003771                                  BLE     2$
000034  104402                          3$:     TRAP    2                               ;                               2617
000036  012746  000200                          MOV     #200,-(SP)                      ;                               2620
000042  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000046  005726                                  TST     (SP)+                           ;                               2617
000050  104467                                  TRAP    67                              ;                               2620
000052  006000                                  ROR     R0
000054  103767                                  BLO     3$
000056  005301                                  DEC     R1                              ; INDEX1                        2614
000060  002355                                  BGE     1$
000062  005001                                  CLR     R1                              ; INDEX1                        2639
000064  005000                          4$:     CLR     R0                              ; INDEX                         2641
000066  105060  000000G                 5$:     CLRB    XMIT.BUFFER(R0)                 ; *(INDEX)                      2642
000072  005200                                  INC     R0                              ; INDEX                         2641
000074  020027  000177                          CMP     R0,#177                         ; INDEX,*
000100  003772                                  BLE     5$
000102  112761  000377  000000G                MOVB    #377,XMIT.BUFFER(R1)            ; *,*(INDEX1)                   2643
000110  104402                          6$:     TRAP    2
000112  012746  000200                          MOV     #200,-(SP)                      ;                               2646
000116  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000122  005726                                  TST     (SP)+                           ;                               2643
000124  104467                                  TRAP    67                              ;                               2646
000126  006000                                  ROR     R0
000130  103767                                  BLO     6$
000132  005000                                  CLR     R0                              ; INDEX                         2649
000134  000411                                  BR      8$
000136  012702  177777                  7$:     MOV     #-1,R2                          ;                               2650
000142  005003                                  CLR     R3
000144  156003  000000G                         BISB    XMIT.BUFFER(R0),R3              ; *(INDEX),*
000150  160302                                  SUB     R3,R2
000152  110260  000000G                         MOVB    R2,XMIT.BUFFER(R0)              ; *,*(INDEX)
000156  005200                                  INC     R0                              ; INDEX                         2649
000160  020037  000000G                 8$:     CMP     R0,P3                           ; INDEX,*
000164  003764                                  BLE     7$
000166  104402                          9$:     TRAP    2                               ;                               2650
000170  012746  000200                          MOV     #200,-(SP)                      ;                               2653
000174  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000200  005726                                  TST     (SP)+                           ;                               2650
000202  104467                                  TRAP    67                              ;                               2653
000204  006000                                  ROR     R0
000206  103767                                  BLO     9$
000210  005201                                  INC     R1                              ; INDEX1                        2639
000212  020127  000177                          CMP     R1,#177                         ; INDEX1,*
000216  003722                                  BLE     4$
000220  000207                                  RTS     PC                              ;                               2561

; Routine Size:  73 words,      Routine Base:  AB$CODE$ + 7070
; Maximum stack depth per invocation:  6 words
```

```
                                        .SBTTL  T8 TEST 8 - STATION ADDRESS RAM TEST
000000  004737  007070'            T8::
000000                              1$:     JSR     PC,$T8                          ;                       2656
000004  104466                              TRAP    66
000006  006000                              ROR     R0
000010  103773                              BLO     1$
000012  000207                              RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 7312
; Maximum stack depth per invocation:  2 words


;   2658  1
```

```
;   2659  1     %SBTTL 'TEST 9 - PROMISCUOUS STATION ADDRESS TEST'
;   2660  1     !++
;   2661  1     !
;   2662  1     !  TEST 9:      PROMISCUOUS STATION ADDRESS TEST
;   2663  1     !
;   2664  1     !  DESCRIPTION:
;   2665  1     !
;   2666  1     !       This test verifies that DEQNA promiscuous addressing mode functions
;   2667  1     !       as specified. Bit patterns and addresses in and out of the range of
;   2668  1     !       setup addresses are used to assure that there is true promiscuity.
;   2669  1     !       If the operator specifies loop on error, the program re-executes the
;   2670  1     !       code that detected the error until ^C is entered.
;   2671  1     !
;   2672  1     !       Hardware tested:        Promiscuous addressing mode logic
;   2673  1     !
;   2674  1     !       Set of Target Addresses in HEXADECIMAL.
;   2675  1     !
;   2676  1     !               00-00-00-00-00-00
;   2677  1     !               AA-AA-AA-AA-AA-AA
;   2678  1     !               55-55-55-55-55-55
;   2679  1     !               FF-FF-FF-FF-FF-FF
;   2680  1     !               Walking 1, shifting 1 across the Target Station Address
;   2681  1     !               Walking 0, shifting 0 across the Target Station Address
;   2682  1     !
;   2683  1     !       Processing:
;   2684  1     !
;   2685  1     !            BEGIN
;   2686  1     !               reset device
;   2687  1     !               select internal loopback mode
;   2688  1     !               set mode to Setup
;   2689  1     !               set 'promiscuous' addressing mode bit
;   2690  1     !               REPEAT for each Target Address
;   2691  1     !                  load Target Address of the packet
;   2692  1     !                  disable receiver
;   2693  1     !                  transmit loopback packet
;   2694  1     !                  enable receiver
;   2695  1     !                  check for expected loopback status
;   2696  1     !                  IF error
;   2697  1     !                  THEN
;   2698  1     !                     print error message if not inhibited
;   2699  1     !                  ENDIF
;   2700  1     !                  call compare_packets
;   2701  1     !               ENDREPEAT
;   2702  1     !             END
;   2703  1     !--
```

```
;  2704  3     BGNTST;
;  2705  3
;  2706  3           !++
;  2707  3           !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM IF EXECUTING
;  2708  3           !  TESTS IN EXTERNAL LOOPBACK MODE.
;  2709  3           !--
;  2710  3
;  2711  3           RESET_DEQNA ( );
;  2712  3           PREP_FOR_SETUP ( );
;  2713  3           INCR INDEX1 FROM 1 TO 14 DO
;  2714  3             WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  2715  3
;  2716  5           BGNSUB;
;  2717  5             XMIT_SETUP_PACKET ( P_MODE );
;  2718  3           ENDSUB;
;  2719  3
;  2720  3           !++
;  2721  3           !  NOW LOOPBACK 6 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;  2722  3           !--
;  2723  3
;  2724  3           RBUF_LENGTH = 6;
;  2725  3           XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  2726  3
;  2727  3           INCR INDEX1 FROM 0 TO 99 DO
;  2728  4             BEGIN
;  2729  4               SELECTONE .INDEX1 OF
;  2730  4                 SET
;  2731  4                 [  0 TO 3  ]:
;  2732  4                              WRT_STATION_ADR ( ZERO, .INDEX1 );
;  2733  4                 [  4 TO 51 ]:
;  2734  4                              WALKING_BIT ( ZERO, .INDEX1 - 4, 5 );
;  2735  4                 [ 52 TO 99 ]:
;  2736  4                              WALKING_BIT ( ONE, .INDEX1 - 52, 5 );
;  2737  4                 TES;
;  2738  4
;  2739  4               WRT_STATION_ADR ( ZERO, ZERO );
;  2740  4
;  2741  6               BGNSUB;
;  2742  6                 XMIT_ILOOP_PACKET ( ZERO );
;  2743  4               ENDSUB;
;  2744  4
;  2745  3             END;
;  2746  3
;  2747  3           INCR INDEX FROM 0 TO 5 DO
;  2748  3             TARGET_ADR [ .INDEX ] = ZERO;
;  2749  1     ENDTST;


                                        .SBTTL  $T9 TEST 9 - PROMISCUOUS STATION ADDRESS TEST
000000  010146                  $T9:    MOV     R1,-(SP)                        ;                       2657
000002  004737  000000G                 JSR     PC,RESET.DEQNA                  ;                       2711
000006  004737  000000G                 JSR     PC,PREP.FOR.SETUP               ;                       2712
000012  012701  000001                  MOV     #1,R1                           ; *,INDEX1              2713
```

```
000016  010146                      1$:   MOV   R1,-(SP)                  ; INDEX1,*                  2714
000020  012746  000023                    MOV   #23,-(SP)
000024  004737  000000G                   JSR   PC,WRT.STATION.ADR
000030  022626                            CMP   (SP)+,(SP)+
000032  005201                            INC   R1                        ; INDEX1                    2713
000034  020127  000016                    CMP   R1,#16                    ; INDEX1,*
000040  003766                            BLE   1$
000042  104402                      2$:   TRAP  2                         ;                           2714
000044  012746  000202                    MOV   #202,-(SP)                ;                           2717
000050  004737  000000G                   JSR   PC,XMIT.SETUP.PACKET
000054  005726                            TST   (SP)+                     ;                           2714
000056  104467                            TRAP  67                        ;                           2717
000060  006000                            ROR   R0
000062  103767                            BLO   2$
000064  012737  000006  000000G           MOV   #6,RBUF.LENGTH            ;                           2724
000072  012700  000006                    MOV   #6,R0                     ;                           2725
000076  006200                            ASR   R0
000100  005400                            NEG   R0
000102  010037  000000G                   MOV   R0,XBUF.LENGTH
000106  005001                            CLR   R1                        ; INDEX1                    2727
000110  005701                      3$:   TST   R1                        ; INDEX1                    2731
000112  002411                            BLT   4$
000114  020127  000003                    CMP   R1,#3                     ; INDEX1,*
000120  003006                            BGT   4$
000122  005046                            CLR   -(SP)                     ;                           2732
000124  010146                            MOV   R1,-(SP)                  ; INDEX1,*
000126  004737  000000G                   JSR   PC,WRT.STATION.ADR
000132  022626                            CMP   (SP)+,(SP)+
000134  000434                            BR    7$                        ;                           2729
000136  020127  000004                4$:   CMP   R1,#4                     ; INDEX1,*                  2733
000142  002410                            BLT   5$
000144  020127  000063                    CMP   R1,#63                    ; INDEX1,*
000150  003005                            BGT   5$
000152  005046                            CLR   -(SP)                     ;                           2734
000154  010146                            MOV   R1,-(SP)                  ; INDEX1,*
000156  162716  000004                    SUB   #4,(SP)
000162  000413                            BR    6$
000164  020127  000064                5$:   CMP   R1,#64                    ; INDEX1,*                  2735
000170  002416                            BLT   7$
000172  020127  000143                    CMP   R1,#143                   ; INDEX1,*
000176  003013                            BGT   7$
000200  012746  000001                    MOV   #1,-(SP)                  ;                           2736
000204  010146                            MOV   R1,-(SP)                  ; INDEX1,*
000206  162716  000064                    SUB   #64,(SP)
000212  012746  000005                6$:   MOV   #5,-(SP)
000216  004737  000000G                   JSR   PC,WALKING.BIT
000222  062706  000006                    ADD   #6,SP
000226  005046                        7$:   CLR   -(SP)                     ;                           2739
000230  005046                            CLR   -(SP)
000232  004737  000000G                   JSR   PC,WRT.STATION.ADR
000236  104402                        8$:   TRAP  2
000240  005016                            CLR   (SP)                      ;                           2742
000242  004737  000000G                   JSR   PC,XMIT.ILOOP.PACKET
```

```
000246  104467                                  TRAP    67
000250  006000                                  ROR     R0
000252  103771                                  BLO     8$
000254  022626                                  CMP     (SP)+,(SP)+                     ;                               2728
000256  005201                                  INC     R1                              ; INDEX1                        2727
000260  020127  000143                          CMP     R1,#143                         ; INDEX1,*
000264  003711                                  BLE     3$
000266  005000                                  CLR     R0                              ; INDEX                         2747
000270  105060  000000G                 9$:     CLRB    TARGET.ADR(R0)                  ; *(INDEX)                      2748
000274  005200                                  INC     R0                              ; INDEX                         2747
000276  020027  000005                          CMP     R0,#5                           ; INDEX,*
000302  003772                                  BLE     9$
000304  012601                                  MOV     (SP)+,R1                        ;                               2657
000306  000207                                  RTS     PC

; Routine Size:  100 words,     Routine Base:  AB$CODE$ + 7326
; Maximum stack depth per invocation:  5 words




                                                .SBTTL  T9 TEST 9 - PROMISCUOUS STATION ADDRESS TEST
000000  004737  007326'                 T9::
000000                                  1$:     JSR     PC,$T9                          ;                               2748
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 7636
; Maximum stack depth per invocation:  2 words


;   2750  1
```

```
;    2751  1     %SBTTL 'TEST 10 - TRANSMIT AND RECEIVE FIFO MEMORY TEST'
;    2752  1     !++
;    2753  1     !
;    2754  1     !  TEST 10:     TRANSMIT AND RECEIVE FIFO MEMORY TEST
;    2755  1     !
;    2756  1     !  DESCRIPTION:
;    2757  1     !
;    2758  1     !       This test verifies that link memory (receive FIFO and transmit
;    2759  1     !       buffer) has no static faults. The host writes and then reads
;    2760  1     !       a sequence of data patterns to the link memory. The data is then
;    2761  1     !       checked to see that the data pattern received is the same as the
;    2762  1     !       data pattern transmitted. This test continues until all the data
;    2763  1     !       patterns are exhausted. If the operator specifies loop on error, the
;    2764  1     !       program re-executes the code that detected the error until ^C is
;    2765  1     !       entered.
;    2766  1     !
;    2767  1     !       Hardware tested:        Transmit buffer address logic
;    2768  1     !                               Transmit buffer memory ( first 1512 bytes )
;    2769  1     !                               Receive FIFO address logic
;    2770  1     !                               Receive FIFO memory ( first 1512 bytes )
;    2771  1     !
;    2772  1     !       The following BINARY patterns are used:
;    2773  1     !
;    2774  1     !                       11111111        00000000
;    2775  1     !                       10101010        01010101
;    2776  1     !                       11001100        00110011
;    2777  1     !                       11110000        00001111
;    2778  1     !
;    2779  1     !       Processing:
;    2780  1     !
;    2781  1     !            BEGIN
;    2782  1     !               reset device
;    2783  1     !               select internal/extended loopback mode
;    2784  1     !               REPEAT for each pattern
;    2785  1     !                  write link memory with pattern - transmit loopback packet
;    2786  1     !                  read link memory with pattern - receive loopback packet
;    2787  1     !                  check for expected loopback status
;    2788  1     !                  IF error
;    2789  1     !                  THEN
;    2790  1     !                     print error message if not inhibited
;    2791  1     !                  ENDIF
;    2792  1     !                  call compare_packets
;    2793  1     !               ENDREPEAT
;    2794  1     !            END
;    2795  1     !--
```

```
;    2796  3     BGNTST;
;    2797  3
;    2798  3     !++
;    2799  3     !  LOOPBACK 1514 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;    2800  3     !--
;    2801  3
;    2802  3     RBUF_LENGTH = LONGEST_PACKET;
;    2803  3     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    2804  3
;    2805  3     INCR INDEX FROM 0 TO 7 DO
;    2806  4       BEGIN
;    2807  4         RESET_DEQNA ( );
;    2808  4         TEMP1 = 0;
;    2809  4         INCR INDEX1 FROM 0 TO 189 DO
;    2810  4           INCR INDEX2 FROM 0 TO 7 DO
;    2811  5             BEGIN
;    2812  5               XMIT_BUFFER [ .TEMP1 ] = .PTRN_TABLE [ .INDEX2 ];
;    2813  5               TEMP1 = .TEMP1 + 1;
;    2814  4             END;
;    2815  4
;    2816  4         !++
;    2817  4         !  ROTATE PATTERN TABLE
;    2818  4         !--
;    2819  4
;    2820  4         TEMP2 = .PTRN_TABLE [ 0 ];
;    2821  4         INCR INDEX3 FROM 0 TO 6 DO
;    2822  4           PTRN_TABLE [ .INDEX3 ] = .PTRN_TABLE [ .INDEX3 + 1 ];
;    2823  4         PTRN_TABLE [ 7 ] = .TEMP2;
;    2824  4
;    2825  6         BGNSUB;
;    2826  6           SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2827  6           SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    2828  6           SEND_ELOOP_PACKET ( ZERO );
;    2829  6           COMPARE_PACKETS ( );
;    2830  4         ENDSUB;
;    2831  4
;    2832  3       END;
;    2833  3
;    2834  3     !  INCR INDEX1 FROM 0 TO LONGEST_PACKET - 1 DO
;    2835  3     !    BEGIN
;    2836  3     !      INCR INDEX FROM 0 TO LONGEST_PACKET - 1 DO
;    2837  3     !        XMIT_BUFFER [ .INDEX ] = ZERO;
;    2838  3     !      XMIT_BUFFER [  .INDEX1 ] = %X'FF';
;    2839  3     !
;    2840  3     !      BGNSUB;
;    2841  3     !        SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2842  3     !        SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    2843  3     !        SEND_ELOOP_PACKET ( ZERO );
;    2844  3     !        COMPARE_PACKETS ( );
;    2845  3     !      ENDSUB;
;    2846  3     !
;    2847  3     !      INCR INDEX FROM 0 TO .P3 DO
;    2848  3     !        XMIT_BUFFER [ .INDEX ] = ( - .XMIT_BUFFER [ .INDEX ] ) - 1;
```

```
;   2849  3     !
;   2850  3     !       BGNSUB;
;   2851  3     !         SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;   2852  3     !         SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;   2853  3     !         SEND_ELOOP_PACKET ( ZERO );
;   2854  3     !         COMPARE_PACKETS ( );
;   2855  3     !       ENDSUB;
;   2856  3     !
;   2857  3     !     END;
;   2858  3
;   2859  1     ENDTST;


                                                .SBTTL  $T10 TEST 10 - TRANSMIT AND RECEIVE FIFO MEMORY TEST
000000  004137  000000G                  $T10:  JSR     R1,$SAVE3                          ;                                  2749
000004  012737  002752  000000G                 MOV     #2752,RBUF.LENGTH                  ;                                  2802
000012  012700  002752                          MOV     #2752,R0                           ;                                  2803
000016  006200                                  ASR     R0
000020  005400                                  NEG     R0
000022  010037  000000G                         MOV     R0,XBUF.LENGTH
000026  012703  000010                          MOV     #10,R3                             ; *,INDEX                          2805
000032  004737  000000G                  1$:    JSR     PC,RESET.DEQNA                     ;                                  2807
000036  005037  000000G                         CLR     TEMP1                              ;                                  2808
000042  012702  000276                          MOV     #276,R2                            ; *,INDEX1                         2809
000046  005000                           2$:    CLR     R0                                 ; INDEX2                           2810
000050  013701  000000G                  3$:    MOV     TEMP1,R1                           ;                                  2812
000054  116061  000000G 000000G                MOVB    PTRN.TABLE(R0),XMIT.BUFFER(R1)     ; *(INDEX2),*
000062  005237  000000G                         INC     TEMP1                              ;                                  2813
000066  005200                                  INC     R0                                 ; INDEX2                           2810
000070  020027  000007                          CMP     R0,#7                              ; INDEX2,*
000074  003765                                  BLE     3$
000076  077215                                  SOB     R2,2$                              ; INDEX1,*                         2809
000100  005037  000000G                         CLR     TEMP2                              ;                                  2820
000104  113737  000000G 000000G                MOVB    PTRN.TABLE,TEMP2
000112  005000                                  CLR     R0                                 ; INDEX3                           2821
000114  116060  000001G 000000G          4$:    MOVB    PTRN.TABLE+1(R0),PTRN.TABLE(R0)    ; *(INDEX3),*(INDEX3)              2822
000122  005200                                  INC     R0                                 ; INDEX3                           2821
000124  020027  000006                          CMP     R0,#6                              ; INDEX3,*
000130  003771                                  BLE     4$
000132  113737  000000G 000007G                MOVB    TEMP2,PTRN.TABLE+7                 ;                                  2823
000140  104402                           5$:    TRAP    2
000142  013746  000000G                         MOV     XBUF.LENGTH,-(SP)                  ;                                  2826
000146  012746  120000                          MOV     #-60000,-(SP)
000152  004737  000000G                         JSR     PC,SET.RDESCR.LIST
000156  013716  000000G                         MOV     XBUF.LENGTH,(SP)                   ;                                  2827
000162  012746  120000                          MOV     #-60000,-(SP)
000166  004737  000000G                         JSR     PC,SET.XDESCR.LIST
000172  005016                                  CLR     (SP)                               ;                                  2828
000174  004737  000000G                         JSR     PC,SEND.ELOOP.PACKET
000200  004737  000000G                         JSR     PC,COMPARE.PACKETS                 ;                                  2829
000204  062706  000006                          ADD     #6,SP                              ;                                  2823
000210  104467                                  TRAP    67                                 ;                                  2829
000212  006000                                  ROR     R0
```

```
000214  103751                                  BLO     5$
000216  077373                                  SOB     R3,1$                   ; INDEX,*                                       2805
000220  000207                                  RTS     PC                      ;                                               2749

; Routine Size:  73 words,      Routine Base:  AB$CODE$ + 7652
; Maximum stack depth per invocation:  8 words



                                                .SBTTL  T10 TEST 10 - TRANSMIT AND RECEIVE FIFO MEMORY TEST
000000  004737  007652'                  T10::
000000                                  1$:     JSR     PC,$T10                 ;                                               2832
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 10074
; Maximum stack depth per invocation:  2 words


;   2860  1
```

```
;    2861  1     %SBTTL 'TEST 11 - PACKET LENGTH TEST'
;    2862  1     !++
;    2863  1     !
;    2864  1     !  TEST 11:     PACKET LENGTH TEST
;    2865  1     !
;    2866  1     !  DESCRIPTION:
;    2867  1     !
;    2868  1     !       This test verifies that DEQNA can transmit and receive variable
;    2869  1     !       length packets ( equal to or greater than 60 bytes and equal to or
;    2870  1     !       less than 1514 bytes without the CRC ) without losing any data
;    2871  1     !       in the process. This test also verifies that the 9th bit of the
;    2872  1     !       FIFO memory is not static (stuck at 1/stuck at 0). If the operator
;    2873  1     !       specifies loop on error, the program re-executes the code that
;    2874  1     !       detected the error until ^C is entered.
;    2875  1     !
;    2876  1     !       Hardware tested:        Transmit and Receive RAM
;    2877  1     !
;    2878  1     !       Processing:
;    2879  1     !
;    2880  1     !            BEGIN
;    2881  1     !               reset device
;    2882  1     !               select internal/extended loopback mode
;    2883  1     !               set down_count to max. packet length
;    2884  1     !               set up_count to min. packet length
;    2885  1     !               REPEAT until down_count = min. packet length
;    2886  1     !                  transmit loopback packet (packet length = down_count)
;    2887  1     !                  check for expected loopback status and packet length
;    2888  1     !                  IF error
;    2889  1     !                  THEN
;    2890  1     !                       print error message if not inhibited
;    2891  1     !                  ENDIF
;    2892  1     !                  call compare_packets
;    2893  1     !                  transmit loopback packet (packet length = up_count)
;    2894  1     !                  check for expected loopback status and packet length
;    2895  1     !                  IF error
;    2896  1     !                  THEN
;    2897  1     !                     print error message if not inhibited
;    2898  1     !                  ENDIF
;    2899  1     !                  call compare_packets
;    2900  1     !                  decrement down_count by 2
;    2901  1     !                  increment up count by 2
;    2902  1     !               ENDREPEAT
;    2903  1     !            END
;    2904  1     !--
```

```
;    2905  3     BGNTST;
;    2906  3
;    2907  3     !++
;    2908  3     !  LOOPBACK PACKETS OF INCREASING AND DECREASING LENGTH THEN CHECK IF PROPERLY
;    2909  3     !  RECEIVED
;    2910  3     !--
;    2911  3
;    2912  3     COUNTER      = ZERO;
;    2913  3     UP_COUNTER   = SHORTEST_PACKET;
;    2914  3     DOWN_COUNTER = LONGEST_PACKET;
;    2915  3
;    2916  3     INCR INDEX1 FROM SHORTEST_PACKET TO MAX_LENGTH BY STEP1 DO
;    2917  4       BEGIN
;    2918  4          RESET_DEQNA ( );
;    2919  4          IF .COUNTER EQLU ZERO
;    2920  4            THEN
;    2921  5              BEGIN
;    2922  5                RBUF_LENGTH = .UP_COUNTER;
;    2923  5                XBUF_LENGTH = - ( .RBUF_LENGTH ^ -1 );
;    2924  5                INCR INDEX FROM 0 TO .UP_COUNTER - 1 DO
;    2925  5                  XMIT_BUFFER [ .INDEX ] = %B'01010101';
;    2926  5                INCR INDEX FROM .UP_COUNTER TO MAX_LENGTH - 1 DO
;    2927  5                  XMIT_BUFFER [ .INDEX ] = ZERO;
;    2928  5                UP_COUNTER = .UP_COUNTER + STEP1;
;    2929  5                COUNTER = ONE;
;    2930  5              END
;    2931  4            ELSE
;    2932  5              BEGIN
;    2933  5                RBUF_LENGTH = .DOWN_COUNTER;
;    2934  5                XBUF_LENGTH = - ( .RBUF_LENGTH ^ -1 );
;    2935  5                INCR INDEX FROM 0 TO .DOWN_COUNTER  - 1 DO
;    2936  5                  XMIT_BUFFER [ .INDEX ] =  %B'10101010';
;    2937  5                INCR INDEX FROM .DOWN_COUNTER TO MAX_LENGTH - 1 DO
;    2938  5                  XMIT_BUFFER [ .INDEX ] = ZERO;
;    2939  5                DOWN_COUNTER = .DOWN_COUNTER - STEP1;
;    2940  5                COUNTER = ZERO;
;    2941  4              END;
;    2942  4
;    2943  6          BGNSUB;
;    2944  6            SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2945  6            SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    2946  6            SEND_ELOOP_PACKET ( ZERO );
;    2947  6            COMPARE_PACKETS ( );
;    2948  4          ENDSUB;
;    2949  4
;    2950  3       END;
;    2951  1     ENDTST;


                                                .SBTTL  $T11 TEST 11 - PACKET LENGTH TEST
000000  004137  000000G                 $T11:   JSR     R1,$SAVE2                       ;                                2859
000004  005037  000000G                         CLR     COUNTER                         ;                                2912
000010  012737  000074  000000G                 MOV     #74,UP.COUNTER                  ;                                2913
```

```
000016  012737  002752  000000G             MOV     #2752,DOWN.COUNTER      ;                               2914
000024  012702  000074                      MOV     #74,R2                  ; *,INDEX1                      2916
000030  004737  000000G             1$:     JSR     PC,RESET.DEQNA          ;                               2918
000034  005737  000000G                     TST     COUNTER                 ;                               2919
000040  001033                             BNE     6$
000042  013700  000000G                     MOV     UP.COUNTER,R0           ;                               2922
000046  010037  000000G                     MOV     R0,RBUF.LENGTH
000052  005001                             CLR     R1                      ; INDEX                         2924
000054  000404                             BR      3$
000056  112761  000125  000000G     2$:     MOVB    #125,XMIT.BUFFER(R1)    ; *,*(INDEX)                    2925
000064  005201                             INC     R1                      ; INDEX                         2924
000066  020100                     3$:     CMP     R1,R0                   ; INDEX,*
000070  002772                             BLT     2$
000072  005300                             DEC     R0                      ;                               2926
000074  000402                             BR      5$
000076  105060  000000G             4$:     CLRB    XMIT.BUFFER(R0)         ; *(INDEX)                      2927
000102  005200                     5$:     INC     R0                      ; INDEX                         2926
000104  020027  002775                     CMP     R0,#2775                ; INDEX,*
000110  003772                             BLE     4$
000112  062737  000002  000000G             ADD     #2,UP.COUNTER           ;                               2928
000120  012737  000001  000000G             MOV     #1,COUNTER              ;                               2929
000126  000431                             BR      11$                     ;                               2919
000130  013700  000000G             6$:     MOV     DOWN.COUNTER,R0         ;                               2933
000134  010037  000000G                     MOV     R0,RBUF.LENGTH
000140  005001                             CLR     R1                      ; INDEX                         2935
000142  000404                             BR      8$
000144  112761  000252  000000G     7$:     MOVB    #252,XMIT.BUFFER(R1)    ; *,*(INDEX)                    2936
000152  005201                             INC     R1                      ; INDEX                         2935
000154  020100                     8$:     CMP     R1,R0                   ; INDEX,*
000156  002772                             BLT     7$
000160  005300                             DEC     R0                      ;                               2937
000162  000402                             BR      10$
000164  105060  000000G             9$:     CLRB    XMIT.BUFFER(R0)         ; *(INDEX)                      2938
000170  005200                     10$:    INC     R0                      ; INDEX                         2937
000172  020027  002775                     CMP     R0,#2775                ; INDEX,*
000176  003772                             BLE     9$
000200  162737  000002  000000G             SUB     #2,DOWN.COUNTER         ;                               2939
000206  005037  000000G                     CLR     COUNTER                 ;                               2940
000212  013700  000000G             11$:    MOV     RBUF.LENGTH,R0          ;                               2923
000216  006200                             ASR     R0
000220  005400                             NEG     R0
000222  010037  000000G                     MOV     R0,XBUF.LENGTH
000226  104402                     12$:    TRAP    2                       ;                               2941
000230  013746  000000G                     MOV     XBUF.LENGTH,-(SP)       ;                               2944
000234  012746  120000                     MOV     #-60000,-(SP)
000240  004737  000000G                     JSR     PC,SET.RDESCR.LIST
000244  013716  000000G                     MOV     XBUF.LENGTH,(SP)        ;                               2945
000250  012746  120000                     MOV     #-60000,-(SP)
000254  004737  000000G                     JSR     PC,SET.XDESCR.LIST
000260  005016                             CLR     (SP)                    ;                               2946
000262  004737  000000G                     JSR     PC,SEND.ELOOP.PACKET
000266  004737  000000G                     JSR     PC,COMPARE.PACKETS      ;                               2947
000272  062706  000006                     ADD     #6,SP                   ;                               2941
```

```
000276  104467                          TRAP    67                      ;                               2947
000300  006000                          ROR     R0
000302  103751                          BLO     12$
000304  062702  000002                  ADD     #2,R2                   ; *,INDEX1                      2916
000310  020227  002776                  CMP     R2,#2776                ; INDEX1,*
000314  003645                          BLE     1$
000316  000207                          RTS     PC                      ;                               2859

; Routine Size:  104 words,     Routine Base:  AB$CODE$ + 10110
; Maximum stack depth per invocation:  7 words



                                        .SBTTL  T11 TEST 11 - PACKET LENGTH TEST
000000  004737  010110'          T11::
000000                           1$:    JSR     PC,$T11                 ;                               2950
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 10430
; Maximum stack depth per invocation:  2 words


;   2952  1
```

```
;  2953  1     %SBTTL 'TEST 12 - NXM INTERRUPT TEST'
;  2954  1     !++
;  2955  1     !
;  2956  1     !  TEST 12:     NXM INTERRUPT TEST
;  2957  1     !
;  2958  1     !  DESCRIPTION:
;  2959  1     !
;  2960  1     !       This test verifies that Transmit and Receive List Invalid bits
;  2961  1     !       (CSR bits 4 and 5) can be set and reset as specified and that both,
;  2962  1     !       Transmit and Receive Descriptor List addresses in the I/O page have
;  2963  1     !       to be valid to succesfully loopback a packet.
;  2964  1     !
;  2965  1     !       After a software reset Transmit and Receive List Invalid bits are
;  2966  1     !       checked for their initial condition state (both set). Then these bits
;  2967  1     !       are cleared by writing Transmit and Receive Descriptor List addresses
;  2968  1     !       into Transmit and Receive Buffer Descriptor Registers.
;  2969  1     !
;  2970  1     !       First, valid loopback packet is sent to verify that UUT properly
;  2971  1     !       transmits and receives loopback packets. Then, a Non-Existant
;  2972  1     !       Memory  Access ( NI ) bit is forced to " 1 " each time an invalid
;  2973  1     !       loopback packet is sent.
;  2974  1     !
;  2975  1     !       If the operator specifies loop on error, the program re-executes the
;  2976  1     !       code that detected the error until ^C is entered.
;  2977  1     !
;  2978  1     !       Hardware tested:        Q-Bus to QTDC interface
;  2979  1     !                                       - Valid and invalid host memory
;  2980  1     !                                         address processing
;  2981  1     !                               CSR register - NXM access (bit 2)
;  2982  1     !                                       - Interrupt Enable (bit 6)
;  2983  1     !                                       - XMIT List Invalid (bit 4)
;  2984  1     !                                       - RCV  List Invalid (bit 5)
;  2985  1     !
;  2986  1     !       Use following Descriptor List and buffer addresses:
;  2987  1     !
;  2988  1     !
;  2989  1     !               TRANSMIT                                RECEIVE
;  2990  1     !               ********                                *******
;  2991  1     !
;  2992  1     !    DESCR LIST ADR     BUFFER ADR          DESCR LIST ADR     BUFFER ADR
;  2993  1     !    ----------------------------           ----------------------------
;  2994  1     !
;  2995  1     !        VALID              VALID                VALID              VALID
;  2996  1     !        INVALID            DON'T CARE           DON'T CARE         DON'T CARE
;  2997  1     !        VALID              INVALID              DON'T CARE         DON'T CARE
;  2998  1     !        VALID              VALID                INVALID            DON'T CARE
;  2999  1     !        VALID              VALID                VALID              INVALID
;  3000  1     !
;  3001  1     !     -  --  ---------------- ----------------------------------------------------
```

```
;    3002  1      !
;    3003  1      !       Processing:
;    3004  1      !
;    3005  1      !            BEGIN
;    3006  1      !               reset device ( disables device interrupt )
;    3007  1      !               select internal loopback mode
;    3008  1      !               read CSR
;    3009  1      !               IF XMIT and RCV List Invalid bits not = 1
;    3010  1      !               THEN
;    3011  1      !                  print error message if not inhibited
;    3012  1      !               ENDIF
;    3013  1      !               enable device interrupt (set CSR bit 6)
;    3014  1      !               transmit valid loopback packet
;    3015  1      !               check for expected loopback status
;    3016  1      !               IF error
;    3017  1      !               THEN
;    3018  1      !                  print error message if not inhibited
;    3019  1      !               ENDIF
;    3020  1      !               call compare_packets
;    3021  1      !               REPEAT for each set of addresses in the set
;    3022  1      .                  transmit invalid loopback packet
;    3023  1      !                  IF NXM interrupt didn't occured
;    3024  1      !                  THEN
;    3025  1      !                     print error message if not inhibited
;    3026  1      !                  ENDIF
;    3027  1      !                  check for expected loopback status
;    3028  1      !                  IF error
;    3029  1      !                  THEN
;    3030  1      !                     print error message if not inhibited
;    3031  1      !                  ENDIF
;    3032  1      !               ENDREPEAT
;    3033  1      !            END
;    3034  1      !--
```

```
;    3035  3     BGNTST;
;    3036  3
;    3037  3     !++
;    3038  3     !  RESET DEQNA AND SELECT LOOPBACK MODE
;    3039  3     !--
;    3040  3
;    3041  3     RESET_DEQNA ( );
;    3042  3
;    3043  3     PREP_FOR_SETUP ( );
;    3044  3     INCR INDEX FROM 1 TO 14 DO
;    3045  3       WRT_STATION_ADR ( .INDEX, PHA_INDEX );
;    3046  3
;    3047  5     BGNSUB;
;    3048  5       XMIT_SETUP_PACKET ( N_MODE );
;    3049  3     ENDSUB;
;    3050  3
;    3051  3     RBUF_LENGTH = 6;
;    3052  3     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3053  3
;    3054  3     CLR_BUFFERS ( B_SIZE );
;    3055  3     ERR_NUMBER = ZERO;
;    3056  3
;    3057  3     !++
;    3058  3     !  LOOPBACK A PACKET, VALID DESCRIPTORS AND BUFFER ADDRESSES, THEN CHECK IF
;    3059  3     !  LOOPBACK PACKET WAS PROPERLY RECEIVED AND NI BIT IN CSR = 0
;    3060  3     !--
;    3061  3
;    3062  3     RESET_DEQNA ( );
;    3063  3     WRT_STATION_ADR ( ZERO, PHA_INDEX );
;    3064  3
;    3065  5     BGNSUB;
;    3066  5       XMIT_ILOOP_PACKET ( ZERO );
;    3067  5       IF GET_BIT ( CSR, NI )
;    3068  5         THEN
;    3069  6           BEGIN
;    3070  6             CSR_WORD = GET_BIT ( CSR_ALL );
;    3071  6             PRINTB ( MSG59 );
;    3072  6             PRINTB ( MSG29 );
;    3073  6             PRINTB ( MSG28 );
;    3074  6             ERRDF  ( 1201, MSG00, ERROR$REPORT );
;    3075  5           END;
;    3076  3     ENDSUB;
;    3077  3
;    3078  3     !++
;    3079  3     !  TRY TO LOOPBACK A PACKET WITH INVALID TRANSMIT DESCRIPTOR ADDRESS,
;    3080  3     !  THEN CHECK FOR NON-EXISTANT MEMORY INTERRUPT ( NI ) BIT IS SET TO 1
;    3081  3     ! -
;    3082  3
;    3083  5     BGNSUB;
;    3084  5       RESET_DEQNA ( );
;    3085  5       .IOP_TABLE [ XLO_ADR ] = NXM_LO_ADR;
;    3086  5       .IOP_TABLE [ XHI_ADR ] = NXM_HI_ADR;
;    3087  5       IF NOT GET_BIT ( CSR, NI )
```

```
;  3088  5          THEN
;  3089  5            IF ( .XMIT_D_LIST [ FLGWD ] AND XFLG_MASK ) NEQU XFLG_MASK
;  3090  5              THEN
;  3091  6                BEGIN
;  3092  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;  3093  6                  PRINTB ( MSG59 );
;  3094  6                  PRINTB ( MSG29 );
;  3095  6                  PRINTB ( MSG27 );
;  3096  6                  ERRDF  ( 1202, MSG00, ERROR$REPORT );
;  3097  5                END;
;  3098  3      ENDSUB;
;  3099  3
;  3100  3      !++
;  3101  3      !  TRY TO LOOPBACK A PACKET WITH INVALID RECEIVE DESCRIPTOR ADDRESS,
;  3102  3      !  THEN CHECK IF NON-EXISTANT MEMORY INTERRUPT ( NI ) BIT IS SET TO 1
;  3103  3      !--
;  3104  3
;  3105  5      BGNSUB;
;  3106  5        RESET_DEQNA ( );
;  3107  5        WRT_STATION_ADR ( ZERO, PHA_INDEX );
;  3108  5
;  3109  5        .IOP_TABLE [ RLO_ADR ] = NXM_LO_ADR;
;  3110  5        .IOP_TABLE [ RHI_ADR ] = NXM_HI_ADR;
;  3111  5
;  3112  5        SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  3113  5        .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;  3114  5        .IOP_TABLE [ XHI_ADR ] = ZERO;
;  3115  5
;  3116  5        CHK_RIXI_STATUS ( ONE );
;  3117  5
;  3118  5        CHK_CSR_STATUS  ( %O'000220', %O'000220' );
;  3119  5        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );        ! O'140000', O'000400'
;  3120  5
;  3121  5        .IOP_TABLE [ CSR ] = EENABLE;
;  3122  5
;  3123  5        DELAY ( 20 );
;  3124  5        IF NOT GET_BIT ( CSR, NI )
;  3125  5          THEN
;  3126  5            IF ( .RCV_D_LIST [ FLGWD ] AND RFLG_MASK ) NEQU RFLG_MASK
;  3127  5              THEN
;  3128  6                BEGIN
;  3129  6                  .IOP_TABLE [ CSR ] = DISABLE;
;  3130  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;  3131  6                  PRINTB ( MSG59 );
;  3132  6                  PRINTB ( MSG29 );
;  3133  6                  PRINTB ( MSG27 );
;  3134  6                  ERRDF  ( 1203, MSG00, ERROR$REPORT );
;  3135  5                END;
;  3136  5        .IOP_TABLE [ CSR ] = DISABLE;
;  3137  3      ENDSUB;
;  3138  3
;  3139  3      !++
;  3140  3      !  TRY TO LOOPBACK A PACKET WITH INVALID TRANSMIT BUFFER ADDRESS,
```

```
;  3141  3     !  THEN CHECK IF NON-EXISTANT MEMORY INTERRUPT ( NI ) BIT IS SET TO 1
;  3142  3     !--
;  3143  3
;  3144  5     BGNSUB;
;  3145  5       RESET_DEQNA ( );
;  3146  5       SET_XDESCR_LIST ( .XBUF_LENGTH, VENXM );
;  3147  5       XMIT_D_LIST [ LOADR ] = NXM_LO_ADR;
;  3148  5       .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;  3149  5       .IOP_TABLE [ XHI_ADR ] = ZERO;
;  3150  5       DELAY ( 20 );
;  3151  5       IF NOT GET_BIT ( CSR, NI )
;  3152  5         THEN
;  3153  5           IF ( .XMIT_D_LIST [ FLGWD ] AND XFLG_MASK ) NEQU XFLG_MASK
;  3154  5             THEN
;  3155  6               BEGIN
;  3156  6                 CSR_WORD = GET_BIT ( CSR_ALL );
;  3157  6                 PRINTB ( MSG59 );
;  3158  6                 PRINTB ( MSG29 );
;  3159  6                 PRINTB ( MSG27 );
;  3160  6                 ERRDF  ( 1204, MSG00, ERROR$REPORT );
;  3161  5               END;
;  3162  3     ENDSUB;
;  3163  3
;  3164  3     !++
;  3165  3     !  TRY TO LOOPBACK A PACKET WITH INVALID RECEIVE BUFFER ADDRESS,
;  3166  3     !  THEN CHECK IF NON-EXISTANT MEMORY INTERRUPT ( NI ) BIT IS SET TO 1
;  3167  3     !--
;  3168  3
;  3169  5     BGNSUB;
;  3170  5       RESET_DEQNA ( );
;  3171  5
;  3172  5       SET_RDESCR_LIST ( .XBUF_LENGTH, VENXM );
;  3173  5       RCV_D_LIST [ LOADR ] = NXM_LO_ADR;
;  3174  5       .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;  3175  5       .IOP_TABLE [ RHI_ADR ] = ZERO;
;  3176  5
;  3177  5       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  3178  5       .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;  3179  5       .IOP_TABLE [ XHI_ADR ] = ZERO;
;  3180  5
;  3181  5       CHK_RIXI_STATUS ( ONE );
;  3182  5
;  3183  5       CHK_CSR_STATUS  ( %O'000220', %O'000220' );
;  3184  5       CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );        ! O'140000', O'000400'
;  3185  5
;  3186  5       .IOP_TABLE [ CSR ] = EENABLE;
;  3187  5
;  3188  5       DELAY ( 20 );
;  3189  5       IF NOT GET_BIT ( CSR, NI )
;  3190  5         THEN
;  3191  5           IF ( .RCV_D_LIST [ FLGWD ] AND RFLG_MASK ) NEQU RFLG_MASK
;  3192  5             THEN
;  3193  6               BEGIN
```

```
;   3194  6                 CSR_WORD = GET_BIT ( CSR_ALL );
;   3195  6                 .IOP_TABLE [ CSR ] = DISABLE;
;   3196  6                 PRINTB ( MSG59 );
;   3197  6                 PRINTB ( MSG29 );
;   3198  6                 PRINTB ( MSG27 );
;   3199  6                 ERRDF  ( 1205, MSG00, ERROR$REPORT );
;   3200  5               END;
;   3201  5       .IOP_TABLE [ CSR ] = DISABLE;
;   3202  3     ENDSUB;
;   3203  3
;   3204  1     ENDTST;


                                          .SBTTL  $T12 TEST 12 - NXM INTERRUPT TEST
000000  010146                    $T12:  MOV     R1,-(SP)                    ;                          2951
000002  162706  000026                    SUB     #26,SP
000006  004737  000000G                   JSR     PC,RESET.DEQNA              ;                          3041
000012  004737  000000G                   JSR     PC,PREP.FOR.SETUP           ;                          3043
000016  012701  000001                    MOV     #1,R1                       ; *,INDEX                  3044
000022  010146                    1$:    MOV     R1,-(SP)                    ; INDEX,*                  3045
000024  012746  000023                    MOV     #23,-(SP)
000030  004737  000000G                   JSR     PC,WRT.STATION.ADR
000034  022626                            CMP     (SP)+,(SP)+
000036  005201                            INC     R1                          ; INDEX                    3044
000040  020127  000016                    CMP     R1,#16                      ; INDEX,*
000044  003766                            BLE     1$
000046  104402                    2$:    TRAP    2                           ;                          3045
000050  012746  000200                    MOV     #200,-(SP)                  ;                          3048
000054  004737  000000G                   JSR     PC,XMIT.SETUP.PACKET
000060  005726                            TST     (SP)+                       ;                          3045
000062  104467                            TRAP    67                          ;                          3048
000064  006000                            ROR     R0
000066  103767                            BLO     2$
000070  012737  000006  000000G           MOV     #6,RBUF.LENGTH              ;                          3051
000076  012700  000006                    MOV     #6,R0                       ;                          3052
000102  006200                            ASR     R0
000104  005400                            NEG     R0
000106  010037  000000G                   MOV     R0,XBUF.LENGTH
000112  012746  004000                    MOV     #4000,-(SP)                 ;                          3054
000116  004737  000000G                   JSR     PC,CLR.BUFFERS
000122  005037  000000G                   CLR     ERR.NUMBER                  ;                          3055
000126  004737  000000G                   JSR     PC,RESET.DEQNA              ;                          3062
000132  005016                            CLR     (SP)                        ;                          3063
000134  012746  000023                    MOV     #23,-(SP)
000140  004737  000000G                   JSR     PC,WRT.STATION.ADR
000144  104402                    3$:    TRAP    2
000146  005016                            CLR     (SP)                        ;                          3066
000150  004737  000000G                   JSR     PC,XMIT.ILOOP.PACKET
000154  013700  000000G                   MOV     REG.ADR,R0                  ;                          3067
000160  016066  000016  000004           MOV     16(R0),4(SP)                ; *,TMP.LOCATION
000166  031766  000004                    BIT     (PC),4(SP)                  ; *,TMP.LOCATION
000172  001436                            BEQ     4$
000174  016666  000004  000006           MOV     4(SP),6(SP)                 ; *,TMP.LOCATION           3070
```

```
000202  016637  000006  000000G          MOV     6(SP),CSR.WORD          ; TMP.LOCATION,*
000210  012716  000000G                   MOV     #MSG59,(SP)             ;                       3071
000214  012746  000001                   MOV     #1,-(SP)
000220  010600                           MOV     SP,R0                   ; SP,*
000222  104414                           TRAP    14
000224  012716  000000G                   MOV     #MSG29,(SP)             ;                       3072
000230  012746  000001                   MOV     #1,-(SP)
000234  010600                           MOV     SP,R0                   ; SP,*
000236  104414                           TRAP    14
000240  012716  000000G                   MOV     #MSG28,(SP)             ;                       3073
000244  012746  000001                   MOV     #1,-(SP)
000250  010600                           MOV     SP,R0                   ; SP,*
000252  104414                           TRAP    14
000254  104455                           TRAP    55                      ;                       3074
000256  002261                           .WORD   2261
000260  000000G                         .WORD   MSG00
000262  000000G                         .WORD   ERROR$REPORT
000264  062706  000006                   ADD     #6,SP                   ;                       3069
000270  104467                   4$:     TRAP    67                      ;                       3075
000272  006000                           ROR     R0
000274  103723                           BLO     3$
000276  104402                   5$:     TRAP    2                       ;                       3076
000300  004737  000000G                   JSR     PC,RESET.DEQNA          ;                       3084
000304  012777  160000  000010G          MOV     #-20000,@IOP.TABLE+10   ;                       3085
000312  012777  000077  000012G          MOV     #77,@IOP.TABLE+12       ;                       3086
000320  013700  000000G                   MOV     REG.ADR,R0              ;                       3087
000324  016066  000016  000010           MOV     16(R0),10(SP)           ; *,TMP.LOCATION
000332  032766  000004  000010           BIT     #4,10(SP)               ; *,TMP.LOCATION
000340  001045                           BNE     6$
000342  013701  000000G                   MOV     XMIT.D.LIST,R1          ;                       3089
000346  042701  037777                   BIC     #37777,R1
000352  020127  140000                   CMP     R1,#-40000
000356  001436                           BEQ     6$
000360  016666  000010  000012           MOV     10(SP),12(SP)           ; *,TMP.LOCATION        3092
000366  016637  000012  000000G          MOV     12(SP),CSR.WORD         ; TMP.LOCATION,*
000374  012716  000000G                   MOV     #MSG59,(SP)             ;                       3093
000400  012746  000001                   MOV     #1,-(SP)
000404  010600                           MOV     SP,R0                   ; SP,*
000406  104414                           TRAP    14
000410  012716  000000G                   MOV     #MSG29,(SP)             ;                       3094
000414  012746  000001                   MOV     #1,-(SP)
000420  010600                           MOV     SP,R0                   ; SP,*
000422  104414                           TRAP    14
000424  012716  000000G                   MOV     #MSG27,(SP)             ;                       3095
000430  012746  000001                   MOV     #1,-(SP)
000434  010600                           MOV     SP,R0                   ; SP,*
000436  104414                           TRAP    14
000440  104455                           TRAP    55                      ;                       3096
000442  002262                           .WORD   2262
000444  000000G                         .WORD   MSG00
000446  000000G                         .WORD   ERROR$REPORT
000450  062706  000006                   ADD     #6,SP                   ;                       3091
000454  104467                   6$:     TRAP    67                      ;                       3097
```

```
000456  006000                                  ROR     R0
000460  103706                                  BLO     5$
000462  104402                          7$:     TRAP    2                       ;                               3098
000464  004737  000000G                         JSR     PC,RESET.DEQNA          ;                               3106
000470  005016                                  CLR     (SP)                    ;                               3107
000472  012746  000023                          MOV     #23,-(SP)
000476  004737  000000G                         JSR     PC,WRT.STATION.ADR
000502  012777  160000  000004G                 MOV     #-20000,@IOP.TABLE+4    ;                               3109
000510  012777  000077  000006G                 MOV     #77,@IOP.TABLE+6        ;                               3110
000516  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                               3112
000522  012746  120000                          MOV     #-60000,-(SP)
000526  004737  000000G                         JSR     PC,SET.XDESCR.LIST
000532  012777  000000G 000010G                 MOV     #XMIT.D.LIST,@IOP.TABLE+10  ;                           3113
000540  005077  000012G                         CLR     @IOP.TABLE+12           ;                               3114
000544  012716  000001                          MOV     #1,(SP)                 ;                               3116
000550  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
000554  012716  000220                          MOV     #220,(SP)               ;                               3118
000560  011646                                  MOV     (SP),-(SP)
000562  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000566  012716  140000                          MOV     #-40000,(SP)            ;                               3119
000572  012746  000400                          MOV     #400,-(SP)
000576  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
000602  012777  000001  000016G                 MOV     #1,@IOP.TABLE+16        ;                               3121
000610  012701  000024                          MOV     #24,R1                  ; *,$$TMP2                      3123
000614  001410                          8$:     BEQ     11$
000616  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000622  001403                                  BEQ     10$
000624  005066  000040                  9$:     CLR     40(SP)                  ; $$TMP
000630  077003                                  SOB     R0,9$                   ; $$TMP1,*
000632  005301                          10$:    DEC     R1                      ; $$TMP2
000634  000767                                  BR      8$
000636  013700  000000G                 11$:    MOV     REG.ADR,R0              ;                               3124
000642  016066  000016  000024                  MOV     16(R0),24(SP)           ; *,TMP.LOCATION
000650  032766  000004  000024                  BIT     #4,24(SP)               ; *,TMP.LOCATION
000656  001047                                  BNE     12$
000660  013701  000000G                         MOV     RCV.D.LIST,R1           ;                               3126
000664  042701  037777                          BIC     #37777,R1
000670  020127  140000                          CMP     R1,#-40000
000674  001440                                  BEQ     12$
000676  005077  000016G                         CLR     @IOP.TABLE+16           ;                               3129
000702  016666  000024  000026                  MOV     24(SP),26(SP)           ; *,TMP.LOCATION                3130
000710  016637  000026  000000G                 MOV     26(SP),CSR.WORD         ; TMP.LOCATION,*
000716  012716  000000G                         MOV     #MSG59,(SP)             ;                               3131
000722  012746  000001                          MOV     #1,-(SP)
000726  010600                                  MOV     SP,R0                   ; SP,*
000730  104414                                  TRAP    14
000732  012716  000000G                         MOV     #MSG29,(SP)             ;                               3132
000736  012746  000001                          MOV     #1,-(SP)
000742  010600                                  MOV     SP,R0                   ; SP,*
000744  104414                                  TRAP    14
000746  012716  000000G                         MOV     #MSG27,(SP)             ;                               3133
000752  012746  000001                          MOV     #1,-(SP)
000756  010600                                  MOV     SP,R0                   ; SP,*
```

```
000760  104414                                  TRAP    14
000762  104455                                  TRAP    55                      ;                               3134
000764  002263                                  .WORD   2263
000766  000000G                                 .WORD   MSG00
000770  000000G                                 .WORD   ERROR$REPORT
000772  062706  000006                          ADD     #6,SP                   ;                               3128
000776  005077  000016G                 12$:    CLR     @IOP.TABLE+16           ;                               3136
001002  062706  000010                          ADD     #10,SP                  ;                               3098
001006  104467                                  TRAP    67                      ;                               3136
001010  006000                                  ROR     R0
001012  103623                                  BLO     7$
001014  104402                          13$:    TRAP    2                       ;                               3137
001016  004737  000000G                         JSR     PC,RESET.DEQNA          ;                               3145
001022  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                               3146
001026  012746  120077                          MOV     #-57701,-(SP)
001032  004737  000000G                         JSR     PC,SET.XDESCR.LIST
001036  012737  160000  000004G                 MOV     #-20000,XMIT.D.LIST+4   ;                               3147
001044  012777  000000G 000010G                 MOV     #XMIT.D.LIST,@IOP.TABLE+10 ;                            3148
001052  005077  000012G                         CLR     @IOP.TABLE+12           ;                               3149
001056  012701  000024                          MOV     #24,R1                  ; *,$$TMP2                      3150
001062  001410                          14$:    BEQ     17$
001064  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
001070  001403                                  BEQ     16$
001072  005066  000032                  15$:    CLR     32(SP)                  ; $$TMP
001076  077003                                  SOB     R0,15$                  ; $$TMP1,*
001100  005301                          16$:    DEC     R1                      ; $$TMP2
001102  000767                                  BR      14$
001104  013700  000000G                 17$:    MOV     REG.ADR,R0              ;                               3151
001110  016066  000016  000022                  MOV     16(R0),22(SP)           ; *,TMP.LOCATION
001116  032766  000004  000022                  BIT     #4,22(SP)               ; *,TMP.LOCATION
001124  001045                                  BNE     18$
001126  013701  000000G                         MOV     XMIT.D.LIST,R1          ;                               3153
001132  042701  037777                          BIC     #37777,R1
001136  020127  140000                          CMP     R1,#-40000
001142  001436                                  BEQ     18$
001144  016666  000022  000024                  MOV     22(SP),24(SP)           ; *,TMP.LOCATION                3156
001152  016637  000024  000000G                 MOV     24(SP),CSR.WORD         ; TMP.LOCATION,*
001160  012716  000000G                         MOV     #MSG59,(SP)             ;                               3157
001164  012746  000001                          MOV     #1,-(SP)
001170  010600                                  MOV     SP,R0                   ; SP,*
001172  104414                                  TRAP    14
001174  012716  000000G                         MOV     #MSG29,(SP)             ;                               3158
001200  012746  000001                          MOV     #1,-(SP)
001204  010600                                  MOV     SP,R0                   ; SP,*
001206  104414                                  TRAP    14
001210  012716  000000G                         MOV     #MSG27,(SP)             ;                               3159
001214  012746  000001                          MOV     #1,-(SP)
001220  010600                                  MOV     SP,R0                   ; SP,*
001222  104414                                  TRAP    14
001224  104455                                  TRAP    55                      ;                               3160
001226  002264                                  .WORD   2264
001230  000000G                                 .WORD   MSG00
001232  000000G                                 .WORD   ERROR$REPORT
```

```
001234  062706  000006                          ADD     #6,SP                           ;                                       3155
001240  005726                          18$:    TST     (SP)+                           ;                                       3137
001242  104467                                  TRAP    67                              ;                                       3161
001244  006000                                  ROR     R0
001246  103662                                  BLO     13$
001250  104402                          19$:    TRAP    2                               ;                                       3162
001252  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                                       3170
001256  013716  000000G                         MOV     XBUF.LENGTH,(SP)                ;                                       3172
001262  012746  120077                          MOV     #-57701,-(SP)
001266  004737  000000G                         JSR     PC,SET.RDESCR.LIST
001272  012737  160000  000004G                 MOV     #-20000,RCV.D.LIST+4            ;                                       3173
001300  012777  000000G 000004G                 MOV     #RCV.D.LIST,@IOP.TABLE+4        ;                                       3174
001306  005077  000006G                         CLR     @IOP.TABLE+6                    ;                                       3175
001312  013716  000000G                         MOV     XBUF.LENGTH,(SP)                ;                                       3177
001316  012746  120000                          MOV     #-60000,-(SP)
001322  004737  000000G                         JSR     PC,SET.XDESCR.LIST
001326  012777  000000G 000010G                 MOV     #XMIT.D.LIST,@IOP.TABLE+10      ;                                       3178
001334  005077  000012G                         CLR     @IOP.TABLE+12                   ;                                       3179
001340  012716  000001                          MOV     #1,(SP)                         ;                                       3181
001344  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
001350  012716  000220                          MOV     #220,(SP)                       ;                                       3183
001354  011646                                  MOV     (SP),-(SP)
001356  004737  000000G                         JSR     PC,CHK.CSR.STATUS
001362  012716  140000                          MOV     #-40000,(SP)                    ;                                       3184
001366  012746  000400                          MOV     #400,-(SP)
001372  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
001376  012777  000001  000016G                 MOV     #1,@IOP.TABLE+16                ;                                       3186
001404  012701  000024                          MOV     #24,R1                          ; *,$$TMP2                              3188
001410  001410                          20$:    BEQ     23$
001412  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
001416  001403                                  BEQ     22$
001420  005066  000040                  21$:    CLR     40(SP)                          ; $$TMP
001424  077003                                  SOB     R0,21$                          ; $$TMP1,*
001426  005301                          22$:    DEC     R1                              ; $$TMP2
001430  000767                                  BR      20$
001432  013700  000000G                 23$:    MOV     REG.ADR,R0                      ;                                       3189
001436  016066  000016  000034                  MOV     16(R0),34(SP)                   ; *,TMP.LOCATION
001444  032766  000004  000034                  BIT     #4,34(SP)                       ; *,TMP.LOCATION
001452  001047                                  BNE     24$
001454  013701  000000G                         MOV     RCV.D.LIST,R1                   ;                                       3191
001460  042701  037777                          BIC     #37777,R1
001464  020127  140000                          CMP     R1,#-40000
001470  001440                                  BEQ     24$
001472  016666  000034  000036                  MOV     34(SP),36(SP)                   ; *,TMP.LOCATION                        3194
001500  016637  000036  000000G                 MOV     36(SP),CSR.WORD                 ; TMP.LOCATION,*
001506  005077  000016G                         CLR     @IOP.TABLE+16                   ;                                       3195
001512  012716  000000G                         MOV     #MSG59,(SP)                     ;                                       3196
001516  012746  000001                          MOV     #1,-(SP)
001522  010600                                  MOV     SP,R0                           ; SP,*
001524  104414                                  TRAP    14
001526  012716  000000G                         MOV     #MSG29,(SP)                     ;                                       3197
001532  012746  000001                          MOV     #1,-(SP)
001536  010600                                  MOV     SP,R0                           ; SP,*
```

```
001540  104414                                  TRAP    14
001542  012716  000000G                         MOV     #MSG27,(SP)                 ;                                       3198
001546  012746  000001                          MOV     #1,-(SP)
001552  010600                                  MOV     SP,R0                       ; SP,*
001554  104414                                  TRAP    14
001556  104455                                  TRAP    55                          ;                                       3199
001560  002265                                  .WORD   2265
001562  000000G                                 .WORD   MSG00
001564  000000G                                 .WORD   ERROR$REPORT
001566  062706  000006                          ADD     #6,SP                       ;                                       3193
001572  005077  000016G                 24$:    CLR     @IOP.TABLE+16               ;                                       3201
001576  062706  000010                          ADD     #10,SP                      ;                                       3162
001602  104467                                  TRAP    67                          ;                                       3201
001604  006000                                  ROR     R0
001606  103620                                  BLO     19$
001610  062706  000032                          ADD     #32,SP                                                              2951
001614  012601                                  MOV     (SP)+,R1
001616  000207                                  RTS     PC

; Routine Size:  456 words,     Routine Base:  AB$CODE$ + 10444
; Maximum stack depth per invocation:  23 words




                                                .SBTTL  T12 TEST 12 - NXM INTERRUPT TEST
000000  004737  010444'                 T12::
000000                                  1$:     JSR     PC,$T12                     ;                                       3202
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 12264
; Maximum stack depth per invocation:  2 words


;   3205  1
```

```
;    3206  1     %SBTTL 'TEST 13 - MULTIPLE AND CHAINED PACKET TEST'
;    3207  1     !++
;    3208  1     !
;    3209  1     !  TEST 13:     MULTIPLE AND CHAINED PACKET TEST
;    3210  1     !
;    3211  1     !  DESCRIPTION:
;    3212  1     !
;    3213  1     !       This test verifies that the DEQNA can transmit and receive multiple,
;    3214  1     !       linked and chained loopback packets.
;    3215  1     !
;    3216  1     !       If the operator specifies loop on error, the program re-executes the
;    3217  1     !       code that detected the  error until ^C is entered.
;    3218  1     !
;    3219  1     !       Hardware tested:
;    3220  1     !
;    3221  1     !       Processing:
;    3222  1     !
;    3223  1     !            BEGIN
;    3224  1     !               reset device
;    3225  1     !               select internal/extended loopback mode
;    3226  1     !               transmit simple loopback packet
;    3227  1     !               check for expected loopback status
;    3228  1     !               IF error
;    3229  1     !               THEN
;    3230  1     !                  print error message if not inhibited
;    3231  1     !               ENDIF
;    3232  1     !               call compare_packets
;    3233  1     !
;    3234  1     !               transmit multiple, linked and chaind loopback packet
;    3235  1     !               check for expected loopback status
;    3236  1     !               IF error
;    3237  1     !               THEN
;    3238  1     !                  print error message if not inhibited
;    3239  1     !               ENDIF
;    3240  1     !               call compare_packets
;    3241  1     !            END
;    3242  1     !--
```

```
;  3243  3     BGNTST;
;  3244  3
;  3245  3     RBUF_LENGTH = 64;
;  3246  3     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  3247  3
;  3248  3     !++
;  3249  3     !  LOOPBACK UNCHAINED PACKET, THEN CHECK IF IT WAS PROPERLY RECEIVED
;  3250  3     !--
;  3251  3
;  3252  3     RESET_DEQNA ( );
;  3253  3     INCR INDEX FROM 0 TO 63 DO
;  3254  3       XMIT_BUFFER [ .INDEX ] = .INDEX;
;  3255  3
;  3256  5     BGNSUB;
;  3257  5       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  3258  5       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  3259  5       SEND_ELOOP_PACKET ( ZERO );
;  3260  5       COMPARE_PACKETS ( );
;  3261  3     ENDSUB;
;  3262  3
;  3263  3     RESET_DEQNA ( );
;  3264  3     CLR_BUFFERS ( 512 );
;  3265  3     INCR INDEX FROM 0 TO 383 DO
;  3266  3       XMIT_BUFFER [ .INDEX ] = .INDEX;
;  3267  3
;  3268  3
;  3269  5     BGNSUB;
;  3270  5       INCR INDEX FROM 0 TO 63 DO
;  3271  5         RCV_D_LIST [ .INDEX, W_LEN ] = .RD13 [ .INDEX ];
;  3272  5       INCR INDEX FROM 0 TO 31 DO
;  3273  5         XMIT_D_LIST [ .INDEX, W_LEN ] = .TD13 [ .INDEX ];
;  3274  5
;  3275  5       XMIT_D_LIST [  7, W_LEN ] = VE;
;  3276  5       XMIT_D_LIST [ 13, W_LEN ] = E;
;  3277  5
;  3278  5       PUT_BIT [ CSR, LB, INX_LOOPBACK ];
;  3279  5       XMIT_AND_RCV_PACKET ( );
;  3280  5       CHK_RIXI_STATUS ( ZERO );
;  3281  5       CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );     ! 0'100220', 0'100220'
;  3282  5
;  3283  5       XMIT_D_LIST [  7, W_LEN ] = V;
;  3284  5       XMIT_D_LIST [ 12, W_LEN ] = NEWB;
;  3285  5       XMIT_D_LIST [ 13, W_LEN ] = V;
;  3286  5
;  3287  5       .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST + 24;
;  3288  5       .IOP_TABLE [ XHI_ADR ] = ZERO;
;  3289  5
;  3290  5       CHK_RIXI_STATUS ( ZERO );
;  3291  5       CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );     ! 0'100220', 0'100220'
;  3292  5
;  3293  5       !++
;  3294  5       !  CHECK IF RECEIVE BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;  3295  5       !--
```

```
;  3296  5
;  3297  5        INCR INDEX FROM 0 TO 53 DO
;  3298  5          IF .RCV_D_LIST [ .INDEX, W_LEN ] NEQU .RD13 [ .INDEX ]
;  3299  5             AND ( .RCV_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;  3300  5             AND .RCV_D_LIST [ .INDEX, W_LEN ] NEQU %C'020600'
;  3301  5               THEN
;  3302  6                 BEGIN
;  3303  6                   CSR_WORD = GET_BIT ( CSR_ALL );
;  3304  6                   PRINTB ( MSG59 );
;  3305  6                   PRINTB ( MSG48 );
;  3306  6                   PRINTB ( MSG50, .RCV_D_LIST [ .INDEX, W_LEN ], .RD13 [ .INDEX ], .INDEX );
;  3307  6                   ERRDF  ( 1301, MSG00, ERROR$REPORT );
;  3308  5                 END;
;  3309  5
;  3310  5        !++
;  3311  5        !  CHECK IF TRANSMIT BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;  3312  5        !--
;  3313  5
;  3314  5        INCR INDEX FROM 0 TO 23 DO
;  3315  5          IF .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU .TD13 [ .INDEX ]
;  3316  5             AND ( .XMIT_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;  3317  5             AND .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU %O'020414'
;  3318  5             AND .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU %O'004140'
;  3319  5               THEN
;  3320  6                 BEGIN
;  3321  6                   CSR_WORD = GET_BIT ( CSR_ALL );
;  3322  6                   PRINTB ( MSG59 );
;  3323  6                   PRINTB ( MSG49 );
;  3324  6                   PRINTB ( MSG50, .XMIT_D_LIST [ .INDEX, W_LEN ], .TD13 [ .INDEX ], .INDEX );
;  3325  6                   ERRDF  ( 1302, MSG00, ERROR$REPORT );
;  3326  5                 END;
;  3327  5
;  3328  5        INCR INDEX FROM 0 TO 5 DO
;  3329  6          BEGIN
;  3330  6            XMIT_D_LIST [ .INDEX, W_LEN ] = .XMIT_D_LIST [ .INDEX + 24, W_LEN ];
;  3331  6            RCV_D_LIST  [ .INDEX, W_LEN ] = .RCV_D_LIST [ .INDEX + 54, W_LEN ];
;  3332  5          END;
;  3333  5
;  3334  5        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );  ! O'140000', O'000400'
;  3335  5        CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS  );  ! O'140000', O'020000'
;  3336  5
;  3337  5        INCR INDEX FROM 0 TO 383 DO
;  3338  5          IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;  3339  5            THEN
;  3340  6              BEGIN
;  3341  6                CSR_WORD = GET_BIT ( CSR_ALL );
;  3342  6                PRINTB ( MSG59 );
;  3343  6                PRINTB ( MSG51 );
;  3344  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;  3345  6                ERRDF  ( 1303, MSG00, ERROR$REPORT );
;  3346  5              END;
;  3347  3      ENDSUB;
;  3348  3
```

```
;  3349  1     ENDTST;


                                          .SBTTL  $T13 TEST 13 - MULTIPLE AND CHAINED PACKET TEST
000000  004137  000000G                $T13:   JSR     R1,$SAVE3                       ;                                       3204
000004  162706  000006                        SUB     #6,SP
000010  012737  000100  000000G               MOV     #100,RBUF.LENGTH                ;                                       3245
000016  012700  000100                        MOV     #100,R0                         ;                                       3246
000022  006200                                ASR     R0
000024  005400                                NEG     R0
000026  010037  000000G                       MOV     R0,XBUF.LENGTH
000032  004737  000000G                       JSR     PC,RESET.DEQNA                  ;                                       3252
000036  005000                                CLR     R0                              ; INDEX                                 3253
000040  110060  000000G               1$:     MOVB    R0,XMIT.BUFFER(R0)              ; INDEX,*(INDEX)                        3254
000044  005200                                INC     R0                              ; INDEX                                 3253
000046  020027  000077                        CMP     R0,#77                          ; INDEX,*
000052  003772                                BLE     1$
000054  104402                        2$:     TRAP    2                               ;                                       3254
000056  013746  000000G                       MOV     XBUF.LENGTH,-(SP)               ;                                       3257
000062  012746  120000                        MOV     #-60000,-(SP)
000066  004737  000000G                       JSR     PC,SET.RDESCR.LIST
000072  013716  000000G                       MOV     XBUF.LENGTH,(SP)                ;                                       3258
000076  012746  120000                        MOV     #-60000,-(SP)
000102  004737  000000G                       JSR     PC,SET.XDESCR.LIST
000106  005016                                CLR     (SP)                            ;                                       3259
000110  004737  000000G                       JSR     PC,SEND.ELOOP.PACKET
000114  004737  000000G                       JSR     PC,COMPARE.PACKETS              ;                                       3260
000120  062706  000006                        ADD     #6,SP                           ;                                       3254
000124  104467                                TRAP    67                              ;                                       3260
000126  006000                                ROR     R0
000130  103751                                BLO     2$
000132  004737  000000G                       JSR     PC,RESET.DEQNA                  ;                                       3263
000136  012746  001000                        MOV     #1000,-(SP)                     ;                                       3264
000142  004737  000000G                       JSR     PC,CLR.BUFFERS
000146  005000                                CLR     R0                              ; INDEX                                 3265
000150  110060  000000G               3$:     MOVB    R0,XMIT.BUFFER(R0)              ; INDEX,*(INDEX)                        3266
000154  005200                                INC     R0                              ; INDEX                                 3265
000156  020027  000577                        CMP     R0,#577                         ; INDEX,*
000162  003772                                BLE     3$
000164  104402                        4$:     TRAP    2                               ;                                       3266
000166  005000                                CLR     R0                              ; INDEX                                 3270
000170  016060  000000G 000000G       5$:     MOV     RD13(R0),RCV.D.LIST(R0)         ; *(INDEX),*(INDEX)                     3271
000176  062700  000002                        ADD     #2,R0                           ; *,INDEX                               3270
000202  020027  000176                        CMP     R0,#176                         ; INDEX,*
000206  003770                                BLE     5$
000210  005000                                CLR     R0                              ; INDEX                                 3272
000212  016060  000000G 000000G       6$:     MOV     TD13(R0),XMIT.D.LIST(R0)        ; *(INDEX),*(INDEX)                     3273
000220  062700  000002                        ADD     #2,R0                           ; *,INDEX                               3272
000224  020027  000076                        CMP     R0,#76                          ; INDEX,*
000230  003770                                BLE     6$
000232  012737  120000  000016G               MOV     #-60000,XMIT.D.LIST+16          ;                                       3275
000240  012737  020000  000032G               MOV     #20000,XMIT.D.LIST+32           ;                                       3276
000246  013700  000000G                       MOV     REG.ADR,R0                      ;                                       3278
```

```
000252  042760  001400  000016                  BIC     #1400,16(R0)
000260  052760  001000  000016                  BIS     #1000,16(R0)
000266  004737  000000G                         JSR     PC,XMIT.AND.RCV.PACKET          ;                               3279
000272  005016                                  CLR     (SP)                            ;                               3280
000274  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
000300  012716  100220                          MOV     #-77560,(SP)                    ;                               3281
000304  011646                                  MOV     (SP),-(SP)
000306  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000312  012737  100000  000016G                 MOV     #-100000,XMIT.D.LIST+16         ;                               3283
000320  012737  100000  000030G                 MOV     #-100000,XMIT.D.LIST+30         ;                               3284
000326  012737  100000  000032G                 MOV     #-100000,XMIT.D.LIST+32         ;                               3285
000334  012777  000030G 000010G                 MOV     #XMIT.D.LIST+30,@IOP.TABLE+10   ;                               3287
000342  005077  000012G                         CLR     @IOP.TABLE+12                   ;                               3288
000346  005016                                  CLR     (SP)                            ;                               3290
000350  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
000354  012716  100220                          MOV     #-77560,(SP)                    ;                               3291
000360  011646                                  MOV     (SP),-(SP)
000362  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000366  005003                                  CLR     R3                              ; INDEX                         3297
000370  010301                          7$:     MOV     R3,R1                           ; INDEX,*                       3298
000372  006301                                  [illegible] R1
000374  016100  000000G                         [illegible] RCV.D.LIST(R1),R0
000400  020061  000000G                         [illegible] R0,RD13(R1)
000404  001456                                  BEQ     8$
000406  010002                                  MOV     R0,R2                           ;                               3299
000410  042702  037777                          BIC     #37777,R2
000414  020227  140000                          CMP     R2,#-40000
000420  001450                                  BEQ     8$
000422  020027  020600                          CMP     R0,#20600                       ;                               3300
000426  001445                                  BEQ     8$
000430  013700  000000G                         MOV     REG.ADR,R0                      ;                               3303
000434  016066  000016  000006                  MOV     16(R0),6(SP)                    ; *,TMP.LOCATION
000442  016637  000006  000000G                 MOV     6(SP),CSR.WORD                  ; TMP.LOCATION,*
000450  012716  000000G                         MOV     #MSG59,(SP)                     ;                               3304
000454  012746  000001                          MOV     #1,-(SP)
000460  010600                                  MOV     SP,R0                           ; SP,*
000462  104414                                  TRAP    14
000464  012716  000000G                         MOV     #MSG48,(SP)                     ;                               3305
000470  012746  000001                          MOV     #1,-(SP)
000474  010600                                  MOV     SP,R0                           ; SP,*
000476  104414                                  TRAP    14
000500  010316                                  MOV     R3,(SP)                         ; INDEX,*                       3306
000502  016146  000000G                         MOV     RD13(R1),-(SP)
000506  016146  000000G                         MOV     RCV.D.LIST(R1),-(SP)
000512  012746  000000G                         MOV     #MSG50,-(SP)
000516  012746  000004                          MOV     #4,-(SP)
000522  010600                                  MOV     SP,R0                           ; SP,*
000524  104414                                  TRAP    14
000526  104455                                  TRAP    55                              ;                               3307
000530  002425                                  .WORD   2425
000532  000000G                                 .WORD   MSG00
000534  000000G                                 .WORD   ERROR$REPORT
000536  062706  000014                          ADD     #14,SP                          ;                               3302
```

```
000542  005203                          8$:     INC     R3                              ; INDEX                         3297
000544  020327  000065                          CMP     R3,#65                          ; INDEX,*
000550  003707                                  BLE     7$
000552  005003                                  CLR     R3                              ; INDEX                         3314
000554  010301                          9$:     MOV     R3,R1                           ; INDEX,*                       3315
000556  006301                                  ASL     R1
000560  016100  000000G                         MOV     XMIT.D.LIST(R1),R0
000564  020061  000000G                         CMP     R0,TD13(R1)
000570  001461                                  BEQ     10$
000572  010002                                  MOV     R0,R2                           ;                               3316
000574  042702  037777                          BIC     #37777,R2
000600  020227  140000                          CMP     R2,#-40000
000604  001453                                  BEQ     10$
000606  020027  020414                          CMP     R0,#20414                       ;                               3317
000612  001450                                  BEQ     10$
000614  020027  004140                          CMP     R0,#4140                        ;                               3318
000620  001445                                  BEQ     10$
000622  013700  000000G                         MOV     REG.ADR,R0                      ;                               3321
000626  016066  000016  000010                  MOV     16(R0),10(SP)                   ; *,TMP.LOCATION
000634  016637  000010  000000G                 MOV     10(SP),CSR.WORD                 ; TMP.LOCATION,*
000642  012716  000000G                         MOV     #MSG59,(SP)                     ;                               3322
000646  012746  000001                          MOV     #1,-(SP)
000652  010600                                  MOV     SP,R0                           ; SP,*
000654  104414                                  TRAP    14
000656  012716  000000G                         MOV     #MSG49,(SP)                     ;                               3323
000662  012746  000001                          MOV     #1,-(SP)
000666  010600                                  MOV     SP,R0                           ; SP,*
000670  104414                                  TRAP    14
000672  010316                                  MOV     R3,(SP)                         ; INDEX,*                       3324
000674  016146  000000G                         MOV     TD13(R1),-(SP)
000700  016146  000000G                         MOV     XMIT.D.LIST(R1),-(SP)
000704  012746  000000G                         MOV     #MSG50,-(SP)
000710  012746  000004                          MOV     #4,-(SP)
000714  010600                                  MOV     SP,R0                           ; SP,*
000716  104414                                  TRAP    14
000720  104455                                  TRAP    55                              ;                               3325
000722  002426                                  .WORD   2426
000724  000000G                                 .WORD   MSG00
000726  000000G                                 .WORD   ERROR$REPORT
000730  062706  000014                          ADD     #14,SP                          ;                               3320
000734  005203                          10$:    INC     R3                              ; INDEX                         3314
000736  020327  000027                          CMP     R3,#27                          ; INDEX,*
000742  003704                                  BLE     9$
000744  005002                                  CLR     R2                              ; INDEX                         3328
000746  010200                          11$:    MOV     R2,R0                           ; INDEX,*                       3330
000750  006300                                  ASL     R0
000752  010201                                  MOV     R2,R1                           ; INDEX,*
000754  006301                                  ASL     R1
000756  016160  000060G 000000G                 MOV     XMIT.D.LIST+60(R1),XMIT.D.LIST(R0) ;
000764  010201                                  MOV     R2,R1                           ; INDEX,*                       3331
000766  006301                                  ASL     R1
000770  016160  000154G 000000G                 MOV     RCV.D.LIST+154(R1),RCV.D.LIST(R0) ;
000776  005202                                  INC     R2                              ; INDEX                         3328
```

```
001000  020227  000005                          CMP     R2,#5                           ; INDEX,*
001004  003760                                  BLE     11$
001006  012716  140000                          MOV     #-40000,(SP)                    ;                                       3334
001012  012746  000400                          MOV     #400,-(SP)
001016  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
001022  012716  140000                          MOV     #-40000,(SP)                    ;                                       3335
001026  012746  020000                          MOV     #20000,-(SP)
001032  004737  000000G                         JSR     PC,CHK.RCV.STATUS
001036  005001                                  CLR     R1                              ; INDEX                                 3337
001040  126161  000000G 000000G          12$:   CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                     3338
001046  001447                                  BEQ     13$
001050  013700  000000G                         MOV     REG.ADR,R0                      ;                                       3341
001054  016066  000016  000016                  MOV     16(R0),16(SP)                   ; *,TMP.LOCATION
001062  016637  000016  000000G                 MOV     16(SP),CSR.WORD                 ; TMP.LOCATION,*
001070  012716  000000G                         MOV     #MSG59,(SP)                     ;                                       3342
001074  012746  000001                          MOV     #1,-(SP)
001100  010600                                  MOV     SP,R0                           ; SP,*
001102  104414                                  TRAP    14
001104  012716  000000G                         MOV     #MSG51,(SP)                     ;                                       3343
001110  012746  000001                          MOV     #1,-(SP)
001114  010600                                  MOV     SP,R0                           ; SP,*
001116  104414                                  TRAP    14
001120  010116                                  MOV     R1,(SP)                         ; INDEX,*                               3344
001122  005046                                  CLR     -(SP)
001124  116116  000000G                         MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
001130  005046                                  CLR     -(SP)
001132  116116  000000G                         MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
001136  012746  000000G                         MOV     #MSG50,-(SP)
001142  012746  000004                          MOV     #4,-(SP)
001146  010600                                  MOV     SP,R0                           ; SP,*
001150  104414                                  TRAP    14
001152  104455                                  TRAP    55                              ;                                       3345
001154  002427                                  .WORD   2427
001156  000000G                                 .WORD   MSG00
001160  000000G                                 .WORD   ERROR$REPORT
001162  062706  000014                          ADD     #14,SP                          ;                                       3340
001166  005201                          13$:    INC     R1                              ; INDEX                                 3337
001170  020127  000577                          CMP     R1,#577                         ; INDEX,*
001174  003721                                  BLE     12$
001176  062706  000010                          ADD     #10,SP                          ;                                       3266
001202  104467                                  TRAP    67                              ;                                       3346
001204  006000                                  ROR     R0
001206  103002                                  BHIS    14$
001210  000137  012464'                         JMP     4$
001214  062706  000010                  14$:    ADD     #10,SP                          ;                                       3204
001220  000207                                  RTS     PC

; Routine Size:  329 words,     Routine Base:  AB$CODE$ + 12300
; Maximum stack depth per invocation:  20 words
```

```
                                        .SBTTL  T13 TEST 13 - MULTIPLE AND CHAINED PACKET TEST
000000  004737  012300'                T13::
000000                                  1$:     JSR     PC,$T13                         ;                       3347
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 13522
; Maximum stack depth per invocation:  2 words


;   3350  1
```

```
;    3351  1     %SBTTL 'TEST 14 - DMA TIMING TEST'
;    3352  1     !++
;    3353  1     !
;    3354  1     !  TEST 14:     DMA TIMING TEST
;    3355  1     !
;    3356  1     !  DESCRIPTION:
;    3357  1     !
;    3358  1     !       This test verifies that the DMA transfer completes within 'X' msec.
;    3359  1     !       Chained and linked 1514 byte loopback packet is used to accomplish
;    3360  1     !       this test. If the operator specifies loop on error, the program
;    3361  1     !       re-executes the code that detected the error until ^C is entered.
;    3362  1     !
;    3363  1     !       NOTE: An answer to the following software question
;    3364  1     !
;    3365  1     !             SYSTEM HAS BLOCK MODE MEMORY (L)?
;    3366  1     !
;    3367  1     !             determines the value for 'X'.
;    3368  1     !
;    3369  1     !       Hardware tested:   Internal/Extended loopback
;    3370  1     !                          Transmit status - last descriptor in chain (bit 15)
;    3371  1     !                          Receive  status - last descriptor in chain (bit 15)
;    3372  1     !                                          - error summary (bit 14)
;    3373  1     !       Processing:
;    3374  1     !
;    3375  1     !            BEGIN
;    3376  1     !               reset device
;    3377  1     !               select internal/extended loopback mode
;    3378  1     !               set the timeout timer to 'X' msec
;    3379  1     !               transmit chained loopback packet
;    3380  1     !               start the timer
;    3381  1     !               IF timeout
;    3382  1     !               THEN
;    3383  1     !                  print error message if not inhibited
;    3384  1     !               ENDIF
;    3385  1     !               check for expected loopback status
;    3386  1     !               IF error
;    3387  1     !               THEN
;    3388  1     !                  print error message if not inhibited
;    3389  1     !               ENDIF
;    3390  1     !               call compare_packets
;    3391  1     !            END
;    3392  1     !--
```

```
;    3393  3      BGNTST;
;    3394  3
;    3395  3      RBUF_LENGTH = LEGAL_LENGTH;
;    3396  3      XBUF_LENGTH = - ( .RBUF_LENGTH ^ -1 );
;    3397  3      INCR INDEX FROM 0 TO LEGAL_LENGTH - 1 DO
;    3398  3        XMIT_BUFFER [ .INDEX ] = .INDEX;
;    3399  3
;    3400  5      BGNSUB;
;    3401  5        RESET DEQNA ( );
;    3402  5        INCR INDEX FROM 0 TO 63 DO
;    3403  5          RCV_D_LIST [ .INDEX, W_LEN ] = .RD13 [ .INDEX ];
;    3404  5        INCR INDEX FROM 0 TO 31 DO
;    3405  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .TD13 [ .INDEX ];
;    3406  5
;    3407  5        TEMP5 = .XMIT_D_LIST [ 27, W_LEN ];
;    3408  5        TEMP6 = .RCV_D_LIST  [ 51, W_LEN ];
;    3409  5        TEMP7 = .RCV_D_LIST  [ 56, W_LEN ];
;    3410  5
;    3411  5        XMIT_D_LIST [ 27, W_LEN ] = -628;
;    3412  5        RCV_D_LIST  [ 51, W_LEN ] = -625;
;    3413  5        RCV_D_LIST  [ 56, W_LEN ] = RCV_BUFFER + LEGAL_LENGTH - 2;
;    3414  5
;    3415  5        PUT_BIT [ CSR, LB, INX_LOOPBACK ];
;    3416  5        XMIT_AND_RCV_PACKET ( );
;    3417  5
;    3418  5        CHK_RIXI_STATUS ( ONE );
;    3419  5
;    3420  5        IF .SWP_BLOCK_MEM EQLU ONE
;    3421  5          THEN
;    3422  5            TEMP4 = %O'367'            ! ADDED 25% TO "305" TO GET "367". FIX FOR $$$
;    3423  5          ELSE                         ! CHANGE FROM 15 MHZ TO 18 MHZ CPU, BY HLM. $$$
;    3424  5            TEMP4 = 4 * %O'367';       ! $$$
;    3425  5
;    3426  5        IF .TEMP1 GTRU .TEMP4
;    3427  5          THEN
;    3428  6            BEGIN
;    3429  6              CSR_WORD = GET_BIT ( CSR_ALL );
;    3430  6              PRINTB ( MSG59 );
;    3431  6              PRINTB ( MSG52 );
;    3432  6              ERRDF  ( 1401, MSG00, ERROR$REPORT );
;    3433  5            END;
;    3434  5
;    3435  5        CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );     ! O'100220', O'100220'
;    3436  5
;    3437  5        XMIT_D_LIST [ 27, W_LEN ] = .TEMP5;
;    3438  5        RCV_D_LIST  [ 51, W_LEN ] = .TEMP6;
;    3439  5        RCV_D_LIST  [ 56, W_LEN ] = .TEMP7;
;    3440  5
;    3441  5        !++
;    3442  5        !  CHECK IF TRANSMIT BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;    3443  5        !--
;    3444  5        INCR INDEX FROM 0 TO 23 DO
;    3445  5          IF .XMIT_D LIST [ .INDEX, W_LEN ] NEQU .TD13 [ .INDEX ]
```

```
;    3446  5            AND ( .XMIT_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;    3447  5              THEN
;    3448  6                BEGIN
;    3449  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;    3450  6                  PRINTB ( MSG59 );
;    3451  6                  PRINTB ( MSG49 );
;    3452  6                  PRINTB ( MSG50, .XMIT_D_LIST [ .INDEX, W_LEN ], .TD13 [ .INDEX ], .INDEX );
;    3453  6                  ERRDF  ( 1402, MSG00, ERROR$REPORT );
;    3454  5                END;
;    3455  5
;    3456  5          !++
;    3457  5          !  CHECK IF RECEIVE BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;    3458  5          !--
;    3459  5          INCR INDEX FROM 0 TO 53 DO
;    3460  5            IF .RCV_D_LIST [ .INDEX, W_LEN ] NEQU .RD13 [ .INDEX ]
;    3461  5              AND ( .RCV_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;    3462  5                THEN
;    3463  6                  BEGIN
;    3464  6                    CSR_WORD = GET_BIT ( CSR_ALL );
;    3465  6                    PRINTB ( MSG59 );
;    3466  6                    PRINTB ( MSG48 );
;    3467  6                    PRINTB ( MSG50, .RCV_D_LIST [ .INDEX, W_LEN ], .RD13 [ .INDEX ], .INDEX );
;    3468  6                    ERRDF  ( 1403, MSG00, ERROR$REPORT );
;    3469  5                  END;
;    3470  5
;    3471  5          INCR INDEX FROM 0 TO 5 DO
;    3472  6            BEGIN
;    3473  6              TEMP1 = .INDEX + 24;
;    3474  6              TEMP2 = .INDEX + 54;
;    3475  6              XMIT_D_LIST [ .INDEX, W_LEN ] = .XMIT_D_LIST [ .TEMP1, W_LEN ];
;    3476  6              RCV_D_LIST  [ .INDEX, W_LEN ] = .RCV_D_LIST  [ .TEMP2, W_LEN ];
;    3477  5            END;
;    3478  5
;    3479  5          RBUF_LENGTH = 1514;
;    3480  5          CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );  ! O'140000', O'000400'
;    3481  5          CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS  );  ! O'140000', O'020000'
;    3482  5
;    3483  5          INCR INDEX FROM 0 TO LEGAL_LENGTH - 1 DO
;    3484  5            IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;    3485  5              THEN
;    3486  6                BEGIN
;    3487  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;    3488  6                  PRINTB ( MSG59 );
;    3489  6                  PRINTB ( MSG51 );
;    3490  6                  PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;    3491  6                  ERRDF  ( 1404, MSG00, ERROR$REPORT );
;    3492  5              END;
;    3493  3          ENDSUB;
;    3494  3
;    3495  1      ENDTST;

                                                .SBTTL  $T14 TEST 14 - DMA TIMING TEST
```

```
000000  004137  000000G                $T14:   JSR     R1,$SAVE2                       ;                               3349
000004  162706  000010                          SUB     #10,SP
000010  012737  002752  000000G                MOV     #2752,RBUF.LENGTH               ;                               3395
000016  012700  002752                         MOV     #2752,R0                        ;                               3396
000022  006200                                 ASR     R0
000024  005400                                 NEG     R0
000026  010037  000000G                        MOV     R0,XBUF.LENGTH
000032  005000                                 CLR     R0                              ; INDEX                         3397
000034  110060  000000G                1$:     MOVB    R0,XMIT.BUFFER(R0)              ; INDEX,*(INDEX)                3398
000040  005200                                 INC     R0                              ; INDEX                         3397
000042  020027  002751                         CMP     R0,#2751                        ; INDEX,*
000046  003772                                 BLE     1$
000050  104402                        2$:     TRAP    2                               ;                               3398
000052  004737  000000G                        JSR     PC,RESET.DEQNA                  ;                               3401
000056  005000                                 CLR     R0                              ; INDEX                         3402
000060  016060  000000G 000000G        3$:     MOV     RD13(R0),RCV.D.LIST(R0)         ; *(INDEX),*(INDEX)             3403
000066  062700  000002                         ADD     #2,R0                           ; *,INDEX                       3402
000072  020027  000176                         CMP     R0,#176                         ; INDEX,*
000076  003770                                 BLE     3$
000100  005000                                 CLR     R0                              ; INDEX                         3404
000102  016060  000000G 000000G        4$:     MOV     TD13(R0),XMIT.D.LIST(R0)        ; *(INDEX),*(INDEX)             3405
000110  062700  000002                         ADD     #2,R0                           ; *,INDEX                       3404
000114  020027  000076                         CMP     R0,#76                          ; INDEX,*
000120  003770                                 BLE     4$
000122  013737  000066G 000000G                MOV     XMIT.D.LIST+66,TEMP5            ;                               3407
000130  013737  000146G 000000G                MOV     RCV.D.LIST+146,TEMP6            ;                               3408
000136  013737  000160G 000000G                MOV     RCV.D.LIST+160,TEMP7            ;                               3409
000144  012737  176614  000066G                MOV     #-1164,XMIT.D.LIST+66           ;                               3411
000152  012737  176617  000146G                MOV     #-1161,RCV.D.LIST+146           ;                               3412
000160  012737  002750G 000160G                MOV     #RCV.BUFFER+2750,RCV.D.LIST+160 ;                               3413
000166  013700  000000G                        MOV     REG.ADR,R0                      ;                               3415
000172  042760  001400  000016                BIC     #1400,16(R0)
000200  052760  001000  000016                BIS     #1000,16(R0)
000206  004737  000000G                        JSR     PC,XMIT.AND.RCV.PACKET          ;                               3416
000212  012746  000001                         MOV     #1,-(SP)                        ;                               3418
000216  004737  000000G                        JSR     PC,CHK.RIXI.STATUS
000222  023727  000000G 000001                CMP     SWP.BLOCK.MEM,#1                ;                               3420
000230  001004                                 BNE     5$
000232  012737  000367  000000G                MOV     #367,TEMP4                      ;                               3422
000240  000403                                 BR      6$                              ;                               3420
000242  012737  001734  000000G        5$:     MOV     #1734,TEMP4                     ;                               3424
000250  023737  000000G 000000G        6$:     CMP     TEMP1,TEMP4                     ;                               3426
000256  101431                                 BLOS    7$
000260  013700  000000G                        MOV     REG.ADR,R0                      ;                               3429
000264  016066  000016  000002                MOV     16(R0),2(SP)                    ; *,TMP.LOCATION
000272  016637  000002  000000G                MOV     2(SP),CSR.WORD                  ; TMP.LOCATION,*
000300  012716  000000G                        MOV     @MSG59,(SP)                     ;                               3430
000304  012746  000001                         MOV     #1,-(SP)
000310  010600                                 MOV     SP,R0                           ; SP,*
000312  104414                                 TRAP    14
000314  012716  000000G                        MOV     @MSG52,(SP)                     ;                               3431
000320  012746  000001                         MOV     #1,-(SP)
000324  010600                                 MOV     SP,R0                           ; SP,*
```

```
000326  104414                          TRAP    14
000330  104455                          TRAP    55                              ;                               3432
000332  002571                          .WORD   2571
000334  000000G                         .WORD   MSG00
000336  000000G                         .WORD   ERROR$REPORT
000340  022626                          CMP     (SP)+,(SP)+                     ;                               3428
000342  012716  100220          7$:     MOV     #-77560,(SP)                    ;                               3435
000346  011646                          MOV     (SP),-(SP)
000350  004737  000000G                 JSR     PC,CHK.CSR.STATUS
000354  013737  000000G 000066G         MOV     TEMP5,XMIT.D.LIST+66            ;                               3437
000362  013737  000000G 000146G         MOV     TEMP6,RCV.D.LIST+146            ;                               3438
000370  013737  000000G 000160G         MOV     TEMP7,RCV.D.LIST+160            ;                               3439
000376  005002                          CLR     R2                              ; INDEX                         3444
000400  010201                  8$:     MOV     R2,R1                           ; INDEX,*                       3445
000402  006301                          ASL     R1
000404  026161  000000G 000000G         CMP     XMIT.D.LIST(R1),TD13(R1)
000412  001454                          BEQ     9$
000414  016100  000000G                 MOV     XMIT.D.LIST(R1),R0              ;                               3446
000420  042700  037777                  BIC     #37777,R0
000424  020027  140000                  CMP     R0,#-40000
000430  001445                          BEQ     9$
000432  013700  000000G                 MOV     REG.ADR,R0                      ;                               3449
000436  016066  000016  000006         MOV     16(R0),6(SP)                    ; *,TMP.LOCATION
000444  016637  000006  000000G         MOV     6(SP),CSR.WORD                  ; TMP.LOCATION,*
000452  012716  000000G                 MOV     #MSG59,(SP)                     ;                               3450
000456  012746  000001                  MOV     #1,-(SP)
000462  010600                          MOV     SP,R0                           ; SP,*
000464  104414                          TRAP    14
000466  012716  000000G                 MOV     #MSG49,(SP)                     ;                               3451
000472  012746  000001                  MOV     #1,-(SP)
000476  010600                          MOV     SP,R0                           ; SP,*
000500  104414                          TRAP    14
000502  010216                          MOV     R2,(SP)                         ; INDEX,*                       3452
000504  016146  000000G                 MOV     TD13(R1),-(SP)
000510  016146  000000G                 MOV     XMIT.D.LIST(R1),-(SP)
000514  012746  000000G                 MOV     #MSG50,-(SP)
000520  012746  000004                  MOV     #4,-(SP)
000524  010600                          MOV     SP,R0                           ; SP,*
000526  104414                          TRAP    14
000530  104455                          TRAP    55                              ;                               3453
000532  002572                          .WORD   2572
000534  000000G                         .WORD   MSG00
000536  000000G                         .WORD   ERROR$REPORT
000540  062706  000014                  ADD     #14,SP                          ;                               3448
000544  005202                  9$:     INC     R2                              ; INDEX                         3444
000546  020227  000027                  CMP     R2,#27                          ; INDEX,*
000552  003712                          BLE     8$
000554  005002                          CLR     R2                              ; INDEX                         3459
000556  010201                  10$:    MOV     R2,R1                           ; INDEX,*                       3460
000560  006301                          ASL     R1
000562  026161  000000G 000000G         CMP     RCV.D.LIST(R1),RD13(R1)
000570  001454                          BEQ     11$
000572  016100  000000G                 MOV     RCV.D.LIST(R1),R0               ;                               3461
```

```
000576  042700  037777                          BIC     #37777,R0
000602  020027  140000                          CMP     R0,#-40000
000606  001445                                  BEQ     11$
000610  013700  000000G                         MOV     REG.ADR,R0                      ;                               3464
000614  016066  000016  000010                  MOV     16(R0),10(SP)                   ; *,TMP.LOCATION
000622  016637  000010  000000G                 MOV     10(SP),CSR.WORD                 ; TMP.LOCATION,*
000630  012716  000000G                         MOV     #MSG59,(SP)                     ;                               3465
000634  012746  000001                          MOV     #1,-(SP)
000640  010600                                  MOV     SP,R0                           ; SP,*
000642  104414                                  TRAP    14
000644  012716  000000G                         MOV     #MSG48,(SP)                     ;                               3466
000650  012746  000001                          MOV     #1,-(SP)
000654  010600                                  MOV     SP,R0                           ; SP,*
000656  104414                                  TRAP    14
000660  010216                                  MOV     R2,(SP)                         ; INDEX,*                       3467
000662  016146  000000G                         MOV     RD13(R1),-(SP)
000666  016146  000000G                         MOV     RCV.D.LIST(R1),-(SP)
000672  012746  000000G                         MOV     #MSG50,-(SP)
000676  012746  000004                          MOV     #4,-(SP)
000702  010600                                  MOV     SP,R0                           ; SP,*
000704  104414                                  TRAP    14
000706  104455                                  TRAP    55                              ;                               3468
000710  002573                                  .WORD   2573
000712  000000G                                 .WORD   MSG00
000714  000000G                                 .WORD   ERROR$REPORT
000716  062706  000014                          ADD     #14,SP                          ;                               3463
000722  005202                          11$:    INC     R2                              ; INDEX                         3459
000724  020227  000065                          CMP     R2,#65                          ; INDEX,*
000730  003712                                  BLE     10$
000732  005002                                  CLR     R2                              ; INDEX                         3471
000734  010237  000000G                 12$:    MOV     R2,TEMP1                        ; INDEX,*                       3473
000740  062737  000030  000000G                 ADD     #30,TEMP1
000746  010237  000000G                         MOV     R2,TEMP2                        ; INDEX,*                       3474
000752  062737  000066  000000G                 ADD     #66,TEMP2
000760  010200                                  MOV     R2,R0                           ; INDEX,*                       3475
000762  006300                                  ASL     R0
000764  013701  000000G                         MOV     TEMP1,R1
000770  006301                                  ASL     R1
000772  016160  000000G 000000G                 MOV     XMIT.D.LIST(R1),XMIT.D.LIST(R0)
001000  013701  000000G                         MOV     TEMP2,R1                        ;                               3476
001004  006301                                  ASL     R1
001006  016160  000000G 000000G                 MOV     RCV.D.LIST(R1),RCV.D.LIST(R0)
001014  005202                                  INC     R2                              ; INDEX                         3471
001016  020227  000005                          CMP     R2,#5                           ; INDEX,*
001022  003744                                  BLE     12$
001024  012737  002752  000000G                 MOV     #2752,RBUF.LENGTH               ;                               3479
001032  012716  140000                          MOV     #-40000,(SP)                    ;                               3480
001036  012746  000400                          MOV     #400,-(SP)
001042  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
001046  012716  140000                          MOV     #-40000,(SP)                    ;                               3481
001052  012746  020000                          MOV     #20000,-(SP)
001056  004737  000000G                         JSR     PC,CHK.RCV.STATUS
001062  005001                                  CLR     R1                              ; INDEX                         3483
```

```
001064  126161  000000G 000000G         13$:    CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                    3484
001072  001447                                  BEQ     14$
001074  013700  000000G                         MOV     REG.ADR,R0                      ;                               3487
001100  016066  000016  000016                  MOV     16(R0),16(SP)                   ; *,TMP.LOCATION
001106  016637  000016  000000G                 MOV     16(SP),CSR.WORD                 ; TMP.LOCATION,*
001114  012716  000000G                         MOV     #MSG59,(SP)                     ;                               3488
001120  012746  000001                          MOV     #1,-(SP)
001124  010600                                  MOV     SP,R0                           ; SP,*
001126  104414                                  TRAP    14
001130  012716  000000G                         MOV     #MSG51,(SP)                     ;                               3489
001134  012746  000001                          MOV     #1,-(SP)
001140  010600                                  MOV     SP,R0                           ; SP,*
001142  104414                                  TRAP    14
001144  010116                                  MOV     R1,(SP)                         ; INDEX,*                       3490
001146  005046                                  CLR     -(SP)
001150  116116  000000G                         MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
001154  005046                                  CLR     -(SP)
001156  116116  000000G                         MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
001162  012746  000000G                         MOV     #MSG50,-(SP)
001166  012746  000004                          MOV     #4,-(SP)
001172  010600                                  MOV     SP,R0                           ; SP,*
001174  104414                                  TRAP    14
001176  104455                                  TRAP    55                              ;                               3491
001200  002574                                  .WORD   2574
001202  000000G                                 .WORD   MSG00
001204  000000G                                 .WORD   ERROR$REPORT
001206  062706  000014                          ADD     #14,SP                          ;                               3486
001212  005201                          14$:    INC     R1                              ; INDEX                         3483
001214  020127  002751                          CMP     R1,#2751                        ; INDEX,*
001220  003721                                  BLE     13$
001222  062706  000010                          ADD     #10,SP                          ;                               3398
001226  104467                                  TRAP    67                              ;                               3492
001230  006000                                  ROR     R0
001232  103002                                  BHIS    15$
001234  000137  013606'                         JMP     2$
001240  062706  000010                  15$:    ADD     #10,SP                          ;                               3349
001244  000207                                  RTS     PC

; Routine Size:  339 words,     Routine Base:  AB$CODE$ + 13536
; Maximum stack depth per invocation:  19 words




                                                .SBTTL  T14 TEST 14 - DMA TIMING TEST
000000  004737  013536'                 T14::
000000                                  1$:     JSR     PC,$T14                         ;                               3493
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 15004
```

```
; Maximum stack depth per invocation:  2 words

;   3496  1
```

```
;  3497  1     %SBTTL 'TEST 15 - LONG PACKET TEST'
;  3498  1     !++
;  3499  1     !
;  3500  1     !  TEST 15:     LONG PACKET TEST
;  3501  1     !
;  3502  1     !  DESCRIPTION:
;  3503  1     !
;  3504  1     !       This test verifies that DEQNA can detect long packets ( 1600 bytes
;  3505  1     !       or more with the CRC ) when transmitted in internal/extended
;  3506  1     !       loopback mode. If the operator specifies loop on error, the
;  3507  1     !       program re-executes the code that detected the error until ^C is
;  3508  1     !       entered.
;  3509  1     !
;  3510  1     !       Hardware tested:        RCV Status - error summary (long packet-bit 14)
;  3511  1     !
;  3512  1     !       Processing:
;  3513  1     !
;  3514  1     !            BEGIN
;  3515  1     !               reset device
;  3516  1     !               select internal/extended loopback mode
;  3517  1     !               transmit loopback packet (legal packet length)
;  3518  1     !               check for expected loopback status
;  3519  1     !               IF error
;  3520  1     !               THEN
;  3521  1     !                  print error message if not inhibited
;  3522  1     !               ENDIF
;  3523  1     !               call compare_packets
;  3524  1     !               transmit loopback packet ( packet length > legal max. )
;  3525  1     !               IF Error Summary bit ( Receice Status Word 1, bit 14 ) = 1
;  3526  1     !                  AND ( receive packet length is truncated )
;  3527  1     !               THEN
;  3528  1     !                  print error message if not inhibited
;  3529  1     !               ENDIF
;  3530  1     !            END
;  3531  1     !--
```

```
;    3532  3     BGNTST;
;    3533  3
;    3534  3     !++
;    3535  3     !  LOOPBACK 1534 BYTE PACKET AND THEN CHECK IF PROPERLY RECEIVED.
;    3536  3     !  THIS IS THE LONGEST PACKET LENGTH WHICH DOESN'T SET 'LONGP' BIT IN
;    3537  3     !  THE RECEIVE STATUS WORD 1 ( BIT 14 ).
;    3538  3     !--
;    3539  3
;    3540  3     RBUF_LENGTH = 1534;
;    3541  3     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3542  3
;    3543  5     BGNSUB;
;    3544  5       RESET_DEQNA ( );
;    3545  5       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    3546  5       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    3547  5       SEND_ELOOP_PACKET ( ZERO );
;    3548  5       COMPARE_PACKETS ( );
;    3549  3     ENDSUB;
;    3550  3
;    3551  3     !++
;    3552  3     !  LOOPBACK 1536 BYTE PACKET AND THEN CHECK IF BITS 13 AND 14 ARE SET IN
;    3553  3     !--
;    3554  3
;    3555  3
;    3556  3     RBUF_LENGTH = 1536;
;    3557  3     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3558  3
;    3559  5     BGNSUB;
;    3560  5       RESET_DEQNA ( );
;    3561  5       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    3562  5       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    3563  5       SEND_ELOOP_PACKET ( ONE );
;    3564  5       COMPARE_PACKETS ( );
;    3565  3     ENDSUB;
;    3566  3
;    3567  1     ENDTST;


                                                .SBTTL  $T15 TEST 15 - LONG PACKET TEST
000000  012737  002776  000000G        $T15:   MOV     #2776,RBUF.LENGTH               ;    3540
000006  012700  002776                        MOV     #2776,R0                        ;    3541
000012  006200                                ASR     R0
000014  005400                                NEG     R0
000016  010037  000000G                       MOV     R0,XBUF.LENGTH
000022  104402                        1$:     TRAP    2
000024  004737  000000G                       JSR     PC,RESET.DEQNA                  ;    3544
000030  013746  000000G                       MOV     XBUF.LENGTH,-(SP)               ;    3545
000034  012746  120000                        MOV     #-60000,-(SP)
000040  004737  000000G                       JSR     PC,SET.RDESCR.LIST
000044  013716  000000G                       MOV     XBUF.LENGTH,(SP)                ;    3546
000050  012746  120000                        MOV     #-60000,-(SP)
000054  004737  000000G                       JSR     PC,SET.XDESCR.LIST
000060  005016                                CLR     (SP)                            ;    3547
```

```
000062  004737  000000G                        JSR     PC,SEND.ELOOP.PACKET
000066  004737  000000G                        JSR     PC,COMPARE.PACKETS      ;                       3548
000072  062706  000006                         ADD     #6,SP                   ;                       3541
000076  104467                                 TRAP    67                      ;                       3548
000100  006000                                 ROR     R0
000102  103747                                 BLO     1$
000104  012737  003000  000000G                MOV     #3000,RBUF.LENGTH       ;                       3556
000112  012700  003000                         MOV     #3000,R0                ;                       3557
000116  006200                                 ASR     R0
000120  005400                                 NEG     R0
000122  010037  000000G                        MOV     R0,XBUF.LENGTH
000126  104402                         2$:     TRAP    2
000130  004737  000000G                        JSR     PC,RESET.DEQNA          ;                       3560
000134  013746  000000G                        MOV     XBUF.LENGTH,-(SP)       ;                       3561
000140  012746  120000                         MOV     #-60000,-(SP)
000144  004737  000000G                        JSR     PC,SET.RDESCR.LIST
000150  013716  000000G                        MOV     XBUF.LENGTH,(SP)        ;                       3562
000154  012746  120000                         MOV     #-60000,-(SP)
000160  004737  000000G                        JSR     PC,SET.XDESCR.LIST
000164  012716  000001                         MOV     #1,(SP)                 ;                       3563
000170  004737  000000G                        JSR     PC,SEND.ELOOP.PACKET
000174  004737  000000G                        JSR     PC,COMPARE.PACKETS      ;                       3564
000200  062706  000006                         ADD     #6,SP                   ;                       3557
000204  104467                                 TRAP    67                      ;                       3564
000206  006000                                 ROR     R0
000210  103746                                 BLO     2$
000212  000207                                 RTS     PC                      ;                       3495

; Routine Size:  70 words,      Routine Base:  AB$CODE$ + 15020
; Maximum stack depth per invocation:  4 words



                                               .SBTTL  T15 TEST 15 - LONG PACKET TEST
000000  004737  015020'                 T15::
000000                                 1$:     JSR     PC,$T15                 ;                       3565
000004  104466                                 TRAP    66
000006  006000                                 ROR     R0
000010  103773                                 BLO     1$
000012  000207                                 RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 15234
; Maximum stack depth per invocation:  2 words


;   3568  1
;   3569  1
```

```
;    3570  1     %SBTTL 'TEST 16 - ODD PACKET TEST'
;    3571  1     !++
;    3572  1     !
;    3573  1     !  TEST 16:     ODD PACKET TEST
;    3574  1     !
;    3575  1     !  DESCRIPTION:
;    3576  1     !
;    3577  1     !       This test verifies that DEQNA can transmit and receive odd length
;    3578  1     !       packets and packets starting and/or ending on odd addresses. Chained
;    3579  1     !       and unchained descriptor lists are used to verify this. If the operator
;    3580  1     !       specifies loop on error, the program re-executes the code that detected
;    3581  1     !       the error until ^C is entered.
;    3582  1     !
;    3583  1     !       Hardware tested:        CSR register - XMIT List Invalid (bit 4)
;    3584  1     !                                    - RCV  List Invalid (bit 5)
;    3585  1     !                               Transmit Descriptor bits
;    3586  1     !                                    - XMIT buffer ends on odd byte
;    3587  1     !                                    - XMIT buffer ends on even byte
;    3588  1     !
;    3589  1     !       Set of addresses and packet lengths:
;    3590  1     !
;    3591  1     !               PACKET ADDRESS              PACKET LENGTH
;    3592  1     !               ----------------            -------------
;    3593  1     !
;    3594  1     !               odd begin                       odd
;    3595  1     !               odd begin and end               even
;    3596  1     !               odd end                         odd
;    3597  1     !
;    3598  1     !       Processing:
;    3599  1     !
;    3600  1     !            BEGIN
;    3601  1     !               reset device
;    3602  1     !               REPEAT for internal and internal/extended loopback mode
;    3603  1     !                 REPEAT for each packet address and length from set
;    3604  1     !                   check for expected loopback status
;    3605  1     !                   IF error
;    3606  1     !                   THEN
;    3607  1     !                     print error message if not inhibited
;    3608  1     !                   ENDIF
;    3609  1     !                   call compare_packets
;    3610  1     !                 ENDREPEAT
;    3611  1     !               ENDREPEAT
;    3612  1     !            END
;    3613  1     !--
```

```
;    3614  3     BGNTST;
;    3615  3
;    3616  3     !++
;    3617  3     !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM
;    3618  3     !--
;    3619  3
;    3620  3     RESET_DEQNA ( );
;    3621  3     PREP_FOR_SETUP ( );
;    3622  3     INCR INDEX1 FROM 1 TO 14 DO
;    3623  3       WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;    3624  3
;    3625  5     BGNSUB;
;    3626  5       XMIT_SETUP_PACKET ( P_MODE );
;    3627  3     ENDSUB;
;    3628  3
;    3629  3     RBUF_LENGTH = 6;
;    3630  3     XBUF_LENGTH = - ( .RBUF_LENGTH ^ -1 );
;    3631  3
;    3632  3     !++
;    3633  3     !  LOOPBACK A PACKET, THEN CHECK IF LOOPBACK PACKET WAS PROPERLY
;    3634  3     !  RECEIVED
;    3635  3     !--
;    3636  3
;    3637  3     CLR_BUFFERS ( 32 );
;    3638  3     CLR_DESCR ( );
;    3639  3     INCR INDEX FROM 0 TO 5 DO
;    3640  3       XMIT_BUFFER [ .INDEX ] = .INDEX;
;    3641  3
;    3642  5     BGNSUB;
;    3643  5       INCR INDEX FROM 0 TO 43 DO
;    3644  5         XMIT_D_LIST [ .INDEX, W_LEN ] = .TD16 [ .INDEX ];
;    3645  5       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    3646  5       PUT_BIT [ CSR, LB, INT_LOOPBACK ];
;    3647  5
;    3648  5       XMIT_AND_RCV_PACKET ( );
;    3649  5       CHK_RIXI_STATUS ( ONE );
;    3650  5       .IOP_TABLE [ CSR ] = ONE;
;    3651  5       CHK_RIXI_STATUS ( ZERO );
;    3652  5       .IOP_TABLE [ CSR ] = ZERO;
;    3653  5
;    3654  5       CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );     ! 0'100220', 0'100220'
;    3655  5
;    3656  5       !++
;    3657  5       !  CHECK IF TRANSMIT BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;    3658  5       !--
;    3659  5
;    3660  5       INCR INDEX FROM 0 TO 17 DO
;    3661  5         IF .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU .TD16 [ .INDEX ]
;    3662  5           AND ( .XMIT_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %C '40000'
;    3663  5             THEN
;    3664  6               BEGIN
;    3665  6                 CSR_WORD = GET_BIT ( CSR_ALL );
;    3666  6                 PRINTB ( MSG59 );
```

```
;  3667  6                PRINTB ( MSG49 );
;  3668  6                PRINTB ( MSG50, .XMIT_D_LIST [ .INDEX, W_LEN ], .TD16 [ .INDEX ], .INDEX );
;  3669  6                ERRDF  ( 1602, MSG00, ERROR$REPORT );
;  3670  5              END;
;  3671  5
;  3672  5
;  3673  5        INCR INDEX FROM 0 TO 5 DO
;  3674  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .XMIT_D_LIST [ .INDEX + 18, W_LEN ];
;  3675  5
;  3676  5        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );  ! 0'140000', 0'000400'
;  3677  5        CHK_RCV_STATUS  ( RFLG_STATUS, RWD13_STATUS );  ! 0'140000', 0'000000'
;  3678  5
;  3679  5        INCR INDEX FROM 0 TO 5 DO
;  3680  5          IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;  3681  5            THEN
;  3682  6              BEGIN
;  3683  6                CSR_WORD = GET_BIT ( CSR_ALL );
;  3684  6                PRINTB ( MSG59 );
;  3685  6                PRINTB ( MSG51 );
;  3686  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;  3687  6                ERRDF  ( 1603, MSG00, ERROR$REPORT );
;  3688  5              END;
;  3689  3      ENDSUB;
;  3690  3
;  3691  3      RESET_DEQNA ( );
;  3692  3      CLR_BUFFERS ( 32 );
;  3693  3      RBUF_LENGTH = 16;
;  3694  3      XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  3695  3      INCR INDEX FROM 0 TO 19 DO
;  3696  3        XMIT_BUFFER [ .INDEX ] = .INDEX;
;  3697  3
;  3698  5      BGNSUB;
;  3699  5        INCR INDEX FROM 0 TO 43 DO
;  3700  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .TD16 [ .INDEX ];
;  3701  5
;  3702  5        XMIT_D_LIST [ 19, W_LEN ] = V;
;  3703  5        XMIT_D_LIST [ 25, W_LEN ] = C;
;  3704  5
;  3705  5        SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  3706  5        PUT_BIT [ CSR, LB, INX_LOOPBACK ];
;  3707  5        XMIT_AND_RCV_PACKET ( );
;  3708  5        CHK_RIXI_STATUS ( ZERO );
;  3709  5
;  3710  5        CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );     ! 0'100220', 0'100220'
;  3711  5
;  3712  5        XMIT_D_LIST [ 19, W_LEN ] = VE;
;  3713  5        XMIT_D_LIST [ 25, W_LEN ] = E;
;  3714  5
;  3715  5        !++
;  3716  5        !  CHECK IF TRANSMIT BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;  3717  5        !--
;  3718  5
;  3719  5        INCR INDEX FROM 0 TO 35 DO
```

```
;  3720  5          IF .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU .TD16 [ .INDEX ]
;  3721  5            AND ( .XMIT_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;  3722  5              THEN
;  3723  6                BEGIN
;  3724  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;  3725  6                  PRINTB ( MSG59 );
;  3726  6                  PRINTB ( MSG49 );
;  3727  6                  PRINTB ( MSG50, .XMIT_D_LIST [ .INDEX, W_LEN ], .TD16 [ .INDEX ], .INDEX );
;  3728  6                  ERRDF  ( 1604, MSG00, ERROR$REPORT );
;  3729  5                END;
;  3730  5
;  3731  5        INCR INDEX FROM 0 TO 5 DO
;  3732  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .XMIT_D_LIST [ .INDEX + 36, W_LEN ];
;  3733  5
;  3734  5        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );  ! O'140000', O'000400'
;  3735  5        CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS );   ! O'140000', O'020000'
;  3736  5
;  3737  5
;  3738  5        INCR INDEX FROM 0 TO 5 DO
;  3739  5          IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;  3740  5            THEN
;  3741  6              BEGIN
;  3742  6                CSR_WORD = GET_BIT ( CSR_ALL );
;  3743  6                PRINTB ( MSG59 );
;  3744  6                PRINTB ( MSG51 );
;  3745  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;  3746  6                ERRDF  ( 1605, MSG00, ERROR$REPORT );
;  3747  5              END;
;  3748  5
;  3749  5        INCR INDEX FROM 6 TO 9 DO
;  3750  5          IF .RCV_BUFFER [ .INDEX ] NEQU ZERO
;  3751  5            THEN
;  3752  6              BEGIN
;  3753  6                CSR_WORD = GET_BIT ( CSR_ALL );
;  3754  6                PRINTB ( MSG59 );
;  3755  6                PRINTB ( MSG51 );
;  3756  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;  3757  6                ERRDF  ( 1606, MSG00, ERROR$REPORT );
;  3758  5              END;
;  3759  5
;  3760  5        INCR INDEX FROM 0 TO 5 DO
;  3761  5          IF .RCV_BUFFER [ .INDEX + 10 ] NEQU .TARGET_ADR [ .INDEX + 114 ]
;  3762  5            THEN
;  3763  6              BEGIN
;  3764  6                CSR_WORD = GET_BIT ( CSR_ALL );
;  3765  6                PRINTB ( MSG59 );
;  3766  6                PRINTB ( MSG51 );
;  3767  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;  3768  6                ERRDF  ( 1607, MSG00, ERROR$REPORT );
;  3769  5              END;
;  3770  3      ENDSUB;
;  3771  3
;  3772  1      ENDTST;
```

```
                                                .SBTTL  $T16 TEST 16 - ODD PACKET TEST
000000  004137  000000G                 $T16:  JSR     R1,$SAVE2                       ;                                3567
000004  162706  000014                        SUB     #14,SP
000010  004737  000000G                        JSR     PC,RESET.DEQNA                  ;                                3620
000014  004737  000000G                        JSR     PC,PREP.FOR.SETUP               ;                                3621
000020  012701  000001                        MOV     #1,R1                           ; *,INDEX1                       3622
000024  010146                          1$:    MOV     R1,-(SP)                        ; INDEX1,*                       3623
000026  012746  000023                        MOV     #23,-(SP)
000032  004737  000000G                        JSR     PC,WRT.STATION.ADR
000036  022626                                CMP     (SP)+,(SP)+
000040  005201                                INC     R1                              ; INDEX1                         3622
000042  020127  000016                        CMP     R1,#16                          ; INDEX1,*
000046  003766                                BLE     1$
000050  104402                          2$:    TRAP    2                               ;                                3623
000052  012746  000202                        MOV     #202,-(SP)                      ;                                3626
000056  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000062  005726                                TST     (SP)+                           ;                                3623
000064  104467                                TRAP    67                              ;                                3626
000066  006000                                ROR     R0
000070  103767                                BLO     2$
000072  012737  000006  000000G               MOV     #6,RBUF.LENGTH                  ;                                3629
000100  012700  000006                        MOV     #6,R0                           ;                                3630
000104  006200                                ASR     R0
000106  005400                                NEG     R0
000110  010037  000000G                        MOV     R0,XBUF.LENGTH
000114  012746  000040                        MOV     #40,-(SP)                       ;                                3637
000120  004737  000000G                        JSR     PC,CLR.BUFFERS
000124  004737  000000G                        JSR     PC,CLR.DESCR                    ;                                3638
000130  005000                                CLR     R0                              ; INDEX                          3639
000132  110060  000000G                 3$:    MOVB    R0,XMIT.BUFFER(R0)              ; INDEX,*(INDEX)                 3640
000136  005200                                INC     R0                              ; INDEX                          3639
000140  020027  000005                        CMP     R0,#5                           ; INDEX,*
000144  003772                                BLE     3$
000146  104402                          4$:    TRAP    2                               ;                                3640
000150  005000                                CLR     R0                              ; INDEX                          3643
000152  016060  000000G 000000G         5$:    MOV     TD16(R0),XMIT.D.LIST(R0)        ; *(INDEX),*(INDEX)              3644
000160  062700  000002                        ADD     #2,R0                           ; *,INDEX                        3643
000164  020027  000126                        CMP     R0,#126                         ; INDEX,*
000170  003770                                BLE     5$
000172  013716  000000G                        MOV     XBUF.LENGTH,(SP)                ;                                3645
000176  012746  120000                        MOV     #-60000,-(SP)
000202  004737  000000G                        JSR     PC,SET.RDESCR.LIST
000206  013700  000000G                        MOV     REG.ADR,R0                      ;                                3646
000212  042760  001400  000016               BIC     #1400,16(R0)
000220  004737  000000G                        JSR     PC,XMIT.AND.RCV.PACKET          ;                                3648
000224  012716  000001                        MOV     #1,(SP)                         ;                                3649
000230  004737  000000G                        JSR     PC,CHK.RIXI.STATUS
000234  012777  000001  000016G              MOV     #1,@IOP.TABLE+16                ;                                3650
000242  005016                                CLR     (SP)                            ;                                3651
000244  004737  000000G                        JSR     PC,CHK.RIXI.STATUS
000250  005077  000016G                        CLR     @IOP.TABLE+16                   ;                                3652
```

```
000254  012716  100220                          MOV     #-77560,(SP)            ;                               3654
000260  011646                                  MOV     (SP),-(SP)
000262  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000266  005002                                  CLR     R2                      ; INDEX                         3660
000270  010201                          6$:     MOV     R2,R1                   ; INDEX,*                       3661
000272  006301                                  ASL     R1
000274  026161  000000G 000000G                 CMP     XMIT.D.LIST(R1),TD16(R1)
000302  001454                                  BEQ     7$
000304  016100  000000G                         MOV     XMIT.D.LIST(R1),R0      ;                               3662
000310  042700  037777                          BIC     #37777,R0
000314  020027  140000                          CMP     R0,#-40000
000320  001445                                  BEQ     7$
000322  013700  000000G                         MOV     REG.ADR,R0              ;                               3665
000326  016066  000016  000006                 MOV     16(R0),6(SP)            ; *,TMP.LOCATION
000334  016637  000006  000000G                 MOV     6(SP),CSR.WORD          ; TMP.LOCATION,*
000342  012716  000000G                         MOV     #MSG59,(SP)             ;                               3666
000346  012746  000001                          MOV     #1,-(SP)
000352  010600                                  MOV     SP,R0                   ; SP,*
000354  104414                                  TRAP    14
000356  012716  000000G                         MOV     #MSG49,(SP)             ;                               3667
000362  012746  000001                          MOV     #1,-(SP)
000366  010600                                  MOV     SP,R0                   ; SP,*
000370  104414                                  TRAP    14
000372  010216                                  MOV     R2,(SP)                 ; INDEX,*                       3668
000374  016146  000000G                         MOV     TD16(R1),-(SP)
000400  016146  000000G                         MOV     XMIT.D.LIST(R1),-(SP)
000404  012746  000000G                         MOV     #MSG50,-(SP)
000410  012746  000004                          MOV     #4,-(SP)
000414  010600                                  MOV     SP,R0                   ; SP,*
000416  104414                                  TRAP    14
000420  104455                                  TRAP    55                      ;                               3669
000422  003102                                  .WORD   3102
000424  000000G                                 .WORD   MSG00
000426  000000G                                 .WORD   ERROR$REPORT
000430  062706  000014                          ADD     #14,SP                  ;                               3664
000434  005202                          7$:     INC     R2                      ; INDEX                         3660
000436  020227  000021                          CMP     R2,#21                  ; INDEX,*
000442  003712                                  BLE     6$
000444  005002                                  CLR     R2                      ; INDEX                         3673
000446  010201                          8$:     MOV     R2,R1                   ; INDEX,*                       3674
000450  006301                                  ASL     R1
000452  010200                                  MOV     R2,R0                   ; INDEX,*
000454  006300                                  ASL     R0
000456  016061  000044G 000000G                 MOV     XMIT.D.LIST+44(R0),XMIT.D.LIST(R1) ;
000464  005202                                  INC     R2                      ; INDEX                         3673
000466  020227  000005                          CMP     R2,#5                   ; INDEX,*
000472  003765                                  BLE     8$
000474  012716  140000                          MOV     #-40000,(SP)            ;                               3676
000500  012746  000400                          MOV     #400,-(SP)
000504  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
000510  012716  140000                          MOV     #-40000,(SP)            ;                               3677
000514  005046                                  CLR     -(SP)
000516  004737  000000G                         JSR     PC,CHK.RCV.STATUS
```

```
000522  005001                          CLR     R1                              ; INDEX                                 3679
000524  126161  000000G 000000G   9$:   CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                   3680
000532  001447                          BEQ     10$
000534  013700  000000G                 MOV     REG.ADR,R0                      ;                                       3683
000540  016066  000016  000014          MOV     16(R0),14(SP)                   ; *,TMP.LOCATION
000546  016637  000014  000000G         MOV     14(SP),CSR.WORD                 ; TMP.LOCATION,*
000554  012716  000000G                 MOV     #MSG59,(SP)                     ;                                       3684
000560  012746  000001                  MOV     #1,-(SP)
000564  010600                          MOV     SP,R0                           ; SP,*
000566  104414                          TRAP    14
000570  012716  000000G                 MOV     #MSG51,(SP)                     ;                                       3685
000574  012746  000001                  MOV     #1,-(SP)
000600  010600                          MOV     SP,R0                           ; SP,*
000602  104414                          TRAP    14
000604  010116                          MOV     R1,(SP)                         ; INDEX,*                               3686
000606  005046                          CLR     -(SP)
000610  116116  000000G                 MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
000614  005046                          CLR     -(SP)
000616  116116  000000G                 MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
000622  012746  000000G                 MOV     #MSG50,-(SP)
000626  012746  000004                  MOV     #4,-(SP)
000632  010600                          MOV     SP,R0                           ; SP,*
000634  104414                          TRAP    14
000636  104455                          TRAP    55                              ;                                       3687
000640  003103                          .WORD   3103
000642  000000G                         .WORD   MSG00
000644  000000G                         .WORD   ERROR$REPORT
000646  062706  000014                  ADD     #14,SP                          ;                                       3682
000652  005201                  10$:    INC     R1                              ; INDEX                                 3679
000654  020127  000005                  CMP     R1,#5                           ; INDEX,*
000660  003721                          BLE     9$
000662  062706  000010                  ADD     #10,SP                          ;                                       3640
000666  104467                          TRAP    67                              ;                                       3688
000670  006000                          ROR     R0
000672  103002                          BHIS    11$
000674  000137  015416'                 JMP     4$
000700  004737  000000G         11$:    JSR     PC,RESET.DEQNA                  ;                                       3691
000704  012716  000040                  MOV     #40,(SP)                        ;                                       3692
000710  004737  000000G                 JSR     PC,CLR.BUFFERS
000714  012737  000020  000000G         MOV     #20,RBUF.LENGTH                 ;                                       3693
000722  012700  000020                  MOV     #20,R0                          ;                                       3694
000726  006200                          ASR     R0
000730  005400                          NEG     R0
000732  010037  000000G                 MOV     R0,XBUF.LENGTH
000736  005000                          CLR     R0                              ; INDEX                                 3695
000740  110060  000000G         12$:    MOVB    R0,XMIT.BUFFER(R0)              ; INDEX,*(INDEX)                        3696
000744  005200                          INC     R0                              ; INDEX                                 3695
000746  020027  000023                  CMP     R0,#23                          ; INDEX,*
000752  003772                          BLE     12$
000754  104402                  13$:    TRAP    2                               ;                                       3696
000756  005000                          CLR     R0                              ; INDEX                                 3699
000760  016060  000000G 000000G   14$:   MOV     TD16(R0),XMIT.D.LIST(R0)        ; *(INDEX),*(INDEX)                     3700
000766  062700  000002                  ADD     #2,R0                           ; *,INDEX                               3699
```

```
000772  020027  000126                          CMP     R0,#126                 ; INDEX,*
000776  003770                                   BLE     14$
001000  012737  100000  000046G                 MOV     #-100000,XMIT.D.LIST+46 ;                       3702
001006  012737  040000  000062G                 MOV     #40000,XMIT.D.LIST+62   ;                       3703
001014  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                       3705
001020  012746  120000                          MOV     #-60000,-(SP)
001024  004737  000000G                         JSR     PC,SET.RDESCR.LIST
001030  013700  000000G                         MOV     REG.ADR,R0              ;                       3706
001034  042760  001400  000016                 BIC     #1400,16(R0)
001042  052760  001000  000016                 BIS     #1000,16(R0)
001050  004737  000000G                         JSR     PC,XMIT.AND.RCV.PACKET  ;                       3707
001054  005016                                   CLR     (SP)                    ;                       3708
001056  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
001062  012716  100220                          MOV     #-77560,(SP)            ;                       3710
001066  011646                                   MOV     (SP),-(SP)
001070  004737  000000G                         JSR     PC,CHK.CSR.STATUS
001074  012737  120000  000046G                 MOV     #-60000,XMIT.D.LIST+46  ;                       3712
001102  012737  020000  000062G                 MOV     #20000,XMIT.D.LIST+62   ;                       3713
001110  005002                                   CLR     R2                      ; INDEX                 3719
001112  010201                          15$:     MOV     R2,R1                   ; INDEX,*               3720
001114  006301                                   ASL     R1
001116  026161  000000G 000000G                 CMP     XMIT.D.LIST(R1),TD16(R1)
001124  001454                                   BEQ     16$
001126  016100  000000G                         MOV     XMIT.D.LIST(R1),R0      ;                       3721
001132  042700  037777                          BIC     #37777,R0
001136  020027  140000                          CMP     R0,#-40000
001142  001445                                   BEQ     16$
001144  013700  000000G                         MOV     REG.ADR,R0              ;                       3724
001150  016066  000016  000012                 MOV     16(R0),12(SP)           ; *,TMP.LOCATION
001156  016637  000012  000000G                 MOV     12(SP),CSR.WORD         ; TMP.LOCATION,*
001164  012716  000000G                         MOV     #MSG59,(SP)             ;                       3725
001170  012746  000001                          MOV     #1,-(SP)
001174  010600                                   MOV     SP,R0                   ; SP,*
001176  104414                                   TRAP    14
001200  012716  000000G                         MOV     #MSG49,(SP)             ;                       3726
001204  012746  000001                          MOV     #1,-(SP)
001210  010600                                   MOV     SP,R0                   ; SP,*
001212  104414                                   TRAP    14
001214  010216                                   MOV     R2,(SP)                 ; INDEX,*               3727
001216  016146  000000G                         MOV     TD16(R1),-(SP)
001222  016146  000000G                         MOV     XMIT.D.LIST(R1),-(SP)
001226  012746  000000G                         MOV     #MSG50,-(SP)
001232  012746  000004                          MOV     #4,-(SP)
001236  010600                                   MOV     SP,R0                   ; SP,*
001240  104414                                   TRAP    14
001242  104455                                   TRAP    55                      ;                       3728
001244  003104                                   .WORD   3104
001246  000000G                                 .WORD   MSG00
001250  000000G                                 .WORD   ERROR$REPORT
001252  062706  000014                          ADD     #14,SP                  ;                       3723
001256  005202                          16$:     INC     R2                      ; INDEX                 3719
001260  020227  000043                          CMP     R2,#43                  ; INDEX,*
001264  003712                                   BLE     15$
```

```
001266  005C02                          CLR     R2                              ; INDEX                         3731
001270  010201                  17$:    MOV     R2,R1                           ; INDEX *                       3732
001272  006301                          ASL     R1
001274  010200                          MOV     R2,R0                           ; INDEX,*
001276  006300                          ASL     R0
001300  016061  000110G 000000G         MOV     XMIT.D.LIST+110(R0),XMIT.D.LIST(R1) ;
001306  005202                          INC     R2                              ; INDEX                         3731
001310  020227  000005                  CMP     R2,#5                           ; INDEX,*
001314  003765                          BLE     17$
001316  012716  140000                  MOV     #-40000,(SP)                    ;                               3734
001322  012746  000400                  MOV     #400,-(SP)
001326  004737  000000G                 JSR     PC,CHK.XMIT.STATUS
001332  012716  140000                  MOV     #-40000,(SP)                    ;                               3735
001336  012746  020000                  MOV     #20000,-(SP)
001342  004737  000000G                 JSR     PC,CHK.RCV.STATUS
001346  005001                          CLR     R1                              ; INDEX                         3738
001350  126161  000000G 000000G 18$:    CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)             3739
001356  001447                          BEQ     19$
001360  013700  000000G                 MOV     REG.ADR,R0                      ;                               3742
001364  016066  000016  000020         MOV     16(R0),20(SP)                   ; *,TMP.LOCATION
001372  016637  000020  000000G         MOV     20(SP),CSR.WORD                 ; TMP.LOCATION,*
001400  012716  000000G                 MOV     #MSG59,(SP)                     ;                               3743
001404  012746  000001                  MOV     #1,-(SP)
001410  010600                          MOV     SP,R0                           ; SP,*
001412  104414                          TRAP    14
001414  012716  000000G                 MOV     #MSG51,(SP)                     ;                               3744
001420  012746  000001                  MOV     #1,-(SP)
001424  010600                          MOV     SP,R0                           ; SP,*
001426  104414                          TRAP    14
001430  010116                          MOV     R1,(SP)                         ; INDEX,*                       3745
001432  005046                          CLR     -(SP)
001434  116116  000000G                 MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
001440  005046                          CLR     -(SP)
001442  116116  000000G                 MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
001446  012746  000000G                 MOV     #MSG50,-(SP)
001452  012746  000004                  MOV     #4,-(SP)
001456  010600                          MOV     SP,R0                           ; SP,*
001460  104414                          TRAP    14
001462  104455                          TRAP    55                              ;                               3746
001464  003105                          .WORD   3105
001466  000000G                         .WORD   MSG00
001470  000000G                         .WORD   ERROR$REPORT
001472  062706  000014                  ADD     #14,SP                          ;                               3741
001476  005201                  19$:    INC     R1                              ; INDEX                         3738
001500  020127  000005                  CMP     R1,#5                           ; INDEX,*
001504  003721                          BLE     18$
001506  012701  000006                  MOV     #6,R1                           ; *,INDEX                       3749
001512  105761  000000G         20$:    TSTB    RCV.BUFFER(R1)                  ; *(INDEX)                      3750
001516  001447                          BEQ     21$
001520  013700  000000G                 MOV     REG.ADR,R0                      ;                               3753
001524  016066  000016  000022         MOV     16(R0),22(SP)                   ; *,TMP.LOCATION
001532  016637  000022  000000G         MOV     22(SP),CSR.WORD                 ; TMP.LOCATION,*
001540  012716  000000G                 MOV     #MSG59,(SP)                     ;                               3754
```

```
001544  012746  000001                          MOV     #1,-(SP)
001550  010600                                  MOV     SP,R0                           ; SP,*
001552  104414                                  TRAP    14
001554  012716  000000G                         MOV     #MSG51,(SP)                     ;                                       3755
001560  012746  000001                          MOV     #1,-(SP)
001564  010600                                  MOV     SP,R0                           ; SP,*
001566  104414                                  TRAP    14
001570  010116                                  MOV     R1,(SP)                         ; INDEX,*                               3756
001572  005046                                  CLR     -(SP)
001574  116116  000000G                         MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
001600  005046                                  CLR     -(SP)
001602  116116  000000G                         MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
001606  012746  000000G                         MOV     #MSG50,-(SP)
001612  012746  000004                          MOV     #4,-(SP)
001616  010600                                  MOV     SP,R0                           ; SP,*
001620  104414                                  TRAP    14
001622  104455                                  TRAP    55                              ;                                       3757
001624  003106                                  .WORD   3106
001626  000000G                                 .WORD   MSG00
001630  000000G                                 .WORD   ERROR$REPORT
001632  062706  000014                          ADD     #14,SP                          ;                                       3752
001636  005201                          21$:    INC     R1                              ; INDEX                                 3749
001640  020127  000011                          CMP     R1,#11                          ; INDEX,*
001644  003722                                  BLE     20$
001646  005001                                  CLR     R1                              ; INDEX                                 3760
001650  126161  000012G 000162G          22$:    CMPB    RCV.BUFFER+12(R1),TARGET.ADR+162(R1) ;
                                                                                        ; *(INDEX),*(INDEX)                     3761
001656  001447                                  BEQ     23$
001660  013700  000000G                         MOV     REG.ADR,R0                      ;                                       3764
001664  016066  000016  000024                  MOV     16(R0),24(SP)                   ; *,TMP.LOCATION
001672  016637  000024  000000G                 MOV     24(SP),CSR.WORD                 ; TMP.LOCATION,*
001700  012716  000000G                         MOV     #MSG59,(SP)                     ;                                       3765
001704  012746  000001                          MOV     #1,-(SP)
001710  010600                                  MOV     SP,R0                           ; SP,*
001712  104414                                  TRAP    14
001714  012716  000000G                         MOV     #MSG51,(SP)                     ;                                       3766
001720  012746  000001                          MOV     #1,-(SP)
001724  010600                                  MOV     SP,R0                           ; SP,*
001726  104414                                  TRAP    14
001730  010116                                  MOV     R1,(SP)                         ; INDEX,*                               3767
001732  005046                                  CLR     -(SP)
001734  116116  000000G                         MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
001740  005046                                  CLR     -(SP)
001742  116116  000000G                         MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
001746  012746  000000G                         MOV     #MSG50,-(SP)
001752  012746  000004                          MOV     #4,-(SP)
001756  010600                                  MOV     SP,R0                           ; SP,*
001760  104414                                  TRAP    14
001762  104455                                  TRAP    55                              ;                                       3768
001764  003107                                  .WORD   3107
001766  000000G                                 .WORD   MSG00
001770  000000G                                 .WORD   ERROR$REPORT
001772  062706  000014                          ADD     #14,SP                          ;                                       3763
```

```
001776  005201                          23$:    INC     R1                              ; INC                           3760
002000  020127  000005                          CMP     R1,#5                           ; INC  ,*
002004  003721                                  BLE     22$
002006  062706  000010                          ADD     #10,SP                          ;                               3696
002012  104467                                  TRAP    67                              ;                               3769
002014  006000                                  ROR     R0
002016  103002                                  BHIS    24$
002020  000137  016224'                        JMP     13$
002024  062706  000016                  24$:    ADD     #16,SP                          ;                               3567
002030  000207                                  RTS     PC

; Routine Size:  525 words,     Routine Base:  AB$CODE$ + 15250
; Maximum stack depth per invocation:  22 words



                                                .SBTTL  T16 TEST 16 - ODD PACKET TEST
000000  004737  015250'                  T16::
000000                                  1$:     JSR     PC,$T16                         ;                               3770
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 17302
; Maximum stack depth per invocation:  2 words


;   3773  1
```

```
;  3774  1     %SBTTL 'TEST 17 - STATION ADDRESS TEST'
;  3775  1     !++
;  3776  1     !
;  3777  1     !  TEST 17:     STATION ADDRESS TEST
;  3778  1     !
;  3779  1     !  DESCRIPTION:
;  3780  1     !
;  3781  1     !       This test verifies that DEQNA accepts only packets with legitimate
;  3782  1     !       'multicast' and 'non-multicast' addresses and discards those with
;  3783  1     !       illegitimate 'multicast' and 'non-multicast' addresses.
;  3784  1     !
;  3785  1     !       Station Address RAM is loaded with a set of Target Addresses and
;  3786  1     !       Mode bits. Target Addresses in and out of the set are used to
;  3787  1     !       loopback packets. If the operator specifies loop on error, the
;  3788  1     !       program re-executes the code that detected the error until ^C is
;  3789  1     !       entered.
;  3790  1     !
;  3791  1     !       Hardware tested:        Address Filter Circuitry
;  3792  1     !
;  3793  1     !       Set of 'multicast' addresses in HEXADECIMAL:
;  3794  1     !
;  3795  1     !               01-00-00-00-00-00
;  3796  1     !               AB-AA-AA-AA-AA-AA
;  3797  1     !               55-55-55-55-55-55
;  3798  1     !               FF-FF-FF-FF-FF-FF
;  3799  1     !               Walking 1
;  3800  1     !
;  3801  1     !       Processing:
;  3802  1     !
;  3803  1     !            BEGIN
;  3804  1     !               reset device
;  3805  1     !               select internal loopback mode
;  3806  1     !               set mode to Setup
;  3807  1     !               load Station Address RAM with 'multicast' addresses
;  3808  1     !               REPEAT for each complemented and uncomplemented 'multicast'
;  3809  1     !                      address in the set
;  3810  1     !                  load address
;  3811  1     !                  disable receiver
;  3812  1     !                  transmit loopback packet
;  3813  1     !                  enable receiver
;  3814  1     !                  check for expected loopback status
;  3815  1     !                  IF error
;  3816  1                        THEN
;  3817  1                           print error message if not inhibited
;  3818  1                        ENDIF
;  3819  1     !                  call compare_packets
;  3820  1     !               ENDREPEAT
;  3821  1     !            END
;  3822  1     !--
```

```
;    3823  3     BGNTST;
;    3824  3
;    3825  3       !++
;    3826  3       !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM TO ALL MULTICAST
;    3827  3       !  MODE.
;    3828  3       !--
;    3829  3
;    3830  3       RESET_DEQNA ( );
;    3831  3       PREP_FOR_SETUP ( );
;    3832  3       INCR INDEX1 FROM 6 TO 19 DO
;    3833  3         WRT_STATION_ADR ( .INDEX1 - 5, .INDEX1 );
;    3834  3
;    3835  5       BGNSUB;
;    3836  5         XMIT_SETUP_PACKET ( N_MODE );
;    3837  3       ENDSUB;
;    3838  3
;    3839  3       !++
;    3840  3       !  NOW LOOPBACK 6 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;    3841  3       !--
;    3842  3
;    3843  3       RBUF_LENGTH = 6;
;    3844  3       XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3845  3
;    3846  3       INCR INDEX1 FROM 6 TO 19 DO
;    3847  4         BEGIN
;    3848  4           WRT_STATION_ADR ( ZERO, .INDEX1 );
;    3849  4
;    3850  6           BGNSUB;
;    3851  6             XMIT_ILOOP_PACKET ( ZERO );
;    3852  4           ENDSUB;
;    3853  4
;    3854  4           INCR INDEX2 FROM 0 TO 5 DO
;    3855  5             BEGIN
;    3856  5               XMIT_BUFFER [ .INDEX2 ] = ( -.XMIT_BUFFER [ .INDEX2 ] ) - 1;
;    3857  5               TARGET_ADR [ .INDEX2 ] = .XMIT_BUFFER [ .INDEX2 ];
;    3858  4             END;
;    3859  4
;    3860  6           BGNSUB;
;    3861  6             XMIT_ILOOP_PACKET ( ONE );
;    3862  4           ENDSUB;
;    3863  3         END;
;    3864  3
;    3865  3       TEMP4 = 14;
;    3866  3       INCR INDEX3 FROM 0 TO 3 DO
;    3867  4         BEGIN
;    3868  4           IF .INDEX3 EQLU 3
;    3869  4             THEN
;    3870  4               TEMP4 = 6;
;    3871  4           RESET_DEQNA ( );
;    3872  4           PREP_FOR_SETUP ( );
;    3873  4           INCR INDEX4 FROM 1 TO .TEMP4 DO
;    3874  5             BEGIN
;    3875  5               WALKING_BIT ( ZERO, .INDEX4 + ( .INDEX3 * 14 ) - 1, 5 );
```

```
;    3876  5            WRT_STATION_ADR ( .INDEX4, ZERO );
;    3877  4          END;
;    3878  4
;    3879  6        BGNSUB;
;    3880  6          XMIT_SETUP_PACKET ( N_MODE );
;    3881  4        ENDSUB;
;    3882  4
;    3883  4        RBUF_LENGTH = 6;
;    3884  4        XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3885  4
;    3886  4        INCR INDEX4 FROM 1 TO .TEMP4 DO
;    3887  5          BEGIN
;    3888  5            WALKING_BIT ( ZERO, .INDEX4 + ( .INDEX3 * 14 )   1, 5 );
;    3889  5            WRT_STATION_ADR ( ZERO, ZERO );
;    3890  5
;    3891  7            BGNSUB;
;    3892  7              XMIT_ILOOP_PACKET ( ZERO );
;    3893  5            ENDSUB;
;    3894  4          END;
;    3895  4
;    3896  4        INCR INDEX2 FROM 0 TO 5 DO
;    3897  5          BEGIN
;    3898  5            XMIT_BUFFER [ .INDEX2 ] = ( -.XMIT_BUFFER [ .INDEX2 ] ) - 1;
;    3899  5            TARGET_ADR [ .INDEX2 ] = .XMIT_BUFFER [ .INDEX2 ];
;    3900  5
;    3901  7            BGNSUB;
;    3902  7              XMIT_ILOOP_PACKET ( ONE );
;    3903  5            ENDSUB;
;    3904  4          END;
;    3905  3      END;
;    3906  3
;    3907  3      INCR INDEX2 FROM 0 TO 5 DO
;    3908  3        TARGET_ADR [ .INDEX2 ] = ZERO;
;    3909  3
;    3910  1    ENDTST;


                                        .SBTTL  $T17 TEST 17 - STATION ADDRESS TEST
000000  004137  000000G          $T17:  JSR     R1,$SAVE4                    ;                          3772
000004  004737  000000G                 JSR     PC,RESET.DEQNA               ;                          3830
000010  004737  000000G                 JSR     PC,PREP.FOR.SETUP            ;                          3831
000014  012701  000006                  MOV     #6,R1                        ; *,INDEX1                 3832
000020  010146                  1$:     MOV     R1,-(SP)                     ; INDEX1,*                 3833
000022  162716  000005                  SUB     #5,(SP)
000026  010146                          MOV     R1,-(SP)                     ; INDEX1,*
000030  004737  000000G                 JSR     PC,WRT.STATION.ADR
000034  022626                          CMP     (SP)+,(SP)+
000036  005201                          INC     R1                           ; INDEX1                   3832
000040  020127  000023                  CMP     R1,#23                       ; INDEX1,*
000044  003765                          BLE     1$
000046  104402                  2$:     TRAP    2                            ;                          3833
000050  012746  000200                  MOV     #200,-(SP)                   ;                          3836
000054  004737  000000G                 JSR     PC,XMIT.SETUP.PACKET
```

```
000060  005726                          TST     (SP)+                   ;                               3833
000062  104467                          TRAP    67                      ;                               3836
000064  006000                          ROR     R0
000066  103767                          BLO     2$
000070  012737  000006  000000G         MOV     #6,RBUF.LENGTH          ;                               3843
000076  012700  000006                  MOV     #6,R0                   ;                               3844
000102  006200                          ASR     R0
000104  005400                          NEG     R0
000106  010037  000000G                 MOV     R0,XBUF.LENGTH
000112  012702  000006                  MOV     #6,R2                   ; *,INDEX1                      3846
000116  005046                  3$:     CLR     -(SP)                   ;                               3848
000120  010246                          MOV     R2,-(SP)                ; INDEX1,*
000122  004737  000000G                 JSR     PC,WRT.STATION.ADR
000126  104402                  4$:     TRAP    2
000130  005016                          CLR     (SP)                    ;                               3851
000132  004737  000000G                 JSR     PC,XMIT.ILOOP.PACKET
000136  104467                          TRAP    67
000140  006000                          ROR     R0
000142  103771                          BLO     4$
000144  005000                          CLR     R0                      ; INDEX2                        3854
000146  012701  000000G         5$:     MOV     #XMIT.BUFFER,R1         ;                               3856
000152  060001                          ADD     R0,R1                   ; INDEX2,*
000154  012703  177777                  MOV     #-1,R3
000160  005004                          CLR     R4
000162  151104                          BISB    (R1),R4
000164  160403                          SUB     R4,R3
000166  110311                          MOVB    R3,(R1)
000170  110360  000000G                 MOVB    R3,TARGET.ADR(R0)       ; *,*(INDEX2)                   3857
000174  005200                          INC     R0                      ; INDEX2                        3854
000176  020027  000005                  CMP     R0,#5                   ; INDEX2,*
000202  003761                          BLE     5$
000204  104402                  6$:     TRAP    2                       ;                               3858
000206  012716  000001                  MOV     #1,(SP)                 ;                               3861
000212  004737  000000G                 JSR     PC,XMIT.ILOOP.PACKET
000216  104467                          TRAP    67
000220  006000                          ROR     R0
000222  103770                          BLO     6$
000224  022626                          CMP     (SP)+,(SP)+             ;                               3847
000226  005202                          INC     R2                      ; INDEX1                        3846
000230  020227  000023                  CMP     R2,#23                  ; INDEX1,*
000234  003730                          BLE     3$
000236  012737  000016  000000G         MOV     #16,TEMP4               ;                               3865
000244  005004                          CLR     R4                      ; INDEX3                        3866
000246  022727  000000  000003         CMP     #0,#3                   ;                               3868
000254  001003                  7$:     BNE     8$
000256  012737  000006  000000G         MOV     #6,TEMP4                ;                               3870
000264  004737  000000G         8$:     JSR     PC,RESET.DEQNA          ;                               3871
000270  004737  000000G                 JSR     PC,PREP.FOR.SETUP       ;                               3872
000274  013702  000000G                 MOV     TEMP4,R2                ;                               3873
000300  010401                          MOV     R4,R1                   ; INDEX3,*                      3875
000302  070127  000016                  MUL     #16,R1
000306  005003                          CLR     R3                      ; INDEX4                        3873
000310  000417                          BR      10$
```

```
000312  005046                          9$:     CLR     -(SP)                   ;                               3875
000314  010100                                  MOV     R1,R0
000316  060300                                  ADD     R3,R0                   ; INDEX4,*
000320  010046                                  MOV     R0,-(SP)
000322  005316                                  DEC     (SP)
000324  012746  000005                         MOV     #5,-(SP)
000330  004737  000000G                       JSR     PC,WALKING.BIT
000334  010316                                  MOV     R3,(SP)                 ; INDEX4,*                      3876
000336  005046                                  CLR     -(SP)
000340  004737  000000G                       JSR     PC,WRT.STATION.ADR
000344  062706  000010                        ADD     #10,SP                  ;                               3874
000350  005203                          10$:    INC     R3                      ; INDEX4                        3873
000352  020302                                  CMP     R3,R2                   ; INDEX4,*
000354  003756                                  BLE     9$
000356  104402                          11$.    TRAP    2                       ;                               3877
000360  012746  000200                        MOV     #200,-(SP)              ;                               3880
000364  004737  000000G                       JSR     PC,XMIT.SETUP.PACKET
000370  005726                                  TST     (SP)+                   ;                               3877
000372  104467                                  TRAP    67                      ;                               3880
000374  006000                                  ROR     R0
000376  103767                                  BLO     11$
000400  012737  000006  000000G                MOV     #6,RBUF.LENGTH          ;                               3883
000406  012700  000006                        MOV     #6,R0                   ;                               3884
000412  006200                                  ASR     R0
000414  005400                                  NEG     R0
000416  010037  000000G                       MOV     R0,XBUF.LENGTH
000422  013703  000000G                       MOV     TEMP4,R3                ;                               3886
000426  005002                                  CLR     R2                      ; INDEX4
000430  000426                                  BR      14$
000432  005046                          12$:    CLR     -(SP)                   ;                               3888
000434  010100                                  MOV     R1,R0
000436  060200                                  ADD     R2,R0                   ; INDEX4,*
000440  010046                                  MOV     R0,-(SP)
000442  005316                                  DEC     (SP)
000444  012746  000005                         MOV     #5,-(SP)
000450  004737  000000G                       JSR     PC,WALKING.BIT
000454  005016                                  CLR     (SP)                    ;                               3889
000456  005046                                  CLR     -(SP)
000460  004737  000000G                       JSR     PC,WRT.STATION.ADR
000464  104402                          13$:    TRAP    2
000466  005016                                  CLR     (SP)                    ;                               3892
000470  004737  000000G                       JSR     PC,XMIT.ILOOP.PACKET
000474  104467                                  TRAP    67
000476  006000                                  ROR     R0
000500  103771                                  BLO     13$
000502  062706  000010                        ADD     #10,SP                  ;                               3887
000506  005202                          14$:    INC     R2                      ; INDEX4                        3886
000510  020203                                  CMP     R2,R3                   ; INDEX4,*
000512  003747                                  BLE     12$
000514  005001                                  CLR     R1                      ; INDEX2                        3896
000516  012700  000000G                15$:    MOV     #XMIT.BUFFER,R0         ;                               3898
000522  060100                                  ADD     R1,R0                   ; INDEX2,*
000524  012702  177777                        MOV     #-1,R2
```

```
000530  005003                                  CLR     R3
000532  151003                                  BISB    (R0),R3
000534  160302                                  SUB     R3,R2
000536  110210                                  MOVB    R2,(R0)
000540  110261  000000G                         MOVB    R2,TARGET.ADR(R1)       ; *,*(INDEX2)                   3899
000544  104402                          16$:    TRAP    2
000546  012746  000001                          MOV     #1,-(SP)                ;                               3902
000552  004737  000000G                         JSR     PC,XMIT.ILOOP.PACKET
000556  005726                                  TST     (SP)+                   ;                               3899
000560  104467                                  TRAP    67                      ;                               3902
000562  006000                                  ROR     R0
000564  103767                                  BLO     16$
000566  005201                                  INC     R1                      ; INDEX2                        3896
000570  020127  000005                          CMP     R1,#5                   ; INDEX2,*
000574  003750                                  BLE     15$
000576  005204                                  INC     R4                      ; INDEX3                        3866
000600  020427  000003                          CMP     R4,#3                   ; INDEX3,*
000604  003623                                  BLE     7$
000606  005000                                  CLR     R0                      ; INDEX2                        3907
000610  105060  000000G                 17$:    CLRB    TARGET.ADR(R0)          ; *(INDEX2)                     3908
000614  005200                                  INC     R0                      ; INDEX2                        3907
000616  020027  000005                          CMP     R0,#5                   ; INDEX2,*
000622  003772                                  BLE     17$
000624  000207                                  RTS     PC                      ;                               3772

; Routine Size:  203 words,     Routine Base:  AB$CODE$ + 17316
; Maximum stack depth per invocation:  11 words




                                        .SBTTL  T17 TEST 17 - STATION ADDRESS TEST
000000  004737  017316'                 T17::
000000                                  1$:     JSR     PC,$T17                 ;                               3908
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 20144
; Maximum stack depth per invocation:  2 words


;   3911  1
;   3912  1
```

```
ZQNA3           CZQNADO DEQNA FUNCTIONAL TEST                    14-Mar-1985 13:11:16    VAX-11 Bliss-16 V4 1-582          SEQ 0200
V01.0           TEST 18 - ALL MULTICAST STATION ADDRESS TEST     14-Mar-1985 13:05:35    DISK$USER2:[MARSHALL.DEQNA]ZQNA3.BLI;4  (41)

;    3913  1     %SBTTL 'TEST 18 - ALL MULTICAST STATION ADDRESS TEST'
;    3914  1     !++
;    3915  1     !
;    3916  1     !  TEST 18:     ALL MULTICAST STATION ADDRESS TEST
;    3917  1     !
;    3918  1     !  DESCRIPTION:
;    3919  1     !
;    3920  1     !       This test vefifies that DEQNA recognizes 'all multicast' addresses of
;    3921  1     !       the node and discards loopback packets with non-enabled addresses.
;    3922  1     !       If the operator specifies loop on error, the program re-executes the
;    3923  1     !       code that detected the error until ^C is entered.
;    3924  1     !
;    3925  1     !       Hardware tested:        All Multicast Addressing
;    3926  1     !                               I8051 Microprocessor
;    3927  1     !                               Address Filter Circuitry
;    3928  1     !
;    3929  1     !       Set of  'all multicast' addresses:
;    3930  1     !
;    3931  1     !               DEQNA Physical Addr     FF-FF-FF-FF-FF-FF
;    3932  1     !               AA-00-00-00-00-00       55-55-55-55-55-55
;    3933  1     !               AA-00-02-AA-AA-AA       AA-AA-AA-AA-AA-AA
;    3934  1     !               AA-00-05-55-55-55       01-00-00-00-00-00
;    3935  1     !               AA-00-04-FF-FF-FF       AB-AA-AA-AA-AA-AA
;    3936  1     !               AA-00-04-00-00-00       FF-00-01-02-03-04
;    3937  1     !               AA-00-04-18-81-18       00-F4-FA-44-44-55
;    3938  1     !
;    3939  1     !       Processing:
;    3940  1     !
;    3941  1     !            BEGIN
;    3942  1     !               reset device
;    3943  1     !               select internal loopback mode
;    3944  1     !               set mode to Setup
;    3945  1     !               load Station Address RAM with 'all multicast' addresses
;    3946  1     !               REPEAT for 'all multicast' addresses in and out of set
;    3947  1     !                  load 'all multicast' address of the packet
;    3948  1     !                  disable receiver
;    3949  1     !                  transmit loopback packet
;    3950  1     !                   enable receiver
;    3951  1     !                  check for expected loopback status
;    3952  1     !                  IF error
;    3953  1     !                  THEN
;    3954  1     !                     print error message if not inhibited
;    3955  1     !                  ENDIF
;    3956  1     !                  call compare_packets
;    3957  1     !               ENDREPEAT
;    3958  1     !            END
;    3959  1     !--
```

```
;    3960  3     BGNTST;
;    3961  3
;    3962  3       !++
;    3963  3       !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM IF EXECUTING
;    3964  3       !  TESTS IN EXTERNAL LOOPBACK MODE.
;    3965  3       !--
;    3966  3
;    3967  3       RESET_DEQNA ( );
;    3968  3       PREP_FOR_SETUP ( );
;    3969  3       INCR INDEX1 FROM 1 TO 13 DO
;    3970  3         WRT_STATION_ADR ( .INDEX1, .INDEX1 );
;    3971  3       WRT_STATION_ADR ( 14, PHA_INDEX );
;    3972  3
;    3973  5       BGNSUB;
;    3974  5         XMIT_SETUP_PACKET ( A_MODE );
;    3975  3       ENDSUB;
;    3976  3
;    3977  3       !++
;    3978  3       !  NOW LOOPBACK 6 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;    3979  3       !--
;    3980  3
;    3981  3       RBUF_LENGTH = 6;
;    3982  3       XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3983  3
;    3984  3       INCR INDEX FROM 6 TO 19 DO
;    3985  4         BEGIN
;    3986  4           WRT_STATION_ADR ( ZERO, .INDEX );
;    3987  4
;    3988  6           BGNSUB;
;    3989  6             XMIT_ILOOP_PACKET ( ZERO );
;    3990  4           ENDSUB;
;    3991  4
;    3992  4           INCR INDEX2 FROM 0 TO 3 DO
;    3993  5             BEGIN
;    3994  5               XMIT_BUFFER [ .INDEX2 ] = ( -.XMIT_BUFFER [ .INDEX2 ] ) - 1;
;    3995  5               TARGET_ADR [ .INDEX2 ] = .XMIT_BUFFER [ .INDEX2 ];
;    3996  4             END;
;    3997  4
;    3998  4           XMIT_BUFFER [ ZERO ] = .XMIT_BUFFER [ ZERO ] AND %O'177774';
;    3999  4           TARGET_ADR [ ZERO ] = .XMIT_BUFFER [ ZERO ];
;    4000  4
;    4001  6           BGNSUB;
;    4002  6             XMIT_ILOOP_PACKET ( ONE );
;    4003  4           ENDSUB;
;    4004  4
;    4005  3         END;
;    4006  3
;    4007  3         INCR INDEX2 FROM 0 TO 5 DO
;    4008  3           TARGET_ADR [ .INDEX2 ] = ZERO;
;    4009  3
;    4010  1     ENDTST;
```

```
                                        .SBTTL  $T18 TEST 18   ALL MULTICAST STATION ADDRESS TEST
000000  004137  000000G                $T18:   JSR     R1,$SAVE4                       ;                               3910
000004  004737  000000G                        JSR     PC,RESET.DEQNA                  ;                               3967
000010  004737  000000G                        JSR     PC,PREP.FOR.SETUP               ;                               3968
000014  012701  000001                        MOV     #1,R1                           ; *,INDEX1                      3969
000020  010146                          1$:     MOV     R1,-(SP)                        ; INDEX1,*                      3970
000022  010146                                  MOV     R1,-(SP)                        ; INDEX1,*
000024  004737  000000G                        JSR     PC,WRT.STATION.ADR
000030  022626                                CMP     (SP)+,(SP)+
000032  005201                                INC     R1                              ; INDEX1                        3969
000034  020127  000015                        CMP     R1,#15                          ; INDEX1,*
000040  003767                                BLE     1$
000042  012746  000016                        MOV     #16,-(SP)                       ;                               3971
000046  012746  000023                        MOV     #23,-(SP)
000052  004737  000000G                        JSR     PC,WRT.STATION.ADR
000056  104402                          2$:     TRAP    2
000060  012716  000201                        MOV     #201,(SP)                       ;                               3974
000064  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000070  104467                                TRAP    67
000072  006000                                ROR     R0
000074  103770                                BLO     2$
000076  012737  000006  000000G               MOV     #6,RBUF.LENGTH                  ;                               3981
000104  012700  000006                        MOV     #6,R0                           ;                               3982
000110  006200                                ASR     R0
000112  005400                                NEG     R0
000114  010037  000000G                        MOV     R0,XBUF.LENGTH
000120  012702  000006                        MOV     #6,R2                           ; *,INDEX                       3984
000124  005016                          3$:     CLR     (SP)                            ;                               3986
000126  010246                                  MOV     R2,-(SP)                        ; INDEX,*
000130  004737  000000G                        JSR     PC,WRT.STATION.ADR
000134  104402                          4$:     TRAP    2
000136  005016                                CLR     (SP)                            ;                               3989
000140  004737  000000G                        JSR     PC,XMIT.ILOOP.PACKET
000144  104467                                TRAP    67
000146  006000                                ROR     R0
000150  103771                                BLO     4$
000152  005000                                CLR     R0                              ; INDEX2                        3992
000154  012701  000000G                 5$:    MOV     #XMIT.BUFFER,R1                 ;                               3994
000160  060001                                ADD     R0,R1                           ; INDEX2,*
000162  012703  177777                        MOV     #-1,R3
000166  005004                                CLR     R4
000170  151'04                                BISB    (R1),R4
000172  160403                                SUB     R4,R3
000174  110311                                MOVB    R3,(R1)
000176  110360  000000G                        MOVB    R3,TARGET.ADR(R0)               ; *,*(INDEX2)                   3995
000202  005200                                INC     R0                              ; INDEX2                        3992
000204  020027  000005                        CMP     R0,#5                           ; INDEX2,*
000210  003761                                BLE     5$
000212  142737  000003  000000G               BICB    #3,XMIT.BUFFER                  ;                               3998
000220  113737  000000G 000000G               MOVB    XMIT.BUFFER,TARGET.ADR          ;                               3999
000226  104402                          6$:     TRAP    2
000230  012716  000001                        MOV     #1,(SP)                         ;                               4002
000234  004737  000000G                        JSR     PC,XMIT.ILOOP.PACKET
```

```
000240  104467                                  TRAP    67
000242  006000                                  ROR     R0
000244  103770                                  BLO     6$
000246  005726                                  TST     (SP)+                           ;                                       3985
000250  005202                                  INC     R2                              ; INDEX                                 3984
000252  020227  000023                          CMP     R2,#23                          ; INDEX,*
000256  003722                                  BLE     3$
000260  005000                                  CLR     R0                              ; INDEX2                                4007
000262  105060  000000G                 7$:     CLRB    TARGET.ADR(R0)                  ; *(INDEX2)                             4008
000266  005200                                  INC     R0                              ; INDEX2                                4007
000270  020027  000005                          CMP     R0,#5                           ; INDEX2,*
000274  003772                                  BLE     7$
000276  022626                                  CMP     (SP)+,(SP)+                     ;                                       3910
000300  000207                                  RTS     PC

; Routine Size:  97 words,      Routine Base:  AB$CODE$ + 20160
; Maximum stack depth per invocation:  10 words



                                        .SBTTL  T18 TEST 18 - ALL MULTICAST STATION ADDRESS TEST
000000  004737  020160'                 T18::
000000                                  1$:     JSR     PC,$T18                         ;                                       4008
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 20462
; Maximum stack depth per invocation:  2 words


;   4011  1
;   4012  1
```

```
;  4013  1     %SBTTL 'TEST 19   RUNT PACKET TEST'
;  4014  1     !++
;  4015  1     !
;  4016  1     !  TEST 19:     RUNT PACKET TEST
;  4017  1     !
;  4018  1     !  DESCRIPTION:
;  4019  1     !
;  4020  1     !       This test verifies that the DEQNA can detect runt packets in FIFO.
;  4021  1     !       If the operator specifies loop on error, the program re-executes the
;  4022  1     !       code that detected the error until ↑C is entered.
;  4023  1     !
;  4024  1     !       Hardware tested:        EPP
;  4025  1     !                               Address Filter Circuitry
;  4026  1     !
;  4027  1     !       Station Address table:
;  4028  1     !
;  4029  1     !                       DEQNA Physical Addr
;  4030  1     !                       AA-00-00-00-00-00
;  4031  1     !                       AA-00-02-AA-AA-AA
;  4032  1     !                       AA-00-05-55-55-55
;  4033  1     !                       AA-00-04-FF-FF-FF
;  4034  1     !                       AA-00-04-00-00-00
;  4035  1     !                       AA-00-04-18-81-18
;  4036  1     !
;  4037  1     !       Processing:
;  4038  1     !
;  4039  1     !            BEGIN
;  4040  1     !               reset device
;  4041  1     !               select internal loopback mode
;  4042  1     !               load Station Address RAM with Station Addresses from table
;  4043  1     !               load packet with valid Station Address
;  4044  1     !               disable receiver
;  4045  1     !               transmit loopback packet
;  4046  1     !               enable receiver
;  4047  1     !               check for expected loopback status
;  4048  1     !               IF error
;  4049  1     !               THEN
;  4050  1     !                  print error message if not inhibited
;  4051  1     !               ENDIF
;  4052  1     !               load packet with invalid Station Address
;  4053  1     !               disable receiver
;  4054  1     !               transmit loopback packet
;  4055  1     !               enable receiver
;  4056  1     !               check for expected loopback status
;  4057  1     !               IF error
;  4058  1     !               THEN
;  4059  1     !                  print error message if not inhibited
;  4060  1     !               ENDIF
;  4061  1     !            END
;  4062  1     !--
```

```
;    4063  3     BGNTST;
;    4064  3
;    4065  3       !++
;    4066  3       !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM IF EXECUTING
;    4067  3       !  TESTS IN EXTERNAL LOOPBACK MODE.
;    4068  3       !--
;    4069  3
;    4070  3       RESET_DEQNA ( );
;    4071  3       PREP_FOR_SETUP ( );
;    4072  3       INCR INDEX1 FROM 6 TO 19 DO
;    4073  3         WRT_STATION_ADR ( .INDEX1 - 5, PHA_INDEX );
;    4074  3
;    4075  5       BGNSUB;
;    4076  5             XMIT_SETUP_PACKET ( N_MODE );
;    4077  3       ENDSUB;
;    4078  3
;    4079  3       !++
;    4080  3       !  NOW LOOPBACK 6 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;    4081  3       !--
;    4082  3
;    4083  3       RBUF_LENGTH = 6;
;    4084  3       XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    4085  3
;    4086  3       WRT_STATION_ADR ( ZERO, PHA_INDEX );
;    4087  3
;    4088  5       BGNSUB;
;    4089  5         XMIT_ILOOP_PACKET ( ZERO );
;    4090  3       ENDSUB;
;    4091  3
;    4092  5       BGNSUB;
;    4093  5         WRT_STATION_ADR ( ZERO, 2 );
;    4094  5
;    4095  5         .IOP_TABLE [ CSR ] = ONE;
;    4096  5
;    4097  5         SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    4098  5         .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;    4099  5         .IOP_TABLE [ RHI_ADR ] = ZERO;
;    4100  5
;    4101  5         SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    4102  5         .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;    4103  5         .IOP_TABLE [ XHI_ADR ] = ZERO;
;    4104  5
;    4105  5         CHK_RIXI_STATUS ( ZERO );
;    4106  5         CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK      );      ! 0'100220', 0'100220'
;    4107  5         CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );      ! 0'140000', 0'000400'
;    4108  5         CHK_RCV_STATUS  ( RFLG_STATUS, RWD16_STATUS );      ! 0'140000', 0'044000'
;    4109  5
;    4110  5         .IOP_TABLE [ CSR ] = ZERO;
;    4111  3       ENDSUB;
;    4112  3
;    4113  1     ENDTST;
```

```
                                        .SBTTL  $T19 TEST 19 - RUNT PACKET TEST
000000  010146                  $T19:   MOV     R1,-(SP)                ;                               4010
000002  004737  000000G                 JSR     PC,RESET.DEQNA          ;                               4070
000006  004737  000000G                 JSR     PC,PREP.FOR.SETUP       ;                               4071
000012  012701  000006                  MOV     #6,R1                   ; *,INDEX1                      4072
000016  010146                  1$:     MOV     R1,-(SP)                ; INDEX1,*                      4073
000020  162716  000005                  SUB     #5,(SP)
000024  012746  000023                  MOV     #23,-(SP)
000030  004737  000000G                 JSR     PC,WRT.STATION.ADR
000034  022626                          CMP     (SP)+,(SP)+
000036  005201                          INC     R1                      ; INDEX1                        4072
000040  020127  000023                  CMP     R1,#23                  ; INDEX1,*
000044  003764                          BLE     1$
000046  104402                  2$:     TRAP    2                       ;                               4073
000050  012746  000200                  MOV     #200,-(SP)              ;                               4076
000054  004737  000000G                 JSR     PC,XMIT.SETUP.PACKET
000060  005726                          TST     (SP)+                   ;                               4073
000062  104467                          TRAP    67                      ;                               4076
000064  006000                          ROR     R0
000066  103767                          BLO     2$
000070  012737  000006  000000G         MOV     #6,RBUF.LENGTH          ;                               4083
000076  012700  000006                  MOV     #6,R0                   ;                               4084
000102  006200                          ASR     R0
000104  005400                          NEG     R0
000106  010037  000000G                 MOV     R0,XBUF.LENGTH
000112  005046                          CLR     -(SP)                   ;                               4086
000114  012746  000023                  MOV     #23,-(SP)
000120  004737  000000G                 JSR     PC,WRT.STATION.ADR
000124  104402                  3$:     TRAP    2
000126  005016                          CLR     (SP)                    ;                               4089
000130  004737  000000G                 JSR     PC,XMIT.ILOOP.PACKET
000134  104467                          TRAP    67
000136  006000                          ROR     R0
000140  103771                          BLO     3$
000142  104402                  4$:     TRAP    2                       ;                               4090
000144  005016                          CLR     (SP)                    ;                               4093
000146  012746  000002                  MOV     #2,-(SP)
000152  004737  000000G                 JSR     PC,WRT.STATION.ADR
000156  012777  000001  000016G         MOV     #1,@IOP.TABLE+16        ;                               4095
000164  013716  000000G                 MOV     XBUF.LENGTH,(SP)        ;                               4097
000170  012746  120000                  MOV     #-60000,-(SP)
000174  004737  000000G                 JSR     PC,SET.RDESCR.LIST
000200  012777  000000G  000004G        MOV     #RCV.D.LIST,@IOP.TABLE+4        ;                       4098
000206  005077  000006G                 CLR     @IOP.TABLE+6            ;                               4099
000212  013716  000000G                 MOV     XBUF.LENGTH,(SP)        ;                               4101
000216  012746  120000                  MOV     #-60000,-(SP)
000222  004737  000000G                 JSR     PC,SET.XDESCR.LIST
000226  012777  000000G  000010G        MOV     #XMIT.D.LIST,@IOP.TABLE+10      ;                       4102
000234  005077  000012G                 CLR     @IOP.TABLE+12           ;                               4103
000240  005016                          CLR     (SP)                    ;                               4105
000242  004737  000000G                 JSR     PC,CHK.RIXI.STATUS
000246  012716  100220                  MOV     #-77560,(SP)            ;                               4106
000252  011646                          MOV     (SP),-(SP)
```

```
000254  004737  000000G                        JSR     PC,CHK.CSR.STATUS
000260  012716  140000                         MOV     #-40000,(SP)                    ;                4107
000264  012746  000400                         MOV     #400,-(SP)
000270  004737  000000G                        JSR     PC,CHK.XMIT.STATUS
000274  012716  140000                         MOV     #-40000,(SP)                    ;                4108
000300  012746  044000                         MOV     #44000,-(SP)
000304  004737  000000G                        JSR     PC,CHK.RCV.STATUS
000310  005077  000016G                        CLR     @IOP.TABLE+16                   ;                4110
000314  062706  000014                         ADD     #14,SP                          ;                4090
000320  104467                                 TRAP    67                              ;                4110
000322  006000                                 ROR     R0
000324  103706                                 BLO     4$
000326  022626                                 CMP     (SP)+,(SP)+                     ;                4010
000330  012601                                 MOV     (SP)+,R1
000332  000207                                 RTS     PC

; Routine Size:  110 words,     Routine Base:  AB$CODE$ + 20476
; Maximum stack depth per invocation:  10 words




                                               .SBTTL  T19 TEST 19 - RUNT PACKET TEST
000000  004737  020476'                 T19::
000000                                   1$:     JSR     PC,$T19                         ;                4111
000004  104466                                 TRAP    66
000006  006000                                 ROR     R0
000010  103773                                 BLO     1$
000012  000207                                 RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 21032
; Maximum stack depth per invocation:  2 words


;   4114  1
;   4115  1
```

```
;    4116  1     %SBTTL 'TEST 20 - FIFO OVERFLOW TEST'
;    4117  1     !++
;    4118  1     !
;    4119  1     !  TEST 20:     FIFO OVERFLOW TEST
;    4120  1     !
;    4121  1     !  DESCRIPTION:
;    4122  1     !
;    4123  1     !       This test verifies that the Ethernet Protocol Processor can
;    4124  1     !       detect receive FIFO overflow condition. If the operator specifies
;    4125  1     !       loop on error, the program re-executes the code that detected the
;    4126  1     !       error until ^C is entered.
;    4127  1     !
;    4128  1     !       Hardware tested:  RCV Status wd 1 - error summary (bit 14),
;    4129  1     !                                           FIFO overflow (bit 0),
;    4130  1     !                                           Byte FIFO in the EDLC,
;    4131  1     !                                           and discard packet (bit 12)
;    4132  1     !       Processing:
;    4133  1     !
;    4134  1     !            BEGIN
;    4135  1     !               reset device
;    4136  1     !               select loopback mode
;    4137  1     !               enable receiver ( set CSR bit 0)
;    4138  1     !               transmit loopback packet
;    4139  1     !               transmit another loopback packet
;    4140  1     !               check for expected loopback status
;    4141  1     !               IF error
;    4142  1     !               THEN
;    4143  1     !                  print error message if not inhibited
;    4144  1     !               ENDIF
;    4145  1     !
;    4146  1     !               reset device
;    4147  1     !               transmit loopback packet
;    4148  1     !               transmit a packet
;    4149  1     !               setup Receive Descriptor List
;    4150  1     !               enable receiver (set CSR BIT 0)
;    4151  1     !               check for expected loopback status
;    4152  1     !               IF error
;    4153  1     !               THEN
;    4154  1     !                  print error message if not inhibited
;    4155  1     !               ENDIF
;    4156  1     !               turn of 3 LED's on the module
;    4157  1     !            END
;    4158  1     !--
```

```
;  4159  3     BGNTST;
;  4160  3
;  4161  3       !++
;  4162  3       !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM
;  4163  3       !--
;  4164  3
;  4165  3       RESET_DEQNA ( );
;  4166  3       PREP_FOR_SETUP ( );
;  4167  3       INCR INDEX1 FROM 1 TO 14 DO
;  4168  3         WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  4169  3
;  4170  5       BGNSUB;
;  4171  5         XMIT_SETUP_PACKET ( P_MODE );
;  4172  3       ENDSUB;
;  4173  3
;  4174  3       !++
;  4175  3       !  LOOPBACK 2 6-BYTE PACKETS IN INTERNAL LOOPBACK MODE CHECK IF PACKETS
;  4176  3       !  WERE RECEIVED PROPERLY, SHOULD TRANSMIT AND RECEIVE  PROPERLY.
;  4177  3       !--
;  4178  3
;  4179  3       RBUF_LENGTH = 6;
;  4180  3       XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  4181  3
;  4182  3       INCR INDEX FROM 2 TO 3 DO
;  4183  4         BEGIN
;  4184  4           WRT_STATION_ADR ( ZERO, .INDEX );
;  4185  4
;  4186  6           BGNSUB;
;  4187  6             XMIT_ILOOP_PACKET ( ZERO );
;  4188  4           ENDSUB;
;  4189  3         END;
;  4190  3
;  4191  3       !++
;  4192  3       !  FORCE RECEIVE FIFO OVERFLOW ( RCV STATUS WD 1 - BIT 0 ) BY TRANSMITTING
;  4193  3       !  2 ND 6-BYTE PACKET IN INTERNAL LOOPBACK MODE BEFORE RECEIVING FIRST PACKET
;  4194  3       !--
;  4195  3
;  4196  5       BGNSUB;
;  4197  5         .IOP_TABLE [ CSR ] = ZERO;
;  4198  5
;  4199  5         WRT_STATION_ADR ( ZERO, 2 );
;  4200  5
;  4201  5         SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  4202  5         .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;  4203  5         .IOP_TABLE [ XHI_ADR ] = ZERO;
;  4204  5
;  4205  5         CHK_RIXI_STATUS ( ONE );
;  4206  5         WRT_STATION_ADR ( ZERO, 3 );
;  4207  5
;  4208  5         SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  4209  5         .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;  4210  5         .IOP_TABLE [ XHI_ADR ] = ZERO;
;  4211  5
```

```
;  4212  5          SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  4213  5          .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;  4214  5          .IOP_TABLE [ RHI_ADR ] = ZERO;
;  4215  5
;  4216  5          .IOP_TABLE [ CSR ] = ONE;
;  4217  5
;  4218  5          CHK_RIXI_STATUS ( ZERO );
;  4219  5          CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK      );      ! 0'100220', 0'100220'
;  4220  5          CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );      ! 0'140000', 0'000400'
;  4221  5          CHK_RCV_STATUS  ( RFLG_STATUS, RWD15_STATUS );      ! 0'140000', 0'000001'
;  4222  5
;  4223  5          .IOP_TABLE [ CSR ] = ZERO;
;  4224  3        ENDSUB;
;  4225  3
;  4226  3        RESET_DEQNA ( );
;  4227  3
;  4228  3        TURN_OFF_LED ( N_MODE );
;  4229  3        TURN_OFF_LED ( LED1 );
;  4230  3        TURN_OFF_LED ( LED2 );
;  4231  3        TURN_OFF_LED ( LED3 );
;  4232  3
;  4233  1      ENDTST;


                                        .SBTTL  $T20 TEST 20 - FIFO OVERFLOW TEST
000000  010146                  $T20:   MOV     R1,-(SP)                        ;                               4113
000002  004737  000000G                 JSR     PC,RESET.DEQNA                  ;                               4165
000006  004737  000000G                 JSR     PC,PREP.FOR.SETUP               ;                               4166
000012  012701  000001                  MOV     #1,R1                           ; *,INDEX1                      4167
000016  010146                  1$:     MOV     R1,-(SP)                        ; INDEX1,*                      4168
000020  012746  000023                  MOV     #23,-(SP)
000024  004737  000000G                 JSR     PC,WRT.STATION.ADR
000030  022626                          CMP     (SP)+,(SP)+
000032  005201                          INC     R1                              ; INDEX1                        4167
000034  020127  000016                  CMP     R1,#16                          ; INDEX1,*
000040  003766                          BLE     1$
000042  104402                  2$:     TRAP    2                               ;                               4168
000044  012746  000202                  MOV     #202,-(SP)                      ;                               4171
000050  004737  000000G                 JSR     PC,XMIT.SETUP.PACKET
000054  005726                          TST     (SP)+                           ;                               4168
000056  104467                          TRAP    67                              ;                               4171
000060  006000                          ROR     R0
000062  103767                          BLO     2$
000064  012737  000006  000000G         MOV     #6,RBUF.LENGTH                  ;                               4179
000072  012700  000006                  MOV     #6,R0                           ;                               4180
000076  006200                          ASR     R0
000100  005400                          NEG     R0
000102  010037  000000G                 MOV     R0,XBUF.LENGTH
000106  012701  000002                  MOV     #2,R1                           ; *,INDEX                       4182
000112  005046                  3$:     CLR     -(SP)                           ;                               4184
000114  010146                          MOV     R1,-(SP)                        ; INDEX,*
000116  004737  000000G                 JSR     PC,WRT.STATION.ADR
000122  104402                  4$:     TRAP    2
```

```
000124  005016                          CLR     (SP)                    ;                               4187
000126  004737  000000G                 JSR     PC,XMIT.ILOOP.PACKET
000132  104467                          TRAP    67
000134  006000                          ROR     R0
000136  103771                          BLO     4$
000140  022626                          CMP     (SP)+,(SP)+             ;                               4183
000142  005201                          INC     R1                      ; INDEX                         4182
000144  020127  000003                  CMP     R1,#3                   ; INDEX,*
000150  003760                          BLE     3$
000152  104402                  5$:     TRAP    2                       ;                               4189
000154  005077  000016G                 CLR     @IOP.TABLE+16           ;                               4197
000160  005046                          CLR     -(SP)                   ;                               4199
000162  012746  000002                  MOV     #2,-(SP)
000166  004737  000000G                 JSR     PC,WRT.STATION.ADR
000172  013716  000000G                 MOV     XBUF.LENGTH,(SP)        ;                               4201
000176  012746  120000                  MOV     #-60000,-(SP)
000202  004737  000000G                 JSR     PC,SET.XDESCR.LIST
000206  012777  000000G 000010G         MOV     #XMIT.D.LIST,@IOP.TABLE+10  ;                           4202
000214  005077  000012G                 CLR     @IOP.TABLE+12           ;                               4203
000220  012716  000001                  MOV     #1,(SP)                 ;                               4205
000224  004737  000000G                 JSR     PC,CHK.RIXI.STATUS
000230  005016                          CLR     (SP)                    ;                               4206
000232  012746  000003                  MOV     #3,-(SP)
000236  004737  000000G                 JSR     PC,WRT.STATION.ADR
000242  013716  000000G                 MOV     XBUF.LENGTH,(SP)        ;                               4208
000246  012746  120000                  MOV     #-60000,-(SP)
000252  004737  000000G                 JSR     PC,SET.XDESCR.LIST
000256  012777  000000G 000010G         MOV     #XMIT.D.LIST,@IOP.TABLE+10  ;                           4209
000264  005077  000012G                 CLR     @IOP.TABLE+12           ;                               4210
000270  013716  000000G                 MOV     XBUF.LENGTH,(SP)        ;                               4212
000274  012746  120000                  MOV     #-60000,-(SP)
000300  004737  000000G                 JSR     PC,SET.RDESCR.LIST
000304  012777  000000G 000004G         MOV     #RCV.D.LIST,@IOP.TABLE+4    ;                           4213
000312  005077  000006G                 CLR     @IOP.TABLE+6            ;                               4214
000316  012777  000001  000016G         MOV     #1,@IOP.TABLE+16        ;                               4216
000324  005016                          CLR     (SP)                    ;                               4218
000326  004737  000000G                 JSR     PC,CHK.RIXI.STATUS
000332  012716  100220                  MOV     #-77560,(SP)            ;                               4219
000336  011646                          MOV     (SP),-(SP)
000340  004737  000000G                 JSR     PC,CHK.CSR.STATUS
000344  012716  140000                  MOV     #-40000,(SP)            ;                               4220
000350  012746  000400                  MOV     #400,-(SP)
000354  004737  000000G                 JSR     PC,CHK.XMIT.STATUS
000360  012716  140000                  MOV     #-40000,(SP)            ;                               4221
000364  012746  000001                  MOV     #1,-(SP)
000370  004737  000000G                 JSR     PC,CHK.RCV.STATUS
000374  005077  000016G                 CLR     @IOP.TABLE+16           ;                               4223
000400  062706  000022                  ADD     #22,SP                  ;                               4189
000404  104467                          TRAP    67                      ;                               4223
000406  006000                          ROR     R0
000410  103660                          BLO     5$
000412  004737  000000G                 JSR     PC,RESET.DEQNA          ;                               4226
000416  012746  000200                  MOV     #200,-(SP)              ;                               4228
```

```
000422  004737  000000G                        JSR     PC,TURN.OFF.LED
000426  012716  000204                        MOV     #204,(SP)               ;                       4229
000432  004737  000000G                        JSR     PC,TURN.OFF.LED
000436  012716  000210                        MOV     #210,(SP)               ;                       4230
000442  004737  000000G                        JSR     PC,TURN.OFF.LED
000446  012716  000214                        MOV     #214,(SP)               ;                       4231
000452  004737  000000G                        JSR     PC,TURN.OFF.LED
000456  005726                                TST     (SP)+                   ;                       4113
000460  012601                                MOV     (SP)+,R1
000462  000207                                RTS     PC

; Routine Size:  154 words,     Routine Base:  AB$CODE$ + 21046
; Maximum stack depth per invocation:  11 words




                                            .SBTTL  T20 TEST 20 - FIFO OVERFLOW TEST
000000  004737  021046'                T20::
000000                                  1$:     JSR     PC,$T20                 ;                       4231
000004  104466                                TRAP    66
000006  006000                                ROR     R0
000010  103773                                BLO     1$
000012  000207                                RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 21532
; Maximum stack depth per invocation:  2 words


;   4234  1
```

```
;  4235  1     %SBTTL 'TEST 21 - SANITY TIMER TEST'
;  4236  1     !++
;  4237  1     !
;  4238  1     !  TEST 21:     SANITY TIMER TEST
;  4239  1     !
;  4240  1     !  DESCRIPTION:
;  4241  1     !
;  4242  1     !       This test verifies that the Sanity Timer times out after a pre-set
;  4243  1     !       ( supplied by the operator ) timeout period. The Sanity Timer uses
;  4244  1     !       DCOK line on the Q-Bus to force the power_fail interrupt of the
;  4245  1     !        processor which in turn causes the processor to reboot itself.
;  4246  1     !
;  4247  1     !       Hardware tested:  Sanity Timer Logic
;  4248  1     !
;  4249  1     !       Processing:
;  4250  1     !
;  4251  1     !            BEGIN
;  4252  1     !               reset device
;  4253  1     !               store Console Terminal and Power_fail interrupt vectors
;  4254  1     !                 ( location 24 and 60 octal )
;  4255  1     !               enable Console Terminal interrupt
;  4256  1     !               arm for Power_fail interrupt
;  4257  1     !               inform the operator about the test procedure
;  4258  1     !               set the Sanity Timer to timeout value supplied by the
;  4259  1     !                 operator
;  4260  1     !               enable the Sanity Timer
;  4261  1     !               wait
;  4262  1     !               IF Power-fail interrupt occured
;  4263  1     !               THEN
;  4264  1     !                  print 'SANITY TIMER TIMED OUT AS EXPECTED'
;  4265  1     !               ELSE
;  4266  1     !                  force Console Terminal input interrupt by typing "Q"
;  4267  1     !                  print error message if not inhibited
;  4268  1     !               ENDIF
;  4269  1     !               disable Sanity Timer
;  4270  1     !               restore Console Terminal and Power_fail interrupt vectors
;  4271  1     !                 ( location 24 and 60 octal )
;  4272  1     !            END
;  4273  1     !--
```

```
;  4274  3     BGNTST;
;  4275  3
;  4276  3     IF .SWP_TIMER
;  4277  3       THEN
;  4278  4         BEGIN
;  4279  4
;  4280  4           !++
;  4281  4           !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM
;  4282  4           !--
;  4283  4
;  4284  4           RESET_DEQNA ( );
;  4285  4           !++
;  4286  4           !  SETUP FOR POWER FAIL AND CONSOLE TERMINAL INTERRUPTS
;  4287  4           !--
;  4288  4
;  4289  4           SETVEC ( PF_VEC_LOC, PWR_INT, PRI07 );              ! POWER FAIL
;  4290  4           SETVEC ( KB_VEC_LOC, KBD INT, PRI05 );              ! CONSOLE TERMINAL
;  4291  4           SETPRI ( PRI00 );                                   ! SET PROCESSOR PRI LEVEL
;  4292  4           PREP_FOR_SETUP ( );
;  4293  4           INCR INDEX1 FROM 1 TO 14 DO
;  4294  4             WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  4295  4
;  4296  6           BGNSUB;
;  4297  6             PUT_BIT [ CSR, SE, EENABLE ];
;  4298  6             XMIT_SETUP_PACKET ( %O'200' + ( .SWP_TOUT_VAL + 4 ) );
;  4299  6
;  4300  6             SELECTONE .SWP_TOUT_VAL OF
;  4301  6               SET
;  4302  6               [ 0,1 ]:
;  4303  7                       BEGIN
;  4304  7                         TEMP1 = 1;
;  4305  7                         PRINTB ( MSG32, .TEMP1 );
;  4306  6                       END;
;  4307  6               [ 2 ]:
;  4308  7                       BEGIN
;  4309  7                         TEMP1 = 4;
;  4310  7                         PRINTB ( MSG32, .TEMP1 );
;  4311  6                       END;
;  4312  6               [ 3 ]:
;  4313  7                       BEGIN
;  4314  7                         TEMP1 = 16;
;  4315  7                         PRINTB ( MSG32, .TEMP1 );
;  4316  6                       END;
;  4317  6               [ 4 ]:
;  4318  7                       BEGIN
;  4319  7                         TEMP1 = 1;
;  4320  7                         PRINTB ( MSG55, .TEMP1 );
;  4321  6                       END;
;  4322  6               [ 5 ]:
;  4323  7                       BEGIN
;  4324  7                         TEMP1 = 4;
;  4325  7                         PRINTB ( MSG55, .TEMP1 );
;  4326  6                       END;
```

```
;  4327  6              [ 6 ]:
;  4328  7                      BEGIN
;  4329  7                        TEMP1 = 16;
;  4330  7                        PRINTB ( MSG55, .TEMP1 );
;  4331  6                      END;
;  4332  6              [ 7 ]:
;  4333  7                      BEGIN
;  4334  7                        TEMP1 = 1;
;  4335  7                        PRINTB ( MSG56, .TEMP1 );
;  4336  6                      END;
;  4337  6              TES;
;  4338  6
;  4339  6            PRINTB ( MSG57 );
;  4340  6            INTERRUPT_FLG = -1;
;  4341  6            WAIT_FOR_TIMEOUT ( );
;  4342  6
;  4343  6            !++
;  4344  6            !  PUT DEQNA IN NORMAL MODE AND CHECK STATUS
;  4345  6            !--
;  4346  6
;  4347  6            PUT_BIT [ CSR, SE, DISABLE ];
;  4348  6            PREP_FOR_SETUP ( );
;  4349  6            INCR INDEX1 FROM 1 TO 14 DO
;  4350  6              WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  4351  6
;  4352  8            BGNSEG;
;  4353  8              XMIT_SETUP_PACKET ( N_MODE );
;  4354  6            ENDSEG;
;  4355  6
;  4356  6            CLRVEC ( PF_VEC_LOC );
;  4357  6            CLRVEC ( KB_VEC_LOC );
;  4358  6
;  4359  6            IF .INTERRUPT_FLG
;  4360  6              THEN
;  4361  7                BEGIN
;  4362  8                  PRINTB ( MSG33 )
;  4363  7                END
;  4364  6              ELSE
;  4365  7                BEGIN
;  4366  7                  CSR_WORD = GET_BIT ( CSR_ALL );
;  4367  7                  PRINTB ( MSG59 );
;  4368  7                  PRINTB ( MSG34 );
;  4369  7                  ERRDF  ( 2101, MSG00, ERROR$REPORT );
;  4370  6                END;
;  4371  4          ENDSUB;
;  4372  3        END;
;  4373  3
;  4374  1    ENDTST;


                                        .SBTTL  $T21 TEST 21 - SANITY TIMER TEST
000000  010146                  $T21:   MOV     R1,-(SP)                        ;                       4233
000002  005746                          TST      (SP)
```

```
000004  032737  000001  000000G                 BIT     #1,SWP.TIMER                    ;                               4276
000012  001002                                  BNE     1$
000014  000137  022546'                         JMP     17$
000020  004737  000000G                 1$:     JSR     PC,RESET.DEQNA                  ;                               4284
000024  012746  000000G                         MOV     #PRI07,-(SP)                    ;                               4289
000030  012746  000000G                         MOV     #PWR.INT,-(SP)
000034  012746  000024                          MOV     #24, (SP)
000040  012746  000003                          MOV     #3,-(SP)
000044  104437                                  TRAP    37
000046  012716  000000G                         MOV     #PRI05,(SP)                     ;                               4290
000052  012746  000000G                         MOV     #KBD.INT,-(SP)
000056  012746  000060                          MOV     #60,-(SP)
000062  012746  000003                          MOV     #3,-(SP)
000066  104437                                  TRAP    37
000070  012700  000000G                         MOV     #PRI00,R0                       ;                               4291
000074  104441                                  TRAP    41
000076  004737  000000G                         JSR     PC,PREP.FOR.SETUP               ;                               4292
000102  012701  000001                          MOV     #1,R1                           ; *,INDEX1                      4293
000106  010116                          2$:     MOV     R1,(SP)                         ; INDEX1,*                      4294
000110  012746  000023                          MOV     #23,-(SP)
000114  004737  000000G                         JSR     PC,WRT.STATION.ADR
000120  005726                                  TST     (SP)+
000122  005201                                  INC     R1                              ; INDEX1                        4293
000124  020127  000016                          CMP     R1,#16                          ; INDEX1,*
000130  003766                                  BLE     2$
000132  104402                          3$:     TRAP    2                               ;                               4294
000134  013700  000000G                         MOV     REG.ADR,R0                      ;                               4297
000140  052760  002000  000016                  BIS     #2000,16(R0)
000146  013700  000000G                         MOV     SWP.TOUT.VAL,R0                 ;                               4298
000152  072027  000004                          ASH     #4,R0
000156  010016                                  MOV     R0,(SP)
000160  062716  000200                          ADD     #200,(SP)
000164  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000170  013701  000000G                         MOV     SWP.TOUT.VAL,R1                 ;                               4300
000174  002417                                  BLT     4$                              ;                               4302
000176  020127  000001                          CMP     R1,#1
000202  003014                                  BGT     4$
000204  012737  000001  000000G                 MOV     #1,TEMP1                        ;                               4304
000212  012716  000001                          MOV     #1,(SP)                         ;                               4305
000216  012746  000000G                         MOV     #MSG32,-(SP)
000222  012746  000002                          MOV     #2,-(SP)
000226  010600                                  MOV     SP,R0                           ; SP,*
000230  104414                                  TRAP    14
000232  000531                                  BR      10$                             ;                               4303
000234  020127  000002                  4$:     CMP     R1,#2                           ;                               4307
000240  001014                                  BNE     5$
000242  012737  000004  000000G                 MOV     #4,TEMP1                        ;                               4309
000250  012716  000004                          MOV     #4,(SP)                         ;                               4310
000254  012746  000000G                         MOV     #MSG32,-(SP)
000260  012746  000002                          MOV     #2,-(SP)
000264  010600                                  MOV     SP,R0                           ; SP,*
000266  104414                                  TRAP    14
000270  000512                                  BR      10$                             ;                               4308
```

```
000272  020127  000003                  5$:     CMP     R1,#3                   ;                               4312
000276  001014                                  BNE     6$
000300  012737  000020  000000G                MOV     #20,TEMP1               ;                               4314
000306  012716  000020                          MOV     #20,(SP)                ;                               4315
000312  012746  000000G                         MOV     #MSG32,-(SP)
000316  012746  000002                          MOV     #2,-(SP)
000322  010600                                  MOV     SP,R0                   ; SP,*
000324  104414                                  TRAP    14
000326  000473                                  BR      10$                     ;                               4313
000330  020127  000004                  6$:     CMP     R1,#4                   ;                               4317
000334  001014                                  BNE     7$
000336  012737  000001  000000G                MOV     #1,TEMP1                ;                               4319
000344  012716  000001                          MOV     #1,(SP)                 ;                               4320
000350  012746  000000G                         MOV     #MSG55,-(SP)
000354  012746  000002                          MOV     #2,-(SP)
000360  010600                                  MOV     SP,R0                   ; SP,*
000362  104414                                  TRAP    14
000364  000454                                  BR      10$                     ;                               4318
000366  020127  000005                  7$:     CMP     R1,#5                   ;                               4322
000372  001014                                  BNE     8$
000374  012737  000004  000000G                MOV     #4,TEMP1                ;                               4324
000402  012716  000004                          MOV     #4,(SP)                 ;                               4325
000406  012746  000000G                         MOV     #MSG55,-(SP)
000412  012746  000002                          MOV     #2,-(SP)
000416  010600                                  MOV     SP,R0                   ; SP,*
000420  104414                                  TRAP    14
000422  000435                                  BR      10$                     ;                               4323
000424  020127  000006                  8$:     CMP     R1,#6                   ;                               4327
000430  001014                                  BNE     9$
000432  012737  000020  000000G                MOV     #20,TEMP1               ;                               4329
000440  012716  000020                          MOV     #20,(SP)                ;                               4330
000444  012746  000000G                         MOV     #MSG55,-(SP)
000450  012746  000002                          MOV     #2,-(SP)
000454  010600                                  MOV     SP,R0                   ; SP,*
000456  104414                                  TRAP    14
000460  000416                                  BR      10$                     ;                               4328
000462  020127  000007                  9$:     CMP     R1,#7                   ;                               4332
000466  001014                                  BNE     11$
000470  012737  000001  000000G                MOV     #1,TEMP1                ;                               4334
000476  012716  000001                          MOV     #1,(SP)                 ;                               4335
000502  012746  000000G                         MOV     #MSG56,-(SP)
000506  012746  000002                          MOV     #2,-(SP)
000512  010600                                  MOV     SP,R0                   ; SP,*
000514  104414                                  TRAP    14
000516  022626                          10$:    CMP     (SP)+,(SP)+             ;                               4333
000520  012716  000000G                 11$:    MOV     #MSG57,(SP)             ;                               4339
000524  012746  000001                          MOV     #1,-(SP)
000530  010600                                  MOV     SP,R0                   ; SP,*
000532  104414                                  TRAP    14
000534  012737  177777  000000G                MOV     #-1,INTERRUPT.FLG       ;                               4340
000542  004737  000000G                         JSR     PC,WAIT.FOR.TIMEOUT     ;                               4341
000546  013700  000000G                         MOV     REG.ADR,R0              ;                               4347
000552  042760  002000  000016                 BIC     #2000,16(R0)
```

```
000560  004737  000000G                        JSR     PC,PREP.FOR.SETUP               ;                               4348
000564  012701  000001                         MOV     #1,R1                           ; *,INDEX1                      4349
000570  010116                         12$:    MOV     R1,(SP)                         ; INDEX1,*                      4350
000572  012746  000023                         MOV     #23,-(SP)
000576  004737  000000G                        JSR     PC,WRT.STATION.ADR
000602  005726                                 TST     (SP)+
000604  005201                                 INC     R1                              ; INDEX1                        4349
000606  020127  000016                         CMP     R1,#16                          ; INDEX1,*
000612  003766                                 BLE     12$
000614  104404                         13$:    TRAP    4                               ;                               4350
000616  012716  000200                         MOV     #200,(SP)                       ;                               4353
000622  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000626  104470                                 TRAP    70
000630  006000                                 ROR     R0
000632  103770                                 BLO     13$
000634  012700  000024                         MOV     #24,R0                          ;                               4356
000640  104436                                 TRAP    36
000642  012700  000060                         MOV     #60,R0                          ;                               4357
000646  104436                                 TRAP    36
000650  032737  000001  000000G                BIT     #1,INTERRUPT.FLG                ;                               4359
000656  001407                                 BEQ     14$
000660  012716  000000G                        MOV     #MSG33,(SP)                     ;                               4362
000664  012746  000001                         MOV     #1,-(SP)
000670  010600                                 MOV     SP,R0                           ; SP,*
000672  104414                                 TRAP    14
000674  000431                                 BR      15$                             ;                               4359
000676  013700  000000G                14$:    MOV     REG.ADR,R0                      ;                               4366
000702  016066  000016  000020                 MOV     16(R0),20(SP)                   ; *,TMP.LOCATION
000710  016637  000020  000000G                MOV     20(SP),CSR.WORD                 ; TMP.LOCATION,*
000716  012716  000000G                        MOV     #MSG59,(SP)                     ;                               4367
000722  012746  000001                         MOV     #1,-(SP)
000726  010600                                 MOV     SP,R0                           ; SP,*
000730  104414                                 TRAP    14
000732  012716  000000G                        MOV     #MSG34,(SP)                     ;                               4368
000736  012746  000001                         MOV     #1,-(SP)
000742  010600                                 MOV     SP,R0                           ; SP,*
000744  104414                                 TRAP    14
000746  104455                                 TRAP    55                              ;                               4369
000750  004065                                 .WORD   4065
000752  000000G                                .WORD   MSG00
000754  000000G                                .WORD   ERROR$REPORT
000756  005726                                 TST     (SP)+                           ;                               4365
000760  022626                         15$:    CMP     (SP)+,(SP)+                     ;                               4294
000762  104467                                 TRAP    67                              ;                               4370
000764  006000                                 ROR     R0
000766  103002                                 BHIS    16$
000770  000137  021700'                        JMP     3$
000774  062706  000016                 16$:    ADD     #16,SP                          ;                               4278
001000  005726                         17$:    TST     (SP)+                           ;                               4233
001002  012601                                 MOV     (SP)+,R1
001004  000207                                 RTS     PC

; Routine Size:  259 words,    Routine Base:  AB$CODE$ + 21546
```

```
; Maximum stack depth per invocation:  14 words


                                        .SBTTL  T21 TEST 21 - SANITY TIMER TEST
000000  004737  021546'          T21::
000000                           1$:    JSR     PC,$T21                  ;                    4372
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,      Routine Base:  AB$CODE$ + 22554
; Maximum stack depth per invocation:  2 words


;   4375  1
;   4376  1     END
;   4377  0     ELUDOM


;                                   OTS external references
                                            .GLOBL  $SAVE4, $SAVE3, $SAVE2


;                                   PSECT SUMMARY
;
;       Psect Name                  Words     Attributes
;        AB$CODE$                    4796       RO ,  I  ,  LCL,  REL,  CON



;                               Library Statistics
;
;                                        ------ - Symbols -------      Pages       Processing
;       File                             Total    Loaded   Percent     Mapped      Time
;
;  DISK$USER2:[MARSHALL.DEQNA]QNALIB.L16;15
;                                         223       142        63         14        00:00.1



;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZQNA3.BLI/LIST=ZQNA3.LIS/OBJECT=ZQNA3.OBJ/SOURCE=PAGE:53
```

```
; Size:              4796 code + 0 data words
; Run Time:            01:59.9
; Elapsed Time:        07:33.3
; Lines/CPU Min:        2190
; Lexemes/CPU-Min:     26074
; Memory Used:         437 pages
; Compilation Complete
```

```
;    0001  0     MODULE ZQNA4 (%TITLE 'CZQNADO DEQNA FUNCTIONAL TEST'
;    0002  0                   IDENT = 'V01.0',
;    0003  0                   ADDRESSING_MODE(ABSOLUTE)
;    0004  0                   ) =
;    0005  0     %SBTTL 'GLOBAL ROUTINE DECLARATION MODULE'
;    0006  0
;    0007  1     BEGIN
;    0008  1
;    0009  1     LIBRARY 'QNALIB';                       ! QNALIB LIBRARY
;    0010  1     REQUIRE 'BLSMAC.REQ';                   ! DIAGNOSTIC SUPERVISOR LIBRARY
;    1500  1     !<BLF/NOFORMAT>
;    1501  1
```

```
;    1502  1      PSECT
;    1503  1          CODE = AC$CODE$;
;    1504  1
;    1505  1      FORWARD ROUTINE
;    1506  1          XMIT_AND_RCV_PACKET          : NOVALUE;
;    1507  1
;    1508  1      !++
;    1509  1      !       EXTERNAL DATA USED BY THIS MODULE
;    1510  1      !--
;    1511  1
;    1512  1      EXTERNAL
;    1513  1
;    1514  1      !++
;    1515  1      !       COMMUNICATION AREA DECLARATIONS
;    1516  1      !--
;    1517  1
;    1518  1          RCV_D_LIST       : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1519  1          XMIT_D_LIST      : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1520  1          DESCR_LIST       : BLOCK  [ DESCR_SIZE, WORD] FIELD ( DL_FIELDS ),
;    1521  1          RCV_BUFFER       : VECTOR [ B_SIZE, BYTE ],
;    1522  1          XMIT_BUFFER      : VECTOR [ B_SIZE, BYTE ],
;    1523  1          DATA_BUFFER      : VECTOR [ BUF_SIZE, BYTE ],
;    1524  1          SETUP_BUFFER     : VECTOR [ SETUB_SIZE, WORD ],
;    1525  1          IOP_TABLE        : VECTOR [ 8, WORD ],
;    1526  1          BD_PROM_DESCR    : VECTOR [ BD_D_SIZE, WORD ],
;    1527  1          STATION_ADR      : VECTOR [ 4, WORD ],
;    1528  1          TARGET_ADR       : VECTOR [ T_SIZE, BYTE ],
;    1529  1          PHYS_ADR         : VECTOR [ 22, BYTE ],
;    1530  1
;    1531  1      !++
;    1532  1      !       HARDWARE  AND SOFTWARE P-TABLE STORAGE DECLARATIONS
;    1533  1      !--
;    1534  1
;    1535  1          HWP_TABLE    : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;    1536  1          SWP_TABLE    : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    1537  1
;    1538  1          REG_ADR      : REF REG_STR FIELD ( IOP_FIELDS ),
;    1539  1          GET_ADR      : REF ADR_STR FIELD ( IOP_FIELDS ),
;    1540  1          IOP_DATA     : REF REG_STR FIELD ( IOP_FIELDS ),
;    1541  1
```

```
;    1542  1
;    1543  1      !++
;    1544  1      !       MISCELLANEOUS DATA DECLARATIONS
;    1545  1      !--
;    1546  1
;    1547  1          XBUF_LENGTH,        RBUF_LENGTH,    INTERRUPT_FLG,      COUNTER,
;    1548  1          SWP_BLOCK_MEM,      SWP_TOUT_VAL,   SWP_ILOOP,          SWP_TIMER,
;    1549  1          UP_COUNTER,         DOWN_COUNTER,   CHECKSUM,           ERR_NUMBER,
;    1550  1          ERR_COUNT,          ERR_FLAG,       CSR_WORD,           PRI00,
;    1551  1          PRI01,              PRI02,          PRI03,              PRI04,
;    1552  1          PRI05,              PRI06,          PRI07,              DEQNA_NO  : WORD,
;    1553  1
;    1554  1      !++
;    1555  1      !   TEMPORARY STORAGE DATA DECLARATIONS
;    1556  1      !--
;    1557  1
;    1558  1          P1,                 P2,             P3,                 P4,
;    1559  1          TMP_IOP_ADR,        TMP_REG_DATA,   TEMP1,              TEMP2,
;    1560  1          TEMP3,              TEMP4,          TEMP5,              TEMP6,
;    1561  1          TEMP7,              TEMP8,          TEMP9,              TADR1,
;    1562  1          TADR2                                                           : WORD,
;    1563  1          TBYTE1,             TBYTE2,         TBYTE3,             TBYTE4    : BYTE,
;    1564  1
;    1565  1      !++
;    1566  1      !       DIAGNOSTIC ERROR MESSAGES DECLARED EXTERNALLY
;    1567  1      !--
;    1568  1
;    1569  1          MSG00,
;    1570  1          MSG01, MSG02, MSG03, MSG04, MSG05, MSG06, MSG07, MSG08, MSG09, MSG10,
;    1571  1          MSG11, MSG12, MSG13, MSG14, MSG15, MSG16, MSG17, MSG18, MSG19, MSG20,
;    1572  1          MSG21, MSG22, MSG23, MSG24, MSG25, MSG26, MSG27, MSG28, MSG29, MSG30,
;    1573  1          MSG31, MSG32, MSG33, MSG34, MSG35, MSG36, MSG37, MSG38, MSG39, MSG40,
;    1574  1          MSG41, MSG42, MSG43, MSG44, MSG45, MSG46, MSG47, MSG48, MSG49, MSG50,
;    1575  1          MSG51, MSG52, MSG53, MSG54, MSG55, MSG56, MSG57, MSG58, MSG59, MSG60,
;    1576  1          MSG61, MSG62, MSG63, MSG64, MSG65, MSG66, MSG67, MSG68, MSG69, MSG70;
;    1577  1
```

```
;    1578  1     %SBTTL 'GLOBAL ROUTINE - ERROR$REPORT ( )'
;    1579  1
;    1580  1     !++
;    1581  1     !
;    1582  1     !  GLOBAL ROUTINE :     ERROR$REPORT
;    1583  1     !
;    1584  1     !  DESCRIPTION:
;    1585  1     !
;    1586  1     !       This routine reports errors to the operator
;    1587  1     !
;    1588  1     !--
;    1589  1
;    1590  1     %SBTTL 'GLOBAL ROUTINE - ERROR$REPORT ( )'
;    1591  1
;    1592  1     BGNMSG (ERROR$REPORT);
```

```
                                        .TITLE  ZQNA4 CZQNADO DEQNA FUNCTIONAL TEST
                                        .IDENT  /V01.0/
                                        .ENABL  AMA

                                        .GLOBL  RCV.D.LIST, XMIT.D.LIST, DESCR.LIST
                                        .GLOBL  RCV.BUFFER, XMIT.BUFFER, DATA.BUFFER
                                        .GLOBL  SETUP.BUFFER, IOP.TABLE, BD.PROM.DESCR
                                        .GLOBL  STATION.ADR, TARGET.ADR, PHYS.ADR
                                        .GLOBL  HWP.TABLE, SWP.TABLE, REG.ADR
                                        .GLOBL  GET.ADR, IOP.DATA, XBUF.LENGTH
                                        .GLOBL  RBUF.LENGTH, INTERRUPT.FLG, COUNTER
                                        .GLOBL  SWP.BLOCK.MEM, SWP.TOUT.VAL, SWP.ILOOP
                                        .GLOBL  SWP.TIMER, UP.COUNTER, DOWN.COUNTER
                                        .GLOBL  CHECKSUM, ERR.NUMBER, ERR.COUNT
                                        .GLOBL  ERR.FLAG, CSR.WORD, PRI00, PRI01
                                        .GLOBL  PRI02, PRI03, PRI04, PRI05, PRI06
                                        .GLOBL  PRI07, DEQNA.NO, P1, P2, P3, P4
                                        .GLOBL  TMP.IOP.ADR, TMP.REG.DATA, TEMP1
                                        .GLOBL  TEMP2, TEMP3, TEMP4, TEMP5, TEMP6
                                        .GLOBL  TEMP7, TEMP8, TEMP9, TADR1, TADR2
                                        .GLOBL  TBYTE1, TBYTE2, TBYTE3, TBYTE4
                                        .GLOBL  MSG00, MSG01, MSG02, MSG03, MSG04
                                        .GLOBL  MSG05, MSG06, MSG07, MSG08, MSG09
                                        .GLOBL  MSG10, MSG11, MSG12, MSG13, MSG14
                                        .GLOBL  MSG15, MSG16, MSG17, MSG18, MSG19
                                        .GLOBL  MSG20, MSG21, MSG22, MSG23, MSG24
                                        .GLOBL  MSG25, MSG26, MSG27, MSG28, MSG29
                                        .GLOBL  MSG30, MSG31, MSG32, MSG33, MSG34
                                        .GLOBL  MSG35, MSG36, MSG37, MSG38, MSG39
                                        .GLOBL  MSG40, MSG41, MSG42, MSG43, MSG44
                                        .GLOBL  MSG45, MSG46, MSG47, MSG48, MSG49
                                        .GLOBL  MSG50, MSG51, MSG52, MSG53, MSG54
                                        .GLOBL  MSG55, MSG56, MSG57, MSG58, MSG59
                                        .GLOBL  MSG60, MSG61, MSG62, MSG63, MSG64
                                        .GLOBL  MSG65, MSG66, MSG67, MSG68, MSG69
                                        .GLOBL  MSG70
```

```
                                        .SBTTL  ERROR$REPORT GLOBAL ROUTINE - ERROR$REPORT ( )
000000                                  .PSECT  AC$CODE$,  RO

000000  004737  000000V          ERROR$REPORT::
                                        JSR     PC,M$ERROR$REPORT               ;                                       1592
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  AC$CODE$ + 0000
; Maximum stack depth per invocation:  2 words


;    1593  2
;    1594  2     PRINTB ( MSG03 );
;    1595  2     PRINTB ( MSG04, .XMIT_D_LIST [ FLGWD ], .RCV_D_LIST [ FLGWD ] );
;    1596  2     PRINTB ( MSG05, .XMIT_D_LIST [ DBITS ], .RCV_D_LIST [ DBITS ] );
;    1597  2     PRINTB ( MSG06, .XMIT_D_LIST [ LOADR ], .RCV_D_LIST [ LOADR ] );
;    1598  2     PRINTB ( MSG07, .XMIT_D_LIST [ TWDL  ], .RCV_D_LIST [ TWDL  ] );
;    1599  2     PRINTB ( MSG08, .XMIT_D_LIST [ STWD1 ] AND XWD1_MASK, .RCV_D_LIST [ STWD1 ] AND RWD2_MASK );
;    1600  2     PRINTB ( MSG09, .XMIT_D_LIST [ STWD2 ] AND XWD2_MASK, .RCV_D_LIST [ STWD2 ] AND RLL_MASK );
;    1601  2     PRINTB ( MSG10, .CSR_WORD AND %O'133777' );
;    1602  2     PRINTB ( MSG11, .HWP_TABLE [ ADDR ] );
;    1603  2
;    1604  1     ENDMSG;


                                        .SBTTL  M$ERROR$REPORT GLOBAL ROUTINE - ERROR$REPORT ( )
000000  012746  000000G        M$ERROR$REPORT:
                                        MOV     #MSG03,-(SP)                    ;                                       1594
000004  012746  000001                  MOV     #1,-(SP)
000010  010600                          MOV     SP,R0                           ; SP,*
000012  104414                          TRAP    14
000014  013716  000000G                 MOV     RCV.D.LIST,(SP)                 ;                                       1595
000020  013746  000000G                 MOV     XMIT.D.LIST,-(SP)
000024  012746  000000G                 MOV     #MSG04,-(SP)
000030  012746  000003                  MOV     #3,-(SP)
000034  010600                          MOV     SP,R0                           ; SP,*
000036  104414                          TRAP    14
000040  013716  000002G                 MOV     RCV.D.LIST+2,(SP)               ;                                       1596
000044  013746  000002G                 MOV     XMIT.D.LIST+2,-(SP)
000050  012746  000000G                 MOV     #MSG05,-(SP)
000054  012746  000003                  MOV     #3,-(SP)
000060  010600                          MOV     SP,R0                           ; SP,*
000062  104414                          TRAP    14
000064  013716  000004G                 MOV     RCV.D.LIST+4,(SP)               ;                                       1597
000070  013746  000004G                 MOV     XMIT.D.LIST+4,-(SP)
000074  012746  000000G                 MOV     #MSG06,-(SP)
000100  012746  000003                  MOV     #3,-(SP)
000104  010600                          MOV     SP,R0                           ; SP,*
000106  104414                          TRAP    14
000110  013716  000006G                 MOV     RCV.D.LIST+6,(SP)               ;                                       1598
```

```
000114  013746  000006G                        MOV     XMIT.D.LIST+6,-(SP)
000120  012746  000000G                        MOV     #MSG07,-(SP)
000124  012746  000003                         MOV     #3,-(SP)
000130  010600                                 MOV     SP,R0                           ; SP,*
000132  104414                                 TRAP    14
000134  013716  000010G                        MOV     RCV.D.LIST+10,(SP)              ;                                       1599
000140  042716  000360                         BIC     #360,(SP)
000144  013746  000010G                        MOV     XMIT.D.LIST+10,-(SP)
000150  042716  020017                         BIC     #20017,(SP)
000154  012746  000000G                        MOV     #MSG08,-(SP)
000160  012746  000003                         MOV     #3,-(SP)
000164  010600                                 MOV     SP,R0                           ; SP,*
000166  104414                                 TRAP    14
000170  005016                                 CLR     (SP)                            ;                                       1600
000172  113716  000012G                        MOVB    RCV.D.LIST+12,(SP)
000176  013746  000012G                        MOV     XMIT.D.LIST+12,-(SP)
000202  042716  140000                         BIC     #140000,(SP)
000206  012746  000000G                        MOV     #MSG09,-(SP)
000212  012746  000003                         MOV     #3,-(SP)
000216  010600                                 MOV     SP,R0                           ; SP,*
000220  104414                                 TRAP    14
000222  013716  000000G                        MOV     CSR.WORD,(SP)                   ;                                       1601
000226  042716  044000                         BIC     #44000,(SP)
000232  012746  000000G                        MOV     #MSG10,-(SP)
000236  012746  000002                         MOV     #2,-(SP)
000242  010600                                 MOV     SP,R0                           ; SP,*
000244  104414                                 TRAP    14
000246  017716  000000G                        MOV     @HWP.TABLE,(SP)                 ;                                       1602
000252  012746  000000G                        MOV     #MSG11,-(SP)
000256  012746  000002                         MOV     #2,-(SP)
000262  010600                                 MOV     SP,R0                           ; SP,*
000264  104414                                 TRAP    14
000266  062706  000060                         ADD     #60,SP                          ;                                       1592
000272  000207                                 RTS     PC

; Routine Size:  94 words,      Routine Base:  AC$CODE$ + 0010
; Maximum stack depth per invocation:  26 words


;   1605  1
;   1606  1
```

```
;    1607  1     %SBTTL 'GLOBAL ROUTINE - E1$REPORT ( )'
;    1608  1
;    1609  1     !++
;    1610  1     !
;    1611  1     !  GLOBAL ROUTINE :     E1$REPORT
;    1612  1     !
;    1613  1     !  DESCRIPTION:
;    1614  1     !
;    1615  1     !       This routine reports errors to the operator
;    1616  1     !
;    1617  1     !--
;    1618  1
;    1619  1     %SBTTL 'GLOBAL ROUTINE - E1$REPORT ( )'
;    1620  1
;    1621  1     BGNMSG ( E1$REPORT );


                                            .SBTTL  E1$REPORT GLOBAL ROUTINE - E1$REPORT ( )
000000  004737  000000V                E1$REPORT::
                                            JSR     PC,M$E1$REPORT                  ;                       1621
000004  104423                             TRAP    23
000006  000207                             RTS     PC

; Routine Size:  4 words,       Routine Base:  AC$CODE$ + 0304
; Maximum stack depth per invocation:  2 words


;    1622  2
;    1623  2       TEMP1 = 1;
;    1624  2
;    1625  1     ENDMSG;


                                            .SBTTL  M$E1$REPORT GLOBAL ROUTINE - E1$REPORT ( )
000000  012737  000001  000000G       M$E1$REPORT:
                                            MOV     #1,TEMP1                        ;                       1623
000006  000207                             RTS     PC                              ;                       1621

; Routine Size:  4 words,       Routine Base:  AC$CODE$ + 0314
; Maximum stack depth per invocation:  0 words


;    1626  1
;    1627  1
```

```
;    1628  1     %SBTTL 'GLOBAL ROUTINE - RESET_DEQNA ( )'
;    1629  1
;    1630  1     GLOBAL ROUTINE RESET_DEQNA : NOVALUE =
;    1631  1
;    1632  1     !++
;    1633  1     !
;    1634  1     !  GLOBAL ROUTINE :     RESET_DEQNA
;    1635  1     !
;    1636  1     !  DESCRIPTION:
;    1637  1     !
;    1638  1     !       This routine verifies that DEQNA can be reset by setting bit 1 in the
;    1639  1     !       CSR register. After the reset, CSR is checked for nominal
;    1640  1     !       status.
;    1641  1     !
;    1642  1     !       Hardware tested:          Q-Bus DMA Interface
;    1643  1     !
;    1644  1     !       Processing:
;    1645  1     !
;    1646  1     !            BEGIN
;    1647  1     !               set Software Reset (SR) bit in CSR and check for
;    1648  1     !                   expected CSR status
;    1649  1     !               IF error
;    1650  1     !               THEN
;    1651  1     !                  print error message if not inhibited
;    1652  1     !               ENDIF
;    1653  1     !               clear SR bit in CSR and check for expected CSR status
;    1654  1     !               IF error
;    1655  1     !               THEN
;    1656  1     !                  print error message if not inhibited
;    1657  1     !               ENDIF
;    1658  1     !            END
;    1659  1     !
;    1660  1     !  INPUT PARAMETERS:
;    1661  1     !
;    1662  1     !--
```

```
;    1663  1
;    1664  1      !++
;    1665  1      !
;    1666  1      !  RESET THE DEVICE AND CHECK CONTENTS OF CSR FOR NOMINAL STATUS
;    1667  1      !
;    1668  1      !--
;    1669  1
;    1670  2      BEGIN
;    1671  2
;    1672  2      PUT_BIT ( CSR, ALL_BITS, ZERO );
;    1673  2      PUT_BIT ( CSR, SR, SET_IT );
;    1674  2
;    1675  2      DELAY ( TIME6_LIMIT );
;    1676  2      TEMP1 = GET_BIT [ CSR_ALL ] AND CSR2_MASK;
;    1677  2
;    1678  2      IF .TEMP1 NEQU CSR1_STATUS
;    1679  2        THEN
;    1680  3          BEGIN
;    1681  3            ERR_FLAG = ONE;
;    1682  3            CSR_WORD = GET_BIT [ CSR_ALL ];
;    1683  3            PRINTB ( MSG59 );
;    1684  3            PRINTB ( MSG31 );
;    1685  3            PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP1, CSR2_STATUS );
;    1686  3            ERRDF ( 0001, MSG00, E1$REPORT );
;    1687  2          END;
;    1688  2
;    1689  2      !++
;    1690  2      !
;    1691  2      !  CLEAR SOFTWARE RESET BIT IN THE CSR AND CHECK FOR EXPECTED STATUS
;    1692  2      !
;    1693  2      !--
;    1694  2
;    1695  2      PUT_BIT ( CSR, SR, CLR_IT );
;    1696  2      DELAY ( TIME6_LIMIT );
;    1697  2      TEMP2 = GET_BIT [ CSR_ALL ] AND CSR2_MASK;
;    1698  2      IF .TEMP2 NEQU CSR2_STATUS
;    1699  2        THEN
;    1700  3          BEGIN
;    1701  3            ERP_FLAG = ONE;
;    1702  3            CSR_WORD = GET_BIT [ CSR_ALL ];
;    1703  3            PRINTB ( MSG59 );
;    1704  3            PRINTB ( MSG31 );
;    1705  3            PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP1, CSR2_STATUS );
;    1706  3            ERRDF ( 0002, MSG00, E1$REPORT );
;    1707  2          END;
;    1708  2
;    1709  1      END;


                                        .GLOBL  L$DLY

                                        .SBTTL  RESET.DEQNA GLOBAL ROUTINE - RESET_DEQNA ( )
```

```
000000  004137  000000G                RESET.DEQNA::
                                                JSR     R1,$SAVE2               ;                                       1630
000004  162706  000016                          SUB     #16,SP
000010  013700  000000G                        MOV     REG.ADR,R0              ;                                       1672
000014  012702  000016                          MOV     #16,R2
000020  060002                                 ADD     R0,R2
000022  005012                                 CLR     (R2)
000024  152712  000002                         BISB    #2,(R2)                 ;                                       1673
000030  012701  000001                         MOV     #1,R1                   ; *,$$TMP2                              1675
000034  001410                          1$:    BEQ     4$
000036  013700  000000G                        MOV     L$DLY,R0                ; *,$$TMP1
000042  001403                                 BEQ     3$
000044  005066  000014                  2$:    CLR     14(SP)                  ; $$TMP
000050  077003                                 SOB     R0,2$                   ; $$TMP1,*
000052  005301                          3$:    DEC     R1                      ; $$TMP2
000054  000767                                 BR      1$
000056  011216                          4$:    MOV     (R2),(SP)               ; *,TMP.LOCATION                        1676
000060  011637  000000G                        MOV     (SP),TEMP1
000064  042737  010000  000000G               BIC     #10000,TEMP1
000072  023727  000000G 000062                 CMP     TEMP1,#62               ;                                       1678
000100  001453                                 BEQ     5$
000102  012737  000001  000000G               MOV     #1,ERR.FLAG             ;                                       1681
000110  011666  000002                         MOV     (SP),2(SP)              ; *,TMP.LOCATION                        1682
000114  011637  000000G                        MOV     (SP),CSR.WORD
000120  012746  000000G                        MOV     #MSG59,-(SP)            ;                                       1683
000124  012746  000001                         MOV     #1,-(SP)
000130  010600                                 MOV     SP,R0                   ; SP,*
000132  104414                                 TRAP    14
000134  012716  000000G                        MOV     #MSG31,(SP)             ;                                       1684
000140  012746  000001                         MOV     #1,-(SP)
000144  010600                                 MOV     SP,R0                   ; SP,*
000146  104414                                 TRAP    14
000150  012716  000060                         MOV     #60,(SP)                ;                                       1685
000154  013746  000000G                        MOV     TEMP1,-(SP)
000160  013766  000000G 000014                 MOV     GET.ADR,14(SP)          ; *,TMP.LOCATION
000166  062766  000016  000014                 ADD     #16,14(SP)              ; *,TMP.LOCATION
000174  016646  000014                         MOV     14(SP),-(SP)            ; TMP.LOCATION,*
000200  012746  000000G                        MOV     #MSG30,-(SP)
000204  012746  000004                         MOV     #4,-(SP)
000210  010600                                 MOV     SP,R0                   ; SP,*
000212  104414                                 TRAP    14
000214  104455                                 TRAP    55                      ;                                       1686
000216  000001                                 .WORD   1
000220  000000G                                .WORD   MSG00
000222  000304'                                .WORD   E1$REPORT
000224  062706  000016                         ADD     #16,SP                  ;                                       1680
000230  013700  000000G                  5$:    MOV     REG.ADR,R0              ;                                       1695
000234  142760  000002  000016               BICB    #2,16(R0)
000242  012702  000001                         MOV     #1,R2                   ; *,$$TMP2                              1696
000246  001410                          6$:    BEQ     9$
000250  013701  000000G                        MOV     L$DLY,R1                ; *,$$TMP1
000254  001403                                 BEQ     8$
000256  005066  000014                  7$:    CLR     14(SP)                  ; $$TMP
```

L2


```
000262  077103                                  SOB     R1,7$                           ; $$TMP1,*
000264  005302                          8$:     DEC     R2                              ; $$TMP2
000266  000767                                  BR      6$
000270  016066  000016  000006          9$:     MOV     16(R0),6(SP)                    ; *,TMP.LOCATION                        1697
000276  016637  000006  000000G                 MOV     6(SP),TEMP2                     ; TMP.LOCATION,*
000304  042737  010000  000000G                 BIC     #10000,TEMP2
000312  023727  000000G 000060                  CMP     TEMP2,#60                       ;                                       1698
000320  001455                                  BEQ     10$
000322  012737  000001  000000G                 MOV     #1,ERR.FLAG                     ;                                       1701
000330  016666  000006  000010                  MOV     6(SP),10(SP)                    ; *,TMP.LOCATION                        1702
000336  016637  000010  000000G                 MOV     10(SP),CSR.WORD                 ; TMP.LOCATION,*
000344  012746  000000G                         MOV     #MSG59,-(SP)                    ;                                       1703
000350  012746  000001                          MOV     #1,-(SP)
000354  010600                                  MOV     SP,R0                           ; SP,*
000356  104414                                  TRAP    14
000360  012716  000000G                         MOV     #MSG31,(SP)                     ;                                       1704
000364  012746  000001                          MOV     #1,-(SP)
000370  010600                                  MOV     SP,R0                           ; SP,*
000372  104414                                  TRAP    14
000374  012716  000060                          MOV     #60,(SP)                        ;                                       1705
000400  013716  000000G                         MOV     TEMP1,-(SP)
000404  013766  000000G 000022                  MOV     GET.ADR,22(SP)                  ; *,TMP.LOCATION
000412  062766  000016  000022                  ADD     #16,22(SP)                      ; *,TMP.LOCATION
000420  016646  000022                          MOV     22(SP),-(SP)                    ; TMP.LOCATION,*
000424  012746  000000G                         MOV     #MSG30,-(SP)
000430  012746  000004                          MOV     #4,-(SP)
000434  010600                                  MOV     SP,R0                           ; SP,*
000436  104414                                  TRAP    14
000440  104455                                  TRAP    55                              ;                                       1706
000442  000002                                  .WORD   2
000444  000000G                                 .WORD   MSC00
000446  000304'                                 .WORD   E1$REPORT
000450  062706  000016                          ADD     #16,SP                          ;                                       1700
000454  062706  000016                  10$:    ADD     #16,SP                          ;                                       1630
000460  000207                                  RTS     PC

; Routine Size:  153 words,     Routine Base:  AC$CODE$ + 0324
; Maximum stack depth per invocation:  19 words


;   1710  1
```

```
;    1711  1     %SBTTL 'GLOBAL ROUTINE - VER_DESCR_STATUS ( )'
;    1712  1
;    1713  1     GLOBAL ROUTINE VER_DESCR_STATUS : NOVALUE =
;    1714  1
;    1715  1     !++
;    1716  1     !
;    1717  1     !  GLOBAL ROUTINE :     VER_DESCR_STATUS
;    1718  1     !
;    1719  1     !  DESCRIPTION:
;    1720  1     !
;    1721  1     !       This routine compares expected receive descriptor to actual receive
;    1722  1     !       descriptor.
;    1723  1     !
;    1724  1     !  INPUT PARAMETERS:
;    1725  1     !
;    1726  1     !       TEST_NU   test number in which error occurred.
;    1727  1     !
;    1728  1     !--
;    1729  1
;    1730  1
;    1731  2     BEGIN
;    1732  2
;    1733  2     INCR INDEX FROM 0 TO BD_D_SIZE - 1 DO
;    1734  3       BEGIN
;    1735  3         TEMP1 = .DESCR_LIST [ .INDEX, W_LEN ];
;    1736  3         TEMP2 = .DESCR_LIST [ .INDEX, W_LEN ] AND RFLG_MASK;
;    1737  4         IF ( .TEMP2 NEQU RFLG_MASK ) AND ( .TEMP1 NEQU .BD_PROM_DESCR [ .INDEX ] )
;    1738  3           THEN
;    1739  4             BEGIN
;    1740  4               CSR_WORD = GET_BIT [ CSR_ALL ];
;    1741  4               PRINTB ( MSG59 );
;    1742  4               PRINTB ( MSG48 );
;    1743  4               PRINTB ( MSG50, .TEMP1, .BD_PROM_DESCR [ .INDEX ] , .INDEX );
;    1744  4               ERRDF ( 0003, MSG00, ERROR$REPORT );
;    1745  3             END;
;    1746  2       END;
;    1747  2
;    1748  1     END;


                                        .SBTTL  VER.DESCR.STATUS GLOBAL ROUTINE - VER_DESCR_STATUS ( )
000000  004137  000000G               VER.DESCR.STATUS::
                                        JSR     R1,$SAVE2                       ;                       1713
000004  005746                         TST     -(SP)
000006  005002                         CLR     R2                              ; INDEX                 1733
000010  010201                 1$:     MOV     R2,R1                           ; INDEX,*               1735
000012  006301                         ASL     R1
000014  016137  000000G 000000G        MOV     DESCR.LIST(R1),TEMP1
000022  016137  000000G 000000G        MOV     DESCR.LIST(R1),TEMP2            ;                       1736
000030  042737  037777  000000G        BIC     #37777,TEMP2
000036  023727  000000G 140000         CMP     TEMP2,#-40000                   ;                       1737
000044  001447                         BEQ     2$
000046  026161  000000G 000000G        CMP     DESCR.LIST(R1),BD.PROM.DESCR(R1)
```

```
000054  001443                          BEQ     2$
000056  013700  000000G                 MOV     REG.ADR,R0                      ;                               1740
000062  016016  000016                  MOV     16(R0),(SP)                     ; *,TMP.LOCATION
000066  011637  000000G                 MOV     (SP),CSR.WORD                   ; TMP.LOCATION,*
000072  012746  000000G                 MOV     #MSG59,-(SP)                    ;                               1741
000076  012746  000001                  MOV     #1,-(SP)
000102  010600                          MOV     SP,R0                           ; SP,*
000104  104414                          TRAP    14
000106  012716  000000G                 MOV     #MSG48,(SP)                     ;                               1742
000112  012746  000001                  MOV     #1,-(SP)
000116  010600                          MOV     SP,R0                           ; SP,*
000120  104414                          TRAP    14
000122  010216                          MOV     R2,(SP)                         ; INDEX,*                       1743
000124  016146  000000G                 MOV     BD.PROM.DESCR(R1),-(SP)
000130  013746  000000G                 MOV     TEMP1,-(SP)
000134  012746  000000G                 MOV     #MSG50,-(SP)
000140  012746  000004                  MOV     #4,-(SP)
000144  010600                          MOV     SP,R0                           ; SP,*
000146  104414                          TRAP    14
000150  104455                          TRAP    55                              ;                               1744
000152  000003                          .WORD   3
000154  000000G                         .WORD   MSG00
000156  000000'                         .WORD   ERROR$REPORT
000160  062706  000016                  ADD     #16,SP                          ;                               1739
000164  005202                  2$:     INC     R2                              ; INDEX                         1733
000166  020227  000017                  CMP     R2,#17                          ; INDEX,*
000172  003706                          BLE     1$
000174  005726                          TST     (SP)+                           ;                               1713
000176  000207                          RTS     PC

; Routine Size:  64 words,      Routine Base:  AC$CODE$ + 1006
; Maximum stack depth per invocation:  13 words


;   1749  1
```

```
ZQNA4           (ZQNADO DEQNA FUNCTIONAL TEST                14-Mar-1985 13:18:55    VAX-11 Bliss-16 V4.1-582              SEQ 0234
V01.0           GLOBAL ROUTINE - CLR_DESCR ( )               14-Mar-1985 13:06:01    DISK$USER2:[MARSHALL.DEQNA]ZQNA4.BLI;4  Page  14
                                                                                                                              (9)

;    1750  1     %SBTTL 'GLOBAL ROUTINE - CLR_DESCR ( )'
;    1751  1
;    1752  1     GLOBAL ROUTINE CLR_DESCR : NOVALUE =
;    1753  1
;    1754  1     !++
;    1755  1     !
;    1756  1     !  GLOBAL ROUTINE :     CLR_DESCR
;    1757  1     !
;    1758  1     !  DESCRIPTION:
;    1759  1     !
;    1760  1     !       This routine initializes transmit and receive descriptor lists to 0.
;    1761  1     !--
;    1762  1
;    1763  1
;    1764  2     BEGIN
;    1765  2
;    1766  2     INCR INDEX FROM 0 TO D_SIZE - 1 DO
;    1767  3       BEGIN
;    1768  3         XMIT_D_LIST [ .INDEX, W_LEN ] = 0;
;    1769  3         RCV_D_LIST  [ .INDEX, W_LEN ] = 0;
;    1770  2       END;
;    1771  2
;    1772  1     END;


                                                .SBTTL  CLR.DESCR GLOBAL ROUTINE - CLR_DESCR ( )
000000  005000                                CLR.DESCR::
                                                CLR     R0                              ; INDEX                         1766
000002  005060  000000G                       1$:     CLR     XMIT.D.LIST(R0)         ; *(INDEX)                      1768
000006  005060  000000G                               CLR     RCV.D.LIST(R0)          ; *(INDEX)                      1769
000012  062700  000002                                ADD     #2,R0                   ; *,INDEX                       1766
000016  020027  000176                                CMP     R0,#176                 ; INDEX,*
000022  003767                                       BLE     1$
000024  000207                                       RTS     PC                      ;                               1752

; Routine Size:  11 words,      Routine Base:  AC$CODE$ + 1206
; Maximum stack depth per invocation:  0 words


;    1773  1
;    1774  1
```

```
;    1775  1     %SBTTL 'GLOBAL ROUTINE - CLR_BUFFERS ( P1 )'
;    1776  1
;    1777  1     GLOBAL ROUTINE CLR_BUFFERS ( P1 ) : NOVALUE =
;    1778  1
;    1779  1     !++
;    1780  1     !
;    1781  1     !  GLOBAL ROUTINE :     CLR_BUFFERS
;    1782  1     !
;    1783  1     !  DESCRIPTION:
;    1784  1     !
;    1785  1     !       This routine initializes transmit and receive buffers to 0.
;    1786  1     !
;    1787  1     !  INPUT PARAMETERS:
;    1788  1     !
;    1789  1     !       P1 - number of bytes to clear.
;    1790  1     !
;    1791  1     !--
;    1792  1
;    1793  1
;    1794  2     BEGIN
;    1795  2
;    1796  2     INCR INDEX FROM 0 TO .P1 - 1 DO
;    1797  3       BEGIN
;    1798  3         RCV_BUFFER  [ .INDEX ] = 0;
;    1799  3         XMIT_BUFFER [ .INDEX ] = 0;
;    1800  2       END;
;    1801  2
;    1802  1     END;


                                        .SBTTL  CLR.BUFFERS GLOBAL ROUTINE - CLR_BUFFERS ( P1 )
000000  005000                    CLR.BUFFERS::
                                        CLR     R0                      ; INDEX                 1796
000002  000405                            BR      2$
000004  105060  000000G           1$:     CLRB    RCV.BUFFER(R0)          ; *(INDEX)              1798
000010  105060  000000G                   CLRB    XMIT.BUFFER(R0)         ; *(INDEX)              1799
000014  005200                            INC     R0                      ; INDEX                 1796
000016  020066  000002            2$:     CMP     R0,2(SP)                ; INDEX,P1
000022  002770                            BLT     1$
000024  000207                            RTS     PC                      ;                       1777

; Routine Size:  11 words,      Routine Base:  AC$CODE$ + 1234
; Maximum stack depth per invocation:  0 words


;    1803  1
;    1804  1
```

```
;    1805  1     %SBTTL 'GLOBAL ROUTINE - CHK_RIXI_STATUS ( P1 )'
;    1806  1
;    1807  1     GLOBAL ROUTINE CHK_RIXI_STATUS ( P1 ) : NOVALUE  =
;    1808  1
;    1809  1     !++
;    1810  1     !
;    1811  1     !  GLOBAL ROUTINE :     CHK_RIXI_STATUS
;    1812  1     !
;    1813  1     !  DESCRIPTION:
;    1814  1     !
;    1815  1     !       This routine verifies that XI ( bit 7 ) and RI ( bit 15 )
;    1816  1     !       of the CSR status word are set to 1 shortly after transmission of a
;    1817  1     !       loopback packet is complete. If either bit isn't set, an error
;    1818  1     !       message is printed.
;    1819  1     !
;    1820  1     !  INPUT PARAMETERS:
;    1821  1     !
;    1822  1     !       P1 - 0: check XI and RI
;    1823  1     !          - 1: ckeck XI
;    1824  1     !          - 2: check RI
;    1825  1     !
;    1826  1     !       TEST_NO - test number in which error occurred.
;    1827  1     !--
;    1828  1
;    1829  2     BEGIN
;    1830  2
;    1831  2     !++
;    1832  2     !  CHECK TRANSMIT INTERRUPT REQUEST BIT ( XI - BIT 7 ) TO VERIFY THAT DEQNA
;    1833  2     !  ACTUALLY COMPLETED TRANSMISSION OF A LOOPBACK PACKET.
;    1834  2     !--
;    1835  2
;    1836  3     IF ( .P1 EQLU 0 ) OR ( .P1 EQLU 1 )
;    1837  2       THEN
;    1838  2         INCR INDEX FROM 0 TO TIME2_LIMIT DO
;    1839  2           IF GET_BIT [ CSR, XI ] EQLU ONE
;    1840  2             THEN
;    1841  3               BEGIN
;    1842  3                 TEMP1 = .INDEX;
;    1843  3                 EXITLOOP;
;    1844  3               END
;    1845  2             ELSE
;    1846  2               IF .INDEX EQLU TIME3_LIMIT
;    1847  2               THEN
;    1848  3                 BEGIN
;    1849  3                   ERR_FLAG = ONE;
;    1850  3                   CSR_WORD = GET_BIT [ CSR_ALL ];
;    1851  3                   PRINTB ( MSG59 );
;    1852  3                   PRINTB ( MSG29 );
;    1853  3                   PRINTB ( MSG26 );
;    1854  3                   ERRDF ( 0004, MSG00, ERROR$REPORT );
;    1855  2                 END;
;    1856  2
;    1857  2     !++
```

```
;    1858  2     !  CHECK RECEIVE INTERRUPT REQUEST BIT ( RI - BIT 15 ) TO VERIFY THAT DEQNA
;    1859  2     !  ACTUALLY RECEIVED TRANSMITTED LOOPBACK PACKET.
;    1860  2     !--
;    1861  2
;    1862  3     IF ( .P1 EQLU 0 ) OR ( .P1 EQLU 2 )
;    1863  2       THEN
;    1864  2         INCR INDEX FROM 0 TO TIME2_LIMIT DO
;    1865  2           IF GET_BIT [ CSR, RI ] EQLU ONE
;    1866  2             THEN
;    1867  3               BEGIN
;    1868  3                 TEMP2 = .INDEX;
;    1869  3                 EXITLOOP;
;    1870  3               END
;    1871  2             ELSE
;    1872  2               IF .INDEX EQLU TIME2_LIMIT
;    1873  2               THEN
;    1874  3                 BEGIN
;    1875  3                   ERR_FLAG = ONE;
;    1876  3                   CSR_WORD = GET_BIT [ CSR_ALL ];
;    1877  3                   PRINTB ( MSG59 );
;    1878  3                   PRINTB ( MSG29 );
;    1879  3                   PRINTB ( MSG25 );
;    1880  3                   ERRDF ( 0005, MSG00, ERROR$REPORT );
;    1881  2                 END;
;    1882  1     END;


                                        .SBTTL  CHK.RIXI.STATUS GLOBAL ROUTINE - CHK_RIXI_STATUS ( P1 )
000000  004137  000000G           CHK.RIXI.STATUS::
                                        JSR     R1,$SAVE3                       ;                               1807
000004  162706  000010                  SUB     #10,SP
000010  016602  000022                  MOV     22(SP),R2                       ; P1,*                          1836
000014  005003                          CLR     R3
000016  005702                          TST     R2
000020  001002                          BNE     1$
000022  005203                          INC     R3
000024  000403                          BR      2$
000026  020227  000001          1$:     CMP     R2,#1
000032  001062                          BNE     6$
000034  005001                  2$:     CLR     R1                              ; INDEX                         1838
000036  013700  000000G         3$:     MOV     REG.ADR,R0                      ;                               1839
000042  016016  000016                  MOV     16(R0),(SP)                     ; *,TMP.LOCATION
000046  105716                          TSTB    (SP)                            ; TMP.LOCATION
000050  100003                          BPL     4$
000052  010137  000000G                 MOV     R1,TEMP1                        ; INDEX,*                       1842
000056  000450                          BR      6$                              ;                               1841
000060  020127  002000          4$:     CMP     R1,#2000                        ; INDEX,*                       1846
000064  001041                          BNE     5$
000066  012737  000001  000000G         MOV     #1,ERR.FLAG                     ;                               1849
000074  016066  000016  000002         MOV     16(R0),2(SP)                    ; *,TMP.LOCATION                1850
000102  016637  000002  000000G         MOV     2(SP),CSR.WORD                  ; TMP.LOCATION,*
000110  012746  000000G                 MOV     #MSG59,-(SP)                    ;                               1851
000114  012746  000001                  MOV     #1,-(SP)
```

```
000120  010600                          MOV     SP,R0                   ; SP,*
000122  104414                          TRAP    14
000124  012716  000000G                 MOV     #MSG29,(SP)             ;                       1852
000130  012746  000001                  MOV     #1,-(SP)
000134  010600                          MOV     SP,R0                   ; SP,*
000136  104414                          TRAP    14
000140  012716  000000G                 MOV     #MSG26,(SP)             ;                       1853
000144  012746  000001                  MOV     #1,-(SP)
000150  010600                          MOV     SP,R0                   ; SP,*
000152  104414                          TRAP    14
000154  104455                          TRAP    55                      ;                       1854
000156  000004                          .WORD   4
000160  000000G                         .WORD   MSG00
000162  000000'                         .WORD   ERROR$REPORT
000164  062706  000010                  ADD     #10,SP                  ;                       1848
000170  005201                  5$:     INC     R1                      ; INDEX                 1838
000172  020127  002000                  CMP     R1,#2000                ; INDEX,*
000176  003717                          BLE     3$
000200  006003                  6$:     ROR     R3                      ;                       1862
000202  103403                          BLO     7$
000204  020227  000002                  CMP     R2,#2
000210  001062                          BNE     11$
000212  005001                  7$:     CLR     R1                      ; INDEX                 1864
000214  013700  000000G         8$:     MOV     REG.ADR,R0              ;                       1865
000220  016066  000016  000004          MOV     16(R0),4(SP)            ; *,TMP.LOCATION
000226  100003                          BPL     9$
000230  010137  000000G                 MOV     R1,TEMP2                ; INDEX,*               1868
000234  000450                          BR      11$                     ;                       1867
000236  020127  002000          9$:     CMP     R1,#2000                ; INDEX,*               1872
000242  001041                          BNE     10$
000244  012737  000001  000000G         MOV     #1,ERR.FLAG             ;                       1875
000252  016066  000016  000006          MOV     16(R0),6(SP)            ; *,TMP.LOCATION        1876
000260  016637  000006  000000G         MOV     6(SP),CSR.WORD          ; TMP.LOCATION,*
000266  012746  000000G                 MOV     #MSG59,-(SP)            ;                       1877
000272  012746  000001                  MOV     #1,-(SP)
000276  010600                          MOV     SP,R0                   ; SP,*
000300  104414                          TRAP    14
000302  012716  000000G                 MOV     #MSG29,(SP)             ;                       1878
000306  012746  000001                  MOV     #1,-(SP)
000312  010600                          MOV     SP,R0                   ; SP,*
000314  104414                          TRAP    14
000316  012716  000000G                 MOV     #MSG25,(SP)             ;                       1879
000322  012746  000001                  MOV     #1,-(SP)
000326  010600                          MOV     SP,R0                   ; SP,*
000330  104414                          TRAP    14
000332  104455                          TRAP    55                      ;                       1880
000334  000005                          .WORD   5
000336  000000G                         .WORD   MSG00
000340  000000'                         .WORD   ERROR$REPORT
000342  062706  000010                  ADD     #10,SP                  ;                       1874
000346  005201                  10$:    INC     R1                      ; INDEX                 1864
000350  020127  002000                  CMP     R1,#2000                ; INDEX,*
000354  003717                          BLE     8$
```

```
000356  062706  000010                  11$:    ADD     #10,SP                          ;                       1807
000362  000207                                  RTS     PC

; Routine Size:  122 words,     Routine Base:  AC$CODE$ + 1262
; Maximum stack depth per invocation:  14 words


;   1883  1
```

```
;    1884  1     %SBTTL 'GLOBAL ROUTINE - CHK_CSR_STATUS ( P1, P2 )'
;    1885  1
;    1886  1     GLOBAL ROUTINE CHK_CSR_STATUS ( P1, P2 ) : NOVALUE  =
;    1887  1
;    1888  1     !++
;    1889  1     !
;    1890  1     !  GLOBAL ROUTINE :     CHK_CSR_STATUS
;    1891  1     !
;    1892  1     !  DESCRIPTION:
;    1893  1     !
;    1894  1     !       This routine checks CSR status words for expected status.
;    1895  1     !
;    1896  1     !  INPUT PARAMETERS:
;    1897  1     !
;    1898  1     !        P1 - expected CSR status
;    1899  1     !        P2 - CSR mask
;    1900  1     !        TEST_NO - test number in which error occurred.
;    1901  1     !
;    1902  1     !--
;    1903  1
;    1904  2     BEGIN
;    1905  2
;    1906  2     !++
;    1907  2     !  SAVE CSR, RESET TRANSMIT AND RECEIVE REQUEST BITS IN THE CSR
;    1908  2     !--
;    1909  2
;    1910  2     DELAY ( 5 );
;    1911  2
;    1912  2     CSR_WORD = GET_BIT [ CSR_ALL ];
;    1913  2
;    1914  2     PUT_BIT [ CSR, RI, ONE ];
;    1915  2     PUT_BIT [ CSR, XI, ONE ];
;    1916  2
;    1917  2     TEMP1 = .CSR_WORD AND .P2;
;    1918  2
;    1919  2     IF .TEMP1 NEQU .P1
;    1920  2       THEN
;    1921  3         BEGIN
;    1922  3           ERR_FLAG = ONE;
;    1923  3           PRINTB ( MSG59 );
;    1924  3           PRINTB ( MSG12, .TEMP1, .P1 );
;    1925  3           ERRDF  ( 0006, MSG00, ERROR$REPORT );
;    1926  2         END;
;    1927  1     END;


                                        .SBTTL  CHK.CSR.STATUS GLOBAL ROUTINE - CHK_CSR_STATUS ( P1, P2 )
000000  010146                    CHK.CSR.STATUS::
                                        MOV     R1,-(SP)                        ;                                       1886
000002  024646                          CMP     -(SP),-(SP)
000004  012701  000005                  MOV     #5,R1                           ; *,$$TMP2                              1910
000010  001410                  1$:     BEQ     4$
000012  013700  000000G                 MOV     L$DLY,R0                        ; *,$$TMP1
```

```
000016  001403                                  BEQ     3$
000020  005066  000002                  2$:     CLR     2(SP)                   ; $$TMP
000024  077003                                  SOB     R0,2$                   ; $$TMP1,*
000026  005301                          3$:     DEC     R1                      ; $$TMP2
000030  000767                                  BR      1$
000032  013700  000000G                 4$:     MOV     REG.ADR,R0              ;                                               1912
000036  062700  000016                          ADD     #16,R0
000042  011016                                  MOV     (R0),(SP)               ; *,TMP.LOCATION
000044  011637  000000G                         MOV     (SP),CSR.WORD
000050  052710  100200                          BIS     #100200,(R0)            ;                                               1915
000054  011637  000000G                         MOV     (SP),TEMP1              ; CSR.WORD,*                                    1917
000060  016600  000010                          MOV     10(SP),R0               ; P2,*
000064  005100                                  COM     R0
000066  040037  000000G                         BIC     R0,TEMP1
000072  023766  000000G 000012                  CMP     TEMP1,12(SP)            ; *,P1                                          1919
000100  001431                                  BEQ     5$
000102  012737  000001  000000G                 MOV     #1,ERR.FLAG             ;                                               1922
000110  012746  000000G                         MOV     #MSG59,-(SP)            ;                                               1923
000114  012746  000001                          MOV     #1,-(SP)
000120  010600                                  MOV     SP,R0                   ; SP,*
000122  104414                                  TRAP    14
000124  016616  000016                          MOV     16(SP),(SP)             ; P1,*                                          1924
000130  013746  000000G                         MOV     TEMP1,-(SP)
000134  012746  000000G                         MOV     #MSG12,-(SP)
000140  012746  000003                          MOV     #3,-(SP)
000144  010600                                  MOV     SP,R0                   ; SP,*
000146  104414                                  TRAP    14
000150  104455                                  TRAP    55                      ;                                               1925
000152  000006                                  .WORD   6
000154  000000G                                 .WORD   MSG00
000156  000000'                                 .WORD   ERROR$REPORT
000160  062706  000012                          ADD     #12,SP                  ;                                               1921
000164  022626                          5$:     CMP     (SP)+,(SP)+             ;                                               1886
000166  012601                                  MOV     (SP)+,R1
000170  000207                                  RTS     PC

; Routine Size:  61 words,      Routine Base:  AC$CODE$ + 1646
; Maximum stack depth per invocation:  10 words


;   1928  1
;   1929  1
```

```
;    1930  1     %SBTTL 'GLOBAL ROUTINE - CHK_XMIT_STATUS ( P1, P2 )'
;    1931  1
;    1932  1     GLOBAL ROUTINE CHK_XMIT_STATUS ( P1, P2 ) : NOVALUE  =
;    1933  1
;    1934  1     !++
;    1935  1     !
;    1936  1     !  GLOBAL ROUTINE :     CHK_XMIT_STATUS
;    1937  1     !
;    1938  1     !  DESCRIPTION:
;    1939  1     !
;    1940  1     !       This routine checks transmit status words for expected status
;    1941  1     !
;    1942  1     !  INPUT PARAMETERS:
;    1943  1     !
;    1944  1     !       P1      - XMIT flag word
;    1945  1     !       P2      - expected XMIT status word 1
;    1946  1     !       TEST_NO - test number in which error occurred.
;    1947  1     !
;    1948  1     !
;    1949  1     !--
;    1950  1
;    1951  2     BEGIN
;    1952  2
;    1953  2     !++
;    1954  2     !  MASK OUT DON'T CARE BITS IN THE XMIT FLAG WORD AND COMPARE TO EXPECTED
;    1955  2     !  XMIR FLAG STATUS. IF STATUS NOT EQUAL THEN PRINT 'BAD XMIT FLAG WORD
;    1956  2     !  STATUS'
;    1957  2     !--
;    1958  2
;    1959  2     TEMP2 = .XMIT_D_LIST [ FLGWD ] AND XFLG_MASK;           ! O'140000'
;    1960  2
;    1961  2     IF .TEMP2 NEQU .P1
;    1962  2       THEN
;    1963  3         BEGIN
;    1964  3           ERR_FLAG = ONE;
;    1965  3           CSR_WORD = GET_BIT [ CSR_ALL ];
;    1966  3           PRINTB ( MSG59 );
;    1967  3           PRINTB ( MSG13, .TEMP2, XFLG_MASK );
;    1968  3           ERRDF ( 0007, MSG00, ERROR$REPORT );
;    1969  2         END;
;    1970  2
;    1971  2     !++
;    1972  2     !  MASK OUT DON'T CARE BITS IN THE XMIT STATUS WD1 AND COMPARE TO EXPECTED
;    1973  2     !  XMIT STATUS WD1. IF STATUS NOT EQUAL THEN PRINT 'BAD XMIT STATUS WORD 1'
;    1974  2     !--
;    1975  2
;    1976  2     IF .XMIT_D_LIST [ STWD1 ] GTRU ZERO
;    1977  2       THEN
;    1978  2         TEMP3 = .XMIT_D_LIST [ STWD1 ] AND XWD1_MASK      ! O'157760'
;    1979  2       ELSE
;    1980  2         TEMP3 = .XMIT_D_LIST [ STWD1 ] AND X1_MASK;       ! O'100000'
;    1981  2
;    1982  2     IF .TEMP3 NEQU .P2
```

```
;    1983  2        THEN
;    1984  3          BEGIN
;    1985  3            ERR_FLAG = ONE;
;    1986  3            CSR_WORD = GET_BIT ( CSR_ALL );
;    1987  3            PRINTB ( MSG59 );
;    1988  3            PRINTB ( MSG14, .TEMP3, .P2 );
;    1989  3            ERRDF ( 0008, MSG00, ERROR$REPORT );
;    1990  2          END;
;    1991  2
;    1992  1      END,


                                        .SBTTL  CHK.XMIT.STATUS GLOBAL ROUTINE - CHK_XMIT_STATUS ( P1, P2 )
000000  024646                    CHK.XMIT.STATUS::
                                        CMP     -(SP),-(SP)                     ;                                       1932
000002  013737  000000G 000000G         MOV     XMIT.D.LIST,TEMP2               ;                                       1959
000010  042737  037777  000000G         BIC     #37777,TEMP2
000016  023766  000000G 000010          CMP     TEMP2,10(SP)                    ; *,P1                                  1961
000024  001437                          BEQ     1$
000026  012737  000001  000000G         MOV     #1,ERR.FLAG                     ;                                       1964
000034  013700  000000G                 MOV     REG.ADR,R0                      ;                                       1965
000040  016016  000016                  MOV     16(R0),(SP)                     ; *,TMP.LOCATION
000044  011637  000000G                 MOV     (SP),CSR.WORD                   ; TMP.LOCATION,*
000050  012746  000000G                 MOV     #MSG59,-(SP)                    ;                                       1966
000054  012746  000001                  MOV     #1,-(SP)
000060  010600                          MOV     SP,R0                           ; SP,*
000062  104414                          TRAP    14
000064  012716  140000                  MOV     #-40000,(SP)                    ;                                       1967
000070  013746  000000G                 MOV     TEMP2,-(SP)
000074  012746  000000G                 MOV     #MSG13,-(SP)
000100  012746  000003                  MOV     #3,-(SP)
000104  010600                          MOV     SP,R0                           ; SP,*
000106  104414                          TRAP    14
000110  104455                          TRAP    55                              ;                                       1968
000112  000007                          .WORD   7
000114  000000G                         .WORD   MSG00
000116  000000'                         .WORD   ERROR$REPORT
000120  062706  000012                  ADD     #12,SP                          ;                                       1963
000124  013700  000010G         1$:     MOV     XMIT.D.LIST+10,R0               ;                                       1976
000130  001406                          BEQ     2$
000132  010037  000000G                 MOV     R0,TEMP3                        ;                                       1978
000136  042737  020017  000000G         BIC     #20017,TEMP3
000144  000405                          BR      3$                              ;                                       1976
000146  010037  000000G         2$:     MOV     R0,TEMP3                        ;                                       1980
000152  042737  077777  000000G         BIC     #77777,TEMP3
000160  023766  000000G 000006  3$:     CMP     TEMP3,6(SP)                     ; *,P2                                  1982
000166  001441                          BEQ     4$
000170  012737  000001  000000G         MOV     #1,ERR.FLAG                     ;                                       1985
000176  013700  000000G                 MOV     REG.ADR,R0                      ;                                       1986
000202  016066  000016  000002          MOV     16(R0),2(SP)                    ; *,TMP.LOCATION
000210  016637  000002  000000G         MOV     2(SP),CSR.WORD                  ; TMP.LOCATION,*
000216  012746  000000G                 MOV     #MSG59,-(SP)                    ;                                       1987
000222  012746  000001                  MOV     #1,-(SP)
```

L3


```
000226  010600                          MOV     SP,R0                   ; SP,*
000230  104414                          TRAP    14
000232  016616  000012                  MOV     12(SP),(SP)             ; P2,*                  1988
000236  013746  000000G                 MOV     TEMP3,-(SP)
000242  012746  000000G                 MOV     #MSG14,-(SP)
000246  012746  000003                  MOV     #3,-(SP)
000252  010600                          MOV     SP,R0                   ; SP,*
000254  104414                          TRAP    14
000256  104455                          TRAP    55                      ;                       1989
000260  000010                          .WORD   10
000262  000000G                         .WORD   MSG00
000264  000000'                         .WORD   ERROR$REPORT
000266  062706  000012                  ADD     #12,SP                  ;                       1984
000272  022626                  4$:     CMP     (SP)+,(SP)+             ;                       1932
000274  000207                          RTS     PC

; Routine Size:  95 words,      Routine Base:  AC$CODE$ + 2040
; Maximum stack depth per invocation:  9 words


;   1993  1
;   1994  1
```

```
;    1995  1     %SBTTL 'GLOBAL ROUTINE - CHK_RCV_STATUS ( P1, P2 )'
;    1996  1
;    1997  1     GLOBAL ROUTINE CHK_RCV_STATUS ( P1, P2 ) : NOVALUE  =
;    1998  1
;    1999  1     !++
;    2000  1     !
;    2001  1     !  GLOBAL ROUTINE :     CHK_RCV_STATUS
;    2002  1     !
;    2003  1     !  DESCRIPTION:
;    2004  1     !
;    2005  1     !       This routine checks receive status words for expected status.
;    2006  1     !
;    2007  1     !  INPUT PARAMETERS:
;    2008  1     !
;    2009  1     !       P1       - expected RCV flag word
;    2010  1     !       P2       - expected RCV status word 1
;    2011  1     !       TEST_NO - test number in which error occurred.
;    2012  1     !
;    2013  1     !--
;    2014  1
;    2015  2     BEGIN
;    2016  2
;    2017  2     !++
;    2018  2     !  MASK OUT DON'T CARE BITS IN THE RCV FLAG WORD AND COMPARE TO EXPECTED
;    2019  2     !  RCV FLAG STATUS. IF STATUS NOT EQUAL THEN PRINT 'BAD RCV FLAG WORD
;    2020  2     !  STATUS'
;    2021  2     !--
;    2022  2
;    2023  2     TEMP1 = .RCV_D_LIST [ FLGWD ] AND RFLG_MASK;            ! 0'140000'
;    2024  2
;    2025  2     IF .TEMP1 NEQU .P1
;    2026  2       THEN
;    2027  3         BEGIN
;    2028  3           ERR_FLAG = ONE;
;    2029  3           CSR_WORD = GET_BIT [ CSR_ALL ];
;    2030  3           PRINTB ( MSG59 );
;    2031  3           PRINTB ( MSG15, .TEMP1, RFLG_MASK );
;    2032  3           ERRDF ( 0009, MSG00, ERROR$REPORT );
;    2033  2         END;
;    2034  2
;    2035  2     !++
;    2036  2     !  MASK OUT DON'T CARE BITS IN THE RCV STATUS WD1 AND COMPARE TO EXPECTED
;    2037  2     !  RCV STATUS WD1. IF STATUS NOT EQUAL THEN PRINT 'BAD RCV STATUS WORD 1'
;    2038  2     !--
;    2039  2
;    2040  2     IF .RCV_D_LIST [ STWD1 ] GEQU ZERO
;    2041  2       THEN
;    2042  2         TEMP2 = .RCV_D_LIST [ STWD1 ] AND R2_MASK           ! 0'174017'
;    2043  2       ELSE
;    2044  2         TEMP2 = .RCV_D_LIST [ STWD1 ] AND .P2;
;    2045  2
;    2046  2     IF .TEMP2 NEQU .P2
;    2047  2       THEN
```

```
;    2048  3          BEGIN
;    2049  3            ERR_FLAG = ONE;
;    2050  3            CSR_WORD = GET_BIT [ CSR_ALL ];
;    2051  3            PRINTB ( MSG59 );
;    2052  3            PRINTB ( MSG16, .TEMP2, .P2 );
;    2053  3            CRRDF ( 0010, MSG00, ERROR$REPORT );
;    2054  2          END;
;    2055  2
;    2056  1      END;


                                                .SBTTL  CHK.RCV.STATUS GLOBAL ROUTINE - CHK_RCV_STATUS ( P1, P2 )
000000  024646                            CHK.RCV.STATUS::
                                                CMP     -(SP),-(SP)                     ;                                       1997
000002  013737  000000G 000000G                 MOV     RCV.D.LIST,TEMP1                ;                                       2023
000010  042737  037777  000000G                 BIC     #37777,TEMP1
000016  023766  000000G 000010                  CMP     TEMP1,10(SP)                    ; *,P1                                  2025
000024  001437                                  BEQ     1$
000026  012737  000001  000000G                 MOV     #1,ERR.FLAG                     ;                                       2028
000034  013700  000000G                         MOV     REG.ADR,R0                      ;                                       2029
000040  016016  000016                          MOV     16(R0),(SP)                     ; *,TMP.LOCATION
000044  011637  000000G                         MOV     (SP),CSR.WORD                   ; TMP.LOCATION,*
000050  012746  000000G                         MOV     #MSG59,-(SP)                    ;                                       2030
000054  012746  000001                          MOV     #1,-(SP)
000060  010600                                  MOV     SP,R0                           ; SP,*
000062  104414                                  TRAP    14
000064  012716  140000                          MOV     #-40000,(SP)                    ;                                       2031
000070  013746  000000G                         MOV     TEMP1,-(SP)
000074  012746  000000G                         MOV     #MSG15,-(SP)
000100  012746  000003                          MOV     #3,-(SP)
000104  010600                                  MOV     SP,R0                           ; SP,*
000106  104414                                  TRAP    14
000110  104455                                  TRAP    55                              ;                                       2032
000112  000011                                  .WORD   11
000114  000000G                                 .WORD   MSG00
000116  000000'                                 .WORD   ERROR$REPORT
000120  062706  000012                          ADD     #12,SP                          ;                                       2027
000124  013700  000010G                 1$:     MOV     RCV.D.LIST+10,R0                ;                                       2040
000130  010037  000000G                         MOV     R0,TEMP2                        ;                                       2042
000134  042737  003764  000000G                 BIC     #3764,TEMP2
000142  023766  000000G 000006                  CMP     TEMP2,6(SP)                     ; *,P2                                  2046
000150  001441                                  BEQ     2$
000152  012737  000001  000000G                 MOV     #1,ERR.FLAG                     ;                                       2049
000160  013700  000000G                         MOV     REG.ADR,R0                      ;                                       2050
000164  016066  000016  000002                  MOV     16(R0),2(SP)                    ; *,TMP.LOCATION
000172  016637  000002  000000G                 MOV     2(SP),CSR.WORD                  ; TMP.LOCATION,*
000200  012746  000000G                         MOV     #MSG59,-(SP)                    ;                                       2051
000204  012746  000001                          MOV     #1,-(SP)
000210  010600                                  MOV     SP,R0                           ; SP,*
000212  104414                                  TRAP    14
000214  016616  000012                          MOV     12(SP),(SP)                     ; P2,*                                  2052
000220  013746  000000G                         MOV     TEMP2,-(SP)
000224  012746  000000G                         MOV     #MSG16,-(SP)
```

```
000230  012746  000003                          MOV     #3,-(SP)
000234  010600                                  MOV     SP,R0                           ; SP,*
000236  104414                                  TRAP    14
000240  104455                                  TRAP    55                              ;                                       2053
000242  000012                                  .WORD   12
000244  000000G                                 .WORD   MSG00
000246  000000'                                 .WORD   ERROR$REPORT
000250  062706  000012                          ADD     #12,SP                          ;                                       2048
000254  022626                          2$:     CMP     (SP)+,(SP)+                     ;                                       1997
000256  000207                                  RTS     PC

; Routine Size:  88 words,      Routine Base:  AC$CODE$ + 2336
; Maximum stack depth per invocation:  9 words


;   2057  1
```

```
;    2058  1     %SBTTL 'GLOBAL ROUTINE - COMPARE_PACKETS ( )'
;    2059  1
;    2060  1     GLOBAL ROUTINE COMPARE_PACKETS : NOVALUE =
;    2061  1
;    2062  1     !++
;    2063  1     !
;    2064  1     !  GLOBAL ROUTINE :     COMPARE_PACKETS
;    2065  1     !
;    2066  1     !  DESCRIPTION:
;    2067  1     !
;    2068  1     !       This routine compares contents of transmit packet to the contents
;    2069  1     !       of receive packet and prints an error message if the don't compare.
;    2070  1     !--
;    2071  1
;    2072  2     BEGIN
;    2073  2
;    2074  2     !++
;    2075  2     !  GET RECEIVE BYTE LENGTH ( RBL ) FROM RCV DISCRIPTOR AND COMPUTE WORD
;    2076  2     !  LENGTH. THEN COMPARE ACTUAL TO EXPECTED RCV WORD LENGTH.
;    2077  2     !--
;    2078  2
;    2079  2     TEMP3 = 0;
;    2080  2
;    2081  2     IF GET_BIT [ CSR, LB ] GTRU ZERO
;    2082  2       THEN
;    2083  2         TEMP3 = .RCV_D_LIST [ STWD1 ] AND RHL_MASK;                 ! 0'003400'
;    2084  2
;    2085  2     IF ( .CSR_WORD AND %O'01' ) EQLU ZERO
;    2086  2       THEN
;    2087  3         TEMP3 = .TEMP3 + ( .RCV_D_LIST [ STWD2 ] AND RLL_MASK )     ! 0'000377'
;    2088  2       ELSE
;    2089  2         TEMP3 = 6;
;    2090  2
;    2091  2     IF .TEMP3 NEQU .RBUF_LENGTH
;    2092  2       THEN
;    2093  3         BEGIN
;    2094  3           ERR_FLAG = ONE;
;    2095  3           CSR_WORD = GET_BIT [ CSR_ALL ];
;    2096  3           PRINTB ( MSG59 );
;    2097  3           PRINTB ( MSG17, .TEMP3, .RBUF_LENGTH );
;    2098  3           ERRDF ( 0011, MSG00, ERROR$REPORT );
;    2099  2         END;
;    2100  2
;    2101  2     INCR INDEX FROM 0 TO .TEMP3 - 1 DO
;    2102  3       BEGIN
;    2103  3         IF .RCV_D_LIST [ STWD1 ] EQLU NEWB
;    2104  3           THEN
;    2105  3             RCV_BUFFER [ .INDEX ] = ZERO;
;    2106  3
;    2107  3         IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;    2108  3           THEN
;    2109  3             IF .RCV_D_LIST [ LONGP ] EQLU ONE
;    2110  3               THEN
```

```
;    2111  4                BEGIN
;    2112  4                  TEMP5 = .INDEX;
;    2113  4                  EXITLOOP;
;    2114  4                END
;    2115  3              ELSE
;    2116  4                BEGIN
;    2117  4                  ERR_FLAG = ONE;
;    2118  4                  CSR_WORD = GET_BIT [ CSR_ALL ];
;    2119  4                  PRINTB ( MSG59 );
;    2120  4                  PRINTB ( MSG51 );
;    2121  4                  PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ],  .INDEX );
;    2122  4                  ERRDF ( 0012, MSG00, ERROR$REPORT );
;    2123  3                END;
;    2124  2        END;
;    2125  1      END;


                                                .SBTTL  COMPARE.PACKETS GLOBAL ROUTINE - COMPARE_PACKETS ( )
000000  004137  000000G                 COMPARE.PACKETS::
                                                JSR     R1,$SAVE2                       ;                                       2060
000004  024646                                  CMP     -(SP),-(SP)
000006  005037  000000G                         CLR     TEMP3                           ;                                       2079
000012  013700  000000G                         MOV     REG.ADR,R0                      ;                                       2081
000016  016046  000016                          MOV     16(R0),-(SP)                    ; *,TMP.LOCATION
000022  032716  001400                          BIT     #1400,(SP)                      ; *,TMP.LOCATION
000026  001406                                  BEQ     1$
000030  013737  000010G 000000G                 MOV     RCV.D.LIST+10,TEMP3             ;                                       2083
000036  042737  174377  000000G                 BIC     #174377,TEMP3
000044  032737  000001  000000G         1$:     BIT     #1,CSR.WORD                     ;                                       2085
000052  001006                                  BNE     2$
000054  005001                                  CLR     R1                              ;                                       2087
000056  153701  000012G                         BISB    RCV.D.LIST+12,R1
000062  060137  000000G                         ADD     R1,TEMP3
000066  000403                                  BR      3$                              ;                                       2085
000070  012737  000006  000000G         2$:     MOV     #6,TEMP3                        ;                                       2089
000076  023737  000000G 000000G         3$:     CMP     TEMP3,RBUF.LENGTH               ;                                       2091
000104  001437                                  BEQ     4$
000106  012737  000001  000000G                 MOV     #1,ERR.FLAG                     ;                                       2094
000114  016066  000016  000002                  MOV     16(R0),2(SP)                    ; *,TMP.LOCATION                        2095
000122  016637  000002  000000G                 MOV     2(SP),CSR.WORD                  ; TMP.LOCATION,*
000130  012746  000000G                         MOV     #MSG59,-(SP)                    ;                                       2096
000134  012746  000001                          MOV     #1,-(SP)
000140  010600                                  MOV     SP,R0                           ; SP,*
000142  104414                                  TRAP    14
000144  013716  000000G                         MOV     RBUF.LENGTH,(SP)                ;                                       2097
000150  013746  000000G                         MOV     TEMP3,-(SP)
000154  012746  000000G                         MOV     #MSG17,-(SP)
000160  012746  000003                          MOV     #3,-(SP)
000164  010600                                  MOV     SP,R0                           ; SP,*
000166  104414                                  TRAP    14
000170  104455                                  TRAP    55                              ;                                       2098
000172  000013                                  .WORD   13
000174  000000G                                 .WORD   MSG00
```

```
000176  000000'                                .WORD   ERROR$REPORT
000200  062706  000012                        ADD     #12,SP                          ;                                       2093
000204  013702  000000G                 4$:    MOV     TEMP3,R2                        ;                                       2101
000210  005001                                CLR     R1                              ; INDEX
000212  000474                                BR      9$
000214  023727  000010G 100000          5$:    CMP     RCV.D.LIST+10,#-100000          ;                                       2103
000222  001002                                BNE     6$
000224  105061  000000G                       CLRB    RCV.BUFFER(R1)                  ; *(INDEX)                              2105
000230  126161  000000G 000000G         6$:    CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                    2107
000236  001461                                BEQ     8$
000240  032737  040000  000010G               BIT     #40000,RCV.D.LIST+10            ;                                       2109
000246  001403                                BEQ     7$
000250  010137  000000G                       MOV     R1,TEMP5                        ; INDEX,*                               2112
000254  000455                                BR      10$                             ;                                       2111
000256  012737  000001  000000G         7$:    MOV     #1,ERR.FLAG                     ;                                       2117
000264  013700  000000G                       MOV     REG.ADR,R0                      ;                                       2118
000270  016066  000016  000004               MOV     16(R0),4(SP)                    ; *,TMP.LOCATION
000276  016637  000004  000000G               MOV     4(SP),CSR.WORD                  ; TMP.LOCATION,*
000304  012746  000000G                       MOV     #MSG59,-(SP)                    ;                                       2119
000310  012746  000001                        MOV     #1,-(SP)
000314  010600                                MOV     SP,R0                           ; SP,*
000316  104414                                TRAP    14
000320  012716  000000G                       MOV     #MSG51,(SP)                     ;                                       2120
000324  012746  000001                        MOV     #1,-(SP)
000330  010600                                MOV     SP,R0                           ; SP,*
000332  104414                                TRAP    14
000334  010116                                MOV     R1,(SP)                         ; INDEX,*                               2121
000336  005046                                CLR     -(SP)
000340  116116  000000G                       MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
000344  005046                                CLR     -(SP)
000346  116116  000000G                       MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
000352  012746  000000G                       MOV     #MSG50,-(SP)
000356  012746  000004                        MOV     #4,-(SP)
000362  010600                                MOV     SP,R0                           ; SP,*
000364  104414                                TRAP    14
000366  104455                                TRAP    55                              ;                                       2122
000370  000014                                .WORD   14
000372  000000G                               .WORD   MSG0C
000374  000000'                               .WORD   ERROR$REPORT
000376  062706  000016                        ADD     #16,SP                          ;                                       2116
000402  005201                          8$:    INC     R1                              ; INDEX                                 2101
000404  020102                          9$:    CMP     R1,R2                           ; INDEX,*
000406  002702                                BLT     5$
000410  062706  000006                 10$:    ADD     #6,SP                           ;                                       2060
000414  000207                                RTS     PC

; Routine Size:  135 words,    Routine Base:  AC$CODE$ + 261F
; Maximum stack depth per invocation:  15 words


;   2126  1
;   2127  1
```

```
;   2128  1     %SBTTL 'GLOBAL ROUTINE - SET_RDESCR_LIST ( P1, P2)'
;   2129  1
;   2130  1     GLOBAL ROUTINE SET_RDESCR_LIST ( P1, P2 ) : NOVALUE  =
;   2131  1
;   2132  1     !++
;   2133  1     !
;   2134  1     !  GLOBAL ROUTINE :     SET_RDESCR_LIST
;   2135  1     !
;   2136  1     !  DESCRIPTION:
;   2137  1     !
;   2138  1     !       This routine initializes receive descriptor list.
;   2139  1     !
;   2140  1     !  INPUT PARAMETERS:
;   2141  1     !
;   2142  1     !       P1 - expected Ethernet packet length in words
;   2143  1     !       P2 - expected RCV Descriptor List settings
;   2144  1     !
;   2145  1     !--
;   2146  1
;   2147  2     BEGIN
;   2148  2
;   2149  2       RCV_D_LIST [ FLGWD ]  = NEWB;
;   2150  2       RCV_D_LIST [ DBITS ]  = .P2;
;   2151  2       RCV_D_LIST [ LOADR ]  = RCV_BUFFER;
;   2152  2       RCV_D_LIST [ TWDL  ]  = .P1;
;   2153  2       RCV_D_LIST [ STWD1 ]  = 0;
;   2154  2       RCV_D_LIST [ STWD2 ]  = 0;
;   2155  2       RCV_D_LIST [ DLINK ]  = V;
;   2156  2       RCV_D_LIST [ BSTAT ]  = F;
;   2157  2
;   2158  1     END;


                                        .SBTTL  SET.RDESCR.LIST GLOBAL ROUTINE - SET_RDESCR_LIST ( P1, P2)
000000  012737  100000  000000G  SET.RDESCR.LIST::
                                        MOV     #-100000,RCV.D.LIST             ;                       2149
000006  016637  000002  000002G         MOV     2(SP),RCV.D.LIST+2              ; P2,*                  2150
000014  012737  000000G 000004G         MOV     #RCV.BUFFER,RCV.D.LIST+4        ;                       2151
000022  016637  000004  000006G         MOV     4(SP),RCV.D.LIST+6              ; P1,*                  2152
000030  005037  000010G                 CLR     RCV.D.LIST+10                   ;                       2153
000034  005037  000012G                 CLR     RCV.D.LIST+12                   ;                       2154
000040  012737  100000  000014G         MOV     #-100000,RCV.D.LIST+14          ;                       2155
000046  012737  020000  000016G         MOV     #20000,RCV.D.LIST+16            ;                       2156
000054  000207                         RTS     PC                              ;                       2130

; Routine Size:  23 words,      Routine Base:  AC$CODE$ + 3234
; Maximum stack depth per invocation:  0 words


;   2159  1
```

```
;    2160  1     %SBTTL 'GLOBAL ROUTINE - SET_XDESCR_LIST ( P1, P2 )'
;    2161  1
;    2162  1     GLOBAL ROUTINE SET_XDESCR_LIST ( P1, P2 ) : NOVALUE  =
;    2163  1
;    2164  1     !++
;    2165  1     !
;    2166  1     !  GLOBAL ROUTINE :     SET_XDESCR_LIST
;    2167  1     !
;    2168  1     !  DESCRIPTION:
;    2169  1     !
;    2170  1     !       This routine initializes transmit descriptor list.
;    2171  1     !
;    2172  1     !  INPUT PARAMETERS:
;    2173  1     !
;    2174  1     !       P1 - expected Ethernet packet length in words
;    2175  1     !       P2 - expected XMIT Descriptor List settings
;    2176  1     !
;    2177  1     !--
;    2178  1          ←
;    2179  2     BEGIN
;    2180  2
;    2181  2       XMIT_D_LIST [ FLGWD ] = NEWB;
;    2182  2       XMIT_D_LIST [ DBITS ] = .P2;
;    2183  2       XMIT_D_LIST [ LOADR ] = XMIT_BUFFER;
;    2184  2       XMIT_D_LIST [ TWDL  ] = .P1;
;    2185  2       XMIT_D_LIST [ STWD1 ] = 0;
;    2186  2       XMIT_D_LIST [ STWD2 ] = 0;
;    2187  2       XMIT_D_LIST [ DLINK ] = V;
;    2188  2       XMIT_D_LIST [ BSTAT ] = E;
;    2189  2
;    2190  1     END;


                                        .SBTTL  SET.XDESCR.LIST GLOBAL ROUTINE - SET_XDESCR_LIST ( P1, P2 )
000000  012737  100000  000000G  SET.XDESCR.LIST::
                                        MOV     #-100000,XMIT.D.LIST            ;                                   2181
000006  016637  000002  000002G         MOV     2(SP),XMIT.D.LIST+2             ; P2,*                              2182
000014  012737  000000G 000004G         MOV     #XMIT.BUFFER,XMIT.D.LIST+4      ;                                   2183
000022  016637  000004  000006G         MOV     4(SP),XMIT.D.LIST+6             ; P1,*                              2184
000030  005037  000010G                 CLR     XMIT.D.LIST+10                  ;                                   2185
000034  005037  000012G                 CLR     XMIT.D.LIST+12                  ;                                   2186
000040  012737  100000  000014G         MOV     #-100000,XMIT.D.LIST+14         ;                                   2187
000046  012737  020000  000016G         MOV     #20000,XMIT.D.LIST+16           ;                                   2188
000054  000207                         RTS     PC                              ;                                   2162

; Routine Size:  23 words,      Routine Base:  AC$CODE$ + 3312
; Maximum stack depth per invocation:  0 words


;   2191  1
```

```
;    2192  1     %SBTTL 'GLOBAL ROUTINE - WALKING_BIT ( P1, P2, P3 )'
;    2193  1
;    2194  1     GLOBAL ROUTINE WALKING_BIT ( P1, P2, P3 ) : NOVALUE  =
;    2195  1
;    2196  1     !++
;    2197  1     !
;    2198  1     !  GLOBAL ROUTINE :     WALKING_BIT
;    2199  1     !
;    2200  1     !  DESCRIPTION:
;    2201  1     !
;    2202  1     !       This routine sets bit to 0 or 1 in a specified bit position of the
;    2203  1     !       Ethernet Station Address. For example,
;    2204  1     !
;    2205  1     !       if
;    2206  1     !               .P1 = 0  and  .P2 = 15    .P3 = 5
;    2207  1     !       then
;    2208  1     !               Ethernet Station Address  =  FF-FF-FF-FF-7F-FF
;    2209  1     !
;    2210  1     !  INPUT PARAMETERS:
;    2211  1     !
;    2212  1     !       P1 - bit ( 0 or 1 )
;    2213  1     !       P2 - bit position from base address
;    2214  1     !       P3 - # of bytes to be tested using this pattern
;    2215  1     !
;    2216  1     !--
;    2217  1
;    2218  2     BEGIN
;    2219  2
;    2220  2     SELECTONE .P2 OF
;    2221  2       SET
;    2222  2       [  0 TO  7 ]:
;    2223  2                   TEMP1 = 0;
;    2224  2       [  8 TO ( .P3 + 1 ) * 8 ]:
;    2225  2                   TEMP1 = .P2 / 8;
;    2226  2       TES;
;    2227  2
;    2228  2     TEMP2 = .P2 MOD 8;
;    2229  2
;    2230  2     IF .P1 EQLU ZERO
;    2231  2       THEN
;    2232  3         BEGIN
;    2233  3           TBYTE1 = %B'00000000';
;    2234  3           SELECTONE .TEMP2 OF
;    2235  3             SET
;    2236  3             [ 0 ]:  TBYTE3 = %O'001';
;    2237  3             [ 1 ]:  TBYTE3 = %O'002';
;    2238  3             [ 2 ]:  TBYTE3 = %O'004';
;    2239  3             [ 3 ]:  TBYTE3 = %O'010';
;    2210  3             [ 4 ]:  TBYTE3 = %O'020';
;    2241  3             [ 5 ]:  TBYTE3 = %O'040';
;    2242  3             [ 6 ]:  TBYTE3 = %O'100';
;    2243  3             [ 7 ]:  TBYTE3 = %O'200';
;    2244  3             TES;
```

```
;    2245  3          END
;    2246  2        ELSE
;    2247  3          BEGIN
;    2248  3            TBYTE1 = %B'11111111';
;    2249  3            SELECTONE .TEMP2 OF
;    2250  3              SET
;    2251  3              [ 0 ]:  TBYTE3 = %O'376';
;    2252  3              [ 1 ]:  TBYTE3 = %O'375';
;    2253  3              [ 2 ]:  TBYTE3 = %O'373';
;    2254  3              [ 3 ]:  TBYTE3 = %C'367';
;    2255  3              [ 4 ]:  TBYTE3 = %O'357';
;    2256  3              [ 5 ]:  TBYTE3 = %O'337';
;    2257  3              [ 6 ]:  TBYTE3 = %O'277';
;    2258  3              [ 7 ]:  TBYTE3 = %O'177';
;    2259  3              TES;
;    2260  2          END;
;    2261  2
;    2262  2      INCR INDEX FROM 0 TO .P3 DO
;    2263  2          TARGET_ADR  [ .INDEX ] = .TBYTE1;
;    2264  2
;    2265  2      TEMP3 = .P3 - .TEMP1;
;    2266  2      TARGET_ADR  [ .TEMP3 ] = .TBYTE3;
;    2267  2
;    2268  1      END;


                                        .SBTTL  WALKING.BIT GLOBAL ROUTINE - WALKING_BIT ( P1, P2, P3 )
000000  004137  000000G          WALKING.BIT::
                                        JSR     R1,$SAVE2                       ;                       2194
000004  016602  000012                  MOV     12(SP),R2                       ; P2,*                  2220
000010  002406                          BLT     1$                              ;                       2222
000012  020227  000007                  CMP     R2,#7
000016  003003                          BGT     1$
000020  005037  000000G                 CLR     TEMP1                           ;                       2223
000024  000421                          BR      2$                              ;                       2220
000026  020227  000010          1$:     CMP     R2,#10                          ;                       2224
000032  002416                          BLT     2$
000034  016600  000010                  MOV     10(SP),R0                       ; P3,*
000040  072027  000003                  ASH     #3,R0
000044  062700  000010                  ADD     #10,R0
000050  020200                          CMP     R2,R0
000052  003006                          BGT     2$
000054  010201                          MOV     R2,R1                           ;                       2225
000056  006700                          SXT     R0
000060  071027  000010                  DIV     #10,R0
000064  010037  000000G                 MOV     R0,TEMP1
000070  010201          2$:             MOV     R2,R1                           ;                       2228
000072  006700                          SXT     R0
000074  071027  000010                  DIV     #10,R0
000100  010137  000000G                 MOV     R1,TEMP2
000104  010100                          MOV     R1,R0                           ; TEMP2,*               2234
000106  005766  000014                  TST     14(SP)                          ; P1                    2230
000112  001071                          BNE     10$
```

```
000114  005037  000000G                        CLR     TBYTE1                          ;                       2233
000120  005700                                  TST     R0                              ;                       2236
000122  001004                                  BNE     3$
000124  012737  000001  000000G                MOV     #1,TBYTE3
000132  000552                                  BR      18$                             ;                       2234
000134  020027  000001                3$:       CMP     R0,#1                           ;                       2237
000140  001004                                  BNE     4$
000142  012737  000002  000000G                MOV     #2,TBYTE3
000150  000543                                  BR      18$                             ;                       2234
000152  020027  000002                4$:       CMP     R0,#2                           ;                       2238
000156  001004                                  BNE     5$
000160  012737  000004  000000G                MOV     #4,TBYTE3
000166  000534                                  BR      18$                             ;                       2234
000170  020027  000003                5$:       CMP     R0,#3                           ;                       2239
000174  001004                                  BNE     6$
000176  012737  000010  000000G                MOV     #10,TBYTE3
000204  000525                                  BR      18$                             ;                       2234
000206  020027  000004                6$:       CMP     R0,#4                           ;                       2240
000212  001004                                  BNE     7$
000214  012737  000020  000000G                MOV     #20,TBYTE3
000222  000516                                  BR      18$                             ;                       2234
000224  020027  000005                7$:       CMP     R0,#5                           ;                       2241
000230  001004                                  BNE     8$
000232  012737  000040  000000G                MOV     #40,TBYTE3
000240  000507                                  BR      18$                             ;                       2234
000242  020027  000006                8$:       CMP     R0,#6                           ;                       2242
000246  001004                                  BNE     9$
000250  012737  000100  000000G                MOV     #100,TBYTE3
000256  000500                                  BR      18$                             ;                       2234
000260  020027  000007                9$:       CMP     R0,#7                           ;                       2243
000264  001075                                  BNE     18$
000266  012737  000200  000000G                MOV     #200,TBYTE3
000274  000471                                  BR      18$                             ;                       2234
000276  012737  000377  000000G        10$:     MOV     #377,TBYTE1                     ;                       2248
000304  005700                                  TST     R0                              ;                       2251
000306  001004                                  BNE     11$
000310  012737  000376  000000G                MOV     #376,TBYTE3
000316  000460                                  BR      18$                             ;                       2249
000320  020027  000001                11$:      CMP     R0,#1                           ;                       2252
000324  001004                                  BNE     12$
000326  012737  000375  000000G                MOV     #375,TBYTE3
000334  000451                                  BR      18$                             ;                       2249
000336  020027  000002                12$:      CMP     R0,#2                           ;                       2253
000342  001004                                  BNE     13$
000344  012737  000373  000000G                MOV     #373,TBYTE3
000352  000442                                  BR      18$                             ;                       2249
000354  020027  000003                13$:      CMP     R0,#3                           ;                       2254
000360  001004                                  BNE     14$
000362  012737  000367  000000G                MOV     #367,TBYTE3
000370  000433                                  BR      18$                             ;                       2249
000372  020027  000004                14$:      CMP     R0,#4                           ;                       2255
000376  001004                                  BNE     15$
000400  012737  000357  000000G                MOV     #357,TBYTE3
```

```
000406  000424                              BR      18$                             ;                                       2249
000410  020027  000005                 15$:  CMP     R0,#5                           ;                                       2256
000414  001004                              BNE     16$
000416  012737  000337  000000G             MOV     #337,TBYTE3
000424  000415                              BR      18$                             ;                                       2249
000426  020027  000006                 16$:  CMP     R0,#6                           ;                                       2257
000432  001004                              BNE     17$
000434  012737  000277  000000G             MOV     #277,TBYTE3
000442  000406                              BR      18$                             ;                                       2249
000444  020027  000007                 17$:  CMP     R0,#7                           ;                                       2258
000450  001003                              BNE     18$
000452  012737  000177  000000G             MOV     #177,TBYTE3
000460  005000                         18$:  CLR     R0                              ; INDEX                                 2262
000462  000404                              BR      20$
000464  113760  000000G 000000G        19$:  MOVB    TBYTE1,TARGET.ADR(R0)           ; *,*(INDEX)                            2263
000472  005200                              INC     R0                              ; INDEX                                 2262
000474  020066  000010                 20$:  CMP     R0,10(SP)                       ; INDEX,P3
000500  003771                              BLE     19$
000502  016637  000010  000000G             MOV     10(SP),TEMP3                    ; P3,*                                  2265
000510  163737  000000G 000000G             SUB     TEMP1,TEMP3
000516  013700  000000G                      MOV     TEMP3,R0                        ;                                       2266
000522  113760  000000G 000000G             MOVB    TBYTE3,TARGET.ADR(R0)
000530  000207                              RTS     PC                              ;                                       2194

; Routine Size:  173 words,    Routine Base:  AC$CODE$ + 3370
; Maximum stack depth per invocation:  4 words


;   2269  1
```

```
;    2270  1     %SBTTL 'GLOBAL ROUTINE - WRT_STATION_ADR ( P1, P2 )'
;    2271  1
;    2272  1     GLOBAL ROUTINE WRT_STATION_ADR ( P1, P2 ). NOVALUE  =
;    2273  1
;    2274  1     !++
;    2275  1     !
;    2276  1     !  GLOBAL ROUTINE :     WRT_STATION_ADR
;    2277  1     !
;    2278  1     !  DESCRIPTION:
;    2279  1     !
;    2280  1     !       This routine writes Station Address to XMIT_BUFFER.
;    2281  1     !
;    2282  1     !  INPUT PARAMETERS:
;    2283  1     !
;    2284  1     !       P1 - Ethernet Station Address index (1:14) in Station Address RAM
;    2285  1     !       P2 - Ethernet Station Address index ( 0:19 ) in the TARGET_ADR table
;    2286  1     !
;    2287  1     !--
;    2288  1
;    2289  2     BEGIN
;    2290  2
;    2291  2     TEMP1 = .P2 * 6;
;    2292  2
;    2293  2     SELECTONE .P1 OF
;    2294  2       SET
;    2295  2       [  0 TO  7 ]:
;    2296  2                       TEMP2 = .P1;
;    2297  2       [  8 TO 14 ]:
;    2298  2                       TEMP2 = .P1 + 57;
;    2299  2       TES;
;    2300  2
;    2301  2     IF .TEMP2 EQLU ZERO
;    2302  2       THEN
;    2303  2         INCR INDEX FROM 0 TO 5 DO
;    2304  3           BEGIN
;    2305  3             XMIT_BUFFER [ .INDEX ] = .TARGET_ADR [ .INDEX + .TEMP1 ];
;    2306  3           END
;    2307  2       ELSE
;    2308  2         INCR INDEX FROM 0 TO 5 DO
;    2309  3           BEGIN
;    2310  3             TEMP3 = .INDEX * 8 + .TEMP2;
;    2311  3             XMIT_BUFFER [ .TEMP3 ] = .TARGET_ADR [ .INDEX + .TEMP1 ];
;    2312  2           END;
;    2313  1     END;


                                        .SBTTL  WRT.STATION.ADR GLOBAL ROUTINE   WRT_STATION_ADR ( P1, P2 )
000000  004137  000000G          WRT.STATION.ADR::
                                        JSR     R1,$SAVE3                       ;                                2272
000004  016601  000012                  MOV     12(SP),R1                       ; P2,*                           2291
000010  070127  000006                  MUL     #6,R1
000014  010137  000000G                 MOV     R1,TEMP1
000020  016600  000014                  MOV     14(SP),R0                       ; P1,*                           2293
```

```
ZQNA4           CZQNADO DEQNA FUNCTIONAL TEST                    14-Mar-1985 13:18:55    VAX-11 Bliss-16 V4.1-582              SEQ 0258
V01.0           GLOBAL ROUTINE - WRT_STATION_ADR ( P1, P2 )      14-Mar-1985 13:06:01    DISK$USER2:[MARSHALL.DEQNA]ZQNA4.BLI;4  Page  38
                                                                                                                                  (19)

000024  002406                                  BLT     1$                              ;                                       2295
000026  020027  000007                          CMP     R0,#7
000032  003003                                  BGT     1$
000034  010037  000000G                         MOV     R0,TEMP2                        ;                                       2296
000040  000413                                  BR      2$                              ;                                       2293
000042  020027  000010              1$:         CMP     R0,#10                          ;                                       2297
000046  002410                                  BLT     2$
000050  020027  000016                          CMP     R0,#16
000054  003005                                  BGT     2$
000056  010037  000000G                         MOV     R0,TEMP2                        ;                                       2298
000062  062737  000071  000000G                 ADD     #71,TEMP2
000070  013703  000000G              2$:         MOV     TEMP2,R3                        ;                                       2301
000074  001014                                  BNE     4$
000076  005000                                  CLR     R0                              ; INDEX                                 2303
000100  010001              3$:                 MOV     R0,R1                           ; INDEX,*                               2305
000102  063701  000000G                         ADD     TEMP1,R1
000106  116160  000000G 000000G                 MOVB    TARGET.ADR(R1),XMIT.BUFFER(R0)  ; *,*(INDEX)
000114  005200                                  INC     R0                              ; INDEX                                 2303
000116  020027  000005                          CMP     R0,#5                           ; INDEX,*
000122  003766                                  BLE     3$
000124  000207                                  RTS     PC                              ;                                       2301
000126  005002              4$:                 CLR     R2                              ; INDEX                                 2308
000130  010200              5$:                 MOV     R2,R0                           ; INDEX,*                               2310
000132  072027  000003                          ASH     #3,R0
000136  060300                                  ADD     R3,R0
000140  010037  000000G                         MOV     R0,TEMP3
000144  010201                                  MOV     R2,R1                           ; INDEX,*                               2311
000146  063701  000000G                         ADD     TEMP1,R1
000152  116160  000000G 000000G                 MOVB    TARGET.ADR(R1),XMIT.BUFFER(R0)
000160  005202                                  INC     R2                              ; INDEX                                 2308
000162  020227  000005                          CMP     R2,#5                           ; INDEX,*
000166  003760                                  BLE     5$
000170  000207                                  RTS     PC                              ;                                       2272

; Routine Size:  61 words,      Routine Base:  AC$CODE$ + 4122
; Maximum stack depth per invocation:  5 words


;   2314  1
```

```
;    2315  1     %SBTTL 'GLOBAL ROUTINE - PREP_FOR_SETUP ( ) '
;    2316  1
;    2317  1     GLOBAL ROUTINE PREP_FOR_SETUP : NOVALUE  =
;    2318  1
;    2319  1     !++
;    2320  1     !
;    2321  1     !  GLOBAL ROUTINE :     PREP_FOR_SETUP
;    2322  1     !
;    2323  1     !  DESCRIPTION:
;    2324  1     !
;    2325  1     !       This routine retrives Ethernet Station Address from the Ethernet's
;    2326  1     !       Station Address PROM, saves copy of Ethernet Station Address PROM
;    2327  1     !       in the TARGET_ADR vector, initializes transmit and receive buffers
;    2328  1     !       to zero and finally sets buffer length to select promiscuous mode.
;    2329  1     !
;    2330  1     !  INPUT PARAMETERS:
;    2331  1     !
;    2332  1     !       none
;    2333  1     !--
;    2334  1
;    2335  2     BEGIN
;    2336  2
;    2337  2      !++
;    2338  2      !  RETRIEVE ETHERNET PHYSICAL STATION ADDRESS AND SAVE A COPY OF IT IN THE
;    2339  2      !  'TARGET_ADR' VECTOR.
;    2340  2      !--
;    2341  2
;    2342  2      INCR INDEX FROM 0 TO 5 DO
;    2343  3        BEGIN
;    2344  3          TBYTE1 = .REG_ADR [ .INDEX, ST_ADDR ];
;    2345  3          TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX ] = .TBYTE1;
;    2346  2        END;
;    2347  2
;    2348  2       CLR_BUFFERS ( 256 );
;    2349  2
;    2350  1     END;


                                        .SBTTL  PREP.FOR.SETUP GLOBAL ROUTINE - PREP_FOR_SETUP ( )
000000  010146                  PREP.FOR.SETUP::
                                        MOV     R1,-(SP)                        ;                                       2317
000002  005746                         TST     -(SP)
000004  005001                         CLR     R1                              ; INDEX                                 2342
000006  010100                  1$:     MOV     R1,R0                           ; INDEX,*                               2344
000010  006300                         ASL     R0
000012  063700  000000G                ADD     REG.ADR,R0
000016  011016                         MOV     (R0),(SP)                       ; *,TMP.LOCATION
000020  005037  000000G                CLR     TBYTE1
000024  111637  000000G                MOVB    (SP),TBYTE1
000030  111661  000162G                MOVB    (SP),TARGET.ADR+162(R1)         ; *,*(INDEX)                            2345
000034  005201                         INC     R1                              ; INDEX                                 2342
000036  020127  000005                CMP     R1,#5                           ; INDEX,*
000042  003761                         BLE     1$
```

```
000044  012746  000400                          MOV     #400,-(SP)                      ;                                       2348
000050  004737  001234'                        JSR     PC,CLR.BUFFERS
000054  022626                                  CMP     (SP)+,(SP)+                     ;                                       2317
000056  012601                                  MOV     (SP)+,R1
000060  000207                                  RTS     PC

; Routine Size:  25 words,      Routine Base:  AC$CODE$ + 4314
; Maximum stack depth per invocation:  4 words


;   2351  1
;   2352  1
;   2353  1
```

```
;    2354  1     %SBTTL 'GLOBAL ROUTINE - FORM_HEX_ADR ( P3 ) '
;    2355  1
;    2356  1     GLOBAL ROUTINE FORM_HEX_ADR ( P3 ) : NOVALUE  =
;    2357  1
;    2358  1     !++
;    2359  1     !
;    2360  1     !  GLOBAL ROUTINE :     FORM_HEX_ADR
;    2361  1     !
;    2362  1     !  DESCRIPTION:
;    2363  1     !
;    2364  1     !       This routine retrives Ethernet Station Address from the Ethernet's
;    2365  1     !       Station Address PROM, saves its copy in the TARGET_ADR vector.
;    2366  1     !
;    2367  1     !  INPUT PARAMETERS:
;    2368  1     !
;    2369  1     !       P3 - Index to Station Address in the TARGET_ADR vector
;    2370  1     !--
;    2371  1
;    2372  2     BEGIN
;    2373  2
;    2374  2       !++
;    2375  2       !  RETRIEVE ETHERNET PHYSICAL STATION ADDRESS AND SAVE A COPY OF IT IN THE
;    2376  2       !  'TARGET_ADR' AND 'STATION_ADR' VECTORS.
;    2377  2       !--
;    2378  2
;    2379  2       IF .P3 EQLU ZERO
;    2380  2         THEN
;    2381  2           TEMP5 = 0
;    2382  2         ELSE
;    2383  2           TEMP5 = .P3 * 6;
;    2384  2
;    2385  2       INCR INDEX5 FROM 0 TO 5 DO
;    2386  3         BEGIN
;    2387  3           TBYTE1 = .REG_ADR [ .INDEX5, ST_ADDR ];
;    2388  3           TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX5 ] = .TBYTE1;
;    2389  2         END;
;    2390  2
;    2391  2       COUNTER = ZERO;
;    2392  2
;    2393  2       INCR INDEX5 FROM 0 TO 5 BY 2 DO
;    2394  3         BEGIN
;    2395  3           TEMP1  = .TARGET_ADR [ .TEMP5 + .INDEX5 ];
;    2396  3           TEMP1  = .TEMP1 ^ 8;
;    2397  3           TEMP2 =  .TARGET_ADR [ .TEMP5 + .INDEX5 + 1 ];
;    2398  3           STATION_ADR [ .COUNTER ] = .TEMP1 OR ( .TEMP2 AND %O'000377' );
;    2399  3           COUNTER = .COUNTER + 1;
;    2400  2         END;
;    2401  2
;    2402  2       !++
;    2403  2       !  PRINT ETHERNET STATION ADDRESS ON THE CONSOLE
;    2404  2       !--
;    2405  2
;    2406  2       COUNTER  = 18;
```

```
;    2407  2        PHYS_ADR [ 0  ] = %C'%';
;    2408  2        PHYS_ADR [ 1  ] = %C'A';
;    2409  2        PHYS_ADR [ 19 ] = %C' ';
;    2410  2        PHYS_ADR [ 20 ] = %C'%';
;    2411  2        PHYS_ADR [ 21 ] = %C'N';
;    2412  2
;    2413  2        DECR INDEX1 FROM 2 TO 0 DO
;    2414  3          BEGIN
;    2415  3            TEMP3 = .STATION_ADR [ .INDEX1 ];
;    2416  3            INCR INDEX2 FROM 0 TO 1 DO
;    2417  4              BEGIN
;    2418  4                INCR INDEX3 FROM 0 TO 1 DO
;    2419  5                  BEGIN
;    2420  5                    TEMP1 = .TEMP3 AND %X'F';
;    2421  5                    IF .TEMP1 LEQU %DECIMAL'9'
;    2422  5                      THEN
;    2423  5                        TBYTE1 = %C'0' + .TEMP1
;    2424  5                      ELSE
;    2425  5                        TBYTE1 = %C'A' + ( .TEMP1 - %DECIMAL'10' );
;    2426  5                    PHYS_ADR [ .COUNTER ] = .TBYTE1;
;    2427  5                    COUNTER = .COUNTER - 1;
;    2428  5                    TEMP3 = .TEMP3 ^ ( -4 );
;    2429  4                  END;
;    2430  4
;    2431  4                IF .COUNTER GTRU 2
;    2432  4                  THEN
;    2433  4                    PHYS_ADR [ .COUNTER ] = %C'-';
;    2434  4
;    2435  4                COUNTER = .COUNTER - 1;
;    2436  4
;    2437  3              END;
;    2438  2          END;
;    2439  2
;    2440  1        END;


                                        .SBTTL  FORM.HEX.ADR GLOBAL ROUTINE - FORM_HEX_ADR ( P3 )
000000  004137  000000G          FORM.HEX.ADR::
                                        JSR     R1,$SAVE3                       ;                                       2356
000004  005746                         TST     -(SP)
000006  016600  000014                 MOV     14(SP),R0                       ; P3,*                                  2379
000012  001003                         BNE     1$
000014  005037  000000G                CLR     TEMP5                           ;                                       2381
000020  000405                         BR      2$                              ;                                       2379
000022  010001                 1$:     MOV     R0,R1                           ;                                       2383
000024  070127  000006                 MUL     #6,R1
000030  010137  000000G                MOV     R1,TEMP5
000034  005000                 2$:     CLR     R0                              ; INDEX5                                2385
000036  010001                 3$:     MOV     R0,R1                           ; INDEX5,*                              2387
000040  006301                         ASL     R1
000042  063701  000000G                ADD     REG.ADR,R1
000046  011116                         MOV     (R1),(SP)                       ; *,TMP.LOCATION
000050  005037  000000G                CLR     TBYTE1
```

```
000054  111637  000000G                        MOVB    (SP),TBYTE1
000060  111660  000162G                        MOVB    (SP),TARGET.ADR+162(R0)         ; *,*(INDEX5)                   2388
000064  005200                                 INC     R0                              ; INDEX5                        2385
000066  020027  000005                         CMP     R0,#5                           ; INDEX5,*
000072  003761                                 BLE     3$
000074  005037  000000G                        CLR     COUNTER                         ;                               2391
000100  005002                                 CLR     R2                              ; INDEX5                        2393
000102  010201                          4$:    MOV     R2,R1                           ; INDEX5,*                      2395
000104  063701  000000G                        ADD     TEMP5,R1
000110  116137  000000G 000000G               MOVB    TARGET.ADR(R1),TEMP1
000116  105037  000001G                        CLRB    TEMP1+1
000122  013700  000000G                        MOV     TEMP1,R0                        ;                               2396
000126  072027  000010                         ASH     #10,R0
000132  010037  000000G                        MOV     R0,TEMP1
000136  116137  000001G 000000G               MOVB    TARGET.ADR+1(R1),TEMP2          ;                               2397
000144  105037  000001G                        CLRB    TEMP2+1
000150  013701  000000G                        MOV     COUNTER,R1                      ;                               2398
000154  006301                                 ASL     R1
000156  005000                                 CLR     R0
000160  153700  000000G                        BISB    TEMP2,R0
000164  053700  000000G                        BIS     TEMP1,R0
000170  010061  000000G                        MOV     R0,STATION.ADR(R1)
000174  005237  000000G                        INC     COUNTER                         ;                               2399
000200  062702  000002                         ADD     #2,R2                           ; *,INDEX5                      2393
000204  020227  000005                         CMP     R2,#5                           ; INDEX5,*
000210  003734                                 BLE     4$
000212  012737  000022  000000G               MOV     #22,COUNTER                     ;                               2406
000220  112737  000045  000000G               MOVB    #45,PHYS.ADR                    ;                               2407
000226  112737  000101  000001G               MOVB    #101,PHYS.ADR+1                 ;                               2408
000234  112737  000040  000023G               MOVB    #40,PHYS.ADR+23                 ;                               2409
000242  112737  000045  000024G               MOVB    #45,PHYS.ADR+24                 ;                               2410
000250  112737  000116  000025G               MOVB    #116,PHYS.ADR+25                ;                               2411
000256  012701  000004                         MOV     #4,R1                           ; *,INDEX1                      2413
000262  016137  000000G 000000G          5$:    MOV     STATION.ADR(R1),TEMP3           ; *(INDEX1),*                   2415
000270  012703  000002                         MOV     #2,R3                           ; *,INDEX2                      2416
000274  012702  000002                  6$:    MOV     #2,R2                           ; *,INDEX3                      2418
000300  013737  000000G 000000G          7$:    MOV     TEMP3,TEMP1                     ;                               2420
000306  042737  177760  000000G               BIC     #177760,TEMP1
000314  013700  000000G                        MOV     TEMP1,R0                        ;                               2421
000320  020027  000011                         CMP     R0,#11
000324  101006                                 BHI     8$
000326  010037  000000G                        MOV     R0,TBYTE1                       ;                               2423
000332  062737  000060  000000G               ADD     #60,TBYTE1
000340  000405                                 BR      9$                              ;                               2421
000342  010037  000000G                  8$:    MOV     R0,TBYTE1                       ;                               2425
000346  062737  000067  000000G               ADD     #67,TBYTE1
000354  013700  000000G                  9$:    MOV     COUNTER,R0                      ;                               2426
000360  113760  000000G 000000G               MOVB    TBYTE1,PHYS.ADR(R0)
000366  005337  000000G                        DEC     COUNTER                         ;                               2427
000372  013700  000000G                        MOV     TEMP3,R0                        ;                               2428
000376  072027  177774                         ASH     #-4,R0
000402  010037  000000G                        MOV     R0,TEMP3
000406  077244                                 SOB     R2,7$                           ; INDEX3,*                      2418
```

```
000410  013702  000000G                         MOV     COUNTER,R2                      ;                                       2431
000414  020227  000002                          CMP     R2,#2
000420  101403                                  BLOS    10$
000422  112762  000055  000000G                 MOVB    #55,PHYS.ADR(R2)                ;                                       2433
000430  005337  000000G             10$:        DEC     COUNTER                         ;                                       2435
000434  077361                                  SOB     R3,6$                           ; INDEX2,*                              2416
000436  162701  000002                          SUB     #2,R1                           ; *,INDEX1                              2413
000442  100307                                  BPL     5$
000444  005726                                  TST     (SP)+                           ;                                       2356
000446  000207                                  RTS     PC

; Routine Size:  148 words,     Routine Base:  AC$CODE$ + 4376
; Maximum stack depth per invocation:  6 words


;   2441  1
;   2442  1
```

```
;    2443  1      %SBTTL 'GLOBAL ROUTINE - XMIT_SETUP_PACKET ( P1 )'
;    2444  1
;    2445  1      GLOBAL ROUTINE XMIT_SETUP_PACKET ( P1 ) : NOVALUE  =
;    2446  1
;    2447  1      !++
;    2448  1      !
;    2449  1      !  GLOBAL ROUTINE :     XMIT_SETUP_PACKET
;    2450  1      !
;    2451  1      !  DESCRIPTION:
;    2452  1      !
;    2453  1      !       This routine initializes descriptor lists to transmit and receive
;    2454  1      !       unchained Setup loopback packet. After loopback packet has been
;    2455  1      !       received DEQNA CSR, transmit and receive status registers are
;    2456  1      !       checked for proper status. Finally, transmit and receive packets
;    2457  1      !       are compared to verify that they are identical.
;    2458  1      !
;    2459  1      !       XMIT_D_LIST [ 0 ] = NEWB            RCV_D_LIST [ 0 ] = NEWB
;    2460  1      !       XMIT_D_LIST [ 1 ] = VSE             RCV_D_LIST [ 1 ] = VE
;    2461  1      !       XMIT_D_LIST [ 2 ] = XMIT_BUFFER     RCV_D_LIST [ 2 ] = RCV_BUFFER
;    2462  1      !       XMIT_D_LIST [ 3 ] = .XBUF_LENGTH    RCV_D_LIST [ 3 ] = .XBUF_LENGTH
;    2463  1      !       XMIT_D_LIST [ 4 ] = 0               RCV_D_LIST [ 4 ] = 0
;    2464  1      !       XMIT_D_LIST [ 5 ] = 0               RCV_D_LIST [ 5 ] = 0
;    2465  1      !       XMIT_D_LIST [ 6 ] = V               RCV_D_LIST [ 6 ] = V
;    2466  1      !       XMIT_D_LIST [ 7 ] = E               RCV_D_LIST [ 7 ] = E
;    2467  1      !
;    2468  1      !
;    2469  1      !  INPUT PARAMETERS:
;    2470  1      !
;    2471  1      !       P1 - transmit buffer length in bytes
;    2472  1      !
;    2473  1      !--
;    2474  1
;    2475  2      BEGIN
;    2476  2
;    2477  2      CLR_DESCR ( );
;    2478  2      RBUF_LENGTH = .P1;
;    2479  2      XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    2480  2      SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2481  2      SET_XDESCR_LIST ( .XBUF_LENGTH, VSE );
;    2482  2
;    2483  2      IF .P1 EQLU A_MODE
;    2484  2        THEN
;    2485  3          BEGIN
;    2486  3            XBUF_LENGTH = - ( ( .RBUF_LENGTH  + -1 ) + 1 );
;    2487  3            SET_XDESCR_LIST ( .XBUF_LENGTH, VSEL );
;    2488  3            SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2489  2          END;
;    2490  2
;    2491  2        XMIT_AND_RCV_PACKET ( );
;    2492  2
;    2493  2        !++
;    2494  2        !  COMPARE STATUS REGISTERS TO EXPECTED VALUES
;    2495  2        !--
```

```
;   2496  2
;   2497  2       CHK_RIXI_STATUS ( ONE );
;   2498  2       CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );          ! 0'100220', 0'100220'
;   2499  2       CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS );      ! 0'140000', 0'020000'
;   2500  2
;   2501  2       TEMP1 = XWD12_STATUS;                              ! 0'000400'
;   2502  2       IF .XMIT_D_LIST [ STE16 ]
;   2503  2         THEN
;   2504  2            TEMP1 = %O'002400';
;   2505  2       CHK_XMIT_STATUS ( XFLG_STATUS, .TEMP1 );           ! 0'140000', ???????
;   2506  2
;   2507  2       COMPARE_PACKETS ( );
;   2508  2
;   2509  1     END;


                                        .SBTTL  XMIT.SETUP.PACKET GLOBAL ROUTINE   XMIT_SETUP_PACKET ( P1 )
000000  004737  001206'          XMIT.SETUP.PACKET::
                                        JSR     PC,CLR.DESCR                    ;                                       2477
000004  016637  000002  000000G         MOV     2(SP),RBUF.LENGTH               ; P1,*                                  2478
000012  016600  000002                  MOV     2(SP),R0                        ; RBUF.LENGTH,*                         2479
000016  006200                          ASR     R0
000020  005400                          NEG     R0
000022  010037  000000G                 MOV     R0,XBUF.LENGTH
000026  010046                          MOV     R0,-(SP)                        ; XBUF.LENGTH,*                         2480
000030  012746  120000                  MOV     #-60000,-(SP)
000034  004737  003234'                 JSR     PC,SET.RDESCR.LIST
000040  013716  000000G                 MOV     XBUF.LENGTH,(SP)                ;                                       2481
000044  012746  130000                  MOV     #-50000,-(SP)
000050  004737  003312'                 JSR     PC,SET.XDESCR.LIST
000054  026627  000010  000201         CMP     10(SP),#201                     ; P1,*                                  2483
000062  001023                          BNE     1$
000064  013700  000000G                 MOV     RBUF.LENGTH,R0                  ;                                       2486
000070  006200                          ASR     R0
000072  005200                          INC     R0
000074  005400                          NEG     R0
000076  010037  000000G                 MOV     R0,XBUF.LENGTH
000102  010016                          MOV     R0,(SP)                         ; XBUF.LENGTH,*                         2487
000104  012746  130200                  MOV     #-47600,-(SP)
000110  004737  003312'                 JSR     PC,SET.XDESCR.LIST
000114  013716  000000G                 MOV     XBUF.LENGTH,(SP)                ;                                       2488
000120  012746  120000                  MOV     #-60000,-(SP)
000124  004737  003234'                 JSR     PC,SET.RDESCR.LIST
000130  022626                          CMP     (SP)+,(SP)+                     ;                                       2485
000132  004737  000000V         1$:     JSR     PC,XMIT.AND.RCV.PACKET          ;                                       2491
000136  012716  000001                  MOV     #1,(SP)                         ;                                       2497
000142  004737  001262'                 JSR     PC,CHK.RIXI.STATUS
000146  012716  100220                  MOV     #-77560,(SP)                    ;                                       2498
000152  011646                          MOV     (SP),-(SP)
000154  004737  001646'                 JSR     PC,CHK.CSR.STATUS
000160  012716  140000                  MOV     #-40000,(SP)                    ;                                       2499
000164  012746  020000                  MOV     #20000,-(SP)
000170  004737  002336'                 JSR     PC,CHK.RCV.STATUS
```

```
000174  012737  000400  000000G                 MOV     #400,TEMP1                      ;                                       2501
000202  032737  002000  000010G                 BIT     #2000,XMIT.D.LIST+10            ;                                       2502
000210  001403                                  BEQ     2$
000212  012737  002400  000000G                 MOV     #2400,TEMP1                     ;                                       2504
000220  012716  140000                  2$:     MOV     #-40000,(SP)                    ;                                       2505
000224  013746  000000G                         MOV     TEMP1,-(SP)
000230  004737  002040'                         JSR     PC,CHK.XMIT.STATUS
000234  004737  002616'                         JSR     PC,COMPARE.PACKETS              ;                                       2507
000240  062706  000014                          ADD     #14,SP                          ;                                       2475
000244  000207                                  RTS     PC                              ;                                       2445

; Routine Size:  83 words,      Routine Base:  AC$CODE$ + 5046
; Maximum stack depth per invocation:  7 words


;   2510  1
;   2511  1
```

```
;    2512  1      %SBTTL 'GLOBAL ROUTINE - SEND_ELOOP_PACKET ( P3 ) '
;    2513  1
;    2514  1      GLOBAL ROUTINE SEND_ELOOP_PACKET ( P3 ) : NOVALUE  =
;    2515  1
;    2516  1      !++
;    2517  1      !
;    2518  1      !  GLOBAL ROUTINE :     SEND_ELOOP_PACKET
;    2519  1      !
;    2520  1      !  DESCRIPTION:
;    2521  1      !
;    2522  1      !       This routine initializes transmit and receive descriptor lists and
;    2523  1      !       then initiates transmissin of a loopback packet. After
;    2524  1      !       loopback packet is received DEQNA CSR, transmit and receive status r
;    2525  1      !       egisters are checked for proper status. Finally, transmit and receive
;    2526  1      !       packets are compared to verify that they are identical.
;    2527  1      !
;    2528  1      !       XMIT_D_LIST [ 0 ] = NEWB            RCV_D_LIST [ 0 ] = NEWB
;    2529  1      !       XMIT_D_LIST [ 1 ] = VE              RCV_D_LIST [ 1 ] = VE
;    2530  1      !       XMIT_D_LIST [ 2 ] = XMIT_BUFFER     RCV_D_LIST [ 2 ] = RCV_BUFFER
;    2531  1      !       XMIT_D_LIST [ 3 ] = .XBUF_LENGTH    RCV_D_LIST [ 3 ] = .XBUF_LENGTH
;    2532  1      !       XMIT_D_LIST [ 4 ] = 0               RCV_D_LIST [ 4 ] = 0
;    2533  1      !       XMIT_D_LIST [ 5 ] = 0               RCV_D_LIST [ 5 ] = 0
;    2534  1      !       XMIT_D_LIST [ 6 ] = V               RCV_D_LIST [ 6 ] = V
;    2535  1      !       XMIT_D_LIST [ 7 ] = E               RCV_D_LIST [ 7 ] = E
;    2536  1      !
;    2537  1      !
;    2538  1      !  INPUT PARAMETERS:
;    2539  1      !
;    2540  1      !       P3 -
;    2541  1      !--
;    2542  1
;    2543  2      BEGIN
;    2544  2
;    2545  2        PUT_BIT ( CSR, LB, INX_LOOPBACK );
;    2546  2        XMIT_AND_RCV_PACKET ( );
;    2547  2
;    2548  2        !++
;    2549  2        !  COMPARE STATUS REGISTERS TO EXPECTED VALUES
;    2550  2        !--
;    2551  2
;    2552  2        CHK_RIXI_STATUS ( ZERO );
;    2553  2        CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );            ! 0'100220', 0'100220'
;    2554  2        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );       ! 0'140000', 0'000400'
;    2555  2
;    2556  2        IF .P3 EQLU ZERO
;    2557  2          THEN
;    2558  3            BEGIN
;    2559  3              CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS );  ! 0'140000', 0'020000'
;    2560  3            END
;    2561  2          ELSE
;    2562  3            BEGIN
;    2563  3              TEMP1 = RWD14_STATUS;                          ! 0'060000'
;    2564  3              IF .RCV_D_LIST [ STWD1 ] AND %0'070001' EQLU %0'070001'
```

```
;   2565  3               THEN
;   2566  3                 TEMP1 = %O'070001';
;   2567  3             CHK_RCV_STATUS  ( RFLG_STATUS, .TEMP1 );        ! O'140000', ??????
;   2568  2         END;
;   2569  1     END;


                                                .SBTTL  SEND.ELOOP.PACKET GLOBAL ROUTINE - SEND_ELOOP_PACKET ( P3 )
000000  013700  000000G                 SEND.ELOOP.PACKET::
                                                MOV     REG.ADR,R0                      ;                               2545
000004  042760  001400  000016                  BIC     #1400,16(R0)
000012  052760  001000  000016                  BIS     #1000,16(R0)
000020  004737  000000V                         JSR     PC,XMIT.AND.RCV.PACKET          ;                               2546
000024  005046                                  CLR     -(SP)                           ;                               2552
000026  004737  001262'                         JSR     PC,CHK.RIXI.STATUS
000032  012716  100220                          MOV     #-77560,(SP)                    ;                               2553
000036  011646                                  MOV     (SP),-(SP)
000040  004737  001646'                         JSR     PC,CHK.CSR.STATUS
000044  012716  140000                          MOV     #-40000,(SP)                    ;                               2554
000050  012746  000400                          MOV     #400,-(SP)
000054  004737  002040'                         JSR     PC,CHK.XMIT.STATUS
000060  005766  000010                          TST     10(SP)                          ; P3                            2556
000064  001005                                  BNE     1$
000066  012716  140000                          MOV     #-40000,(SP)                    ;                               2559
000072  012746  020000                          MOV     #20000,-(SP)
000076  000416                                  BR      3$
000100  012737  060000  000000G         1$:     MOV     #60000,TEMP1                    ;                               2563
000106  032737  000001  000010G                 BIT     #1,RCV.D.LIST+10                ;                               2564
000114  001403                                  BEQ     2$
000116  012737  070001  000000G                 MOV     #70001,TEMP1                    ;                               2566
000124  012716  140000                  2$:     MOV     #-40000,(SP)                    ;                               2567
000130  013746  000000G                         MOV     TEMP1,-(SP)
000134  004737  002336'                 3$:     JSR     PC,CHK.RCV.STATUS
000140  062706  000010                          ADD     #10,SP                          ;                               2543
000144  000207                                  RTS     PC                              ;                               2514

; Routine Size:  51 words,      Routine Base:  AC$CODE$ + 5314
; Maximum stack depth per invocation:  5 words


;   2570  1
```

```
;  2571  1     %SBTTL 'GLOBAL ROUTINE - SEND_TEST_PACKET '
;  2572  1
;  2573  1     GLOBAL ROUTINE SEND_TEST_PACKET : NOVALUE  =
;  2574  1
;  2575  1     !++
;  2576  1     !
;  2577  1     !  GLOBAL ROUTINE :     SEND_TEST_PACKET
;  2578  1     !
;  2579  1     !  DESCRIPTION:
;  2580  1     !
;  2581  1     !       This routine initializes transmit and receive descriptor lists and
;  2582  1     !       then initiates transmission of an external loopback packet.
;  2583  1     !
;  2584  1     !       XMIT_D_LIST [ 0 ] = NEWB            RCV_D_LIST [ 0 ] = NEWB
;  2585  1     !       XMIT_D_LIST [ 1 ] = VE              RCV_D_LIST [ 1 ] = VE
;  2586  1     !       XMIT_D_LIST [ 2 ] = XMIT_BUFFER     RCV_D_LIST [ 2 ] = RCV_BUFFER
;  2587  1     !       XMIT_D_LIST [ 3 ] = .XBUF_LENGTH    RCV_D_LIST [ 3 ] = .XBUF_LENGTH
;  2588  1     !       XMIT_D_LIST [ 4 ] = 0               RCV_D_LIST [ 4 ] = 0
;  2589  1     !       XMIT_D_LIST [ 5 ] = 0               RCV_D_LIST [ 5 ] = 0
;  2590  1     !       XMIT_D_LIST [ 6 ] = V               RCV_D_LIST [ 6 ] = V
;  2591  1     !       XMIT_D_LIST [ 7 ] = E               RCV_D_LIST [ 7 ] = E
;  2592  1     !
;  2593  1     !
;  2594  1     !  INPUT PARAMETERS:
;  2595  1     !
;  2596  1     !       None
;  2597  1     !--
;  2598  1
;  2599  2     BEGIN
;  2600  2
;  2601  2       !++
;  2602  2       !  WRITE ETHERNET STATION ADDRESS AND DATA PATTERN INTO THE TRANSMIT BUFFER
;  2603  2       !--
;  2604  2
;  2605  2       RESET_DEQNA ( );
;  2606  2
;  2607  2       INCR INDEX FROM 0 TO 5 DO
;  2608  3         BEGIN
;  2609  3           XMIT_BUFFER [ .INDEX ]       = .TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX ];
;  2610  3           XMIT_BUFFER [ .INDEX + 6 ]   = .TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX ];
;  2611  2         END;
;  2612  2
;  2613  2       XMIT_BUFFER [ PKT_TYPE ]     = LPB_PKT;
;  2614  2       XMIT_BUFFER [ PKT_TYPE + 1 ] = SKIP_CNT;
;  2615  2       XMIT_BUFFER [ PKT_TYPE + 2 ] = RFC;
;  2616  2
;  2617  2       !++
;  2618  2       !  CONVERT SETUP PACKET SIZE FROM BYTE COUNT TO WORD COUNT AND SET UP
;  2619  2       !  DESCRIPTOR LISTS
;  2620  2       !--
;  2621  2
;  2622  2       RBUF_LENGTH = PKT_LENGTH + 14;
;  2623  2       XBUF_LENGTH =   ( .RBUF_LENGTH ^ -1 );
```

```
;   2624  2
;   2625  2        SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;   2626  2        SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;   2627  2
;   2628  2        !++
;   2629  2        !  SET DEQNA TO EXTERNAL LOOPBACK MODE AND SEND LOOPBACK PACKET
;   2630  2        !--
;   2631  2
;   2632  2        PUT_BIT ( CSR, LB, EXT_LOOPBACK );
;   2633  2        XMIT_AND_RCV_PACKET ( );
;   2634  2
;   2635  1     END;


                                                .SBTTL  SEND.TEST.PACKET GLOBAL ROUTINE   SEND_TEST PACKET
000000  004737  000324'                    SEND.TEST.PACKET::
                                                JSR     PC,RESET.DEQNA                  ;                                       2605
000004  005000                                  CLR     R0                              ; INDEX                                 2607
000006  116060  000162G 000000G          1$:     MOVB    TARGET ADR+162(R0),XMIT.BUFFER(R0) ;
                                                                                        ; *(INDEX),*(INDEX)                     2609
000014  116060  000162G 000006G                 MOVB    TARGET.ADR+162(R0),XMIT.BUFFER+6(R0) ;
                                                                                        ; *(INDEX),*(INDEX)                     2610
000022  005200                                  INC     R0                              ; INDEX                                 2607
000024  020027  000005                          CMP     R0,#5                           ; INDEX,*
000030  003766                                  BLE     1$
000032  112737  000220  000014G                MOVB    #220,XMIT.BUFFER+14             ;                                       2613
000040  105037  000015G                         CLRB    XMIT.BUFFER+15                  ;                                       2614
000044  112737  000001  000016G                MOVB    #1,XMIT.BUFFER+16               ;                                       2615
000052  012737  002752  000000G                MOV     #2752,RBUF.LENGTH               ;                                       2622
000060  012700  002752                          MOV     #2752,R0                        ;                                       2623
000064  006200                                  ASR     R0
000066  005400                                  NEG     R0
000070  010037  000000G                         MOV     R0,XBUF.LENGTH
000074  010046                                  MOV     R0,-(SP)                        ; XBUF.LENGTH,*                         2625
000076  012746  120000                          MOV     #-60000,-(SP)
000102  004737  003234'                         JSR     PC,SET.RDESCR.LIST
000106  013716  000000G                         MOV     XBUF.LENGTH,(SP)                ;                                       2626
000112  012746  120000                          MOV     #-60000,-(SP)
000116  004737  003312'                         JSR     PC,SET.XDESCR.LIST
000122  013700  000000G                         MOV     REG.ADR,R0                      ;                                       2632
000126  052760  001400  000016                 BIS     #1400,16(R0)
000134  004737  000000V                         JSR     PC,XMIT.AND.RCV.PACKET          ;                                       2633
000140  062706  000006                          ADD     #6,SP                           ;                                       2599
000144  000207                                  RTS     PC                              ;                                       2573

; Routine Size:  51 words,      Routine Base:  AC$CODE$ + 5462
; Maximum stack depth per invocation:  4 words


;   2636  1
```

```
;    2637  1     %SBTTL 'GLOBAL ROUTINE - XMIT_AND_RCV_PACKET
;    2638  1
;    2639  1     GLOBAL ROUTINE XMIT_AND_RCV PACKET : NOVALUE  =
;    2640  1
;    2641  1     !++
;    2642  1     !
;    2643  1     !  GLOBAL ROUTINE :     XMIT_AND_RCV_PACKET
;    2644  1     !
;    2645  1     !  DESCRIPTION:
;    2646  1     !
;    2647  1     !       This routine initiates transmit and receive operations.
;    2648  1     !
;    2649  1     !  INPUT PARAMETERS:
;    2650  1     !
;    2651  1     !
;    2652  1     !
;    2653  1     !
;    2654  1     !--
;    2655  1
;    2656  2     BEGIN
;    2657  2
;    2658  2       .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;    2659  2       .IOP_TABLE [ RHI_ADR ] = 0;
;    2660  2
;    2661  2       .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;    2662  2       .IOP_TABLE [ XHI_ADR ] = 0;
;    2663  2
;    2664  1     END;


                                            .SBTTL  XMIT.AND.RCV.PACKET GLOBAL ROUTINE - XMIT_AND RCV_PACKET
000000  012777  000000G 000004G         XMIT.AND.RCV.PACKET::
                                            MOV     #RCV.D.LIST,@IOP.TABLE+4        ;                                2658
000006  005077  000006G                     CLR     @IOP.TABLE+6                    ;                                2659
000012  012777  000000G 000010G             MOV     #XMIT.D.LIST,@IOP.TABLE+10      ;                                2661
000020  005077  000012G                     CLR     @IOP.TABLE+12                   ;                                2662
000024  000207                              RTS     PC                              ;                                2639

; Routine Size:  11 words,      Routine Base:  AC$CODE$ + 5630
; Maximum stack depth per invocation:  0 words


;    2665  1
;    2666  1
```

```
;    2667  1     %SBTTL 'GLOBAL ROUTINE - XMIT_ILOOP_PACKET ( P3 ) '
;    2668  1
;    2669  1     GLOBAL ROUTINE XMIT_ILOOP_PACKET ( P3 ) : NOVALUE  =
;    2670  1
;    2671  1     !++
;    2672  1     !
;    2673  1     !  GLOBAL ROUTINE :     XMIT_ILOOP_PACKET
;    2674  1     !
;    2675  1     !  DESCRIPTION:
;    2676  1     !
;    2677  1     !       This routine
;    2678  1     !
;    2679  1     !  INPUT PARAMETERS:
;    2680  1     !
;    2681  1     !       P3 - selector
;    2682  1     !
;    2683  1     !--
;    2684  1
;    2685  2     BEGIN
;    2686  2
;    2687  2       CLR_DESCR ( );
;    2688  2
;    2689  2       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2690  2       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    2691  2
;    2692  2       XMIT_AND_RCV_PACKET ( );
;    2693  2
;    2694  2       .IOP_TABLE [ CSR ] = EENABLE;
;    2695  2
;    2696  2       IF .P3 EQLU ONE
;    2697  2         THEN
;    2698  3           BEGIN
;    2699  3             CHK_RIXI_STATUS ( ONE );
;    2700  3             CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );        ! 0'100220', 0'100220'
;    2701  3             CHK_RCV_STATUS  ( RFLG_STATUS, RWD16_STATUS );   ! 0'140000', 0'044000'
;    2702  3           END
;    2703  2         ELSE
;    2704  3           BEGIN
;    2705  3             CHK_RIXI_STATUS ( ZERO );
;    2706  3             CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );        ! 0'100220', 0'100220'
;    2707  3             CHK_RCV_STATUS  ( RFLG_STATUS, RWD13_STATUS );   ! 0'140000', 0'000000'
;    2708  2           END;
;    2709  2
;    2710  2       CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );         ! 0'140000', 0'000400'
;    2711  2       COMPARE_PACKETS ( );
;    2712  2       .IOP_TABLE [ CSR ] = DISABLE;
;    2713  2
;    2714  1     END;


                                        .SBTTL  XMIT.ILOOP.PACKET GLOBAL ROUTINE - XMIT_ILOOP_PACKET ( P3 )
000000  004737  001206'            XMIT.ILOOP.PACKET::
                                        JSR     PC,CLR.DESCR                    ;                                       2687
```

```
000004  013746  000000G                        MOV     XBUF.LENGTH,-(SP)               ;                               2689
000010  012746  120000                         MOV     #-60000,-(SP)
000014  004737  003234'                       JSR     PC,SET.RDESCR.LIST
000020  013716  000000G                        MOV     XBUF.LENGTH,(SP)                ;                               2690
000024  012746  120000                         MOV     #-60000,-(SP)
000030  004737  003312'                       JSR     PC,SET.XDESCR.LIST
000034  004737  005630'                       JSR     PC,XMIT.AND.RCV.PACKET          ;                               2692
000040  012777  000001  000016G               MOV     #1,@IOP.TABLE+16                ;                               2694
000046  026627  000010  000001               CMP     10(SP),#1                       ; P3,*                          2696
000054  001016                               BNE     1$
000056  012716  000001                       MOV     #1,(SP)                         ;                               2699
000062  004737  001262'                       JSR     PC,CHK.RIXI.STATUS
000066  012716  100220                       MOV     #-77560,(SP)                    ;                               2700
000072  011646                               MOV     (SP),-(SP)
000074  004737  001646'                       JSR     PC,CHK.CSR.STATUS
000100  012716  140000                       MOV     #-40000,(SP)                    ;                               2701
000104  012746  044000                       MOV     #44000,-(SP)
000110  000413                               BR      2$
000112  005016                       1$:     CLR     (SP)                            ;                               2705
000114  004737  001262'                       JSR     PC,CHK.RIXI.STATUS
000120  012716  100220                       MOV     #-77560,(SP)                    ;                               2706
000124  011646                               MOV     (SP),-(SP)
000126  004737  001646'                       JSR     PC,CHK.CSR.STATUS
000132  012716  140000                       MOV     #-40000,(SP)                    ;                               2707
000136  005046                               CLR     -(SP)
000140  004737  002336'                       JSR     PC,CHK.RCV.STATUS
000144  012716  140000                       MOV     #-40000,(SP)                    ;                               2710
000150  012746  000400                       MOV     #400,-(SP)
000154  004737  002040'                       JSR     PC,CHK.XMIT.STATUS
000160  004737  002616'                       JSR     PC,COMPARE.PACKETS              ;                               2711
000164  005077  000016G                      CLR     @IOP.TABLE+16                   ;                               2712
000170  062706  000014                       ADD     #14,SP                          ;                               2685
000174  000207                               RTS     PC                              ;                               2669

; Routine Size:  63 words,      Routine Base:  AC$CODE$ + 5656
; Maximum stack depth per invocation:  7 words


;   2715  1
;   2716  1
```

```
;    2717  1     %SBTTL 'GLOBAL ROUTINE - TURN_OFF_LED ( P1 )'
;    2718  1
;    2719  1     GLOBAL ROUTINE TURN_OFF_LED ( P1 ) : NOVALUE  =
;    2720  1
;    2721  1     !++
;    2722  1     !
;    2723  1     !  GLOBAL ROUTINE :     TURN_OFF_LED
;    2724  1     !
;    2725  1     !  DESCRIPTION:
;    2726  1     !
;    2727  1     !       This routine
;    2728  1     !
;    2729  1     !  INPUT PARAMETERS:
;    2730  1     !
;    2731  1     !       P1 -
;    2732  1     !
;    2733  1     !--
;    2734  1
;    2735  2     BEGIN
;    2736  2
;    2737  2     PREP_FOR_SETUP ( );
;    2738  2
;    2739  2     INCR INDEX1 FROM 1 TO 14 DO
;    2740  2       WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;    2741  2
;    2742  2     XMIT_SETUP_PACKET ( .P1 );
;    2743  2
;    2744  2
;    2745  1     END;


                                        .SBTTL  TURN.OFF.LED GLOBAL ROUTINE - TURN_OFF_LED ( P1 )
000000  010146                  TURN.OFF.LED::
                                        MOV     R1,-(SP)                        ;                               2719
000002  004737  004314'                 JSR     PC,PREP.FOR.SETUP               ;                               2737
000006  012701  000001                  MOV     #1,R1                           ; *,INDEX1                      2739
000012  010146                  1$:     MOV     R1,-(SP)                        ; INDEX1,*                      2740
000014  012746  000023                  MOV     #23,-(SP)
000020  004737  004122'                 JSR     PC,WRT.STATION.ADR
000024  022626                          CMP     (SP)+,(SP)+
000026  005201                          INC     R1                              ; INDEX1                        2739
000030  020127  000016                  CMP     R1,#16                          ; INDEX1,*
000034  003766                          BLE     1$
000036  016646  000004                  MOV     4(SP),-(SP)                     ; P1,*                          2742
000042  004737  005046'                 JSR     PC,XMIT.SETUP.PACKET
000046  005726                          TST     (SP)+                           ;                               2735
000050  012601                          MOV     (SP)+,R1                        ;                               2719
000052  000207                          RTS     PC

; Routine Size:  22 words,      Routine Base:  AC$CODE$ + 6054
; Maximum stack depth per invocation:  4 words
```

```
;   2746  1
;   2747  1
;   2748  1     END
;   2749  0     ELUDOM


;                                   OTS external references
                                          .GLOBL  $SAVE3, $SAVE2


;                                   PSECT SUMMARY
;
;       Psect Name                  Words     Attributes
;        AC$CODE$                    1580       RO ,  I  ,  LCL,  REL,  CON



;                               Library Statistics
;
;                                          - ------ Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[MARSHALL.DEQNA]QNALIB.L16;15
;                                           223       133        59        14          00:00.1




;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZQNA4.BLI/LIST=ZQNA4.LIS/OBJECT=ZQNA4.OBJ/SOURCE=PAGE:53

; Size:         1580 code + 0 data words
; Run Time:        00:46.2
; Elapsed Time:    03:40.0
; Lines/CPU Min:    3567
; Lexemes/CPU-Min: 25672
; Memory Used:  235 pages
; Compilation Complete
```

```
;    0001  0     MODULE ZQNA5 (%TITLE 'CZQNADO DEQNA FUNCTIONAL TEST'
;    0002  0                   IDENT = 'V01.0',
;    0003  0                   ADDRESSING_MODE(ABSOLUTE)
;    0004  0                   ) =
;    0005  0     %SBTTL 'LAST ADDRESS AND SETUP SECTION'
;    0006  0
;    0007  1     BEGIN
;    0008  1
;    0009  1     LIBRARY 'QNALIB';                       ! QNALIB LIBRARY
;    0010  1     REQUIRE 'BLSMAC.REQ';                   ! DIAGNOSTIC SUPERVISOR LIBRARY
;    1500  1     !<BLF/NOFORMAT>
;    1501  1
```

```
                                                                                                                    SEQ 0278
ZQNA5           CZQNADO DEQNA FUNCTIONAL TEST                   14 Mar-1985 13:22:42    VAX-11 Bliss-16 V4.1-582          Page   2
V01.0           LAST ADDRESS AND SETUP SECTION                  14-Mar-1985 13:07:52    DISK$USER2:[MARSHALL.DEQNA]ZQNA5.BLI;4   (2)

;   1502  2     LASTAD
;   1503  2     BGNSETUP(1);                                    ! NUMBER OF P TABLES
; P 1504  2     BGNPTAB
; P 1505  2             %o'174440',%o'700'
;   1506  2     ENDPTAB
;   1507  1     ENDSETUP


                                            .TITLE  ZQNA5 CZQNADO DEQNA FUNCTIONAL TEST
                                            .IDENT  /V01.0/
                                            .ENABL  AMA


000000                                      .PSECT  $XYZ$,  RO
000000  000014'                    BL$LAS::.WORD   T$FREE
000002  000000C                             .WORD   <<T$FREE-<BL$LAS+4>>/2>
000004  000000                    P.AAA:  .WORD   0
000006  000002                            .WORD   2                                       ; Plit count word
000010  174440                    P.AAB:  .WORD   -3340
000012  000700                            .WORD   700
000014  000000                    T$FREE::.WORD   0


        000004'                    L$LAST==            BL$LAS+4
        000001                     T$PTHV==            1
        000004'                    $$LAS1=             P.AAA
        000010'                    $REM2=              P.AAB


                                            .SBTTL  $END.LINK LAST ADDRESS AND SETUP SECTION
000000  000207                    $END.LINK::
                                            RTS     PC                                      ;                                       1499

; Routine Size:  1 word,        Routine Base:  $XYZ$ + 0016
; Maximum stack depth per invocation:  0 words


;   1508  1
;   1509  1     END
;   1510  0     ELUDOM


;                                   PSECT SUMMARY
;
;       Psect Name                  Words     Attributes
;        $XYZ$                          8       RO ,  I  ,  LCL,  REL,  CON



;                               Library Statistics
;
```

```
;                                        -----  - Symbols --------      Pages      Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[MARSHALL.DEQNA]QNALIB.L16;15
;                                          223         3         1          14        00:00.1
```

```
;                                    COMMAND QUALIFIERS

;       BLISS/PDP11 ZQNA5.BLI/LIST=ZQNA5.LIS/OBJECT=ZQNA5.OBJ/SOURCE=PAGE:53

; Size:          1 code + 7 data words
; Run Time:        00:06.2
; Elapsed Time:    00:16.5
; Lines/CPU Min:   14707
; Lexemes/CPU-Min: 78262
; Memory Used:  98 pages
; Compilation Complete
```

```
;    0001  0     !++
;    0002  0     !
;    0003  0     !  DEFINE DATA STRUCTURES IN THIS SECTION
;    0004  0     !
;    0005  0     !--
;    0006  0
;    0007  0     STRUCTURE                                       ! DEFINE ACCESS ALGORITHM
;    0008  0        REG_STR [ O, P, S, E ]=
;    0009  1           BEGIN
;    0010  1              LOCAL TMP_LOCATION;
;    0011  1              TMP_LOCATION = .(REG_STR + %UPVAL * O) <O,%BPVAL,O>;
;    0012  1              TMP_LOCATION
;    0013  0           END < P, S, E >;
;    0014  0
;    0015  0
;    0016  0     STRUCTURE                                       ! DEFINE ACCESS ALGORITHM
;    0017  0        ADR_STR [ O, P, S, E ]=
;    0018  1           BEGIN
;    0019  1              LOCAL TMP_LOCATION;
;    0020  1              TMP_LOCATION = (ADR_STR + %UPVAL * O) <O,%BPVAL,O>;
;    0021  1              TMP_LOCATION
;    0022  0           END < P, S, E >;
;    0023  0
;    0024  0     STRUCTURE                                       ! DEFINE ACCESS ALGORITHM
;    0025  0        LBLOCK [ O, P, S, E, I ]=
;    0026  1           BEGIN
;    0027  1             CASE I FROM O TO 2 OF
;    0028  1                SET
;    0029  1                [ O ]:
;    0030  1                     ( LBLOCK + O * %UPVAL );
;    0031  1                [ 1 ]:
;    0032  1                     ( .LBLOCK + O * %UPVAL );
;    0033  1                [ 2 ]:
;    0034  1                     ( .LBLOCK + O * %UPVAL );
;    0035  1                TES;
;    0036  0            END < P, S, E >;
```

```
;    0037  0      !++
;    0038  0      !
;    0039  0      !   MACRO DEFINITIONS
;    0040  0      !
;    0041  0      !--
;    0042  0
;    0043  0      MACRO
;    0044  0
; M  0045  0          TST_BIT ( ADDR, EXPECTED ) =
; M  0046  0              ( IF ( .ADDR AND EXPECTED ) EQLU EXPECTED
; M  0047  0                THEN
; M  0048  0                   TRUE
; M  0049  0                ELSE
;    0050  0                   FALSE )%,
;    0051  0
;    0052  0
; M  0053  0          PUT_BIT ( OFFSET, POSITION, IMAGE ) =
; M  0054  0              BEGIN
; M  0055  0              ( .REG_ADR + %UPVAL * OFFSET )< %FIELDEXPAND ( POSITION ) > = IMAGE;
;    0056  0              END%,
;    0057  0
; M  0058  0          GET_STATION_ADR ( OFFSET, POSITION, IMAGE ) =
; M  0059  0              BEGIN
; M  0060  0              ( .STATION_ADR + OFFSET )< %FIELDEXPAND ( POSITION ) > = IMAGE;
;    0061  0              END%,
;    0062  0
;    0063  0
;    0064  0      !++
;    0065  0      !
;    0066  0      !  THIS MACRO GETS BITS SPECIFIED BY THE FIELD NAME " POSITION "
;    0067  0      !  AND MEMORY LOC SPECIFIED BY (.REG_ADR + %UPVAL * OFFSET)
;    0068  0      !
;    0069  0      !--
;    0070  0
; M  0071  0              GET_BIT ( OFFSET, POSITION ) =
; M  0072  0
;    0073  0                      .REG_ADR [ OFFSET, POSITION ] %;
;    0074  0
```

```
;  0075  0
;  0076  0
;  0077  0     !++
;  0078  0     !
;  0079  0     !   PROGRAM LITERALS
;  0080  0     !
;  0081  0     !--
;  0082  0
;  0083  0     LITERAL
;  0084  0
;  0085  0        NO              = 0,            !
;  0086  0        YES             = 1,            !
;  0087  0        FALSE           = 0,            !
;  0088  0        TRUE            = 1,            !
;  0089  0        ZERO            = 0,            !
;  0090  0        ONE             = 1,            !
;  0091  0        DISABLE         = 0,            !
;  0092  0        EENABLE         = 1,            !
;  0093  0
;  0094  0        P_CLOCK         = 1,            !
;  0095  0        L_CLOCK         = 1,            !
;  0096  0        NO_CLOCK        = 0,            !
;  0097  0        CLEAR_FLG       = 0,            !
;  0098  0        SET_FLG         = 1,            !
;  0099  0        PWR_DELAY       = 10000,        !
;  0100  0        M1_DELAY        = 10,           !
;  0101  0        M2_DELAY        = 20,           !
;  0102  0        M3_DELAY        = 30,           !
;  0103  0        M4_DELAY        = 40,           !
;  0104  0        M5_DELAY        = 50,           !
;  0105  0
;  0106  0        K               = 1024,         !
;  0107  0        TIME1_LIMIT     = 128,          ! DELAY - LOOP ITERATION COUNT
;  0108  0        TIME2_LIMIT     = 1 * K,        ! DELAY - LOOP ITERATION COUNT
;  0109  0        TIME3_LIMIT     = 1 * K,        ! DELAY - LOOP ITERATION COUNT
;  0110  0        TIME4_LIMIT     = 512,          ! DELAY - LOOP ITERATION COUNT
;  0111  0        TIME5_LIMIT     = 16 * K,       ! DELAY - 16K LOOP ITERATION COUNT
;  0112  0        TIME6_LIMIT     = 1,            ! DELAY - LOOP ITERATION COUNT
;  0113  0        TIME7_LIMIT     = 10,           ! DELAY - LOOP ITERATION COUNT
;  0114  0        TIME8_LIMIT     = 50,           ! DELAY - LOOP ITERATION COUNT
;  0115  0        TIME9_LIMIT     = 100,          ! DELAY - LOOP ITERATION COUNT
;  0116  0
;  0117  0        STEP1           = 2,            !
;  0118  0
;  0119  0        RLO_ADR         = 2,            !
;  0120  0        RHI_ADR         = 3,            !
;  0121  0        XLO_ADR         = 4,            !
;  0122  0        XHI_ADR         = 5,            !
;  0123  0        IOP_LO_ADR      = 2,            !
;  0124  0        IOP_HI_ADR      = 3,            !
;  0125  0        IOP_SIZE        = %O'16',       ! I/O PAGE REGISTER SIZE
;  0126  0        IOP_ADR         = 0,            ! OFFSET TO DEVICE ADDRESS
;  0127  0        IOP_VEC         = 2,            ! OFFSET TO DEVICE VECTOR ADDRESS
```

```
;   0128  0         IOP_BRL       = 4,            ! OFFSET TO DEVICE BR LEVEL
;   0129  0         INT_VEC       = 6,            !
;   0130  0
;   0131  0         CSR           = 7,            !
;   0132  0         WORD_LIMIT    = %O 177777',   !
;   0133  0
```

```
;  0134  0     !--
;  0135  0     !
;  0136  0     !  DESCRIPTOR LIST DEFINITIONS
;  0137  0     !
;  0138  0     !--
;  0139  0
;  0140  0         D_FLAG_WD     = 0,            ! STATUS WORD 0, FLAG WORD
;  0141  0         D_DESCR_BITS  = 1,            !
;  0142  0         D_HI_ADR      = 1,            !
;  0143  0         D_LO_ADR      = 2,            !
;  0144  0         D_WD_COUNT    = 3,            !
;  0145  0         D_WD1_STATUS  = 4,            !
;  0146  0         D_WD2_STATUS  = 5,            !
;  0147  0
;  0148  0         D1_OFFSET     = 18,           !
;  0149  0         D2_OFFSET     = 36,           !
;  0150  0
;  0151  0         T_SIZE        = 120,
;  0152  0         DESCR_SIZE    = 128,
;  0153  0         D_SIZE        = DESCR_SIZE / 2,
;  0154  0         BD_D_SIZE     = 16,
;  0155  0         BUF_SIZE      = 4096,
;  0156  0         B_SIZE        = BUF_SIZE / 2,
;  0157  0         SETUB_SIZE    = 256,
;  0158  0         BYTE_COUNT    = - ( BUF_SIZE / 4 ),
;  0159  0         PROM_SIZE     = 4096,
;  0160  0         CHSUM_OFFSET  = 6,
;  0161  0
;  0162  0         SA_RBL        = %O'177775',          ! STATION ADR RCV BUF LENGTH - 3 WDS
;  0163  0
;  0164  0         PKT_LENGTH      = 1500,              ! PACKET LENGTH
;  0165  0         MAX_LENGTH      = 1534,              ! PACKET LENGTH
;  0166  0         LEGAL_LENGTH    = 1514,              ! LEGAL PACKET LENGTH
;  0167  0         ILLEGAL_LENGTH  = 1536,              ! ILLEGAL PACKET LENGTH
;  0168  0         LPB_PKT         = %O'0220',          ! LOOPBACK PACKET
;  0169  0         PKT_TYPE        = 12,                ! PACKET TYPE
;  0170  0         SKIP_CNT        = 0,                 !
;  0171  0         RFC             = 1,                 !
;  0172  0         PKT_DATA        = 15,                !
;  0173  0         SHORTEST_PACKET = 60,                ! SHORTEST SETUP PACKET LENGTH
;  0174  0         LONGEST_PACKET  = 1514,              ! LONGEST SETUP PACKET LENGTH
;  0175  0         LSPL            = 1514,              ! LONGEST SETUP PACKET LENGTH
;  0176  0         PHA_INDEX       = 19,                ! PHYSICAL ADDRESS INDEX IN THE
;  0177  0                                              !  TARGET_ADR VECTOR
;  0178  0
;  0179  0         KB_VEC_LOC      = %O'000060',        ! INPUT CONSOLE TERMINAL VECTOR LOC
;  0180  0         PF_VEC_LOC      = %O'000024',        ! POWER FAIL VECTOR LOCATION
;  0181  0         CPU_LED         = %O'177524',        ! TURN OFF CPU LED LIT ON DCOK
;  0182  0         KB_ADDR         = %O'177560',        ! CONSOLE TERMINAL INPUT ADDRESS
;  0183  0         KB_ENABLE       = %O'000100',        ! ENABLE CONSOLE TERMINAL INPUT
;  0184  0
```

```
;  0185  0     !++
;  0186  0     !
;  0187  0     !  TRANSMIT, RECEIVE AND CSR STATUS AND MASK WORD DEFINITIONS
;  0188  0     !
;  0189  0     !--
;  0190  0
;  0191  0         CSR_STATUS      = %O'100220',       !
;  0192  0         CSR1_STATUS     = %O'000062',       !
;  0193  0         CSR2_STATUS     = %O'000060',       !
;  0194  0         CSR_MASK        = %O'100220',       !
;  0195  0         CSR1_MASK       = %O'010376',       !
;  0196  0         CSR2_MASK       = %O'167777',       ! TRANSCEIVER POWER ( XC - BIT 12 )
;  0197  0         CSR3_MASK       = %O'010000',       ! TRANSCEIVER POWER ( XC - BIT 12 )
;  0198  0
;  0199  0         PATRN1      = %O'001411',           ! CSR STATIC BITS
;  0200  0         PATRN2      = %O'001471',           ! CSR STATIC BITS
;  0201  0
;  0202  0         NXM_LO_ADR  = %O'160000',           ! NXM ADDRESS - LOW ORDER BITS
;  0203  0         NXM_HI_ADR  = %O'000077',           ! NXM ADDRESS - HIGH ORDER BITS
;  0204  0
;  0205  0         XFLG_MASK   = %O'140000',           ! TRANSMIT FLAG WORD MASK BITS
;  0206  0         X1_MASK     = %O'100000',           ! TRANSMIT STATUS WD 1 MASK BITS
;  0207  0         XWD1_MASK   = %O'157760',           ! TRANSMIT STATUS WD 1 MASK BITS
;  0208  0         XWD2_MASK   = %O'037777',           ! TRANSMIT STATUS WD 2 MASK BITS
;  0209  0         XFLG_STATUS = %O'140000',           ! EXPECTED TRANSMIT FLAG WORD
;  0210  0         XWD11_STATUS = %O'000000',          !
;  0211  0         XWD12_STATUS = %O'000400',          ! EXPECTED TRANSMIT STATUS WD 1
;  0212  0                                             !  BIT 8 IS SET IN INTERNAL LOOOPBACK MODES
;  0213  0                                             !  BIT 8 IS RESET IN EXTERNAL LOOOPBACK MODES
;  0214  0         XWD14_STATUS = %O'047600',          ! EXPECTED TRANSMIT STATUS WD 1
;  0215  0
;  0216  0         RFLG_MASK   = %O'140000',           ! RECEIVE FLAG WORD MASK BITS
;  0217  0         R1_MASK     = %O'100000',           ! RECEIVE STATUS WD 1 MASK BITS
;  0218  0         R2_MASK     = %O'174013',           ! RECEIVE STATUS WD 1 MASK BITS ! N.M. CHANGED FROM 174017 TO 174013
;  0219  0         RWD1_MASK   = %O'140000',           ! RECEIVE STATUS WD 1 MASK BITS
;  0220  0         RWD2_MASK   = %O'177417',           ! RECEIVE STATUS WD 1 MASK BITS
;  0221  0         RWD1_STATUS = %O'020000',           ! EXPECTED RECEIVE STATUS WD 1
;  0222  0         RWD11_STATUS = %O'100000',          ! EXPECTED RECEIVE STATUS WD 1
;  0223  0         RWD12_STATUS = %O'160000',          ! EXPECTED RECEIVE STATUS WD 1
;  0224  0         RWD13_STATUS = %O'000000',          ! EXPECTED RECEIVE STATUS WD 1
;  0225  0         RWD14_STATUS = %O'060000',          ! EXPECTED RECEIVE STATUS WD 1
;  0226  0         RWD15_STATUS = %O'000001',          ! EXPECTED RECEIVE STATUS WD 1
;  0227  0         RWD16_STATUS = %O'044000',          ! EXPECTED RECEIVE STATUS WD 1
;  0228  0
;  0229  0         RFLG_STATUS = %O'140000',           ! EXPECTED RECEIVE FLAG WORD
;  0230  0
;  0231  0
;  0232  0         RHL_MASK    = %O'003400',           ! RCV HIGH ORDER LENGTH BITS
;  0233  0         RLL_MASK    = %O'000377',           ! RCV LOW ORDER LENGTH BITS
```

```
;  0234  0     !++
;  0235  0     !
;  0236  0     !  BUFFER DESCRIPTOR / CHAIN DESCRIPTOR BIT DEFINITIONS
;  0237  0     !
;  0238  0     !--
;  0239  0
;  0240  0         V             = %O'100000',          ! VALID ADDRESS IF 1
;  0241  0         C             = %O'040000',          ! CHAIN ADDRESS IF 1
;  0242  0         E             = %O'020000',          ! END OF MESSAGE IF 1
;  0243  0         S             = %O'010000',          ! SETUP MODE PACKET IF 1
;  0244  0
;  0245  0         NEWB          = %O'100000',          ! BUFFER NOT USED IF 1
;  0246  0         LASTD         = %O'100000',          ! LAST DESCRIPTOR IN CHAIN
;  0247  0         VE            = %O'120000',          !
;  0248  0         VL            = %O'100200',          !
;  0249  0         VH            = %O'100100',          !
;  0250  0         VC            = %O'140000',          !
;  0251  0         VHL           = %O'100300',          !
;  0252  0         VSE           = %O'130000',          !
;  0253  0         VSEL          = %O'130200',          !
;  0254  0         VENXM         = %O'120077',          !
;  0255  0
;  0256  0         XLRL_SET      = %B'11',              ! XMIT AND RCV LISTS INVALID
;  0257  0         ILEL_SET      = %B'11',              ! INTERNAL AND EXTERNAL LOOPBACK BITS
;  0258  0         ILEL_CLR      = %B'00',              ! INTERNAL AND EXTERNAL LOOPBACK BITS
;  0259  0
;  0260  0         INT_LOOPBACK  = %B'00',              ! INTERNAL LOOPBACK MODE
;  0261  0         INX_LOOPBACK  = %B'10',              ! INTERNAL/EXTENDED LOOPBACK MODE
;  0262  0         EXT_LOOPBACK  = %B'11',              ! EXTERNAL LOOPBACK MODE
;  0263  0
;  0264  0         N_MODE        = %O'000200',          ! ENABLE NORMAL MODE OF OPERATION
;  0265  0         P_MODE        = %O'000202',          ! ENABLE PROMISCUOUS MODE OF OPERATION
;  0266  0         A_MODE        = %O'000201',          ! ENABLE ALL MULTICAST MODE OF OPERATION
;  0267  0         LED1          = %O'000204',          ! TURN OFF LED 1
;  0268  0         LED2          = %O'000210',          ! TURN OFF LED 2
;  0269  0         LED3          = %O'000214',          ! TURN OFF LED 3
;  0270  0
```

```
;    0271  0     !++
;    0272  0     !  STATION ADDRESS CONSTANTS
;    0273  0     !--
;    0274  0
;    0275  0        SADR1 = 0,                   ! HIGH STATION ADDRESS BITS
;    0276  0        SADR2 = 1,                   ! MIDDLE BITS
;    0277  0        SADR3 = 2,                   ! LOW STATION ADDRESS BITS
;    0278  0        CHSUM = 3,                   ! ACTUAL CHECKSUM INDEX
;    0279  0
;    0280  0     !++
;    0281  0     !   HARDWARE AND SOFTWARE P-TABLE EQUATES
;    0282  0     !--
;    0283  0
;    0284  0         SWP_SIZE     = 5,           ! SOFTWARE P-TABLE SIZE ( WORDS )
;    0285  0         HWP_SIZE     = 2,           ! HARDWARE P TABLE SIZE ( WORDS )
;    0286  0
;    0287  0
;    0288  0             SET_IT   = 1,
;    0289  0             CLR_IT   = 0;
;    0290  0
```

```
;    0291  0     !++
;    0292  0     !
;    0293  0     !   THE CONTROL AND STATUS REGISTER BIT DEFINITIONS
;    0294  0     !
;    0295  0     !--
;    0296  0
;    0297  0     FIELD
;    0298  0         IOP_FIELDS  =
;    0299  0             SET
;    0300  0             RE      = [ 0, 1, 0 ],  ! RECEIVER ENABLE          R/W ( ACTIVE HIGH )
;    0301  0             SR      = [ 1, 1, 0 ],  ! SOFTWARE RESET           R/W ( ACTIVE HIGH )
;    0302  0             NI      = [ 2, 1, 0 ],  ! NXM INTERRUPT            R   ( ACTIVE HIGH )
;    0303  0             BD      = [ 3, 1, 0 ],  ! BOOT/DIAGNOSTIC ROM      R/W ( ACTIVE HIGH )
;    0304  0             XL      = [ 4, 1, 0 ],  ! XMIT LIST INVALID        R   ( ACTIVE HIGH )
;    0305  0             RL      = [ 5, 1, 0 ],  ! RCV LIST INVALID         R   ( ACTIVE HIGH )
;    0306  0             IE      = [ 6, 1, 0 ],  ! INTERRUPT ENABLE         R/W ( ACTIVE HIGH )
;    0307  0             XI      = [ 7, 1, 0 ],  ! XMIT INTERRUPT REQUEST   R/W ( ACTIVE HIGH )
;    0308  0             IL      = [ 8, 1, 0 ],  ! INTERNAL LOOPBACK MODE   R/W ( ACTIVE LOW  )
;    0309  0             EL      = [ 9, 1, 0 ],  ! EXTERNAL LOOPBACK MODE   R/W ( ACTIVE HIGH )
;    0310  0             SE      = [10, 1, 0 ],  ! SANITY TIMER ENABLE      R/W ( ACTIVE HIGH )
;    0311  0             X1      = [11, 1, 0 ],  ! RESERVED, UNUSABLE
;    0312  0             XC      = [12, 1, 0 ],  ! TRANSCEIVER PWR          R   ( ACTIVE HIGH )
;    0313  0             CA      = [13, 1, 0 ],  ! CARRIER                  R   ( ACTIVE HIGH )
;    0314  0             X2      = [14, 1, 0 ],  ! RESERVED, UNUSABLE
;    0315  0             RI      = [15, 1, 0 ],  ! RCV INTERRUPT REQUEST    R/W ( ACTIVE HIGH )
;    0316  0
;    0317  0             LB      = [ 8, 2, 0 ],          ! LOOPBACK BITS
;    0318  0             XLRL    = [ 4, 2, 0 ],          ! XMIT AND RCV LISTS INVALID BITS
;    0319  0             ALL_BITS= [ 0,16, 0 ],          ! FETCH WHOLE WORD
;    0320  0
;    0321  0             LO_NIBBLE = [ 0,  0, 0 ],       !
;    0322  0             HI_NIBBLE = [ 0,  4, 0 ],       !
;    0323  0             LO_BYTE   = [ 0,  8, 0 ],       !
;    0324  0             HI_BYTE   = [ 0, 16, 0 ],       ! GET WORD, ALL BITS
;    0325  0             ST_ADDR   = [ 0,  8, 0 ],       ! STATION ADDRESS LOW BYTE
;    0326  0             ST_WORD   = [ 0, 16, 0 ],       ! GET WORD, ALL BITS
;    0327  0
;    0328  0             RCV_LO  = [ 2, 0, 16, 0 ],      ! RCV BUFFER DESCRIPTOR LIST LOW ADDRESS
;    0329  0             PCV_HI  = [ 3, 0, 8, 0 ],       ! RCV BUFFER DESCRIPTOR LIST HIGH ADDRESS
;    0330  0             XMIT_LO = [ 4, 0, 16, 0 ],      ! XMIT BUFFER DESCRIPTOR LIST LOW ADDRESS
;    0331  0             XMIT_HI = [ 5, 0, 8, 0 ],       ! XMIT BUFFER DESCRIPTOR LIST HIGH ADDRESS
;    0332  0             VEC_ADR = [ 2, 8, 0 ],          ! INTERRUPT VECTOR ADDRESS
;    0333  0             VEC_ALL = [ 6, 0, 16, 0 ],      ! INTERRUPT VECTOR ADDRESS
;    0334  0             CSR_ALL = [ 7, 0, 16, 0 ]       ! CONTROL AND STATUS REGISTER
;    0335  0             TES;
```

```
;    0336  0      !++
;    0337  0      !
;    0338  0      !  TRANSMIT AND RECEIVE DESCRIPTOR LIST FIELDS
;    0339  0      !
;    0340  0      !--
;    0341  0
;    0342  0      FIELD
;    0343  0          DL_FIELDS  =
;    0344  0              SET
;    0345  0              FLGWD   = [ 0, 0, 16, 0 ],      ! XMIT OF RCV FLAG WORD
;    0346  0
;    0347  0              DBITS   = [ 1, 0, 16, 0 ],      ! DESCRIPTOR BITS
;    0348  0              H_BIT   = [ 1,  6, 1, 0 ],      ! XMIT BUFFER BEGINS ON BYTE BOUNDARY
;    0349  0              L_BIT   = [ 1,  7, 1, 0 ],      ! XMIT BUFFER ENDS ON BYTE BOUNDARY
;    0350  0              S_BIT   = [ 1, 12, 1, 0 ],      ! SET-UP PACKET IF 1
;    0351  0              E_BIT   = [ 1, 13, 1, 0 ],      ! LAST DESCRIPTOR IN CHAIN ( END )
;    0352  0              C_BIT   = [ 1, 14, 1, 0 ],      ! DESCRIPTOR HAS CHAIN ADDRESS IF 1
;    0353  0              V_BIT   = [ 1, 15, 1, 0 ],      ! VALID ADDRESS IF 1
;    0354  0
;    0355  0              LOADR   = [ 2, 0, 16, 0 ],      ! LOW 16 BITS OF XMIT OR RCV BUFFER ADDRESS
;    0356  0
;    0357  0              TWDL    = [ 3, 0, 16, 0 ],      ! XMIT OR RCV PACKET WORD LENGTH
;    0358  0
;    0359  0              STWD1   = [ 4, 0, 16, 0 ],      ! XMIT OR RCV STATUS WORD 1
;    0360  0              OVF     = [ 4, 0, 1,  0 ],      ! FIFO BUFFER OVERFLOW
;    0361  0              ABORT   = [ 4,  9, 1, 0 ],      !
;    0362  0              STE16   = [ 4, 10, 1, 0 ],      ! SANITY TIMER ON AT POWER_UP
;    0363  0              NOCAR   = [ 4, 11, 1, 0 ],      ! NO CARRIER
;    0364  0              RUNT    = [ 4, 11, 1, 0 ],      ! RUNT PACKET IN FIFO
;    0365  0              ESETUP  = [ 4, 13, 1, 0 ],      ! CONTROL SET_UP OR LOOPBACK PACKET
;    0366  0              LONGP   = [ 4, 14, 1, 0 ],      ! LONG PACKET
;    0367  0              ERRSU   = [ 4, 14, 1, 0 ],      ! ERROR SUMMARY
;    0368  0              LSTD    = [ 4, 15, 1, 0 ],      ! LAST DESCRIPTOR LIST IN CHAIN
;    0369  0
;    0370  0              STWD2   = [ 5, 0, 16, 0 ],      ! XMIT OR RCV STATUS WORD 2
;    0371  0              TDR     = [ 5, 0, 14, 0 ],      !
;    0372  0              RBLL    = [ 5, 0,  8, 0 ],      ! RECEIVE BYTE LENGTH ( LOW 8 BITS )
;    0373  0
;    0374  0              DLINK   = [ 6, 0, 16, 0 ],      ! DESCRIPTOR LINK PRE-FILL STATUS WD
;    0375  0
;    0376  0              BSTAT   = [ 7, 0, 16, 0 ],      ! BUFFER STATE ! XMIT ODD/EVEN ! HIGH ORDER ADR
;    0377  0
;    0378  0              B_LEN   = [ 0, 8, 0 ],          !
;    0379  0              W_LEN   = [ 0,16, 0 ]           !
;    0380  0              TES;
```

```
;    0381  0     !++
;    0382  0     !
;    0383  0     !  HARDWARE P-TABLE FIELD DEFINITIONS
;    0384  0     !
;    0385  0     !--
;    0386  0
;    0387  0     FIELD
;    0388  0         HWP_FIELDS =
;    0389  0             SET
;    0390  0             ADDR    = [ 0, 0, 16, 0 ],      ! I/O PAGE BASE ADDRESS
;    0391  0             VEC     = [ 1, 0, 16, 0 ],      ! INTERRUPT VECTOR ADDRESS
;    0392  0             BRL     = [ 2, 0, 16, 0 ]       ! BR LEVEL
;    0393  0             TES;
;    0394  0
;    0395  0
;    0396  0     !++
;    0397  0     !
;    0398  0     !  SOFTWARE P-TABLE FIELD DEFINITIONS
;    0399  0     !
;    0400  0     !--
;    0401  0
;    0402  0     FIELD
;    0403  0         SWP_FIELDS =
;    0404  0             SET
;    0405  0             ERR_CNT = [0,0,16,0]            ! # OF ERRORS BEFORE DROPPING DEQNA
;    0406  0             TES;
;    0407  0
;    0408  0


;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 QNALIB.R16/LIST=QNALIB.LIS/LIBRARY=QNALIB.L16/SOURCE=PAGE:53

; Run Time:         00:03.5
; Elapsed Time:     00:11.6
; Lines/CPU Min:     7075
; Lexemes/CPU-Min: 33815
; Memory Used:  44 pages
; Library Precompilation Complete
```

```
ZQNAD0.EXE;1   Memory allocation map  TKB M40.10      Page 1
                    14-MAR-85   13:24


Partition name : DUMMY
Identification : V01.0
Task  UIC      : [202,34]
Task attributes: -HD
Total address windows: 1.
Task  image  size  : 11264. words
Task address limits: 002000 055713
R-W disk blk limits: 000002 000055 000054 00044.

*** Root segment: ZQNA1


R/W mem  limits: 002000 055713 053714 22476.
Disk blk limits: 000002 000055 000054 00044.


Memory allocation synopsis:

Section                                          Title  Ident  File
-------                                          -----  -----  ----
$CODE$:(RO,I,LCL,REL,CON)      002000 000406 00262.
                               002000 000242 00162. ZQNA1  V01.0  ZQNA1.O8J;1
                               002242 000144 00100. ZQNA2  V01.0  ZQNA2.O8J;1
$GLOB$:(RW,D,LCL,REL,CON)      002406 012504 05444.
                               002406 012504 05444. ZQNA1  V01.0  ZQNA1.O8J;1
$PLIT$:(RO,D,LCL,REL,CON)      015112 007036 03614.
                               015112 007036 03614. ZQNA1  V01.0  ZQNA1.O8J;1
AA$COD:(RO,I,LCL,REL,CON)      024150 000370 00248.
                               024150 000370 00248. ZQNA2  V01.0  ZQNA2.O8J;1
AB$COD:(RO,I,LCL,REL,CON)      024540 022570 09592.
                               024540 022570 09592. ZQNA3  V01.0  ZQNA3.O8J;1
AC$COD:(RO,I,LCL,REL,CON)      047330 006130 03160.
                               047330 006130 03160. ZQNA4  V01.0  ZQNA4.O8J;1
. BLK.:(RW,I,LCL,REL,CON)      055460 000000 00000.
$XYZ$ :(RO,I,LCL,REL,CON)      055460 000234 00156.
                               055460 000214 00140. CZQNAA 2.4    B16SAV.O8J;1
                               055674 000020 00016. ZQNA5  V01.0  ZQNA5.O8J;1


Global symbols:

ADR     000020    BIT1   000002    BIT8   000400    COUNTE 015016-R  ERRBLK 002204-R  GP$1   002312-R  INTERR 015012-R
BD.PRO 014300-R   BIT10  002000    BIT9   001000    CSR.WO 015030-R  ERRMSG 002202-R  GP$2   002322-R  IOP.DA 015002-R
BIT0    000001    BIT11  004000    BL$LAS 055674-R  DATA.B 003006-R  ERRNBR 002200-R  GP$3   002336-R  IOP.TA 014034-R
BIT00   000001    BIT12  010000    BOE    000400    DEQNA. 015014-R  ERROR$ 047330-R  GP$4   002346-R  ISR    000100
BIT01   000002    BIT13  020000    BUF.LE 015026-R  DESCR. 002406-R  ERRTYP 002176-R  GP$5   002362-R  IXE    004000
BIT02   000004    BIT14  040000    CHECKS 015024-R  DFSTBL 002210-R  ERR.CO 015040-R  GP$6   002370-R  KBD.IN 055646-R
BIT03   000010    BIT15  100000    CHK.CS 051176-R  DOWN.C 015022-R  ERR.FL 015036-R  GP$7   002376-R  LOE    040000
BIT04   000020    BIT2   000004    CHK.RC 051666-R  D$PCNT 002122-R  ERR.NU 015034-R  HOE    100000    LOT    000010
BIT05   000040    BIT3   000010    CHK.RI 050612-R  EF.CON 000036    ETH.ST 014054-R  HP.TAB 002210-R  L$ACP  002110-R
BIT06   000100    BIT4   000020    CHK.XM 051370-R  EF.NEW 000035    EVL    000004    HWP.TA 014774-R  L$APT  002036-R
BIT07   000200    BIT5   000040    CLR.BU 050564-R  EF.PWR 000034    E1$REP 047634-R  IBE    010000    L$AU   024520-R
BIT08   000400    BIT6   000100    CLR.DE 050536-R  EF.RES 000037    FORM.H 053726-R  IDU    000040    L$AUT  002070-R
BIT09   001000    BIT7   000200    COMPAR 052146-R  EF.STA 000040    GET.AD 015004-R  IER    020000    L$AUTO 024462 R
```

```
ZQNADO.EXE;1   Memory allocation map  TKB M40.10      Page 2
ZQNA1                  14-MAR-85   13:24


L$CCP  002106-R  L$MREV 002050-R  MSG11  016574-R  MSG45  021702-R  PRIO2  000100    SWP.IL 002226 R  T15    041774-R
L$CLEA 024474-R  L$NAME 002000-R  MSG12  016660-R  MSG46  021760-R  PRIO3  000140    SWP.LB 002222-R  T16    044042-R
L$CO   002032-R  L$NDHR 002332-R  MSG13  016724-R  MSG47  022030-R  PRIO4  000200    SWP.TA 014776-R  T17    044704-R
L$DEPO 002011-R  L$NDHW 002214-R  MSG14  017010-R  MSG48  022104-R  PRIO5  000240    SWP.TI 002220-R  T18    045222-R
L$DESC 002260-R  L$NDSF 002404-R  MSG15  017100-R  MSG49  022142-R  PRIO6  000300    SWP.TO 002224-R  T19    045572-R
L$DESP 002076-R  L$NDSW 002232-R  MSG16  017162-R  MSG50  022200-R  PRIO7  000340    TADR1  015106-R  T2     026032-R
L$DEVP 002060-R  L$PRIO 002042-R  MSG17  017250-R  MSG51  022262-R  PTRN.T 014100-R  TADR2  015110-R  T20    046272-R
L$DISP 002124-R  L$PROT 002234-R  MSG18  017336-R  MSG52  022314-R  PWR.IN 055604-R  TARGET 014110-R  T21    047314-R
L$DLY  002116-R  L$PRT  002112-R  MSG19  017362-R  MSG53  022360-R  P1     015070-R  TBYTE1 015102-R  T3     026616-R
L$DTP  002040-R  L$REPP 002062-R  MSG20  017450-R  MSG54  022410-R  P2     015072-R  TBYTE2 015103-R  T4     027664-R
L$DTYP 002034-R  L$REV  002010-R  MSG21  017540-R  MSG55  022460-R  P3     015074-R  TBYTE3 015104-R  T5     030544-R
L$DU   024506-R  L$RPT  024160-R  MSG22  017620-R  MSG56  022522-R  P4     015076-R  TBYTE4 015105-R  T6     031254-R
L$DUT  002072-R  L$SFTL 002334-R  MSG23  017704-R  MSG57  022560-R  P5     015100-R  TD13   014470-R  T7     033614-R
L$DVTY 002242-R  L$SOFT 002336-R  MSG24  017762-R  MSG58  022650-R  QSTO1  015112-R  TD16   014340-R  T8     034052-R
L$EF   002052-R  L$SPC  002056-R  MSG25  020036-R  MSG59  022714-R  QSTO2  015142-R  TEMP1  015046-R  T9     034376-R
L$ENVI 002044-R  L$SPCP 002020-R  MSG26  020100-R  MSG60  023026-R  QSTO3  015172-R  TEMP2  015050-R  UAM    000200
L$ERRT 002176-R  L$SPTP 002024-R  MSG27  020142-R  MSG61  023070-R  QSTO4  015234-R  TEMP3  015052-R  UP.COU 015020-R
L$ETP  002102-R  L$STA  002030-R  MSG28  020204-R  MSG62  023132-R  QSTO5  015276-R  TEMP4  015054-R  VER.DE 050336-R
L$EXP1 002046-R  L$SW   002220-R  MSG29  020250-R  MSG63  023222-R  QSTO6  015340-R  TEMP5  015056-R  WAIT.F 055566-R
L$EXP4 002064-R  L$SWLE 002216-R  MSG30  020276-R  MSG64  023316-R  QSTO7  015402-R  TEMP6  015060-R  WALKIN 052720-R
L$EXP5 002066-R  L$TEST 002114-R  MSG31  020364-R  MSG65  023352-R  RBUF.L 015010-R  TEMP7  015062-R  WRT.ST 053452-R
L$HARD 002312-R  L$TIML 002014-R  MSG32  020450-R  MSG66  023416-R  RCV.BU 003006-R  TEMP8  015064-R  XBUF.L 015006-R
L$HIME 002120-R  L$UNIT 002012-R  MSG33  020512-R  MSG67  023504-R  RCV.D. 002406-R  TEMP9  015066-R  XC.FLA 015032-R
L$HPCP 002016-R  MSG00  015444-R  MSG34  020566-R  MSG68  023606-R  RD13   014574-R  TMP.IO 015042-R  XMIT.A 055160-R
L$HPTP 002022-R  MSG01  015502-R  MSG35  020642-R  MSG69  023704-R  REG.AD 015000-R  TMP.RE 015044-R  XMIT.B 007006-R
L$HRDL 002310-R  MSG02  015564-R  MSG36  020740-R  MSG70  024004-R  RESET. 047654-R  TURN.O 055404-R  XMIT.D 002606-R
L$HW   002210-R  MSG03  015652-R  MSG37  021044-R  MSG71  024070-R  SEND.E 054644-R  T$FREE 055710-R  XMIT.I 055206-R
L$HWLE 002206-R  MSG04  015756-R  MSG38  021136-R  NXM.IN 024530-R  SEND.T 055012-R  T$PTHV 000001    XMIT.S 054376-R
L$ICP  002104-R  MSG05  016050-R  MSG39  021216-R  PHYS.A 013006-R  SETUP. 013034-R  T1     025202-R  $END.L 055712-R
L$INIT 024450-R  MSG06  016142-R  MSG40  021302-R  PNT    001000    SET.RD 052564-R  T10    034634-R  $SAVE2 055460-R
L$LADP 002026-R  MSG07  016234-R  MSG41  021372-R  PREP.F 053644-R  SET.XD 052642-R  T11    035170-R  $SAVE3 055474-R
L$LAST 055700-R  MSG08  016326-R  MSG42  021434-R  PRI    002000    SP.TAB 002220-R  T12    037024-R  $SAVE4 055512-R
L$LOAD 002100-R  MSG09  016420-R  MSG43  021514-R  PRIO0  000000    STATIO 014070-R  T13    040262-R  $SAVE5 055532-R
L$LUN  002074-R  MSG10  016512-R  MSG44  021600-R  PRIO1  000040    SWP.BL 002230-R  T14    041544-R


*** Task builder statistics:

    Total work file references: 87112.
    Work  file  reads: 0.
    Work  file writes: 0.
    Size of core pool: 23454. words (91. pages)
    Size of work file: 3328. words (13. pages)

    Elapsed time:00:00:39
```

ZQNADO CREATED BY TKB ON 14-MAR-85 AT 13:24 PAGE 1 SEQ 0293

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
FDR	000020	# ZQNA1	# ZQNA2	
BD.PRO	014300-R	# ZQNA1	ZQNA3	ZQNA4
BIT0	000001	# ZQNA1	# ZQNA2	
BIT00	000001	# ZQNA1	# ZQNA2	
BIT01	000002	# ZQNA1	# ZQNA2	
BIT02	000004	# ZQNA1	# ZQNA2	
BIT03	000010	# ZQNA1	# ZQNA2	
BIT04	000020	# ZQNA1	# ZQNA2	
BIT05	000040	# ZQNA1	# ZQNA2	
BIT06	000100	# ZQNA1	# ZQNA2	
BIT07	000200	# ZQNA1	# ZQNA2	
BIT08	000400	# ZQNA1	# ZQNA2	
BIT09	001000	# ZQNA1	# ZQNA2	
BIT1	000002	# ZQNA1	# ZQNA2	
BIT10	002000	# ZQNA1	# ZQNA2	
BIT11	004000	# ZQNA1	# ZQNA2	
BIT12	010000	# ZQNA1	# ZQNA2	
BIT13	020000	# ZQNA1	# ZQNA2	
BIT14	040000	# ZQNA1	# ZQNA2	
BIT15	100000	# ZQNA1	# ZQNA2	
BIT2	000004	# ZQNA1	# ZQNA2	
BIT3	000010	# ZQNA1	# ZQNA2	
BIT4	000020	# ZQNA1	# ZQNA2	
BIT5	000040	# ZQNA1	# ZQNA2	
BIT6	000100	# ZQNA1	# ZQNA2	
BIT7	000200	# ZQNA1	# ZQNA2	
BIT8	000400	# ZQNA1	# ZQNA2	
BIT9	001000	# ZQNA1	# ZQNA2	
BL$LAS	055674-R	# ZQNA5		
BOE	000400	# ZQNA1	# ZQNA2	
BUF.LE	015026-R	# ZQNA1		
CHECKS	015024-R	# ZQNA1	ZQNA3	ZQNA4
CHK.CS	051176-R	ZQNA3	# ZQNA4	
CHK.RC	051666-R	ZQNA3	# ZQNA4	
CHK.RI	050612-R	ZQNA3	# ZQNA4	
CHK.XM	051370-R	ZQNA3	# ZQNA4	
CLR.BU	050564-R	ZQNA3	# ZQNA4	
CLR.DE	050536-R	ZQNA3	# ZQNA4	
COMPAR	052146-R	ZQNA3	# ZQNA4	
COUNTE	015016-R	# ZQNA1	ZQNA3	ZQNA4
CSR.WO	015030-R	# ZQNA1	ZQNA3	ZQNA4
DATA.B	003006-R	# ZQNA1	ZQNA3	ZQNA4
DEQNA.	015014-R	# ZQNA1	ZQNA3	ZQNA4
DESCR.	002406-R	# ZQNA1	ZQNA3	ZQNA4
DFSTBL	002210-R	# ZQNA1		
DOWN.C	015022-R	# ZQNA1	ZQNA3	ZQNA4
D$PCNT	002122-R	# ZQNA1		
EF.CON	000036	# ZQNA1	# ZQNA2	
EF.NEW	000035	# ZQNA1	# ZQNA2	
EF.PWR	000034	# ZQNA1	# ZQNA2	
EF.RES	000037	# ZQNA1	# ZQNA2	
EF.STA	000040	# ZQNA1	# ZQNA2	

SEQ 0294

ZQNADO CREATED BY TKB ON 14-MAR-85 AT 13:24 PAGE 2

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...				
ERRBLK	002204-R	# ZQNA1				
ERRMSG	002202-R	# ZQNA1				
ERRNBR	002200-R	# ZQNA1				
ERROR$	047330-R	ZQNA3	# ZQNA4			
ERRTYP	002176-R	# ZQNA1				
ERR.CO	015040-R	# ZQNA1	ZQNA3	ZQNA4		
ERR.FL	015036-R	# ZQNA1	ZQNA3	ZQNA4		
ERR.NU	015034-R	# ZQNA1	ZQNA3	ZQNA4		
ETH.ST	014054-R	# ZQNA1				
EVL	000004	# ZQNA1	# ZQNA2			
E1$REP	047634-R	ZQNA3	# ZQNA4			
FORM.H	053726-R	ZQNA3	# ZQNA4			
GET.AD	015004-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
GP$1	002312-R	# ZQNA2				
GP$2	002322-R	# ZQNA2				
GP$3	002336-R	# ZQNA2				
GP$4	002346-R	# ZQNA2				
GP$5	002362-R	# ZQNA2				
GP$6	002370-R	# ZQNA2				
GP$7	002376-R	# ZQNA2				
HOE	100000	# ZQNA1	# ZQNA2			
HP.TAB	002210-R	# ZQNA1				
HWP.TA	014774-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
IBE	010000	# ZQNA1	# ZQNA2			
IDU	000040	# ZQNA1	# ZQNA2			
IER	020000	# ZQNA1	# ZQNA2			
INTERR	015012-R	CZQNAA	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
IOP.DA	015002-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
IOP.TA	014034-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
ISR	000100	# ZQNA1	# ZQNA2			
IXE	004000	# ZQNA1	# ZQNA2			
KBD.IN	055646-R	# CZQNAA	ZQNA3			
LOE	040000	# ZQNA1	# ZQNA2			
LOT	000010	# ZQNA1	# ZQNA2			
L$ACP	002110-R	# ZQNA1				
L$APT	002036-R	# ZQNA1				
L$AU	024520-R	ZQNA1	# ZQNA2			
L$AUT	002070-R	# ZQNA1				
L$AUTO	024462-R	ZQNA1	# ZQNA2			
L$CCP	002106-R	# ZQNA1				
L$CLEA	024474-R	ZQNA1	# ZQNA2			
L$CO	002032-R	# ZQNA1				
L$DEPO	002011-R	# ZQNA1				
L$DESC	002260-R	ZQNA1	# ZQNA2			
L$DESP	002076-R	# ZQNA1				
L$DEVP	002060-R	# ZQNA1				
L$DISP	002124-R	# ZQNA1				
L$DLY	002116-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
L$DTP	002040-R	# ZQNA1				
L$DTYP	002034-R	# ZQNA1				
L$DU	024506-R	ZQNA1	# ZQNA2			
L$DUT	002072-R	# ZQNA1				

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
L$DVTY	002242-R	ZQNA1	# ZQNA2	
L$EF	002052-R	# ZQNA1		
L$ENVI	002044-R	# ZQNA1		
L$ERRT	002176-R	# ZQNA1		
L$ETP	002102-R	# ZQNA1		
L$EXP1	002046-R	# ZQNA1		
L$EXP4	002064-R	# ZQNA1		
L$EXP5	002066-R	# ZQNA1		
L$HARD	002312-R	ZQNA1	# ZQNA2	
L$HIME	002120-R	# ZQNA1		
L$HPCP	002016-R	# ZQNA1		
L$HPTP	002022-R	# ZQNA1		
L$HRDL	002310-R	# ZQNA2		
L$HW	002210-R	# ZQNA1		
L$HWLE	002206-R	# ZQNA1		
L$ICP	002104-R	# ZQNA1		
L$INIT	024450-R	ZQNA1	# ZQNA2	
L$LADP	002026-R	# ZQNA1		
L$LAST	055700-R	ZQNA1	# ZQNA5	
L$LOAD	002100-R	# ZQNA1		
L$LUN	002074-R	# ZQNA1		
L$MREV	002050-R	# ZQNA1		
L$NAME	002000-R	# ZQNA1		
L$NDHR	002332-P	# ZQNA2		
L$NDHW	002214-R	# ZQNA1		
L$NDSF	002404-R	# ZQNA2		
L$NDSW	002232-R	# ZQNA1		
L$PRIO	002042-R	# ZQNA1		
L$PROT	002234-R	# ZQNA1		
L$PRT	002112-R	# ZQNA1		
L$REPP	002062-R	# ZQNA1		
L$REV	002010-R	# ZQNA1		
L$RPT	024160-R	ZQNA1	# ZQNA2	
L$SFTL	002334-R	# ZQNA2		
L$SOFT	002336-R	ZQNA1	# ZQNA2	
L$SPC	002056-R	# ZQNA1		
L$SPCP	002020-R	# ZQNA1		
L$SPTP	002024-R	# ZQNA1		
L$STA	002030-R	# ZQNA1		
L$SW	002220-R	# ZQNA1		
L$SWLE	002216-R	# ZQNA1		
L$TEST	002114-R	# ZQNA1		
L$TIML	002014-R	# ZQNA1		
L$UNIT	002012-R	# ZQNA1		
MSG00	015444-R	# ZQNA1	ZQNA3	ZQNA4
MSG01	015502-R	# ZQNA1	ZQNA3	ZQNA4
MSG02	015564-R	# ZQNA1	ZQNA3	ZQNA4
MSG03	015652-R	# ZQNA1	ZQNA3	ZQNA4
MSG04	015756-R	# ZQNA1	ZQNA3	ZQNA4
MSG05	016050-R	# ZQNA1	ZQNA3	ZQNA4
MSG06	016142-R	# ZQNA1	ZQNA3	ZQNA4
MSG07	016234-R	# ZQNA1	ZQNA3	ZQNA4

ZQNADO CREATED BY TKB ON 14-MAR-85 AT 13:24 PAGE 4

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...			
MSG08	016326-R	# ZQNA1	ZQNA3	ZQNA4	
MSG09	016420-R	# ZQNA1	ZQNA3	ZQNA4	
MSG10	016512-R	# ZQNA1	ZQNA3	ZQNA4	
MSG11	016574-R	# ZQNA1	ZQNA3	ZQNA4	
MSG12	016660-R	# ZQNA1	ZQNA3	ZQNA4	
MSG13	016724-R	# ZQNA1	ZQNA3	ZQNA4	
MSG14	017010-R	# ZQNA1	ZQNA3	ZQNA4	
MSG15	017100-R	# ZQNA1	ZQNA3	ZQNA4	
MSG16	017162-R	# ZQNA1	ZQNA3	ZQNA4	
MSG17	017250-R	# ZQNA1	ZQNA3	ZQNA4	
MSG18	017336-R	# ZQNA1	ZQNA3	ZQNA4	
MSG19	017362-R	# ZQNA1	ZQNA3	ZQNA4	
MSG20	017450-R	# ZQNA1	ZQNA3	ZQNA4	
MSG21	017540-R	# ZQNA1	ZQNA3	ZQNA4	
MSG22	017620-R	# ZQNA1	ZQNA3	ZQNA4	
MSG23	017704-R	# ZQNA1	ZQNA3	ZQNA4	
MSG24	017762-R	# ZQNA1	ZQNA3	ZQNA4	
MSG25	020036-R	# ZQNA1	ZQNA3	ZQNA4	
MSG26	020100-R	# ZQNA1	ZQNA3	ZQNA4	
MSG27	020142-R	# ZQNA1	ZQNA3	ZQNA4	
MSG28	020204-R	# ZQNA1	ZQNA3	ZQNA4	
MSG29	020250-R	# ZQNA1	ZQNA3	ZQNA4	
MSG30	020276-R	# ZQNA1	ZQNA3	ZQNA4	
MSG31	020364-R	# ZQNA1	ZQNA3	ZQNA4	
MSG32	020450-R	# ZQNA1	ZQNA3	ZQNA4	
MSG33	020512-R	# ZQNA1	ZQNA3	ZQNA4	
MSG34	020566-R	# ZQNA1	ZQNA3	ZQNA4	
MSG35	020642-R	# ZQNA1	ZQNA3	ZQNA4	
MSG36	020740-R	# ZQNA1	ZQNA3	ZQNA4	
MSG37	021044-R	# ZQNA1	ZQNA3	ZQNA4	
MSG38	021136-R	# ZQNA1	ZQNA3	ZQNA4	
MSG39	021216-R	# ZQNA1	ZQNA3	ZQNA4	
MSG40	021302-R	# ZQNA1	ZQNA3	ZQNA4	
MSG41	021372-R	# ZQNA1	ZQNA3	ZQNA4	
MSG42	021434-R	# ZQNA1	ZQNA3	ZQNA4	
MSG43	021514-R	# ZQNA1	ZQNA3	ZQNA4	
MSG44	021600-R	# ZQNA1	ZQNA3	ZQNA4	
MSG45	021702-R	# ZQNA1	ZQNA3	ZQNA4	
MSG46	021760-R	# ZQNA1	ZQNA3	ZQNA4	
MSG47	022030-R	# ZQNA1	ZQNA3	ZQNA4	
MSG48	022104-R	# ZQNA1	ZQNA3	ZQNA4	
MSG49	022142-R	# ZQNA1	ZQNA3	ZQNA4	
MSG50	022200-R	# ZQNA1	ZQNA3	ZQNA4	
MSG51	022262-R	# ZQNA1	ZQNA3	ZQNA4	
MSG52	022314-R	# ZQNA1	ZQNA3	ZQNA4	
MSG53	022360-R	# ZQNA1	ZQNA3	ZQNA4	
MSG54	022410-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
MSG55	022460-R	# ZQNA1	ZQNA3	ZQNA4	
MSG56	022522-R	# ZQNA1	ZQNA3	ZQNA4	
MSG57	022560-R	# ZQNA1	ZQNA3	ZQNA4	
MSG58	022650-R	# ZQNA1	ZQNA3	ZQNA4	
MSG59	022714-R	# ZQNA1	ZQNA3	ZQNA4	

SYMBOL VALUE REFERENCES...

MSG60	023026-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG61	023070-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG62	023132-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG63	023222-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG64	023316-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG65	023352-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG66	023416-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG67	023504-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG68	023606-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG69	023704-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG70	024004-R	#	ZQNA1		ZQNA3		ZQNA4	
MSG71	024070-R	#	ZQNA1		ZQNA3			
NXM.IN	024530-R	#	ZQNA2		ZQNA3			
PHYS.A	013006-R	#	ZQNA1		ZQNA3		ZQNA4	
PNT	001000	#	ZQNA1	#	ZQNA2			
PREP.F	053644-R		ZQNA3	#	ZQNA4			
PRI	002000	#	ZQNA1	#	ZQNA2			
PRI00	000000	#	ZQNA1	#	ZQNA2	ZQNA3	ZQNA4	
PRI01	000040	#	ZQNA1	#	ZQNA2	ZQNA3	ZQNA4	
PRI02	000100	#	ZQNA1	#	ZQNA2	ZQNA3	ZQNA4	
PRI03	000140	#	ZQNA1	#	ZQNA2	ZQNA3	ZQNA4	
PRI04	000200	#	ZQNA1	#	ZQNA2	ZQNA3	ZQNA4	
PRI05	000240	#	ZQNA1	#	ZQNA2	ZQNA3	ZQNA4	
PRI06	000300	#	ZQNA1	#	ZQNA2	ZQNA3	ZQNA4	
PRI07	000340	#	ZQNA1	#	ZQNA2	ZQNA3	ZQNA4	
PTRN.T	014100-R	#	ZQNA1		ZQNA3			
PWR.IN	055604-R	#	CZQNAA		ZQNA3			
P1	015070-R	#	ZQNA1		ZQNA3		ZQNA4	
P2	015072-R	#	ZQNA1		ZQNA3		ZQNA4	
P3	015074-R	#	ZQNA1		ZQNA3		ZQNA4	
P4	015076-R	#	ZQNA1		ZQNA3		ZQNA4	
P5	015100-R	#	ZQNA1					
QST01	015112-R	#	ZQNA1		ZQNA2			
QST02	015142-R	#	ZQNA1		ZQNA2			
QST03	015172-R	#	ZQNA1		ZQNA2			
QST04	015234-R	#	ZQNA1		ZQNA2			
QST05	015276-R	#	ZQNA1		ZQNA2			
QST06	015340-R	#	ZQNA1		ZQNA2			
QST07	015402-R	#	ZQNA1		ZQNA2			
RBUF.L	015010-R	#	ZQNA1		ZQNA3		ZQNA4	
RCV.BU	003006-R	#	ZQNA1		ZQNA3		ZQNA4	
RCV.D.	002406-R	#	ZQNA1		ZQNA3		ZQNA4	
RD13	014574-R	#	ZQNA1		ZQNA3			
REG.AD	015000-R	#	ZQNA1		ZQNA2	ZQNA3	ZQNA4	
RESET.	047654-R		ZQNA2		ZQNA3	# ZQNA4		
SEND.E	054644-R		ZQNA3	#	ZQNA4			
SEND.T	055012 R		ZQNA3	#	ZQNA4			
SETUP.	013034-R	#	ZQNA1		ZQNA4			
SET.RD	052564 R		ZQNA3	#	ZQNA4			
SET.XD	052642 R		ZQNA3	#	ZQNA4			
SP.TAB	002220-R	#	ZQNA1					
STATIO	014070 R	#	ZQNA1		ZQNA3		ZQNA4	

ZQNADO CREATED BY TKB ON 14-MAR-85 AT 13:24 PAGE 6

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...				
SWP.BL	002230-R	# ZQNA1	ZQNA3	ZQNA4		
SWP.IL	002226-R	# ZQNA1	ZQNA3	ZQNA4		
SWP.LB	002222-R	# ZQNA1	ZQNA3			
SWP.TA	014776-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
SWP.TI	002220-R	# ZQNA1	ZQNA3	ZQNA4		
SWP.TO	002224-R	# ZQNA1	ZQNA3	ZQNA4		
TADR1	015106-R	# ZQNA1	ZQNA3	ZQNA4		
TADR2	015110-R	# ZQNA1	ZQNA3	ZQNA4		
TARGET	014110-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE1	015102-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE2	015103-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE3	015104-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE4	015105-R	# ZQNA1	ZQNA3	ZQNA4		
TD13	014470-R	# ZQNA1	ZQNA3			
TD16	014340-R	# ZQNA1	ZQNA3			
TEMP1	015046-R	CZQNAA # ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP2	015050-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP3	015052-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP4	015054-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP5	015056-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP6	015060-R	CZQNAA # ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP7	015062-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP8	015C64-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP9	015066-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TMP.IO	015042-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TMP.RE	015044-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TURN.O	055404-R	ZQNA3 # ZQNA4				
T$FREE	055710-R	# ZQNA5				
T$PTHV	000001	ZQNA1 # ZQNA5				
T1	025202-R	ZQNA1 # ZQNA3				
T10	034634-R	ZQNA1 # ZQNA3				
T11	035170-R	ZQNA1 # ZQNA3				
T12	037024-R	ZQNA1 # ZQNA3				
T13	040262-R	ZQNA1 # ZQNA3				
T14	041544-R	ZQNA1 # ZQNA3				
T15	041774-R	ZQNA1 # ZQNA3				
T16	044042-R	ZQNA1 # ZQNA3				
T17	044704-R	ZQNA1 # ZQNA3				
T18	045222-R	ZQNA1 # ZQNA3				
T19	045572-R	ZQNA1 # ZQNA3				
T2	026032-R	ZQNA1 # ZQNA3				
T20	046272-R	ZQNA1 # ZQNA3				
T21	047314-R	ZQNA1 # ZQNA3				
T3	026616-R	ZQNA1 # ZQNA3				
T4	027664-R	ZQNA1 # ZQNA3				
T5	030544-R	ZQNA1 # ZQNA3				
T6	031254-R	ZQNA1 # ZQNA3				
T7	033614-R	ZQNA1 # ZQNA3				
T8	034052-R	ZQNA1 # ZQNA3				
T9	034376-R	ZQNA1 # ZQNA3				
UAM	000200 #	ZQNA1 # ZQNA2				
UP.COU	015020-R	# ZQNA1	ZQNA3	ZQNA4		

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
VFR.DE	050336-R	ZQNA3	# ZQNA4	
WAIT.F	055566-R	# CZQNAA	ZQNA3	
WALKIN	052720-R	ZQNA3	# ZQNA4	
WRT.ST	053452-R	ZQNA3	# ZQNA4	
XBUF.L	015006-R	# ZQNA1	ZQNA3	ZQNA4
XC.FLA	015032-R	# ZQNA1	ZQNA3	
XMIT.A	055160-R	ZQNA3	# ZQNA4	
XMIT.B	007006-R	# ZQNA1	ZQNA3	ZQNA4
XMIT.D	002606-R	# ZQNA1	ZQNA3	ZQNA4
XMIT.I	055206-R	ZQNA3	# ZQNA4	
XMIT.S	054376-R	ZQNA3	# ZQNA4	
$END.L	055712-R	# ZQNA5		
$SAVE2	055460-R	# CZQNAA	ZQNA3	ZQNA4
$SAVE3	055474-R	# CZQNAA	ZQNA3	ZQNA4
$SAVE4	055512-R	# CZQNAA	ZQNA2	ZQNA3
$SAVE5	055532-R	# CZQNAA		