```
;    0001  0
;    0002  0
;    0003  0
;    0004  0
;    0005  0
;    0006  0
;    0007  1
;    0008  1
; C  0009  1
; C  0010  1                          IDENTIFICATION
; C  0011  1                          ----- ----  --
; C  0012  1
; C  0013  1              PRODUCT CODE:   AC-T614E-MC
; C  0014  1
; C  0015  1              PRODUCT NAME:   CZQNAEO DEQNA FUNCTIONAL TEST
; C  0016  1
; C  0017  1              PRODUCT DATE:   APRIL 2, 1986
; 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) 1986
; C  0025  1
; C  0026  1              DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
; C  0027  1
; C  0028  1              THIS INFORMATION IN THIS  DOCUMENT IS SUBJECT TO  CHANGE WITHOUT
; C  0029  1              NOTICE AND SHOULD NOT  BE CONSTRUED AS  A COMMITMENT  BY DIGITAL
; C  0030  1              EQUIPMENT CORPORATION.  DIGITAL EQUIPMENT CORPORATION ASSUMES NO
; C  0031  1              RESPONSIBILTY FOR ANY ERRORS THAT MAY  APPEAR IN  THIS DOCUMENT.
; C  0032  1
; C  0033  1
; C  0034  1
; C  0035  1              NO RESPONSIBILTY IS ASSUMED  FOR THE  USE OR  RELIABILITY OF
; C  0036  1              SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS
; C  0037  1              AFFILIATED COMPANIES.
; C  0038  1
; C  0039  1
; C  0040  1
; C  0041  1              THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:
; C  0042  1
; C  0043  1
; 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
; C 0061 1
; C 0062 1
; C 0063 1
; C 0064 1
; C 0065 1
; C 0066 1
; C 0067 1
; C 0068 1
; C 0069 1
; C 0070 1
; C 0071 1
; C 0072 1
; C 0073 1
; C 0074 1
; C 0075 1
; C 0076 1
; C 0077 1
; C 0078 1
; C 0079 1

```
; C 0080  1              1.0  GENERAL INFORMATION
; C 0081  1                   -------------------
; 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 wh'ch 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 descr'pt'ons below, a dec'mal 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:

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

M1


```
; 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 CZQNAD0 in March, 1985 by Howard L. Marshall:
; C 0509  1
; C 0510  1          Modified DMA Timing Test, Test #14, to 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
; C 0514  1          Rev. CZQNAD0 changed to CZQNAE0 March 1986 Dave Scoda
; C 0515  1
; C 0516  1          Added ZQNA6.MAC to correct a bug in the generation of the checksum
; C 0517  1          for the Ethernet hardware address rom. Added code to check for reserved
; C 0518  1          and qualified bits that allowed the test to run on a busy network.
; C 0519  1          Fixed several error reports where actual and expected data were
; C 0520  1          reversed. Changed test 6 from an NMI interrupt to a Tx done interrupt;
; C 0521  1          it would fail along with test 12 in a 4MB or a 256KB system. Shortened
; C 0522  1          test 7; it no longer reports a false error. Made test 12 selectable,
; C 0523  1          see test 6 above. Changed error reports for bad descriptors in tests
; C 0524  1          13, 14 and 16.
; C 0525  1
; C 0526  1
;   0527  1     )%
;   0528  1
```

```
;    0529  1
;    0530  1     LIBRARY 'QNALIB';
;    0531  1     REQUIRE 'BLSMAC.REQ';                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;    2021  1
;    2022  1     !++
;    2023  1     !       DEFINE THE NUMBER OF TESTS IN THIS DIAGNOSTIC
;    2024  1     !--
;    2025  1
;    2026  1     PSECT
;    2027  1         CODE = AA$CODE$;
;    2028  1
;    2029  1     LITERAL
;    2030  1         DS$NBR_OF_TESTS = 21;
;    2031  1
;    2032  1     EQUALS;
;    2033  1
;    2034  1     POINTER (ALL);
;    2035  1
;    2036  1     !++
;    2037  1     !        THE PROGRAM HEADER IS THE INTERFACE BETWEEN THE DIAGNOSTIC PROGRAM
;    2038  1     !        AND THE SUPERVISOR.
;    2039  1     !--
;    2040  1
;    2041  1     HEADER (%ASCII'CZQNA ',%ASCII'E',%ASCII'0', 120, 0, PRI00);
;    2042  1
;    2043  1
;    2044  1     !++
;    2045  1     !       NO POINTERS ARE OPTIONAL USING BLISS. MAKE SURE THE FOLLOWING
;    2046  1     !       SECTIONS OF CODE ARE IN PLACE (IN THE CORRECT SKELS),EVEN IF
;    2047  1     !       THE SECTIONS ARE BLANK.
;    2048  1     !
;    2049  1     !       ARGUMENT        FUNCTION
;    2050  1     !       --  ----        ------
;    2051  1     !       RPT             REPORT CODE
;    2052  1     !       SW              SOFTWARE TABLE
;    2053  1     !       SFT             SOFTWARE TABLE QUESTIONS
;    2054  1     !       AU              ADD CODE
;    2055  1     !       DU              DROP CODE
;    2056  1     !       TBL             ERROR TABLE
;    2057  1     !       SETUP           ASSEMBLED P-TABLES
;    2058  1     !
;    2059  1     !       CHANGE THE "HEADER" TO CONTAIN THE PROPER ARGUMENTS.
;    2060  1     !       ARGUMENTS ARE: NAME,REV,PATCH,LONGEST TEST TIME,TYPE
;    2061  1     !       WHERE "TYPE" = 0 FOR SEQUENTIAL DIAGNOSTIC AND =1
;    2062  1     !       FOR EXERCISER.  THERE IS ALSO AN OPTIONAL SIXTH ARGUMENT
;    2063  1     !       WHICH SPECIFIES THE PROCESSOR PRIORITY TO BE SET WHEN
;    2064  1     !       STARTING THE DIAGNOSTIC (DEFAULT IS 0).
;    2065  1     !--
;    2066  1
;    2067  1
```

```
;   2068  1     %SBTTL 'DISPATCH TABLE'
;   2069  1
;   2070  1     DISPATCH (DS$NBR OF TESTS);
;   2071  1
;   2072  1     !++
;   2073  1     !       THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
;   2074  1     !       IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
;   2075  1     !
;   2076  1     !        CHANGE THE LITERAL DECLARATION OF DS$NBR_OF_TESTS TO BE
;   2077  1     !        THE NUMBER OF HARDWARE TESTS IN YOUR PROGRAM.
;   2078  1     !
;   2079  1     !
;   2080  1
;   2081  1     ERRTBL;
;   2082  1
;   2083  1     !++
;   2084  1     !       THE ERRTBL MACRO IS REQUIRED WHETHER OR NOT YOU REPORT ERRORS USING
;   2085  1     !       THE "ERROR" MACRO.  THE ERRTBL MACRO EXPANDS INTO FOUR WORDS THAT
;   2086  1     !       ARE USED BY THE RUNTIME SERVICES DURING AN ERROR CALL: ERROR TYPE,
;   2087  1     !       ERROR NUMBER, ADDRESS OF ERROR MESSAGE AND ADDRESS OF MESSAGE
;   2088  1     !       BLOCK.  THERE MUST BE ONLY ONE ERRTBL IN ANY PROGRAM.  THIS SECTION
;   2089  1     !       IS NOT OPTIONAL.
;   2090  1     !
;   2091  1
```

E2


```
;    2092  1     %SBTTL 'GLOBAL DATA SECTION'
;    2093  1
;    2094  1     PSECT
;    2095  1         PLIT   = $PLIT$,
;    2096  1         OWN    = $OWN$,
;    2097  1         GLOBAL = $GLOB$;
;    2098  1
;    2099  1     !++
;    2100  1     !       THE GLOBAL DATA DEFINED IN THIS SECTION IS USED BY MORE THAN ONE
;    2101  1     !       TEST.
;    2102  1     !--
;    2103  1
;    2104  1     GLOBAL
;    2105  1
;    2106  1     !++
;    2107  1     !       COMMUNICATION AREA DECLARATIONS
;    2108  1     !--
;    2109  1
;    2110  1         RCV_D_LIST      : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    2111  1         XMIT_D_LIST     : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    2112  1         RCV_BUFFER      : VECTOR [ B_SIZE, BYTE ],
;    2113  1         XMIT_BUFFER     : VECTOR [ B_SIZE, BYTE ],
;    2114  1         PHYS_ADR        : VECTOR [ 22, BYTE ],
;    2115  1         SETUP_BUFFER    : VECTOR [ SETUB_SIZE, WORD ],
;    2116  1         IOP_TABLE       : VECTOR [ 8, WORD ],
;    2117  1         ETH_STATION_ADR : VECTOR [ 6, WORD ],
;    2118  1         STATION_ADR     : VECTOR [ 4, WORD ],
;    2119  1         PTRN_TABLE      : VECTOR [ 8, BYTE ] INITIAL ( BYTE (
;    2120  1
;    2121  1                           %B'00000000', %B'11111111', %B'10101010', %B'01010101',
;    2122  1                           %B'11001100', %B'00110011', %B'11110000', %B'00001111')),
```

F2


```
;   2123  1          TARGET_ADR      : VECTOR [ T_SIZE, BYTE ] INITIAL ( BYTE (
;   2124  1
;   2125  1          %X'00', %X'00', %X'00', %X'00', %X'00', %X'00',     !  1 - MEMORY PATTERN
;   2126  1          %X'55', %X'55', %X'55', %X'55', %X'55', %X'55',     !  2
;   2127  1          %X'AA', %X'AA', %X'AA', %X'AA', %X'AA', %X'AA',     !  3 - MEMORY PATTERN
;   2128  1          %X'55', %X'55', %X'55', %X'55', %X'55', %X'55',     !  4   MEMORY PATTERN
;   2129  1          %X'FF', %X'FF', %X'FF', %X'FF', %X'FF', %X'FF',     !  5 - MEMORY PATTERN
;   2130  1          %X'00', %X'F4', %X'FA', %X'44', %X'44', %X'55',     !  6
;   2131  1          %X'AA', %X'00', %X'00', %X'00', %X'00', %X'00',     !  7 - MEMORY PATTERN
;   2132  1          %X'AA', %X'00', %X'02', %X'AA', %X'AA', %X'AA',     !  8
;   2133  1          %X'AA', %X'00', %X'05', %X'55', %X'55', %X'55',     !  9
;   2134  1          %X'AA', %X'00', %X'04', %X'FF', %X'FF', %X'FF',     ! 10
;   2135  1          %X'AA', %X'00', %X'04', %X'00', %X'00', %X'00',     ! 11 - LOW ETHERNET ADR
;   2136  1          %X'AA', %X'00', %X'04', %X'18', %X'81', %X'18',     ! 12 - HIGH ETHERNET ADR
;   2137  1          %X'01', %X'00', %X'00', %X'00', %X'00', %X'00',     ! 13 - ALL MULTICAST
;   2138  1          %X'AB', %X'AA', %X'AA', %X'AA', %X'AA', %X'AA',     ! 14 - ALL MULTICAST
;   2139  1          %X'FF', %X'00', %X'01', %X'02', %X'03', %X'04',     ! 15 - ALL MULTICAST
;   2140  1          %X'55', %X'05', %X'06', %X'07', %X'08', %X'09',     ! 16 - ALL MULTICAST
;   2141  1          %X'CD', %X'36', %X'26', %X'27', %X'27', %X'49',     ! 17
;   2142  1          %X'33', %X'A1', %X'67', %X'B8', %X'4C', %X'9F',     ! 18
;   2143  1          %X'EB', %X'BE', %X'C7', %X'8F', %X'33', %X'FF',     ! 19
;   2144  1          %X'FF', %X'FF', %X'FF', %X'FF', %X'FF', %X'FF' )),  ! 20 - STATION ADDR
```

```
;  2145  1
;  2146  1        BD_PROM_DESCR   : VECTOR [ BD D_SIZE, WORD ] INITIAL ( WORD (
;  2147  1
;  2148  1                            NEWB,                   ! BUFFER NOT USED IF 1
;  2149  1                            V,                      ! VALID ADDRESS IF 1
;  2150  1                            RCV_BUFFER,             ! RCV BUFFER ADDRESS
;  2151  1                            BYTE_COUNT,             ! 1/4 THE BYTE COUNT
;  2152  1                            0,                      ! STATUS WORD 1
;  2153  1                            0,                      ! STATUS WORD 2
;  2154  1
;  2155  1                            NEWB,                   ! BUFFER NOT USED IF 1
;  2156  1                            V,                      ! VALID ADDRESS IF 1
;  2157  1                            XMIT_BUFFER,            ! XMIT BUFFER ADDRESS
;  2158  1                            BYTE_COUNT,             ! 1/4 THE BYTE COUNT
;  2159  1                            0,                      ! STATUS WORD 1
;  2160  1                            0,                      ! STATUS WORD 2
;  2161  1
;  2162  1                            NEWB,                   ! BUFFER NOT USED IF 1
;  2163  1                            E,                      ! VALID ADDRESS IF 1
;  2164  1                            0,                      ! 2 EXTRA WORDS
;  2165  1                            0 )),                   !
;  2166  1
;  2167  1
;  2168  1        TD16: VECTOR [ 44, WORD ] INITIAL ( WORD (
;  2169  1
;  2170  1              NEWB, VL  , XMIT_BUFFER          , -1 , 0, 0,    ! 1 BYTE DESCRIPTOR
;  2171  1              NEWB, VHL , XMIT_BUFFER          , -2 , 0, 0,    ! 2 BYTE DESCRIPTOR
;  2172  1              NEWB, VH  , XMIT_BUFFER + 2      , -1 , 0, 0,    ! 1 BYTE DESCRIPTOR
;  2173  1              NEWB, VE  , XMIT_BUFFER + 4      , -1 , 0, 0,    ! 2 BYTE DESCRIPTOR
;  2174  1              NEWB, E   , XMIT_D_LIST + 60     , -1 , 0, 0,    ! END OF DESCRIPTOR
;  2175  1              NEWB, V   , XMIT_D_LIST + 56     , -2 , 0, 0,    ! 4 BYTE DESCRIPTOR
;  2176  1              NEWB, VE  , TARGET_ADR + 114     , -3 , 0, 0,    ! 6 BYTE DESCRIPTOR
;  2177  1              NEWB, E )),                                      ! END OF DESCRIPTOR
;  2178  1
;  2179  1        TD13: VECTOR [ 34, WORD ] INITIAL ( WORD (
;  2180  1
;  2181  1              NEWB, V   , XMIT_BUFFER          ,  -1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;  2182  1              NEWB, V   , XMIT_BUFFER + 2      ,-127 , 0, 0,   ! 378 BYTE DESCRIPTOR
;  2183  1              NEWB, V   , XMIT_BUFFER + 256    ,  -1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;  2184  1              NEWB, C   , XMIT_D_LIST + 48     ,  -1 , 0, 0,   ! CHAIN DESCRIPTOR
;  2185  1              NEWB, VE  , XMIT_BUFFER + 258    , -63 , 0, 0,   ! 2 BYTE DESCRIPTOR
;  2186  1              NEWB, E )),                                      ! END OF DESCRIPTOR
;  2187  1
```

```
;  2188  1          RD13: VECTOR [ 64, WORD ] INITIAL ( WORD (
;  2189  1
;  2190  1                NEWB, V  , RCV_BUFFER          ,   1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;  2191  1                NEWB, V  , RCV_BUFFER + 2      , -62 , 0, 0,   ! 124 BYTE DESCRIPTOR
;  2192  1                NEWB, V  , RCV_BUFFER + 126    ,  -1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;  2193  1                NEWB, V  , RCV_BUFFER + 128    ,  -2 , 0, 0,   ! 4 BYTE DESCRIPTOR
;  2194  1                NEWB, V  , RCV_BUFFER + 132    , -60 , 0, 0,   ! 120 BYTE DESCRIPTOR
;  2195  1                NEWB, V  , RCV_BUFFER + 252    ,  -2 , 0, 0,   ! 4 BYTE DESCRIPTOR
;  2196  1                NEWB, VC , RCV_D_LIST + 84     ,  -1 , 0, 0,   ! CHAIN DESCRIPTOR
;  2197  1                NEWB, V  , RCV_BUFFER + 256    ,  -3 , 0, 0,   ! 6 BYTE DESCRIPTOR
;  2198  1                NEWB, V  , RCV_BUFFER + 262    , -60 , 0, 0,   ! 120 BYTE DESCRIPTOR
;  2199  1                NEWB, V  , RCV_BUFFER + 382    ,  -1 , 0, 0,   ! 2 BYTE DESCRIPTOR
;  2200  1                NEWB, E )),                                    ! END OF DESCRIPTOR
;  2201  1
```

```
;  2202  1     !++
;  2203  1     !       HARDWARE AND SOFTWARE P-TABLE STORAGE DECLARATIONS
;  2204  1     !
;  2205  1
;  2206  1         HWP_TABLE    : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;  2207  1         SWP_TABLE    : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;  2208  1
;  2209  1         REG_ADR      : REF REG_STR FIELD ( IOP_FIELDS ),
;  2210  1         IOP_DATA     : REF REG_STR FIELD ( IOP_FIELDS ),
;  2211  1         GET_ADR      : REF ADR_STR FIELD ( IOP_FIELDS ),
;  2212  1
;  2213  1     !++
;  2214  1     !
;  2215  1     !       MISCELLANEOUS DATA DECLARATIONS
;  2216  1     !
;  2217  1     ! -
;  2218  1
;  2219  1         XBUF_LENGTH          : WORD,                 ! XMIT BUFFER LENGTH IN WORDS
;  2220  1         RBUF_LENGTH          : WORD,                 ! RCV BUFFER LENGTH IN BYTES
;  2221  1         INTERRUPT FLG        : WORD,                 ! 1 = INTERRUPT OCCURED
;  2222  1         DEQNA_NO             : WORD,                 ! DEQNA UNDER TEST THIS PASS
;  2223  1         COUNTER              : WORD,                 ! ITERATION COUNTER, INDEX
;  2224  1         UP_COUNTER           : WORD,                 ! ITERATION COUNTER, INDEX
;  2225  1         DOWN_COUNTER         : WORD,                 ! ITERATION COUNTER, INDEX
;  2226  1         CHECKSUM             : WORD,                 ! EXPECTED PROM CHECKSUM
;  2227  1         BUF_LENGTH           : WORD,                 ! XMIT BUFFER SIZE IN WORDS
;  2228  1         CSR_WORD             : WORD,                 !
;  2229  1         XC_FLAG              : WORD INITIAL (0),     !
;  2230  1         ERR_NUMBER           : WORD INITIAL (0),     !
;  2231  1         ERR_FLAG             : WORD INITIAL (0),     !
;  2232  1         ERR_COUNT            : WORD INITIAL (0),     !
;  2233  1         tmpr1                : word,                 !scratch var used by romchk
;  2234  1
```

J2


```
;    2235  1     !++
;    2236  1     !
;    2237  1     !   TEMPORARY STORAGE DATA DECLARATIONS
;    2238  1     !
;    2239  1     !--
;    2240  1
;    2241  1         TMP_IOP_ADR         : WORD,          ! I/O PAGE REGISTER ADDRESS
;    2242  1         TMP_REG_DATA        : WORD,          ! I/O PAGE REG CONTENTS
;    2243  1         TEMP1               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2244  1         TEMP2               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2245  1         TEMP3               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2246  1         TEMP4               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2247  1         TEMP5               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2248  1         TEMP6               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2249  1         TEMP7               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2250  1         TEMP8               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2251  1         TEMP9               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2252  1         P1                  : WORD,          ! PARAMETER #1
;    2253  1         P2                  : WORD,          ! PARAMETER #2
;    2254  1         P3                  : WORD,          ! PARAMETER #3
;    2255  1         P4                  : WORD,          ! PARAMETER #4
;    2256  1         P5                  : WORD,          ! PARAMETER #5
;    2257  1         TBYTE1              : BYTE,          !
;    2258  1         TBYTE2              : BYTE,          !
;    2259  1         TBYTE3              : BYTE,          !
;    2260  1         TBYTE4              : BYTE,          !
;    2261  1         TADR1               : WORD,          !
;    2262  1         TADR2               : WORD,          !
;    2263  1         LOGUN               : WORD;          !logical unit # for >1 devices
;    2264  1
```

```
;    2265  1
;    2266  1      %SBTTL 'GLOBAL TEXT SECTION'
;    2267  1
;    2268  1      !++
;    2269  1      !       THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS, MESSAGES,
;    2270  1      !       AND ASCII INFORMATION THAT IS USED IN MORE THAN ONE TEST.
;    2271  1      !-
;    2272  1
;    2273  1      GLOBAL BIND
;    2274  1
;    2275  1          DESCR_LIST    = RCV_D_LIST,
;    2276  1          DATA_BUFFER   = RCV_BUFFER,
;    2277  1
;    2278  1      !++
;    2279  1      !       HARDWARE AND SOFTWARE QUESTIONS
;    2280  1      !
;    2281  1
;    2282  1          QST01 = UPLIT (%ASCIZ'DEQNA I/O PAGE ADR     '),
;    2283  1          QST02 = UPLIT (%ASCIZ'INTERRUPT VECTOR ADR   '),
;    2284  1          QST03 = UPLIT (%ASCIZ'DO YOU WANT TO TEST SANITY TIMER '),
;    2285  1          QST04 = UPLIT (%ASCIZ'IS LOOPBACK CONNECTOR IN DEQNA   '),
;    2286  1          QST05 = UPLIT (%ASCIZ'SANITY TIMER TIME-OUT VALUE      '),
;    2287  1          QST06 = UPLIT (%ASCIZ'EXTERNAL LOOPBACK MODE           '),
;    2288  1          QST07 = UPLIT (%ASCIZ'SYSTEM HAS BLOCK-MODE MEMORY     '),
;    2289  1          QST10 = UPLIT (%ASCIZ'NXM TEST ? MUST HAVE < 4MB MEMORY'),
;    2290  1
;    2291  1
;    2292  1
;    2293  1      !++
;    2294  1      !       DEVICE ERROR MESSAGES
;    2295  1      !++
;    2296  1
;    2297  1          MSG00 = UPLIT (%ASCIZ' DEQNA FATAL ERROR DETECTED '),
;    2298  1          MSG01  = UPLIT (%ASCIZ'%N%N%A   DEQNA ADDRESS: %O6%A,  STATION ADDRESS: '),
;    2299  1          MSG02  = UPLIT (%ASCIZ'%A       ACTUAL DATA = %O6%A     EXPECTED DATA = %O6%N'),
;    2300  1          MSG03  = UPLIT (%ASCIZ'%A                                XMIT DESCRIPTOR    RCV DESCRIPTOR %N'),
;    2301  1          MSG04  = UPLIT (%ASCIZ'%A       FLAG WORD                %O6%A              %O6%N'),
;    2302  1          MSG05  = UPLIT (%ASCIZ'%A       ADDR DESC BITS/HIGH ADDR %O6%A              %O6%N'),
;    2303  1          MSG06  = UPLIT (%ASCIZ'%A       LOW  ORDER ADDR BITS     %O6%A              %O6%N'),
;    2304  1          MSG07  = UPLIT (%ASCIZ'%A       PACKET LENGTH ( WD )     %O6%A              %O6%N'),
;    2305  1          MSG08  = UPLIT (%ASCIZ'%A       STATUS WORD 1            %O6%A              %O6%N'),
;    2306  1          MSG09  = UPLIT (%ASCIZ'%A       STATUS WORD 2            %O6%A              %O6%N'),
;    2307  1          MSG10  = UPLIT (%ASCIZ'%A       DEQNA CSR REGISTER                  %O6%N'),
;    2308  1          MSG11  = UPLIT (%ASCIZ'%A       DEQNA I/O PAGE ADR                  %O6%N%N'),
;    2309  1          MSG12  = UPLIT (%ASCIZ'%A BAD CSR: ACT = %O6%A EXP = %O6%N'),
;    2310  1          MSG13  = UPLIT (%ASCIZ'%A BAD TRANSMIT FLAG WORD: ACT = %O6%A EXP = %O6%N'),
;    2311  1          MSG14  = UPLIT (%ASCIZ'%A BAD TRANSMIT STATUS WORD 1: ACT = %O6%A EXP = %O6%N'),
;    2312  1          MSG15  = UPLIT (%ASCIZ'%A BAD RECEIVE FLAG WORD: ACT = %O6%A EXP = %O6%N'),
;    2313  1          MSG16  = UPLIT (%ASCIZ'%A BAD RECEIVE STATUS WORD 1: ACT = %O6%A EXP = %O6%N'),
;    2314  1          MSG17  = UPLIT (%ASCIZ'%A BAD RECEIVE BUFFER LENGTH: ACT = %O6%A EXP = %O6%N'),
;    2315  1          MSG18  = UPLIT (%ASCIZ'%A BAD CSR = %O6%N'),
;    2316  1          MSG19  = UPLIT (%ASCIZ'%A LOOPBACK PACKET UNABLE TO SET CA BIT, CSR = %O6%N'),
;    2317  1          MSG20  = UPLIT (%ASCIZ'%A LOOPBACK PACKET UNABLE TO CLEAR CA BIT, CSR = %O6%N'),
```

L2


```
;    2318  1          MSG21  = UPLIT (%ASCIZ'%A CA BIT OK, BUT RI BIT IS NOT ON, CSR = %O6%N'),
;    2319  1          MSG22  = UPLIT (%ASCIZ'%A CA BIT IN THE CSR WAS SET TOO EARLY, CSR = %O6%N'),
;    2320  1          MSG23  = UPLIT (%ASCIZ'%A XL AND RL ( BITS 4,5 ) TO BE RESET TO 0%N'),
;    2321  1          MSG24  = UPLIT (%ASCIZ'%A XL AND RL ( BITS 4,5 ) TO BE SET TO 1%N'),
;    2322  1          MSG25  = UPLIT (%ASCIZ'%A RI ( BIT 15 ) TO BE SET TO 1%N'),
;    2323  1          MSG26  = UPLIT (%ASCIZ'%A XI ( BIT 7 ) TO BE SET TO 1%N'),
;    2324  1          MSG27  = UPLIT (%ASCIZ'%A NI ( BIT 2 ) TO BE SET TO 1%N'),
;    2325  1          MSG28  = UPLIT (%ASCIZ'%A NI ( BIT 2 ) TO BE RESET TO 0%N'),
;    2326  1          MSG29  = UPLIT (%ASCIZ'%A BAD CSR, EXPECTED'),
;    2327  1          MSG30  = UPLIT (%ASCIZ'%A CSR ADR = %O6%A  ACTUAL = %O6%A  EXPECTED = %O6%N'),
;    2328  1          MSG31  = UPLIT (%ASCIZ'%N%A UNABLE TO RESET DEQNA: ADR: %O6%A  CSR = %O6%N'),
;    2329  1          MSG32  = UPLIT (%ASCIZ'%N%A WAIT ABOUT %D2%A SECOND(S) -'),
;    2330  1          MSG33  = UPLIT (%ASCIZ'%N%A SANITY TIMER TIMED OUT AS EXPECTED %N'),
;    2331  1          MSG34  = UPLIT (%ASCIZ'%N%A NO SANITY TIMER INTERRUPT DETECTED %N'),
;    2332  1          MSG35  = UPLIT (%ASCIZ'%N%A DISCONNECT TRANSCEIVER CABLE FROM BULKHEAD ASSEMBLY AND'),
;    2333  1          MSG36  = UPLIT (%ASCIZ'%N%A CONNECT LOOPBACK CONNECTOR TO BULKHEAD ASSEMBLY, THEN RETEST%N'),
;    2334  1          MSG37  = UPLIT (%ASCIZ'%N%A DISCONNECT BULKHEAD ASSEMBLY FROM DEQNA AND CONNECT'),
;    2335  1          MSG38  = UPLIT (%ASCIZ'%N%A LOOPBACK CONNECTOR TO DEQNA, THEN RETEST%N'),
;    2336  1          MSG39  = UPLIT (%ASCIZ'%N%A CHECK FOR LOOSE WIRES IN A LOOPBACK CONNECTOR'),
;    2337  1          MSG40  = UPLIT (%ASCIZ'%N%A OR USE DIFFERENT LOOPBACK CONNECTOR, THEN RETEST%N'),
;    2338  1          MSG41  = UPLIT (%ASCIZ'%N%A REPLACE DEQNA, THEN RETEST%N'),
;    2339  1          MSG42  = UPLIT (%ASCIZ'%N%A REPLACE BULKHEAD CONNECTOR, THEN RETEST%N'),
;    2340  1          MSG43  = UPLIT (%ASCIZ'%N%A DISCONNECT TRANSCEIVER CABLE FROM TRANSCEIVER'),
;    2341  1          MSG44  = UPLIT (%ASCIZ'%N%A AND CONNECT IT TO LOOPBACK CONNECTOR AND BULKHEAD ASSEMBLY%N'),
;    2342  1          MSG45  = UPLIT (%ASCIZ'%N%A REPLACE TRANSCEIVER CABLE, THEN RETEST%N'),
;    2343  1          MSG46  = UPLIT (%ASCIZ'%N%A REPLACE TRANSCEIVER, THEN RETEST%N'),
;    2344  1          MSG47  = UPLIT (%ASCIZ'%N%A FUSE OK BIT IN CSR0 CLEAR, NO POWER TO XCVR?%N'),
;    2345  1          MSG48  = UPLIT (%ASCIZ'%N%A BAD RECEIVE DESCRIPTOR:'),
;    2346  1          MSG49  = UPLIT (%ASCIZ'%N%A BAD TRANSMIT DESCRIPTOR:'),
;    2347  1          MSG50  = UPLIT (%ASCIZ'%A ACTUAL = %O6%A EXPECTED = %O6%A INDEX = %D4%N'),
;    2348  1          MSG51  = UPLIT (%ASCIZ'%N%A BAD RECEIVE BUFFER:'),
;    2349  1          MSG52  = UPLIT (%ASCIZ'%N%A DMA OPERATION TAKES TOO LONG%N'),
;    2350  1          MSG53  = UPLIT (%ASCIZ'%N%A TOO MANY DEVICES%N'),
;    2351  1          MSG54  = UPLIT (%ASCIZ'%N%A THERE WAS A POWER FAIL - WAITING%N'),
;    2352  1          MSG55  = UPLIT (%ASCIZ'%N%A WAIT ABOUT %D2%A MINUTE(S) -'),
;    2353  1          MSG56  = UPLIT (%ASCIZ'%N%A WAIT ABOUT %D2%A HOUR -'),
;    2354  1          MSG57  = UPLIT (%ASCIZ'%A IF NO RESET, TYPE ANY CHARACTER TO EXIT FROM TEST%N'),
;    2355  1          MSG58  = UPLIT (%ASCIZ'%N%A TDR VALUE IS EQUAL TO ZERO %N'),
;    2356  1          MSG59  = UPLIT (%ASCIZ'%N%N%A---------------------------------------------------- ------ --       %N'),
```

M2


```
;    2357  1          MSG60 = UPLIT (%ASCIZ'%N%A BAD CSR, BITS STUCK AT 0:%N'),
;    2358  1          MSG61 = UPLIT (%ASCIZ'%N%A BAD CSR, BITS STUCK AT 1:%N'),
;    2359  1          MSG62 = UPLIT (%ASCIZ'%N%A SOFTWARE RESET UNABLE TO CLEAR CSR STATIC BITS:%N'),
;    2360  1          MSG63 = UPLIT (%ASCIZ'%N%A BAD STATION ADDRESS CHECKSUM: ACT = %O6%A EXP = %O6%N'),
;    2361  1          MSG64 = UPLIT (%ASCIZ'%N%A BAD STATION ADDRESS: '),
;    2362  1          MSG65 = UPLIT (%ASCIZ'%N%A BAD DEQNA I/O PAGE REGISTER:%N'),
;    2363  1          MSG66 = UPLIT (%ASCIZ'%N%A BAD CSR, EXPECTED RL ( BIT 5 ) TO BE SET TO 0%N'),
;    2364  1          MSG67 = UPLIT (%ASCIZ'%N%A BAD B/D PROM CHECKSUM: INDEX = %O6%A ACT = %O6%A EXP = %O6%N'),
;    2365  1          MSG68 = UPLIT (%ASCIZ'%N%A B/D PROM CHECKSUM OFFSET = %O6%A ACT = %O6%A EXP = %O6%N'),
;    2366  1          MSG69 = UPLIT (%ASCIZ'%N%A BAD INTERRUPT: ADR = %O6%A ACT LEV = %O6%A EXP LEV = %O6%N'),
;    2367  1          MSG70 = UPLIT (%ASCIZ'%N%A REGISTER FAILED TO RESPOND AT ADDRESS:  %O6%N'),
;    2368  1          MSG71 = UPLIT (%ASCIZ'%N%A BAD TRANSMIT STATUS, TOO MANY COLLISIONS%N'),
;    2369  1          msg72 = UPLIT (%ASCIZ'%N%A DEVICE FAILED TO INTERRUPT: CPU PRIORITY = %O6%N'),
;    2370  1          msg73 = UPLIT (%ASCIZ'%N%A UNEXPECTED DEVICE INTERRUPT: CPU PRIORITY = %O6%N'),
;    2371  1          msg74 = UPLIT (%ASCIZ'%N%A FAILURE IN EXTERNAL LOOPBACK MODE %N'),
;    2372  1          msg75 = UPLIT (%ASCIZ'%N%A Rcv Desc Base = %O6%A INDEX = %D4%A Actual = %O6%N'),
;    2373  1          msg76 = UPLIT (%ASCIZ'%N%A Tx Desc Base - %O6%A INDEX = %D4%A Actual = %O6%N');
;    2374  1
```

N2


```
;   2375  1     %SBTTL  DEFAULT HARDWARE P-TABLE'
;   2376  1
;   2377  1     BGNHW ( HP_TABLE );
;   2378  1
;   2379  1     !++
;   2380  1     !       THE DEFAULT HARDWARE P TABLE CONTAINS DEFAULT VALUES OF THE
;   2381  1     !       TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE IS IDENTICAL TO
;   2382  1     !       THE STRUCTURE OF THE HARDWARE P-TABLES, AND IS USED AS A "TEMPLATE"
;   2383  1     !       FOR BUILDING THE P TABLES.
;   2384  1     !
;   2385  1     !
;   2386  1     !       PLACE YOUR DEFAULT HARDWARE P-TABLE HERE.  THE VALUES AND
;   2387  1     !       SIZE WILL BE USED AS A "TEMPLATE" FOR CREATING ACTUAL P-TABLE
;   2388  1     !       ENTRIES AND THE DEFAULT VALUES IN THE OPERATOR DIALOGUE.
;   2389  1     !       THE ACTUAL P-TABLE BUILT AT RUNTIME IS STORED IN SUPERVISOR
;   2390  1     !       SPACE.
;   2391  1     !-
;   2392  1
;   2393  1         GLOBAL
;   2394  1             DFSTBL  : BLOCK [ HWP_SIZE, WORD ] INITIAL ( %O'174440', %O'700' );
;   2395  1     ENDHW;
;   2396  1
;   2397  1
```

B3


```
;   2398  1     %SBTTL 'SOFTWARE P-TABLE'
;   2399  1
;   2400  1     !++
;   2401  1     !       THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
;   2402  1     !       PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
;   2403  1     !       SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
;   2404  1     !       AT RUN TIME.
;   2405  1     !
;   2406  1     !
;   2407  1     !       PLACE YOUR SOFTWARE P-TABLE HERE,USING GLOBAL OR OWN DECLARATIONS
;   2408  1     !       THIS TABLE IS NOT OPTIONAL.  THIS TABLE, UNLIKE THE HARDWARE TABLE,
;   2409  1     !       WILL CONTAIN THE ACTUAL VALUES ENTERED BY THE OPERATOR.
;   2410  1     !--
;   2411  1
;   2412  1     BGNSW ( SP_TABLE );
;   2413  1
;   2414  1        GLOBAL
;   2415  1             SWP_TIMER       : WORD INITIAL ( NO ),  ! NO SANITY TIMER TEST
;   2416  1             SWP_LBC         : WORD INITIAL ( NO ),  ! NO LOOPBACK IN DEQNA
;   2417  1             SWP_TOUT_VAL    : WORD INITIAL ( 3 ),   ! TIMEOUT VALUE = 16 SEC.
;   2418  1             SWP_ILOOP       : WORD INITIAL ( NO ),  ! EXTERNAL LOOPBACK MODE
;   2419  1             SWP_BLOCK_MEM   : WORD INITIAL ( YES ), ! BLOCK-MODE MEMORY PRESENT
;   2420  1             SWP_NXM         : WORD INITIAL ( NO );  ! do NXM test 12 < max memory
;   2421  1
;   2422  1     ENDSW;
;   2423  1
;   2424  1
```

```
;   2425  1     %SBTTL 'PROTECTION TABLE'
;   2426  1
;   2427  1     !++
;   2428  1     !       THIS TABLE IS USED BY THE RUNTIME SERVICES TO PROTECT THE LOAD MEDIA.
;   2429  1     !
;   2430  1     !       1ST ARG =       OFFSET INTO P-TABLE FOR CSR ADDRESS
;   2431  1     !       2ND ARG =       OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;   2432  1     !       3RD ARG =       OFFSET INTO P-TABLE FOR DRIVE NUMBER
;   2433  1     !
;   2434  1     !       INSERT BYTE OFFSET FOR DATA NOTED IN COMMENTS ABOVE.  (OFFSET
;   2435  1     !       REFERS TO THE NUMBER OF BYTES FROM THE BEGINNING OF A PTABLE
;   2436  1     !       ENTRY TO THE ITEM IN QUESTION.)  IF THE PARTICULAR
;   2437  1     !       ITEM DOES NOT APPLY, LEAVE ENTRY AS -1.  WHEN THE RUNTIME
;   2438  1     !       SERVICES EXECUTES A GPHARD, IT USES THESE OFFSETS (IF NOT
;   2439  1     !       SET TO -1) TO GET THE ITEMS AND COMPARE WITH THOSE SAVED
;   2440  1     !       IN THE XXDP+ MONITOR.  IF THE UNIT BEING REQUESTED MATCHES THE
;   2441  1     !       LOAD DEVICE, THE RUNTIME SERVICES RETURN AN INCOMPLETE FLAG ON
;   2442  1     !       THE GPHARD.
;   2443  1     !--
;   2444  1
;   2445  1     BGNPROT (-1, -1, -1);
;   2446  1
;   2447  1     ENDPROT;
;   2448  1
;   2449  1
;   2450  1
;   2451  1     END
;   2452  0     ELUDOM


                                        .TITLE  ZQNA1 CZQNAEO 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     105                          .ASCII  /E/
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$AUTO
000112  000236'                         L$PRT:: .WORD   L$PROT
000114  000000                          L$TEST::.WORD   0
000116  000000                          L$DLY:: .WORD   0
000120  000000                          L$HIME::.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$SWLEN>/2>
000220  000000                          SWP.TIMER::
                                                .WORD   0
000222  000000                          SWP.LBC::
                                                .WORD   0
000224  000003                          SWP.TOUT.VAL::
                                                .WORD   3
000226  000000                          SWP.ILOOP::
                                                .WORD   0
000230  000001                          SWP.BLOCK.MEM::
                                                .WORD   1
000232  000000                          SWP.NXM::
                                                .WORD   0
000234                                  L$NDSW::.BLKW   1
000236  177777                          L$PROT::.WORD   -1
000240  177777                                  .WORD   -1
000242  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/
```

F3


```
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    116    130    115        P.AAH:  .ASCII  /NXM/
```

```
000335  040  124  105                .ASCII  / TE/
000340  123  124  040                .ASCII  /ST /
000343  077  040  115                .ASCII  /? M/
000346  125  123  124                .ASCII  /UST/
000351  040  110  101                .ASCII  / HA/
000354  126  105  040                .ASCII  /VE /
000357  074  040  064                .ASCII  /< 4/
000362  115  102  040                .ASCII  /MB /
000365  115  105  115                .ASCII  /MEM/
000370  117  122  131                .ASCII  /ORY/
000373  000                         .ASCII  <00>
000374  040  104  105       P.AAI:   .ASCII  / DE/
000377  121  116  101                .ASCII  /QNA/
000402  040  106  101                .ASCII  / FA/
000405  124  101  114                .ASCII  /TAL/
000410  040  105  122                .ASCII  / ER/
000413  122  117  122                .ASCII  /ROR/
000416  040  104  105                .ASCII  / DE/
000421  124  105  103                .ASCII  /TEC/
000424  124  105  104                .ASCII  /TED/
000427  040  000  000                .ASCII  / /<00><00>
000432  045  116  045       P.AAJ:   .ASCII  /%N%/
000435  116  045  101                .ASCII  /N%A/
000440  040  040  040                .ASCII  /   /
000443  104  105  121                .ASCII  /DEQ/
000446  116  101  040                .ASCII  /NA /
000451  101  104  104                .ASCII  /ADD/
000454  122  105  123                .ASCII  /RES/
000457  123  072  040                .ASCII  /S: /
000462  045  117  066                .ASCII  /%O6/
000465  045  101  054                .ASCII  /%A,/
000470  040  040  123                .ASCII  /  S/
000473  124  101  124                .ASCII  /TAT/
000476  111  117  116                .ASCII  /ION/
000501  040  101  104                .ASCII  / AD/
000504  104  122  105                .ASCII  /DRE/
000507  123  123  072                .ASCII  /SS:/
000512  040  000                     .ASCII  / /<00>
000514  045  101  040       P.AAK:   .ASCII  /%A /
000517  040  040  040                .ASCII  /   /
000522  040  040  101                .ASCII  /  A/
000525  103  124  125                .ASCII  /CTU/
000530  101  114  040                .ASCII  /AL /
000533  104  101  124                .ASCII  /DAT/
000536  101  040  075                .ASCII  /A =/
000541  040  045  117                .ASCII  / %O/
000544  066  045  101                .ASCII  /6%A/
000547  040  040  040                .ASCII  /   /
000552  040  040  105                .ASCII  /  E/
000555  130  120  105                .ASCII  /XPE/
000560  103  124  105                .ASCII  /CTE/
000563  104  040  104                .ASCII  /D D/
000566  101  124  101                .ASCII  /ATA/
```

H3


```
000571  040  075  040            .ASCII  / = /
000574  045  117  066            .ASCII  /%06/
000577  045  116  000            .ASCII  /%N/<00>
000602  045  101  040  P.AAL:    .ASCII  /%A /
000605  040  040  040            .ASCII  /   /
000610  040  040  040            .ASCII  /   /
000613  040  040  040            .ASCII  /   /
000616  040  040  040            .ASCII  /   /
000621  040  040  040            .ASCII  /   /
000624  040  040  040            .ASCII  /   /
000627  040  040  040            .ASCII  /   /
000632  040  040  040            .ASCII  /   /
000635  040  040  040            .ASCII  /   /
000640  130  115  111            .ASCII  /XMI/
000643  124  040  104            .ASCII  /T D
000646  105  123  103            .ASCII  /ESC/
000651  122  111  120            .ASCII  /RIP/
000654  124  117  122            .ASCII  /TOR/
000657  040  040  040            .ASCII  /   /
000662  040  122  103            .ASCII  / RC/
000665  126  040  104            .ASCII  /V D/
000670  105  123  103            .ASCII  /ESC/
000673  122  111  120            .ASCII  /RIP/
000676  124  117  122            .ASCII  /TOR/
000701  040  045  116            .ASCII  / %N/
000704  000  000                 .ASCII  <00><00>
000706  045  101  040  P.AAM:    .ASCII  /%A /
000711  040  040  040            .ASCII  /   /
000714  040  040  106            .ASCII  /  F/
000717  114  101  107            .ASCII  /LAG/
000722  040  127  117            .ASCII  / WO/
000725  122  104  040            .ASCII  /RD /
000730  040  040  040            .ASCII  /   /
000733  040  040  040            .ASCII  /   /
000736  040  040  040            .ASCII  /   /
000741  040  040  040            .ASCII  /   /
000744  040  040  040            .ASCII  /   /
000747  040  040  045            .ASCII  /  %/
000752  117  066  045            .ASCII  /06%/
000755  101  040  040            .ASCII  /A  /
000760  040  040  040            .ASCII  /   /
000763  040  040  040            .ASCII  /   /
000766  040  040  040            .ASCII  /   /
000771  040  045  117            .ASCII  / %O/
000774  066  045  116            .ASCII  /6%N/
000777  000                      .ASCII  <00>
001000  045  101  040  P.AAN:    .ASCII  /%A /
001003  040  040  040            .ASCII  /   /
001006  040  040  101            .ASCII  /  A/
001011  104  104  122            .ASCII  /DDR/
001014  040  104  105            .ASCII  / DE/
001017  123  103  040            .ASCII  /SC /
001022  102  111  124            .ASCII  /BIT/
```

```
001025  123  057  110                .ASCII  /S/<57>/H/
001030  111  107  110                .ASCII  /IGH/
001033  040  101  104                .ASCII  / AD/
001036  104  122  040                .ASCII  /DR /
001041  040  040  045                .ASCII  /  %/
001044  117  066  045                .ASCII  /06%/
001047  101  040  040                .ASCII  /A  /
001052  040  040  040                .ASCII  /   /
001055  040  040  040                .ASCII  /   /
001060  040  040  040                .ASCII  /   /
001063  040  045  117                .ASCII  / %O/
001066  066  045  116                .ASCII  /6%N/
001071  000                        .ASCII  <00>
001072  045  101  040        P.AAO:  .ASCII  /%A /
001075  040  040  040                .ASCII  /   /
001100  040  040  114                .ASCII  /  L/
001103  117  127  040                .ASCII  /OW /
001106  040  117  122                .ASCII  / OR/
001111  104  105  122                .ASCII  /DER/
001114  040  101  104                .ASCII  / AD/
001117  104  122  040                .ASCII  /DR /
001122  102  111  124                .ASCII  /BIT/
001125  123  040  040                .ASCII  /S  /
001130  040  040  040                .ASCII  /   /
001133  040  040  045                .ASCII  /  %/
001136  117  066  045                .ASCII  /06%/
001141  101  040  040                .ASCII  /A  /
001144  040  040  040                .ASCII  /   /
001147  040  040  040                .ASCII  /   /
001152  040  040  040                .ASCII  /   /
001155  040  045  117                .ASCII  / %O/
001160  066  045  116                .ASCII  /6%N/
001163  000                        .ASCII  <00>
001164  045  101  040        P.AAP:  .ASCII  /%A /
001167  040  040  040                .ASCII  /   /
001172  040  040  120                .ASCII  /  P/
001175  101  103  113                .ASCII  /ACK/
001200  105  124  040                .ASCII  /ET /
001203  114  105  116                .ASCII  /LEN/
001206  107  124  110                .ASCII  /GTH/
001211  040  050  040                .ASCII  / ( /
001214  127  104  040                .ASCII  /WD /
001217  051  040  040                .ASCII  /)  /
001222  040  040  040                .ASCII  /   /
001225  040  040  045                .ASCII  /  %/
001230  117  066  045                .ASCII  /06%/
001233  101  040  040                .ASCII  /A  /
001236  040  040  040                .ASCII  /   /
001241  040  040  040                .ASCII  /   /
001244  040  040  040                .ASCII  /   /
001247  040  045  117                .ASCII  / %O/
001252  066  045  116                .ASCII  /6%N/
001255  000                        .ASCII  <00>
```

J3


```
001256  045  101  040  P.AAQ:  .ASCII  /%A /
001261  040  040  040          .ASCII  /   /
001264  040  040  123          .ASCII  /  S/
001267  124  101  124          .ASCII  /TAT/
001272  125  123  040          .ASCII  /US /
001275  127  117  122          .ASCII  /WOR/
001300  104  040  061          .ASCII  /D 1/
001303  040  040  040          .ASCII  /   /
001306  040  040  040          .ASCII  /   /
001311  040  040  040          .ASCII  /   /
001314  040  040  040          .ASCII  /   /
001317  040  040  045          .ASCII  /  %/
001322  117  066  045          .ASCII  /06%/
001325  101  040  040          .ASCII  /A  /
001330  040  040  040          .ASCII  /   /
001333  040  040  040          .ASCII  /   /
001336  040  040  040          .ASCII  /   /
001341  040  045  117          .ASCII  / %O/
001344  066  045  116          .ASCII  /6%N/
001347  000                    .ASCII  <00>
001350  045  101  040  P.AAR:  .ASCII  /%A /
001353  040  040  040          .ASCII  /   /
001356  040  040  123          .ASCII  /  S/
001361  124  101  124          .ASCII  /TAT/
001364  125  123  040          .ASCII  /US /
001367  127  117  122          .ASCII  /WOR/
001372  104  040  062          .ASCII  /D 2/
001375  040  040  040          .ASCII  /   /
001400  040  040  040          .ASCII  /   /
001403  040  040  040          .ASCII  /   /
001406  040  040  040          .ASCII  /   /
001411  040  040  045          .ASCII  /  %/
001414  117  066  045          .ASCII  /06%/
001417  101  040  040          .ASCII  /A  /
001422  040  040  040          .ASCII  /   /
001425  040  040  040          .ASCII  /   /
001430  040  040  040          .ASCII  /   /
001433  040  045  117          .ASCII  / %O/
001436  066  045  116          .ASCII  /6%N/
001441  000                    .ASCII  <00>
001442  045  101  040  P.AAS:  .ASCII  /%A /
001445  040  040  040          .ASCII  /   /
001450  040  040  104          .ASCII  /  D/
001453  105  121  116          .ASCII  /EQN/
001456  101  040  103          .ASCII  /A C/
001461  123  122  040          .ASCII  /SR /
001464  122  105  107          .ASCII  /REG/
001467  111  123  124          .ASCII  /IST/
001472  105  122  040          .ASCII  /ER /
001475  040  040  040          .ASCII  /   /
001500  040  040  040          .ASCII  /   /
001503  040  040  040          .ASCII  /   /
001506  040  040  040          .ASCII  /   /
```

K3


```
001511  040  040  040            .ASCII  /   /
001514  040  040  045            .ASCII  /  %/
001517  117  066  045            .ASCII  /06%/
001522  116  000                 .ASCII  /N/<00>
001524  045  101  040  P.AAT:    .ASCII  /%A /
001527  040  040  040            .ASCII  /   /
001532  040  040  104            .ASCII  /  D/
001535  105  121  116            .ASCII  /EQN/
001540  101  040  111            .ASCII  /A I/
001543  057  117  040            .ASCII  <57>/O /
001546  120  101  107            .ASCII  /PAG/
001551  105  040  101            .ASCII  /E A/
001554  104  122  040            .ASCII  /DR /
001557  040  040  040            .ASCII  /   /
001562  040  040  040            .ASCII  /   /
001565  040  040  040            .ASCII  /   /
001570  040  040  040            .ASCII  /   /
001573  040  040  040            .ASCII  /   /
001576  040  040  045            .ASCII  /  %/
001601  117  066  045            .ASCII  /06%/
001604  116  045  116            .ASCII  /N%N/
001607  000                      .ASCII  <00>
001610  045  101  040  P.AAU:    .ASCII  /%A /
001613  102  101  104            .ASCII  /BAD/
001616  040  103  123            .ASCII  / CS/
001621  122  072  040            .ASCII  /R: /
001624  101  103  124            .ASCII  /ACT/
001627  040  075  040            .ASCII  / = /
001632  045  117  066            .ASCII  /%06/
001635  045  101  040            .ASCII  /%A /
001640  105  130  120            .ASCII  /EXP/
001643  040  075  040            .ASCII  / = /
001646  045  117  066            .ASCII  /%06/
001651  045  116  000            .ASCII  /%N/<00>
001654  045  101  040  P.AAV:    .ASCII  /%A /
001657  102  101  104            .ASCII  /BAD/
001662  040  124  122            .ASCII  / TR/
001665  101  116  123            .ASCII  /ANS/
001670  115  111  124            .ASCII  /MIT/
001673  040  106  114            .ASCII  / FL/
001676  101  107  040            .ASCII  /AG /
001701  127  117  122            .ASCII  /WOR/
001704  104  072  040            .ASCII  /D: /
001707  101  103  124            .ASCII  /ACT/
001712  040  075  040            .ASCII  / = /
001715  045  117  066            .ASCII  /%06/
001720  045  101  040            .ASCII  /%A /
001723  105  130  120            .ASCII  /EXP/
001726  040  075  040            .ASCII  / = /
001731  045  117  066            .ASCII  /%06/
001734  045  116  000            .ASCII  /%N/<00>
001737  000                      .ASCII  <00>
001740  045  101  040  P.AAW:    .ASCII  /%A /
```

L3


```
001743  102  101  104              .ASCII  /BAD/
001746  040  124  122              .ASCII  / TR/
001751  101  116  123              .ASCII  /ANS/
001754  115  111  124              .ASCII  /MIT/
001757  040  123  124              .ASCII  / ST/
001762  101  124  125              .ASCII  /ATU/
001765  123  040  127              .ASCII  /S W/
001770  117  122  104              .ASCII  /ORD/
001773  040  061  072              .ASCII  / 1:/
001776  040  101  103              .ASCII  / AC/
002001  124  040  075              .ASCII  /T =/
002004  040  045  117              .ASCII  / %O/
002007  066  045  101              .ASCII  /6%A/
002012  040  105  130              .ASCII  / EX/
002015  120  040  075              .ASCII  /P =/
002020  040  045  117              .ASCII  / %O/
002023  066  045  116              .ASCII  /6%N/
002026  000  000                   .ASCII  <00><00>
002030  045  101  040    P.AAX:    .ASCII  /%A /
002033  102  101  104              .ASCII  /BAD/
002036  040  122  105              .ASCII  / RE/
002041  103  105  111              .ASCII  /CEI/
002044  126  105  040              .ASCII  /VE /
002047  106  114  101              .ASCII  /FLA/
002052  107  040  127              .ASCII  /G W/
002055  117  122  104              .ASCII  /ORD/
002060  072  040  101              .ASCII  /: A/
002063  103  124  040              .ASCII  /CT /
002066  075  040  045              .ASCII  /= %/
002071  117  066  045              .ASCII  /O6%/
002074  101  040  105              .ASCII  /A E/
002077  130  120  040              .ASCII  /XP /
002102  075  040  045              .ASCII  /= %/
002105  117  066  045              .ASCII  /O6%/
002110  116  000                   .ASCII  /N/<00>
002112  045  101  040    P.AAY:    .ASCII  /%A /
002115  102  101  104              .ASCII  /BAD/
002120  040  122  105              .ASCII  / RE/
002123  103  105  111              .ASCII  /CEI/
002126  126  105  040              .ASCII  /VE /
002131  123  124  101              .ASCII  /STA/
002134  124  125  123              .ASCII  /TUS/
002137  040  127  117              .ASCII  / WO/
002142  122  104  040              .ASCII  /RD /
002145  061  072  040              .ASCII  /1: /
002150  101  103  124              .ASCII  /ACT/
002153  040  075  040              .ASCII  / = /
002156  045  117  066              .ASCII  /%O6/
002161  045  101  040              .ASCII  /%A /
002164  105  130  120              .ASCII  /EXP/
002167  040  075  040              .ASCII  / = /
002172  045  117  066              .ASCII  /%O6/
002175  045  116  000              .ASCII  /%N/<00>
```

M3


```
002200  045  101  040        P.AAZ:  .ASCII  /%A /
002203  102  101  104                .ASCII  /BAD/
002206  040  122  105                .ASCII  / RE/
002211  103  105  111                .ASCII  /CEI/
002214  126  105  040                .ASCII  /VE /
002217  102  125  106                .ASCII  /BUF/
002222  106  105  122                .ASCII  /FER/
002225  040  114  105                .ASCII  / LE/
002230  116  107  124                .ASCII  /NGT/
002233  110  072  040                .ASCII  /H: /
002236  101  103  124                .ASCII  /ACT/
002241  040  075  040                .ASCII  / = /
002244  045  117  066                .ASCII  /%06/
002247  045  101  040                .ASCII  /%A /
002252  105  130  120                .ASCII  /EXP/
002255  040  075  040                .ASCII  / = /
002260  045  117  066                .ASCII  /%06/
002263  045  116  000                .ASCII  /%N/<00>
002266  045  101  040        P.ABA:  .ASCII  /%A /
002271  102  101  104                .ASCII  /BAD/
002274  040  103  123                .ASCII  / CS/
002277  122  040  075                .ASCII  /R =/
002302  040  045  117                .ASCII  / %O/
002305  066  045  116                .ASCII  /6%N/
002310  000  000                     .ASCII  <00><00>
002312  045  101  040        P.ABB:  .ASCII  /%A /
002315  114  117  117                .ASCII  /LOO/
002320  120  102  101                .ASCII  /PBA/
002323  103  113  040                .ASCII  /CK /
002326  120  101  103                .ASCII  /PAC/
002331  113  105  124                .ASCII  /KET/
002334  040  125  116                .ASCII  / UN/
002337  101  102  114                .ASCII  /ABL/
002342  105  040  124                .ASCII  /E T/
002345  117  040  123                .ASCII  /O S/
002350  105  124  040                .ASCII  /ET /
002353  103  101  040                .ASCII  /CA /
002356  102  111  124                .ASCII  /BIT/
002361  054  040  103                .ASCII  /, C/
002364  123  122  040                .ASCII  /SR /
002367  075  040  045                .ASCII  /= %/
002372  117  066  045                .ASCII  /06%/
002375  116  000  000                .ASCII  /N/<00><00>
002400  045  101  040        P.ABC:  .ASCII  /%A /
002403  114  117  117                .ASCII  /LOO/
002406  120  102  101                .ASCII  /PBA/
002411  103  113  040                .ASCII  /CK /
002414  120  101  103                .ASCII  /PAC/
002417  113  105  124                .ASCII  /KET/
002422  040  125  116                .ASCII  / UN/
002425  101  102  114                .ASCII  /ABL/
002430  105  040  124                .ASCII  /E T/
002433  117  040  103                .ASCII  /O C/
```

N3


```
002436  114  105  101                .ASCII  /LEA/
002441  122  040  103                .ASCII  /R C/
002444  101  040  102                .ASCII  /A B/
002447  111  124  054                .ASCII  /IT,/
002452  040  103  123                .ASCII  / CS/
002455  122  040  075                .ASCII  /R =/
002460  040  045  117                .ASCII  / %O/
002463  066  045  116                .ASCII  /6%N/
002466  000  000                     .ASCII  <00><00>
002470  045  101  040      P.ABD:    .ASCII  /%A /
002473  103  101  040                .ASCII  /CA /
002476  102  111  124                .ASCII  /BIT/
002501  040  117  113                .ASCII  / OK/
002504  054  040  102                .ASCII  /, B/
002507  125  124  040                .ASCII  /UT /
002512  122  111  040                .ASCII  /RI /
002515  102  111  124                .ASCII  /BIT/
002520  040  111  123                .ASCII  / IS/
002523  040  116  117                .ASCII  / NO/
002526  124  040  117                .ASCII  /T O/
002531  116  054  040                .ASCII  /N, /
002534  103  123  122                .ASCII  /CSR/
002537  040  075  040                .ASCII  / = /
002542  045  117  066                .ASCII  /%O6/
002545  045  116  000                .ASCII  /%N/<00>
002550  045  101  040      P.ABE:    .ASCII  /%A /
002553  103  101  040                .ASCII  /CA /
002556  102  111  124                .ASCII  /BIT/
002561  040  111  116                .ASCII  / IN/
002564  040  124  110                .ASCII  / TH/
002567  105  040  103                .ASCII  /E C/
002572  123  122  040                .ASCII  /SR /
002575  127  101  123                .ASCII  /WAS/
002600  040  123  105                .ASCII  / SE/
002603  124  040  124                .ASCII  /T T/
002606  117  117  040                .ASCII  /OO /
002611  105  101  122                .ASCII  /EAR/
002614  114  131  054                .ASCII  /LY,/
002617  040  103  123                .ASCII  / CS/
002622  122  040  075                .ASCII  /R =/
002625  040  045  117                .ASCII  / %O/
002630  066  045  116                .ASCII  /6%N/
002633  000                         .ASCII  <00>
002634  045  101  040      P.ABF:    .ASCII  /%A /
002637  130  114  040                .ASCII  /XL /
002642  101  116  104                .ASCII  /AND/
002645  040  122  114                .ASCII  / RL/
002650  040  050  040                .ASCII  / ( /
002653  102  111  124                .ASCII  /BIT/
002656  123  040  064                .ASCII  /S 4/
002661  054  065  040                .ASCII  /,5 /
002664  051  040  124                .ASCII  /) T/
002667  117  040  102                .ASCII  /O B/
```

```
002672  105  040  122            .ASCII  /E R/
002675  105  123  105            .ASCII  /ESE/
002700  124  040  124            .ASCII  /T T/
002703  117  040  060            .ASCII  /O 0/
002706  045  116  000            .ASCII  /%N/<00>
002711  000                      .ASCII  <00>
002712  045  101  040  P.ABG:    .ASCII  /%A /
002715  130  114  040            .ASCII  /XL /
002720  101  116  104            .ASCII  /AND/
002723  040  122  114            .ASCII  / RL/
002726  040  050  040            .ASCII  / ( /
002731  102  111  124            .ASCII  /BIT/
002734  123  040  064            .ASCII  /S 4/
002737  054  065  040            .ASCII  /,5 /
002742  051  040  124            .ASCII  /) T/
002745  117  040  102            .ASCII  /O B/
002750  105  040  123            .ASCII  /E S/
002753  105  124  040            .ASCII  /ET /
002756  124  117  040            .ASCII  /TO /
002761  061  045  116            .ASCII  /1%N/
002764  000  000                 .ASCII  <00><00>
002766  045  101  040  P.ABH:    .ASCII  /%A /
002771  122  111  040            .ASCII  /RI /
002774  050  040  102            .ASCII  /( B/
002777  111  124  040            .ASCII  /IT /
003002  061  065  040            .ASCII  /15 /
003005  051  040  124            .ASCII  /) T/
003010  117  040  102            .ASCII  /O B/
003013  105  040  123            .ASCII  /E S/
003016  105  124  040            .ASCII  /ET /
003021  124  117  040            .ASCII  /TO /
003024  061  045  116            .ASCII  /1%N/
003027  000                      .ASCII  <00>
003030  045  101  040  P.ABI:    .ASCII  /%A /
003033  130  111  040            .ASCII  /XI /
003036  050  040  102            .ASCII  /( B/
003041  111  124  040            .ASCII  /IT /
003044  067  040  051            .ASCII  /7 )/
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  123  105            .ASCII  / SE/
```

C4


```
003122  124  040  124              .ASCII  /T T/
003125  117  040  061              .ASCII  /O 1/
003130  045  116  000              .ASCII  /%N/<00>
003133  000                       .ASCII  <00>
003134  045  101  040  P.ABK:     .ASCII  /%A /
003137  116  111  040              .ASCII  /NI /
003142  050  040  102              .ASCII  /( B/
003145  111  124  040              .ASCII  /IT /
003150  062  040  051              .ASCII  /2 )/
003153  040  124  117              .ASCII  / TO/
003156  040  102  105              .ASCII  / BE/
003161  040  122  105              .ASCII  / RE/
003164  123  105  124              .ASCII  /SET/
003167  040  124  117              .ASCII  / TO/
003172  040  060  045              .ASCII  / 0%/
003175  116  000  000              .ASCII  /N/<00><00>
003200  045  101  040  P.ABL:     .ASCII  /%A /
003203  102  101  104              .ASCII  /BAD/
003206  040  103  123              .ASCII  / CS/
003211  122  054  040              .ASCII  /R, /
003214  105  130  120              .ASCII  /EXP/
003217  105  103  124              .ASCII  /ECT/
003222  105  104  000              .ASCII  /ED/<00>
003225  000                       .ASCII  <00>
003226  045  101  040  P.ABM:     .ASCII  /%A /
003231  103  123  122              .ASCII  /CSR/
003234  040  101  104              .ASCII  / AD/
003237  122  040  075              .ASCII  /R =/
003242  040  045  117              .ASCII  / %O/
003245  066  045  101              .ASCII  /6%A/
003250  040  040  101              .ASCII  /  A/
003253  103  124  125              .ASCII  /CTU/
003256  101  114  040              .ASCII  /AL /
003261  075  040  045              .ASCII  /= %/
003264  117  066  045              .ASCII  /O6%/
003267  101  040  040              .ASCII  /A  /
003272  105  130  120              .ASCII  /EXP/
003275  105  103  124              .ASCII  /ECT/
003300  105  104  040              .ASCII  /ED /
003303  075  040  045              .ASCII  /= %/
003306  117  066  045              .ASCII  /O6%/
003311  116  000  000              .ASCII  /N/<00><00>
003314  045  116  045  P.ABN:     .ASCII  /%N%/
003317  101  040  125              .ASCII  /A U/
003322  116  101  102              .ASCII  /NAB/
003325  114  105  040              .ASCII  /LE /
003330  124  117  040              .ASCII  /TO /
003333  122  105  123              .ASCII  /RES/
003336  105  124  040              .ASCII  /ET /
003341  104  105  121              .ASCII  /DEQ/
003344  116  101  072              .ASCII  /NA:/
003347  040  101  104              .ASCII  / AD/
003352  122  072  040              .ASCII  /R: /
```

```
003355  045  117  066                .ASCII  /%06/
003360  045  101  040                .ASCII  /%A /
003363  040  103  123                .ASCII  / CS/
003366  122  040  075                .ASCII  /R =/
003371  040  045  117                .ASCII  / %O/
003374  066  045  116                .ASCII  /6%N/
003377  000                         .ASCII  <00>
003400  045  116  045       P.ABO:   .ASCII  /%N%/
003403  101  040  127                .ASCII  /A W/
003406  101  111  124                .ASCII  /AIT/
003411  040  101  102                .ASCII  / AB/
003414  117  125  124                .ASCII  /OUT/
003417  040  045  104                .ASCII  / %D/
003422  062  045  101                .ASCII  /2%A/
003425  040  123  105                .ASCII  / SE/
003430  103  117  116                .ASCII  /CON/
003433  104  050  123                .ASCII  /D(S/
003436  051  040  055                .ASCII  /) -/
003441  000                         .ASCII  <00>
003442  045  116  045       P.ABP:   .ASCII  /%N%/
003445  101  040  123                .ASCII  /A S/
003450  101  116  111                .ASCII  /ANI/
003453  124  131  040                .ASCII  /TY /
003456  124  111  115                .ASCII  /TIM/
003461  105  122  040                .ASCII  /ER /
003464  124  111  115                .ASCII  /TIM/
003467  105  104  040                .ASCII  /ED /
003472  117  125  124                .ASCII  /OUT/
003475  040  101  123                .ASCII  / AS/
003500  040  105  130                .ASCII  / EX/
003503  120  105  103                .ASCII  /PEC/
003506  124  105  104                .ASCII  /TED/
003511  040  045  116                .ASCII  / %N/
003514  000  000                    .ASCII  <00><00>
003516  045  116  045       P.ABQ:   .ASCII  /%N%/
003521  101  040  116                .ASCII  /A N/
003524  117  040  123                .ASCII  /O S/
003527  101  116  111                .ASCII  /ANI/
003532  124  131  040                .ASCII  /TY /
003535  124  111  115                .ASCII  /TIM/
003540  105  122  040                .ASCII  /ER /
003543  111  116  124                .ASCII  /INT/
003546  105  122  122                .ASCII  /ERR/
003551  125  120  124                .ASCII  /UPT/
003554  040  104  105                .ASCII  / DE/
003557  124  105  103                .ASCII  /TEC/
003562  124  105  104                .ASCII  /TED/
003565  040  045  116                .ASCII  / %N/
003570  000  000                    .ASCII  <00><00>
003572  045  116  045       P.ABR:   .ASCII  /%N%/
003575  101  040  104                .ASCII  /A D/
003600  111  123  103                .ASCII  /ISC/
003603  117  116  116                .ASCII  /ONN/
```

```
003606  105  103  124              .ASCII  /ECT/
003611  040  124  122              .ASCII  / TR/
003614  101  116  123              .ASCII  /ANS/
003617  103  105  111              .ASCII  /CEI/
003622  126  105  122              .ASCII  /VER/
003625  040  103  101              .ASCII  / CA/
003630  102  114  105              .ASCII  /BLE/
003633  040  106  122              .ASCII  / FR/
003636  117  115  040              .ASCII  /OM /
003641  102  125  114              .ASCII  /BUL/
003644  113  110  105              .ASCII  /KHE/
003647  101  104  040              .ASCII  /AD /
003652  101  123  123              .ASCII  /ASS/
003655  105  115  102              .ASCII  /EMB/
003660  114  131  040              .ASCII  /LY /
003663  101  116  104              .ASCII  /AND/
003666  000  000                   .ASCII  <00><00>
003670  045  116  045       P.ABS: .ASCII  /%N%/
003673  101  040  103              .ASCII  /A C/
003676  117  116  116              .ASCII  /ONN/
003701  105  103  124              .ASCII  /ECT/
003704  040  114  117              .ASCII  / LO/
003707  117  120  102              .ASCII  /OPB/
003712  101  103  113              .ASCII  /ACK/
003715  040  103  117              .ASCII  / CO/
003720  116  116  105              .ASCII  /NNE/
003723  103  124  117              .ASCII  /CTO/
003726  122  040  124              .ASCII  /R T/
003731  117  040  102              .ASCII  /O B/
003734  125  114  113              .ASCII  /ULK/
003737  110  105  101              .ASCII  /HEA/
003742  104  040  101              .ASCII  /D A/
003745  123  123  105              .ASCII  /SSE/
003750  115  102  114              .ASCII  /MBL/
003753  131  054  040              .ASCII  /Y, /
003756  124  110  105              .ASCII  /THE/
003761  116  040  122              .ASCII  /N R/
003764  105  124  105              .ASCII  /ETE/
003767  123  124  045              .ASCII  /ST%/
003772  116  000                   .ASCII  /N/<00>
003774  045  116  045       P.ABT: .ASCII  /%N%/
003777  101  040  104              .ASCII  /A D/
004002  111  123  103              .ASCII  /ISC/
004005  117  116  116              .ASCII  /ONN/
004010  105  103  124              .ASCII  /ECT/
004013  040  102  125              .ASCII  / BU/
004016  114  113  110              .ASCII  /LKH/
004021  105  101  104              .ASCII  /EAD/
004024  040  101  123              .ASCII  / AS/
004027  123  105  115              .ASCII  /SEM/
004032  102  114  131              .ASCII  /BLY/
004035  040  106  122              .ASCII  / FR/
004040  117  115  040              .ASCII  /OM /
```

```
004043  104  105  121              .ASCII  /DEQ/
004046  116  101  040              .ASCII  /NA /
004051  101  116  104              .ASCII  /AND/
004054  040  103  117              .ASCII  / CO/
004057  116  116  105              .ASCII  /NNE/
004062  103  124  000              .ASCII  /CT/<00>
004065  000                       .ASCII  <00>
004066  045  116  045  P.ABU:      .ASCII  /%N%/
004071  101  040  114              .ASCII  /A L/
004074  117  117  120              .ASCII  /OOP/
004077  102  101  103              .ASCII  /BAC/
004102  113  040  103              .ASCII  /K C/
004105  117  116  116              .ASCII  /ONN/
004110  105  103  124              .ASCII  /ECT/
004113  117  122  040              .ASCII  /OR /
004116  124  117  040              .ASCII  /TO /
004121  104  105  121              .ASCII  /DEQ/
004124  116  101  054              .ASCII  /NA,/
004127  040  124  110              .ASCII  / TH/
004132  105  116  040              .ASCII  /EN /
004135  122  105  124              .ASCII  /RET/
004140  105  123  124              .ASCII  /EST/
004143  045  116  000              .ASCII  /%N/<00>
004146  045  116  045  P.ABV:      .ASCII  /%N%/
004151  101  040  103              .ASCII  /A C/
004154  110  105  103              .ASCII  /HEC/
004157  113  040  106              .ASCII  /K F/
004162  117  122  040              .ASCII  /OR /
004165  114  117  117              .ASCII  /LOO/
004170  123  105  040              .ASCII  /SE /
004173  127  111  122              .ASCII  /WIR/
004176  105  123  040              .ASCII  /ES /
004201  111  116  040              .ASCII  /IN /
004204  101  040  114              .ASCII  /A L/
004207  117  117  120              .ASCII  /OOP/
004212  102  101  103              .ASCII  /BAC/
004215  113  040  103              .ASCII  /K C/
004220  117  116  116              .ASCII  /ONN/
004223  105  103  124              .ASCII  /ECT/
004226  117  122  000              .ASCII  /OR/<00>
004231  000                       .ASCII  <00>
004232  045  116  045  P.ABW:      .ASCII  /%N%/
004235  101  040  117              .ASCII  /A O/
004240  122  040  125              .ASCII  /R U/
004243  123  105  040              .ASCII  /SE /
004246  104  111  106              .ASCII  /DIF/
004251  106  105  122              .ASCII  /FER/
004254  105  116  124              .ASCII  /ENT/
004257  040  114  117              .ASCII  / LO/
004262  117  120  102              .ASCII  /OPB/
004265  101  103  113              .ASCII  /ACK/
004270  040  103  117              .ASCII  / CO/
004273  116  116  105              .ASCII  /NNE/
```

```
004276  103  124  117              .ASCII  /CTO/
004301  122  054  040              .ASCII  /R, /
004304  124  110  105              .ASCII  /THE/
004307  116  040  122              .ASCII  /N R/
004312  105  124  105              .ASCII  /ETE/
004315  123  124  045              .ASCII  /ST%/
004320  116  000                   .ASCII  /N/<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  104  105              .ASCII  / DE/
004341  121  116  101              .ASCII  /QNA/
004344  054  040  124              .ASCII  /, T/
004347  110  105  116              .ASCII  /HEN/
004352  040  122  105              .ASCII  / RE/
004355  124  105  123              .ASCII  /TES/
004360  124  045  116              .ASCII  /T%N/
004363  000                       .ASCII  <00>
004364  045  116  045    P.ABY:    .ASCII  /%N%/
004367  101  040  122              .ASCII  /A R/
004372  105  120  114              .ASCII  /EPL/
004375  101  103  105              .ASCII  /ACE/
004400  040  102  125              .ASCII  / BU/
004403  114  113  110              .ASCII  /LKH/
004406  105  101  104              .ASCII  /EAD/
004411  040  103  117              .ASCII  / CO/
004414  116  116  105              .ASCII  /NNE/
004417  103  124  117              .ASCII  /CTO/
004422  122  054  040              .ASCII  /R, /
004425  124  110  105              .ASCII  /THE/
004430  116  040  122              .ASCII  /N R/
004433  105  124  105              .ASCII  /ETE/
004436  123  124  045              .ASCII  /ST%/
004441  116  000  000              .ASCII  /N/<00><00>
004444  045  116  045    P.ABZ:    .ASCII  /%N%/
004447  101  040  104              .ASCII  /A D/
004452  111  123  103              .ASCII  /ISC/
004455  117  116  116              .ASCII  /ONN/
004460  105  103  124              .ASCII  /ECT/
004463  040  124  122              .ASCII  / TR/
004466  101  116  123              .ASCII  /ANS/
004471  103  105  111              .ASCII  /CEI/
004474  126  105  122              .ASCII  /VER/
004477  040  103  101              .ASCII  / CA/
004502  102  114  105              .ASCII  /BLE/
004505  040  106  122              .ASCII  / FR/
004510  117  115  040              .ASCII  /OM /
004513  124  122  101              .ASCII  /TRA/
004516  116  123  103              .ASCII  /NSC/
004521  105  111  126              .ASCII  /EIV/
004524  105  122  000              .ASCII  /ER/<00>
004527  000                       .ASCII  <00>
```

```
004530  045  116  045        P.ACA:  .ASCII  /%N%/
004533  101  040  101                .ASCII  /A A/
004536  116  104  040                .ASCII  /ND /
004541  103  117  116                .ASCII  /CON/
004544  116  105  103                .ASCII  /NEC/
004547  124  040  111                .ASCII  /T I/
004552  124  040  124                .ASCII  /T T/
004555  117  040  114                .ASCII  /O L/
004560  117  117  120                .ASCII  /OOP/
004563  102  101  103                .ASCII  /BAC/
004566  113  040  103                .ASCII  /K C/
004571  117  116  116                .ASCII  /ONN/
004574  105  103  124                .ASCII  /ECT/
004577  117  122  040                .ASCII  /OR /
004602  101  116  104                .ASCII  /AND/
004605  040  102  125                .ASCII  / BU/
004610  114  113  110                .ASCII  /LKH/
004613  105  101  104                .ASCII  /EAD/
004616  040  101  123                .ASCII  / AS/
004621  123  105  115                .ASCII  /SEM/
004624  102  114  131                .ASCII  /BLY/
004627  045  116  000                .ASCII  /%N/<00>
004632  045  116  045        P.ACB:  .ASCII  /%N%/
004635  101  040  122                .ASCII  /A R/
004640  105  120  114                .ASCII  /EPL/
004643  101  103  105                .ASCII  /ACE/
004646  040  124  122                .ASCII  / TR/
004651  101  116  123                .ASCII  /ANS/
004654  103  105  111                .ASCII  /CEI/
004657  126  105  122                .ASCII  /VER/
004662  040  103  101                .ASCII  / CA/
004665  102  114  105                .ASCII  /BLE/
004670  054  040  124                .ASCII  /, T/
004673  110  105  116                .ASCII  /HEN/
004676  040  122  105                .ASCII  / RE/
004701  124  105  123                .ASCII  /TES/
004704  124  045  116                .ASCII  /T%N/
004707  000                        .ASCII  <00>
004710  045  116  045        P.ACC:  .ASCII  /%N%/
004713  101  040  122                .ASCII  /A R/
004716  105  120  114                .ASCII  /EPL/
004721  101  103  105                .ASCII  /ACE/
004724  040  124  122                .ASCII  / TR/
004727  101  116  123                .ASCII  /ANS/
004732  103  105  111                .ASCII  /CEI/
004735  126  105  122                .ASCII  /VER/
004740  054  040  124                .ASCII  /, T/
004743  110  105  116                .ASCII  /HEN/
004746  040  122  105                .ASCII  / RE/
004751  124  105  123                .ASCII  /TES/
004754  124  045  116                .ASCII  /T%N/
004757  000                        .ASCII  <00>
004760  045  116  045        P.ACD:  .ASCII  /%N%/
```

```
004763  101  040  106                .ASCII  /A F/
004766  125  123  105                .ASCII  /USE/
004771  040  117  113                .ASCII  / OK/
004774  040  102  111                .ASCII  / BI/
004777  124  040  111                .ASCII  /T I/
005002  116  040  103                .ASCII  /N C/
005005  123  122  060                .ASCII  /SR0/
005010  040  103  114                .ASCII  / CL/
005013  105  101  122                .ASCII  /EAR/
005016  054  040  116                .ASCII  /, N/
005021  117  040  120                .ASCII  /O P/
005024  117  127  105                .ASCII  /OWE/
005027  122  040  124                .ASCII  /R T/
005032  117  040  130                .ASCII  /O X/
005035  103  126  122                .ASCII  /CVR/
005040  077  045  116                .ASCII  /?%N/
005043  000                         .ASCII  <00>
005044  045  116  045       P.ACE:   .ASCII  /%N%/
005047  101  040  102                .ASCII  /A B/
005052  101  104  040                .ASCII  /AD /
005055  122  105  103                .ASCII  /REC/
005060  105  111  126                .ASCII  /EIV/
005063  105  040  104                .ASCII  /E D/
005066  105  123  103                .ASCII  /ESC/
005071  122  111  120                .ASCII  /RIP/
005074  124  117  122                .ASCII  /TOR/
005077  072  000  000                .ASCII  /:/<00><00>
005102  045  116  045       P.ACF:   .ASCII  /%N%/
005105  101  040  102                .ASCII  /A B/
005110  101  104  040                .ASCII  /AD /
005113  124  122  101                .ASCII  /TRA/
005116  116  123  115                .ASCII  /NSM/
005121  111  124  040                .ASCII  /IT /
005124  104  105  123                .ASCII  /DES/
005127  103  122  111                .ASCII  /CRI/
005132  120  124  117                .ASCII  /PTO/
005135  122  072  000                .ASCII  /R:/<00>
005140  045  101  040       P.ACG:   .ASCII  /%A /
005143  101  103  124                .ASCII  /ACT/
005146  125  101  114                .ASCII  /UAL/
005151  040  075  040                .ASCII  / = /
005154  045  117  066                .ASCII  /%06/
005157  045  101  040                .ASCII  /%A /
005162  105  130  120                .ASCII  /EXP/
005165  105  103  124                .ASCII  /ECT/
005170  105  104  040                .ASCII  /ED /
005173  075  040  045                .ASCII  /= %/
005176  117  066  045                .ASCII  /06%/
005201  101  040  111                .ASCII  /A I/
005204  116  104  105                .ASCII  /NDE/
005207  130  040  075                .ASCII  /X =/
005212  040  045  104                .ASCII  / %D/
005215  064  045  116                .ASCII  /4%N/
```

J4


```
005220  000  000                    .ASCII  <00><00>
005222  045  116  045       P.ACH:  .ASCII  /%N%/
005225  101  040  102               .ASCII  /A B/
005230  101  104  040               .ASCII  /AD /
005233  122  105  103               .ASCII  /REC/
005236  105  111  126               .ASCII  /EIV/
005241  105  040  102               .ASCII  /E B/
005244  125  106  106               .ASCII  /UFF/
005247  105  122  072               .ASCII  /ER:/
005252  000  000                    .ASCII  <00><00>
005254  045  116  045       P.ACI:  .ASCII  /%N%/
005257  101  040  104               .ASCII  /A D/
005262  115  101  040               .ASCII  /MA /
005265  117  120  105               .ASCII  /OPE/
005270  122  101  124               .ASCII  /RAT/
005273  111  117  116               .ASCII  /ION/
005276  040  124  101               .ASCII  / TA/
005301  113  105  123               .ASCII  /KES/
005304  040  124  117               .ASCII  / TO/
005307  117  040  114               .ASCII  /O L/
005312  117  116  107               .ASCII  /ONG/
005315  045  116  000               .ASCII  /%N/<00>
005320  045  116  045       P.ACJ:  .ASCII  /%N%/
005323  101  040  124               .ASCII  /A T/
005326  117  117  040               .ASCII  /OO /
005331  115  101  116               .ASCII  /MAN/
005334  131  040  104               .ASCII  /Y D/
005337  105  126  111               .ASCII  /EVI/
005342  103  105  123               .ASCII  /CES/
005345  045  116  000               .ASCII  /%N/<00>
005350  045  116  045       P.ACK:  .ASCII  /%N%/
005353  101  040  124               .ASCII  /A T/
005356  110  105  122               .ASCII  /HER/
005361  105  040  127               .ASCII  /E W/
005364  101  123  040               .ASCII  /AS /
005367  101  040  120               .ASCII  /A P/
005372  117  127  105               .ASCII  /OWE/
005375  122  040  106               .ASCII  /R F/
005400  101  111  114               .ASCII  /AIL/
005403  040  055  040               .ASCII  / - /
005406  127  101  111               .ASCII  /WAI/
005411  124  111  116               .ASCII  /TIN/
005414  107  045  116               .ASCII  /G%N/
005417  000                       .ASCII  <00>
005420  045  116  045       P.ACL:  .ASCII  /%N%/
005423  101  040  127               .ASCII  /A W/
005426  101  111  124               .ASCII  /AIT/
005431  040  101  102               .ASCII  / AB/
005434  117  125  124               .ASCII  /OUT/
005437  040  045  104               .ASCII  / %D/
005442  062  045  101               .ASCII  /2%A/
005445  040  115  111               .ASCII  / MI/
005450  116  125  124               .ASCII  /NUT/
```

K4


```
005453  105  050  123                .ASCII  /E(S/
005456  051  040  055                .ASCII  /) -/
005461  000                         .ASCII  <00>
005462  045  116  045        P.ACM:  .ASCII  /%N%/
005465  101  040  127                .ASCII  /A W/
005470  101  111  124                .ASCII  /AIT/
005473  040  101  102                .ASCII  / AB/
005476  117  125  124                .ASCII  /OUT/
005501  040  045  104                .ASCII  / %D/
005504  062  045  101                .ASCII  /2%A/
005507  040  110  117                .ASCII  / HO/
005512  125  122  040                .ASCII  /UR /
005515  055  000  000                .ASCII  /-/<00><00>
005520  045  101  040        P.ACN:  .ASCII  /%A /
005523  111  106  040                .ASCII  /IF /
005526  116  117  040                .ASCII  /NO /
005531  122  105  123                .ASCII  /RES/
005534  105  124  054                .ASCII  /ET,/
005537  040  124  131                .ASCII  / TY/
005542  120  105  040                .ASCII  /PE /
005545  101  116  131                .ASCII  /ANY/
005550  040  103  110                .ASCII  / CH/
005553  101  122  101                .ASCII  /ARA/
005556  103  124  105                .ASCII  /CTE/
005561  122  040  124                .ASCII  /R T/
005564  117  040  105                .ASCII  /O E/
005567  130  111  124                .ASCII  /XIT/
005572  040  106  122                .ASCII  / FR/
005575  117  115  040                .ASCII  /OM /
005600  124  105  123                .ASCII  /TES/
005603  124  045  116                .ASCII  /T%N/
005606  000  000                     .ASCII  <00><00>
005610  045  116  045        P.ACO:  .ASCII  /%N%/
005613  101  040  124                .ASCII  /A T/
005616  104  122  040                .ASCII  /DR /
005621  126  101  114                .ASCII  /VAL/
005624  125  105  040                .ASCII  /UE /
005627  111  123  040                .ASCII  /IS /
005632  105  121  125                .ASCII  /EQU/
005635  101  114  040                .ASCII  /AL /
005640  124  117  040                .ASCII  /TO /
005643  132  105  122                .ASCII  /ZER/
005646  117  040  045                .ASCII  /O %/
005651  116  000  000                .ASCII  /N/<00><00>
005654  045  116  045        P.ACP:  .ASCII  /%N%/
005657  116  045  101                .ASCII  /N%A/
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  055              .ASCII  /- -/
005712  055  055  055              .ASCII  /- -/
005715  055  055  055              .ASCII  /---/
005720  055  055  055              .ASCII  /---/
005723  055  055  055              .ASCII  /- -/
005726  055  055  055              .ASCII  /- -/
005731  055  055  055              .ASCII  /---/
005734  055  055  055              .ASCII  /---/
005737  055  055  055              .ASCII  /---/
005742  055  055  055              .ASCII  /---/
005745  055  055  055              .ASCII  /---/
005750  055  055  055              .ASCII  /---/
005753  055  055  055              .ASCII  /---/
005756  055  055  055              .ASCII  /---/
005761  055  055  045              .ASCII  /--%/
005764  116  000                  .ASCII  /N/<00>
005766  045  116  045      P.ACQ:  .ASCII  /%N%/
005771  101  040  102              .ASCII  /A B/
005774  101  104  040              .ASCII  /AD /
005777  103  123  122              .ASCII  /CSR/
006002  054  040  102              .ASCII  /, B/
006005  111  124  123              .ASCII  /ITS/
006010  040  123  124              .ASCII  / ST/
006013  125  103  113              .ASCII  /UCK/
006016  040  101  124              .ASCII  / AT/
006021  040  060  072              .ASCII  / 0:/
006024  045  116  000              .ASCII  /%N/<00>
006027  000                      .ASCII  <00>
006030  045  116  045      P.ACR:  .ASCII  /%N%/
006033  101  040  102              .ASCII  /A B/
006036  101  104  040              .ASCII  /AD /
006041  103  123  122              .ASCII  /CSR/
006044  054  040  102              .ASCII  /, B/
006047  111  124  123              .ASCII  /ITS/
006052  040  123  124              .ASCII  / ST/
006055  125  103  113              .ASCII  /UCK/
006060  040  101  124              .ASCII  / AT/
006063  040  061  072              .ASCII  / 1:/
006066  045  116  000              .ASCII  /%N/<00>
006071  000                      .ASCII  <00>
006072  045  116  045      P.ACS:  .ASCII  /%N%/
006075  101  040  123              .ASCII  /A S/
006100  117  106  124              .ASCII  /OFT/
006103  127  101  122              .ASCII  /WAR/
006106  105  040  122              .ASCII  /E R/
006111  105  123  105              .ASCII  /ESE/
006114  124  040  125              .ASCII  /T U/
006117  116  101  102              .ASCII  /NAB/
006122  114  105  040              .ASCII  /LE /
006125  124  117  040              .ASCII  /TO /
006130  103  114  105              .ASCII  /CLE/
006133  101  122  040              .ASCII  /AR /
006136  103  123  122              .ASCII  /CSR/
```

M4


```
006141  040  123  124              .ASCII  / ST/
006144  101  124  111              .ASCII  /ATI/
006147  103  040  102              .ASCII  /C B/
006152  111  124  123              .ASCII  /ITS/
006155  072  045  116              .ASCII  /:%N/
006160  000  000                   .ASCII  <00><00>
006162  045  116  045    P.ACT:    .ASCII  /%N%/
006165  101  040  102              .ASCII  /A B/
006170  101  104  040              .ASCII  /AD /
006173  123  124  101              .ASCII  /STA/
006176  124  111  117              .ASCII  /TIO/
006201  116  040  101              .ASCII  /N A/
006204  104  104  122              .ASCII  /DDR/
006207  105  123  123              .ASCII  /ESS/
006212  040  103  110              .ASCII  / CH/
006215  105  103  113              .ASCII  /ECK/
006220  123  125  115              .ASCII  /SUM/
006223  072  040  101              .ASCII  /: A/
006226  103  124  040              .ASCII  /CT /
006231  075  040  045              .ASCII  /= %/
006234  117  066  045              .ASCII  /06%/
006237  101  040  105              .ASCII  /A E/
006242  130  120  040              .ASCII  /XP /
006245  075  040  045              .ASCII  /= %/
006250  117  066  045              .ASCII  /06%/
006253  116  000  000              .ASCII  /N/<00><00>
006256  045  116  045    P.ACU:    .ASCII  /%N%/
006261  101  040  102              .ASCII  /A B/
006264  101  104  040              .ASCII  /AD /
006267  123  124  101              .ASCII  /STA/
006272  124  111  117              .ASCII  /TIO/
006275  116  040  101              .ASCII  /N A/
006300  104  104  122              .ASCII  /DDR/
006303  105  123  123              .ASCII  /ESS/
006306  072  040  000              .ASCII  /: /<00>
006311  000                       .ASCII  <00>
006312  045  116  045    P.ACV:    .ASCII  /%N%/
006315  101  040  102              .ASCII  /A B/
006320  101  104  040              .ASCII  /AD /
006323  104  105  121              .ASCII  /DEQ/
006326  116  101  040              .ASCII  /NA /
006331  111  057  117              .ASCII  /I/<57>/O/
006334  040  120  101              .ASCII  / PA/
006337  107  105  040              .ASCII  /GE /
006342  122  105  107              .ASCII  /REG/
006345  111  123  124              .ASCII  /IST/
006350  105  122  072              .ASCII  /ER:/
006353  045  116  000              .ASCII  /%N/<00>
006356  045  116  045    P.ACW:    .ASCII  /%N%/
006361  101  040  102              .ASCII  /A B/
006364  101  104  040              .ASCII  /AD /
006367  103  123  122              .ASCII  /CSR/
006372  054  040  105              .ASCII  /, E/
```

```
006375  130  120  105                .ASCII  /XPE/
006400  103  124  105                .ASCII  /CTE/
006403  104  040  122                .ASCII  /D R/
006406  114  040  050                .ASCII  /L (/
006411  040  102  111                .ASCII  / BI/
006414  124  040  065                .ASCII  /T 5/
006417  040  051  040                .ASCII  / ) /
006422  124  117  040                .ASCII  /TO /
006425  102  105  040                .ASCII  /BE /
006430  123  105  124                .ASCII  /SET/
006433  040  124  117                .ASCII  / TO/
006436  040  060  045                .ASCII  / 0%/
006441  116  000  000                .ASCII  /N/<00><00>
006444  045  116  045          P.ACX:  .ASCII  /%N%/
006447  101  040  102                .ASCII  /A B/
006452  101  104  040                .ASCII  /AD /
006455  102  057  104                .ASCII  /B/<57>/D/
006460  040  120  122                .ASCII  / PR/
006463  117  115  040                .ASCII  /OM /
006466  103  110  105                .ASCII  /CHE/
006471  103  113  123                .ASCII  /CKS/
006474  125  115  072                .ASCII  /UM:/
006477  040  111  116                .ASCII  / IN/
006502  104  105  130                .ASCII  /DEX/
006505  040  075  040                .ASCII  / = /
006510  045  117  066                .ASCII  /%O6/
006513  045  101  040                .ASCII  /%A /
006516  101  103  124                .ASCII  /ACT/
006521  040  075  040                .ASCII  / = /
006524  045  117  066                .ASCII  /%O6/
006527  045  101  040                .ASCII  /%A /
006532  105  130  120                .ASCII  /EXP/
006535  040  075  040                .ASCII  / = /
006540  045  117  066                .ASCII  /%O6/
006543  045  116  000                .ASCII  /%N/<00>
006546  045  116  045          P.ACY:  .ASCII  /%N%/
006551  101  040  102                .ASCII  /A B/
006554  057  104  040                .ASCII  <57>/D /
006557  120  122  117                .ASCII  /PRO/
006562  115  040  103                .ASCII  /M C/
006565  110  105  103                .ASCII  /HEC/
006570  113  123  125                .ASCII  /KSU/
006573  115  040  117                .ASCII  /M O/
006576  106  106  123                .ASCII  /FFS/
006601  105  124  040                .ASCII  /ET /
006604  075  040  045                .ASCII  /= %/
006607  117  066  045                .ASCII  /O6%/
006612  101  040  101                .ASCII  /A A/
006615  103  124  040                .ASCII  /CT /
006620  075  040  045                .ASCII  /= %/
006623  117  066  045                .ASCII  /O6%/
006626  101  040  105                .ASCII  /A E/
006631  130  120  040                .ASCII  /XP /
```

```
006634  075  040  045              .ASCII  /= %/
006637  117  066  045              .ASCII  /06%/
006642  116  000                   .ASCII  /N/<00>
006644  045  116  045  P.ACZ:      .ASCII  /%N%/
006647  101  040  102              .ASCII  /A B/
006652  101  104  040              .ASCII  /AD /
006655  111  116  124              .ASCII  /INT/
006660  105  122  122              .ASCII  /ERR/
006663  125  120  124              .ASCII  /UPT/
006666  072  040  101              .ASCII  /: A/
006671  104  122  040              .ASCII  /DR /
006674  075  040  045              .ASCII  /= %/
006677  117  066  045              .ASCII  /06%/
006702  101  040  101              .ASCII  /A A/
006705  103  124  040              .ASCII  /CT /
006710  114  105  126              .ASCII  /LEV/
006713  040  075  040              .ASCII  / = /
006716  045  117  066              .ASCII  /%06/
006721  045  101  040              .ASCII  /%A /
006724  105  130  120              .ASCII  /EXP/
006727  040  114  105              .ASCII  / LE/
006732  126  040  075              .ASCII  /V =/
006735  040  045  117              .ASCII  / %O/
006740  066  045  116              .ASCII  /6%N/
006743  000                       .ASCII  <00>
006744  045  116  045  P.ADA:      .ASCII  /%N%/
006747  101  040  122              .ASCII  /A R/
006752  105  107  111              .ASCII  /EGI/
006755  123  124  105              .ASCII  /STE/
006760  122  040  106              .ASCII  /R F/
006763  101  111  114              .ASCII  /AIL/
006766  105  104  040              .ASCII  /ED /
006771  124  117  040              .ASCII  /TO /
006774  122  105  123              .ASCII  /RES/
006777  120  117  116              .ASCII  /PON/
007002  104  040  101              .ASCII  /D A/
007005  124  040  101              .ASCII  /T A/
007010  104  104  122              .ASCII  /DDR/
007013  105  123  123              .ASCII  /ESS/
007016  072  040  040              .ASCII  /:  /
007021  045  117  066              .ASCII  /%06/
007024  045  116  000              .ASCII  /%N/<00>
007027  000                       .ASCII  <00>
007030  045  116  045  P.ADB:      .ASCII  /%N%/
007033  101  040  102              .ASCII  /A B/
007036  101  104  040              .ASCII  /AD /
007041  124  122  101              .ASCII  /TRA/
007044  116  123  115              .ASCII  /NSM/
007047  111  124  040              .ASCII  /IT /
007052  123  124  101              .ASCII  /STA/
007055  124  125  123              .ASCII  /TUS/
007060  054  040  124              .ASCII  /, T/
007063  117  117  040              .ASCII  /00 /
```

```
007066  115  101  116                .ASCII  /MAN/
007071  131  040  103                .ASCII  /Y C/
007074  117  114  114                .ASCII  /OLL/
007077  111  123  111                .ASCII  /ISI/
007102  117  116  123                .ASCII  /ONS/
007105  045  116  000                .ASCII  /%N/<00>
007110  045  116  045          P.ADC:  .ASCII  /%N%/
007113  101  040  104                .ASCII  /A D/
007116  105  126  111                .ASCII  /EVI/
007121  103  105  040                .ASCII  /CE /
007124  106  101  111                .ASCII  /FAI/
007127  114  105  104                .ASCII  /LED/
007132  040  124  117                .ASCII  / TO/
007135  040  111  116                .ASCII  / IN/
007140  124  105  122                .ASCII  /TER/
007143  122  125  120                .ASCII  /RUP/
007146  124  072  040                .ASCII  /T: /
007151  103  120  125                .ASCII  /CPU/
007154  040  120  122                .ASCII  / PR/
007157  111  117  122                .ASCII  /IOR/
007162  111  124  131                .ASCII  /ITY/
007165  040  075  040                .ASCII  / = /
007170  045  117  066                .ASCII  /%O6/
007173  045  116  000                .ASCII  /%N/<00>
007176  045  116  045          P.ADD:  .ASCII  /%N%/
007201  101  040  125                .ASCII  /A U/
007204  116  105  130                .ASCII  /NEX/
007207  120  105  103                .ASCII  /PEC/
007212  124  105  104                .ASCII  /TED/
007215  040  104  105                .ASCII  / DE/
007220  126  111  103                .ASCII  /VIC/
007223  105  040  111                .ASCII  /E I/
007226  116  124  105                .ASCII  /NTE/
007231  122  122  125                .ASCII  /RRU/
007234  120  124  072                .ASCII  /PT:/
007237  040  103  120                .ASCII  / CP/
007242  125  040  120                .ASCII  /U P/
007245  122  111  117                .ASCII  /RIO/
007250  122  111  124                .ASCII  /RIT/
007253  131  040  075                .ASCII  /Y =/
007256  040  045  117                .ASCII  / %O/
007261  066  045  116                .ASCII  /6%N/
007264  000  000                     .ASCII  <00><00>
007266  045  116  045          P.ADE:  .ASCII  /%N%/
007271  101  040  106                .ASCII  /A F/
007274  101  111  114                .ASCII  /AIL/
007277  125  122  105                .ASCII  /URE/
007302  040  111  116                .ASCII  / IN/
007305  040  105  130                .ASCII  / EX/
007310  124  105  122                .ASCII  /TER/
007313  116  101  114                .ASCII  /NAL/
007316  040  114  117                .ASCII  / LO/
007321  117  120  102                .ASCII  /OPB/
```

```
007324  101  103  113                    .ASCII  /ACK/
007327  040  115  117                    .ASCII  / MO/
007332  104  105  040                    .ASCII  /DE /
007335  045  116  000                    .ASCII  /%N/<00>
007340  045  116  045          P.ADF:    .ASCII  /%N%/
007343  101  040  122                    .ASCII  /A R/
007346  143  166  040                    .ASCII  /cv /
007351  104  145  163                    .ASCII  /Des/
007354  143  040  102                    .ASCII  /c B/
007357  141  163  145                    .ASCII  /ase/
007362  040  075  040                    .ASCII  / = /
007365  045  117  066                    .ASCII  /%06/
007370  045  101  040                    .ASCII  /%A /
007373  111  116  104                    .ASCII  /IND/
007376  105  130  040                    .ASCII  /EX /
007401  075  040  045                    .ASCII  /= %/
007404  104  064  045                    .ASCII  /D4%/
007407  101  040  101                    .ASCII  /A A/
007412  143  164  165                    .ASCII  /ctu/
007415  141  154  040                    .ASCII  /al /
007420  075  040  045                    .ASCII  /= %/
007423  117  066  045                    .ASCII  /06%/
007426  116  000                        .ASCII  /N/<00>
007430  045  116  045          P.ADG:    .ASCII  /%N%/
007433  101  040  124                    .ASCII  /A T/
007436  170  040  104                    .ASCII  /x D/
007441  145  163  143                    .ASCII  /esc/
007444  040  102  141                    .ASCII  / Ba/
007447  163  145  040                    .ASCII  /se /
007452  075  040  045                    .ASCII  /= %/
007455  117  066  045                    .ASCII  /06%/
007460  101  040  111                    .ASCII  /A I/
007463  116  104  105                    .ASCII  /NDE/
007466  130  040  075                    .ASCII  /X =/
007471  040  045  104                    .ASCII  / %D/
007474  064  045  101                    .ASCII  /4%A/
007477  040  101  143                    .ASCII  / Ac/
007502  164  165  141                    .ASCII  /tua/
007505  154  040  075                    .ASCII  /l =/
007510  040  045  117                    .ASCII  / %O/
007513  066  045  116                    .ASCII  /6%N/
007516  000  000                        .ASCII  <00><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  0000C0                  ERR.NUMBER::
                                        .WORD   0
012430  000000                  ERR.FLAG::
                                        .WORD   0
012432  000000                  ERR.COUNT::
                                        .WORD   0
012434                          TMPR1:: .BLKW   1
012436                          TMP.IOP.ADR::
                                        .BLKW   1
012440                          TMP.REG.DATA::
                                        .BLKW   1
012442                          TEMP1:: .BLKW   1
012444                          TEMP2:: .BLKW   1
012446                          TEMP3:: .BLKW   1
012450                          TEMP4:: .BLKW   1
012452                          TEMP5:: .BLKW   1
012454                          TEMP6:: .BLKW   1
012456                          TEMP7:: .BLKW   1
012460                          TEMP8:: .BLKW   1
012462                          TEMP9:: .BLKW   1
012464                          P1::    .BLKW   1
012466                          P2::    .BLKW   1
012470                          P3::    .BLKW   1
012472                          P4::    .BLKW   1
012474                          P5::    .BLKW   1
012476                          TBYTE1::.BLKB   1
012477                          TBYTE2::.BLKB   1
012500                          TBYTE3::.BLKB   1
012501                          TBYTE4::.BLKB   1
012502                          TADR1:: .BLKW   1
012504                          TADR2:: .BLKW   1
012506                          LOGUN:: .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                  PRIC3==             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
```

M5


```
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'                 QST10==         P.AAH
000374'                 MSG00==         P.AAI
000432'                 MSG01==         P.AAJ
000514'                 MSG02==         P.AAK
000602'                 MSG03==         P.AAL
000706'                 MSG04==         P.AAM
001000'                 MSG05==         P.AAN
001072'                 MSG06==         P.AAO
001164'                 MSG07==         P.AAP
001256'                 MSG08==         P.AAQ
001350'                 MSG09==         P.AAR
001442'                 MSG10==         P.AAS
001524'                 MSG11==         P.AAT
001610'                 MSG12==         P.AAU
001654'                 MSG13==         P.AAV
001740'                 MSG14==         P.AAW
002030'                 MSG15==         P.AAX
002112'                 MSG16==         P.AAY
002200'                 MSG17==         P.AAZ
002266'                 MSG18==         P.ABA
002312'                 MSG19==         P.ABB
002400'                 MSG20==         P.ABC
002470'                 MSG21==         P.ABD
002550'                 MSG22==         P.ABE
002634'                 MSG23==         P.ABF
002712'                 MSG24==         P.ABG
002766'                 MSG25==         P.ABH
003030'                 MSG26==         P.ABI
003072'                 MSG27==         P.ABJ
003134'                 MSG28==         P.ABK
003200'                 MSG29==         P.ABL
003226'                 MSG30==         P.ABM
003314'                 MSG31==         P.ABN
003400'                 MSG32==         P.ABO
003442'                 MSG33==         P.ABP
003516'                 MSG34==         P.ABQ
```

N5


```
003572'                 MSG35==         P.ABR
003670'                 MSG36==         P.ABS
003774'                 MSG37==         P.ABT
004066'                 MSG38==         P.ABU
004146'                 MSG39==         P.ABV
004232'                 MSG40==         P.ABW
004322'                 MSG41==         P.ABX
004364'                 MSG42==         P.ABY
004444'                 MSG43==         P.ABZ
004530'                 MSG44==         P.ACA
004632'                 MSG45==         P.ACB
004710'                 MSG46==         P.ACC
004760'                 MSG47==         P.ACD
005044'                 MSG48==         P.ACE
005102'                 MSG49==         P.ACF
005140'                 MSG50==         P.ACG
005222'                 MSG51==         P.ACH
005254'                 MSG52==         P.ACI
005320'                 MSG53==         P.ACJ
005350'                 MSG54==         P.ACK
005420'                 MSG55==         P.ACL
005462'                 MSG56==         P.ACM
005520'                 MSG57==         P.ACN
005610'                 MSG58==         P.ACO
005654'                 MSG59==         P.ACP
005766'                 MSG60==         P.ACQ
006030'                 MSG61==         P.ACR
006072'                 MSG62==         P.ACS
006162'                 MSG63==         P.ACT
006256'                 MSG64==         P.ACU
006312'                 MSG65==         P.ACV
006356'                 MSG66==         P.ACW
006444'                 MSG67==         P.ACX
006546'                 MSG68==         P.ACY
006644'                 MSG69==         P.ACZ
006744'                 MSG70==         P.ADA
007030'                 MSG71==         P.ADB
007110'                 MSG72==         P.ADC
007176'                 MSG73==         P.ADD
007266'                 MSG74==         P.ADE
007340'                 MSG75==         P.ADF
007430'                 MSG76==         P.ADG
000210'                 HP.TABLE==      L$HWLEN+2
000220'                 SP.TABLE==      L$SWLEN+2


;                       PSECT SUMMARY
;
;       Psect Name      Words     Attributes
;         $CODE$          82       RO ,  I  ,  LCL,  REL,  CON
;         $GLOB$        2724       RW ,  D  ,  LCL,  REL,  CON
;         $PLIT$        1960       RO ,  D  ,  LCL,  REL,  CON
```

```
;                         Library Statistics
;
;                                              -------- Symbols --------      Pages      Processing
;      File                                    Total    Loaded   Percent      Mapped     Time
;
;  DISK2:[SCODA.QNA.ZQNA]QNALIB.L16;2           224       88        39          14        00:00.1



;                              COMMAND QUALIFIERS

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

; Size:           0 code + 4766 data words
; Run Time:          00:17.3
; Elapsed Time:      00:19.5
; Lines/CPU Min:      8518
; Lexemes/CPU-Min: 53892
; Memory Used:  242 pages
; Compilation Complete
```

C6


```
;    0001  0     MODULE ZQNA2 (%TITLE 'CZQNAE0 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
```

D6


```
;    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$ISR NOVALUE,
;    1510  1         QNA_INT             : L$ISR NOVALUE;
;    1511  1
;    1512  1     EXTERNAL ROUTINE
;    1513  1         RESET_DEQNA         : NOVALUE;
;    1514  1
```

E6


```
;    1515  1      EXTERNAL
;    1516  1
;    1517  1      !++
;    1518  1      !       COMMUNICATION AREA DECLARATIONS
;    1519  1      !-
;    1520  1
;    1521  1          IOP_TABLE       : VECTOR [ 8, WORD ],
;    1522  1
;    1523  1
;    1524  1      !++
;    1525  1      !       HARDWARE AND SOFTWARE P-TABLE STORAGE DECLARATIONS
;    1526  1      ! -
;    1527  1
;    1528  1          HWP_TABLE   : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;    1529  1          SWP_TABLE   : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    1530  1
;    1531  1          INTERRUPT_FLG       : WORD,                 ! 1 = INTERRUPT OCCURED
;    1532  1
;    1533  1          REG_ADR             : REF REG_STR FIELD ( IOP_FIELDS ),
;    1534  1          IOP_DATA            : REF REG_STR FIELD ( IOP_FIELDS ),
;    1535  1          GET_ADR             : REF ADR_STR FIELD ( IOP_FIELDS ),
;    1536  1
;    1537  1      !++
;    1538  1      !   TEMPORARY STORAGE DATA DECLARATIONS
;    1539  1      !--
;    1540  1
;    1541  1          TMP_IOP_ADR         : WORD,                 ! I/O PAGE REGISTER ADDRESS
;    1542  1          TMP_REG_DATA        : WORD,                 ! I/O PAGE REG CONTENTS
;    1543  1          TEMP1               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1544  1          TEMP2               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1545  1          TEMP3               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1546  1          TEMP4               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1547  1          TEMP5               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1548  1          TEMP6               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1549  1          TEMP7               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1550  1          TEMP8               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1551  1          TEMP9               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1552  1
;    1553  1      ! added location(s)
;    1554  1
;    1555  1          logun               :WORD,                  !logical unit # (unit under test)
;    1556  1
;    1557  1
;    1558  1      !++
;    1559  1      !       QUESTIONS AND ERROR MESSAGEES DECLARED EXTERNALLY
;    1560  1      !--
;    1561  1
;    1562  1          QST01, QST02, QST03, QST04, QST05, QST06, QST07, QST10, MSG54;
;    1563  1
```

F6


```
;   1564  1     %SBTTL 'TYPE AND DESCRIPTION'
;   1565  1
;   1566  1     !++
;   1567  1     !       NAMES OF DEVICES SUPPORTED BY PROGRAM
;   1568  1     !--
;   1569  1
;   1570  1     EQUALS;
;   1571  1     DEVTYP (%ASCIZ'DEQNA/M7504');
;   1572  1
;   1573  1     !++
;   1574  1     !       TEST DESCRIPTION
;   1575  1     !--
;   1576  1
;   1577  1     DESCRIPT (%ASCIZ'DEQNA FUNCTIONAL TEST');
;   1578  1
```

```
;    1579  1     %SBTTL 'HARDWARE PARAMETER CODING SECTION'
;    1580  1
;    1581  1     !++
;    1582  1     !       THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;    1583  1     !       THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    1584  1     !       MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    1585  1     !       INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    1586  1     !       MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    1587  1     !       WITH THE OPERATOR.
;    1588  1     !
;    1589  1     !       THIS CODE IS USED BY THE SUPERVISOR TO INTERROGATE THE OPERATOR
;    1590  1     !       FOR DEVICE INFORMATION TO PUT IN THE P TABLE.  THIS CODE IS USED
;    1591  1     !       IN CONJUNCTION WITH THE DEFAULT P-TABLE TEMPLATE.  THE MACROS
;    1592  1     !       USED IN THIS SECTION ARE "GPRMD", "GPRMA".
;    1593  1     !++
;    1594  1
;    1595  1     BGNHRD;
;    1596  1     GPRMA (QST01, %O'0', 0, %O'174440', %O'174460', YES, 1);  ! I/O PAGE  ADDRESS ?
;    1597  1     GPRMA (QST02, %O'2', 0, %O'700',    %O'704',    YES, 1);  ! INTERRUPT VECTOR ADDR ?
;    1598  1     ENDHRD;
;    1599  1
;    1600  1
```

```
;   1601  1     %SBTTL 'SOFTWARE PARAMETER CODING SECTION'
;   1602  1
;   1603  1     !++
;   1604  1     !       THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;   1605  1     !       THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;   1606  1     !       MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;   1607  1     !       INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;   1608  1     !       MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;   1609  1     !       WITH THE OPERATOR.
;   1610  1     !--
;   1611  1
;   1612  1     BGNSFT;
;   1613  1
;   1614  1     GPRML ( QST03, %O'0', -1, YES, 1);       ! DO YOU WANT TO TEST SANITY TIMER ?
;   1615  1     XFERF(NOTIMER);
;   1616  1     GPRMD ( QST05, %O'4', D, -1, 0, 7, YES, 1);
;   1617  1                                              ! SANITY TIMER TIME-OUT VALUE ?
;   1618  1     $L(NOTIMER);
;   1619  1
;   1620  1     GPRML ( QST06, %O'6',  -1, YES, 1);      ! EXTERNAL LOOPBACK MODE ?
;   1621  1     GPRML ( QST07, %O'10', -1, YES, 1);      ! SYSTEM HAS BLOCK-MODE MEMORY ?
;   1622  1     GPRML ( QST04, %O'2', -1, YES, 1);       ! LOOPBACK CONNECTOR IN DEQNA ?
;   1623  1     GPRML ( QST10, %O'12', -1, YES, 1);      ! run NXM test 21, sys has < 4M mem
;   1624  1
;   1625  1     ENDSFT;
;   1626  1
;   1627  1
```

```
;   1628  1     %SBTTL 'REPORT CODING SECTION'
;   1629  1
;   1630  1     !++
;   1631  1     !
;   1632  1     !       THE REPORT CODING SECTION CONTAINS THE
;   1633  1     !       "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;   1634  1     !
;   1635  1     !       THIS SECTION CONTAINS THE CODE FOR PRINTING
;   1636  1     !       STATISTICAL INFORMATION GATHERED BY THE DIAGNOSTIC.  IT IS
;   1637  1     !       EXECUTED BY THE OPERATOR COMMAND "PRINT" OR BY THE MACRO CALL
;   1638  1     !       "DORPT".  USE THE PRINTS MACRO TO PRINT THE INFORMATION.
;   1639  1     !       USE FORMAT STATEMENTS AS IN THE PRINTB/PRINTX MACROS.  IT IS
;   1640  1     !       THE PROGRAMMER'S RESPONSIBILTY TO DEVISE AND IMPLEMENT THE
;   1641  1     !       FORM AND CONTENT OF THE STATISTICS.
;   1642  1     !--
;   1643  1
;   1644  1
;   1645  2     BGNRPT;
;   1646  2
;   1647  2       TEMP1 = 1;
;   1648  2
;   1649  1     ENDRPT;


                                            .TITLE  ZQNA2 CZQNAEO 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  005130                  GP$8::  .WORD   5130
000144  000000G                         .WORD   QST10
000146  177777                         .WORD   -1
000150                          L$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, LOGUN, QST01, QST02, QST03
                                        .GLOBL  QST04, QST05, QST06, QST07, QST10
                                        .GLOBL  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
```

K6


```
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                ;                       1647
000006  000207                         RTS     PC                      ;                       1625
```

```
; 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                 ;                       1647
000004  104425                        TRAP    25
000006  000207                        RTS     PC

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


;   1650  1
;   1651  1
;   1652  1
;   1653  1
```

```
;  1654  1     %SBTTL 'INITIALIZE SECTION'
;  1655  1
;  1656  1     !++
;  1657  1     !       THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
;  1658  1     !       AT THE BEGINNING OF EACH PASS.
;  1659  1     !
;  1660  1     !       THE INITIALIZE CODE IS EXECUTED UNDER FIVE CONDITIONS.  THERE
;  1661  1     !       ARE SUPERVISOR EVENT FLAGS THAT ARE USED TO LET THE
;  1662  1     !       DIAGNOSTIC KNOW UNDER WHICH CONDITION THE EXECUTION IS TAKING
;  1663  1     !       PLACE.  THE EVENT FLAGS ARE READ USING THE "READEF" MACRO.
;  1664  1     !       THE CONDITIONS UNDER WHICH THE INIT CODE IS EXECUTED AND THE
;  1665  1     !       CORRESPONDING EVENT FLAGS ARE:
;  1666  1     !               START COMMAND           EF.START
;  1667  1     !               RESTART COMMAND         EF.RESTART
;  1668  1     !               CONTINUE COMMAND        EF.CONTINUE
;  1669  1     !               POWERDOWN/POWERUP       EF.PWR
;  1670  1     !               NEW PASS                EF.NEW
;  1671  1     !       EXAMPLE OF EVENT FLAG USE:
;  1672  1     !               IF READEF(EF.START) THEN
;  1673  1     !                   START_FLAG = 1;
;  1674  1     !       DURING THE INIT CODE, USE THE "GPHARD" MACRO TO OBTAIN P-TABLE
;  1675  1     !       INFORMATION FOR DEVICE TESTING.  GET ONE UNIT'S INFORMATION IF
;  1676  1     !       THIS IS A SEQUENTIAL DIAGNOSTIC. NUMBER OF UNITS AVAILABLE IS IN
;  1677  1     !       A HEADER LOCATION: "L$UNIT".
;  1678  1     !--
;  1679  1
;  1680  2     BGNINIT;
;  1681  2
;  1682  2     LOCAL
;  1683  2       START_FLAG,                           ! SET IF THIS PASS IS A START
;  1684  2       DELAY_MULT,                           ! CONTAINS DELAY FACTOR
;  1685  2       cont_flag;                            ! set if event flag ef_continue
;  1686  2
;  1687  2     SETPRI (PRI07);                         ! PRIORITY 7 - NO INTERRUPTS ALLOWED
;  1688  2     START_FLAG = CLEAR_FLG;                 ! CLEAR FLAG BEFORE TESTING IT
;  1689  2     cont_flag = clear_flg;                  ! same, clear continue flag before use
;  1690  2
;  1691  2     IF READEF (EF_PWR)                      ! ARE WE HERE BECAUSE OF POWER FAIL?
;  1692  2       THEN
;  1693  3         BEGIN
;  1694  3           PRINTF ( MSG54 );                 ! "THERE WAS POWER FAILURE - WAITING"
;  1695  3
;  1696  3           INCR COUNT FROM 0 TO 60 DO        ! WAIT APPROX. 60 SECONDS
;  1697  4             BEGIN
;  1698  4               DELAY_MULT = 10000;
;  1699  4               DELAY (.DELAY_MULT);
;  1700  4               BREAK;                        ! BREAK FOR APT
;  1701  3             END;
;  1702  2         END;
;  1703  2
;  1704  2     IF READEF (EF_START)                    ! IS THIS A START ?
;  1705  2       THEN
;  1706  3         BEGIN
```

```
;   1707  3          START_FLAG = TRUE;
;   1708  2        END;
;   1709  2
;   1710  2      !++
;   1711  2      !       CLEAR HARDWARE P-TABLE ON A START BEFORE DOING THE GPHARDS
;   1712  2      !--
;   1713  2
;   1714  2      IF .START_FLAG OR READEF (EF_NEW) OR READEF (EF_CONTINUE)
;   1715  2        THEN                                  ! IF THIS IS A START
;   1716  3          BEGIN
;   1717  3
;   1718  3            INCR INDEX FROM 0 TO HWP_SIZE BY 2 DO     ! ZERO OUT THE TABLES
;   1719  3              (HWP_TABLE + .INDEX) = 0;
;   1720  3            logun = -1 ;
;   1721  2          END;
;   1722  2
;   1723  2
;   1724  2      !++
;   1725  2      !   GET BASE ADDRESS OF HARDWARE P-TABLE AND DEQNA I/O PAGE
;   1726  2      ! -
;   1727  2
;   1728  2           logun = .logun +1;               !advance to next unit 0 or 1 2 = done
;   1729  2            if .logun NEQU 2
;   1730  2            then
;   1731  3            begin
;   1732  3             LOCAL TABLE_POINTER;
;   1733  3             IF GPHARD ( .logun, TABLE_POINTER ) NEQU 0     ! GET P-TABLE ADDRESS
;   1734  3              THEN
;   1735  4                BEGIN
;   1736  4                 IOP_DATA  = .HWP_TABLE [ ADDR ];
;   1737  4                 HWP_TABLE = .TABLE_POINTER;                ! SAVE HW P-TABLE ADDRESS
;   1738  4                 REG_ADR   = .HWP_TABLE [ ADDR ];           ! SAVE I/O PAGE BASE ADDRESS
;   1739  4                 GET_ADR   = .HWP_TABLE [ ADDR ];           ! SAVE I/O PAGE BASE ADDRESS
;   1740  4                 TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;   1741  4                 INCR INDEX FROM 0 TO 7 DO
;   1742  5                  BEGIN
;   1743  5                   IOP_TABLE [ .INDEX ] = .TMP_IOP_ADR;
;   1744  5                   TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;   1745  4                  END;
;   1746  3                END;
;   1747  2              END;
;   1748  2           if .logun EQLU 2
;   1749  2              then
;   1750  3              begin
;   1751  3              logun =  1 ;
;   1752  2              end;
;   1753  2      RETURN;
;   1754  1      ENDINIT;

                                        .GLOBL  L$DLY
```

```
                                        .SBTTL  LINIT INITIALIZE SECTION
000000 004137 000000G          LINIT:   JSR     R1,$SAVE4              ;                                1649
000004 005746                           TST     -(SP)
000006 012700 000340                    MOV     #340,R0                ;                                1687
000012 104441                           TRAP    41
000014 005004                           CLR     R4                     ; START.FLAG                     1688
000016 012700 000034                    MOV     #34,R0                 ;                                1691
000022 104447                           TRAP    47
000024 103027                           BHIS    6$
000026 012746 000000G                   MOV     #MSG54,-(SP)           ;                                1694
000032 012746 000001                    MOV     #1,-(SP)
000036 010600                           MOV     SP,R0                  ; SP,*
000040 104417                           TRAP    17
000042 012702 000075                    MOV     #75,R2                 ; *,COUNT                        1696
000046 012703 023420           1$:      MOV     #23420,R3              ; *,DELAY.MULT                   1698
000052 010301                           MOV     R3,R1                  ; DELAY.MULT,$$TMP2              1699
000054 001410                  2$:      BEQ     5$
000056 013700 000000G                   MOV     L$DLY,R0               ; *,$$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,*                        1696
000102 022626                           CMP     (SP)+,(SP)+            ;                                1693
000104 012700 000040           6$:      MOV     #40,R0                 ;                                1704
000110 104447                           TRAP    47
000112 103002                           BHIS    7$
000114 012704 000001                    MOV     #1,R4                  ; *,START.FLAG                   1707
000120 006004                  7$:      ROR     R4                     ; START.FLAG                     1714
000122 103410                           BLO     8$
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 103013                           BHIS    10$
000144 005000                  8$:      CLR     R0                     ; INDEX                          1718
000146 005060 000000G          9$:      CLR     HWP.TABLE(R0)          ; *(INDEX)                       1719
000152 062700 000002                    ADD     #2,R0                  ; *,INDEX                        1718
000156 020027 000002                    CMP     R0,#2                  ; INDEX,*
000162 003771                           BLE     9$
000164 012737 177777 000000G            MOV     #-1,LOGUN              ;                                1720
000172 005237 000000G          10$:     INC     LOGUN                  ;                                1728
000176 023727 000000G 000002            CMP     LOGUN,#2               ;                                1729
000204 001441                           BEQ     13$
000206 013700 000000G                   MOV     LOGUN,R0               ;                                1733
000212 104442                           TRAP    42
000214 005700                           TST     R0                     ; TABLE.POINTER
000216 001430                           BEQ     12$
000220 017737 000000G 000000G           MOV     @HWP.TABLE,IOP.DATA    ;                                1736
000226 010037 000000G                   MOV     R0,HWP.TABLE           ; TABLE.POINTER,*                1737
```

C7


```
000232  011000                                  MOV     (R0),R0                 ; HWP.TABLE,*                   1738
000234  010037  000000G                         MOV     R0,REG.ADR
000240  010037  000000G                         MOV     R0,GET.ADR              ;                               1739
000244  010037  000000G                         MOV     R0,TMP.IOP.ADR          ;                               1740
000250  005000                                  CLR     R0                      ; INDEX                         1741
000252  013760  000000G 000000G         11$:    MOV     TMP.IOP.ADR,IOP.TABLE(R0)       ; *,*(INDEX)            1743
000260  062737  000002  000000G                 ADD     #2,TMP.IOP.ADR          ;                               1744
000266  062700  000002                          ADD     #2,R0                   ; *,INDEX                       1741
000272  020027  000016                          CMP     R0,#16                  ; INDEX,*
000276  003765                                  BLE     11$
000300  023727  000000G 000002          12$:    CMP     LOGUN,#2                ;                               1748
000306  001003                                  BNE     14$
000310  012737  177777  000000G         13$:    MOV     #-1,LOGUN               ;                               1751
000316  005726                          14$:    TST     (SP)+                   ;                               1649
000320  000207                                  RTS     PC

; Routine Size:  105 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                ;                               1753
000004  104411                                  TRAP    11
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0342
; Maximum stack depth per invocation:  2 words


;   1755  1
;   1756  1
;   1757  1
```

```
;   1758  1     %SBTTL 'AUTODROP SECTION'
;   1759  1
;   1760  1     !++
;   1761  1     !
;   1762  1     !       THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;   1763  1     !       THE "ADR" FLAG WAS SET.  THE UNIT UNDER TEST IS CHECKED TO
;   1764  1     !       SEE IF IT WILL RESPOND.  IF IT DOESN'T IT IS IMMEDIATELY
;   1765  1     !       DROPPED FROM TESTING.
;   1766  1     !
;   1767  1     !--
;   1768  1
;   1769  2     BGNAUTO;
;   1770  2
;   1771  2       RETURN;
;   1772  2
;   1773  1     ENDAUTO;


                                        .SBTTL  LAUTO AUTODROP SECTION
000000  000207                  LAUTO:  RTS     PC                      ;                       1754

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0352
; Maximum stack depth per invocation:  0 words




                                        .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  000352'          L$AUTO::JSR     PC,LAUTO                ;                       1771
000004  104461                          TRAP    61
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0354
; Maximum stack depth per invocation:  2 words


;   1774  1
;   1775  1
```

E7


```
;   1776  1     %SBTTL 'CLEANUP CODING SECTION'
;   1777  1
;   1778  1     !++
;   1779  1     !
;   1780  1     !       THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
;   1781  1     !       AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
;   1782  1     !
;   1783  1     !       INSERT YOUR CLEANUP CODING.  THIS CODING SHOULD
;   1784  1     !       RESTORE YOUR TEST DEVICE TO A NEUTRAL STATE.
;   1785  1     !       THIS CODE WILL BE EXECUTED AFTER EACH PASS AND AFTER THE
;   1786  1     !       PROGRAM IS INTERRUPTED BY "^C".
;   1787  1     !--
;   1788  1
;   1789  2     BGNCLN;
;   1790  2
;   1791  2       clrvec (4);           !give trap 4 vector back to supervisor
;   1792  2       RETURN;
;   1793  2
;   1794  1     ENDCLN;


                                        .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  012700  000004          LCLEAN: MOV     #4,R0                           ;                       1791
000004  104436                          TRAP    36
000006  000207                          RTS     PC                              ;                       1773

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0364
; Maximum stack depth per invocation:  2 words



                                        .SBTTL  L$CLEAN CLEANUP CODING SECTION
000000  004737  000364'        L$CLEAN::
                                        JSR     PC,LCLEAN                       ;                       1792
000004  104412                          TRAP    12
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0374
; Maximum stack depth per invocation:  2 words


;   1795  1
;   1796  1
```

F7


```
;   1797  1     %SBTTL 'DROP UNIT SECTION'
;   1798  1
;   1799  1     !++
;   1800  1     !
;   1801  1     !       THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;   1802  1     !       TO NO LONGER BE TESTED.
;   1803  1     !
;   1804  1     !       INSERT DROP CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
;   1805  1     !       A "DROP" COMMAND OR A "DODU" MACRO EXECUTION.  THE PURPOSE
;   1806  1     !       OR THIS CODE IS TO DO ANY NECESSARY HOUSEKEEPING AFTER A
;   1807  1     !       UNIT HAS BEEN DROPPED.
;   1808  1     !
;   1809  1     !--
;   1810  1
;   1811  2     BGNDU;
;   1812  2
;   1813  2       RETURN;
;   1814  2
;   1815  1     ENDDU;


                                            .SBTTL  LDU DROP UNIT SECTION
000000  000207                         LDU:    RTS     PC                      ;                               1794

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0404
; Maximum stack depth per invocation:  0 words



                                            .SBTTL  L$DU DROP UNIT SECTION
000000  004737  000404'                 L$DU::  JSR     PC,LDU                  ;                               1813
000004  104453                                  TRAP    53
000006  000207                                 RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0406
; Maximum stack depth per invocation:  2 words


;   1816  1
;   1817  1
```

```
;   1818  1     %SBTTL 'ADD UNIT SECTION'
;   1819  1
;   1820  1     !++
;   1821  1     !
;   1822  1     !       THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;   1823  1     !       TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;   1824  1     !       TO THE TEST CYCLE.
;   1825  1     !
;   1826  1     !       INSERT ADD CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
;   1827  1     !       AN "ADD" COMMAND.  THE PURPOSE OF THIS CODE IS TO DO ANY
;   1828  1     !       HOUSEKEEPING THAT MAY BE NECESSARY AFTER A UNIT HAS BEEN ADDED.
;   1829  1     !
;   1830  1     !--
;   1831  1
;   1832  2     BGNAU;
;   1833  2
;   1834  2       RETURN;
;   1835  2
;   1836  1     ENDAU;


                                        .SBTTL  LAU ADD UNIT SECTION
000000  000207                  LAU:    RTS     PC                      ;                       1815

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0416
; Maximum stack depth per invocation:  0 words



                                        .SBTTL  L$AU ADD UNIT SECTION
000000  004737  000416'          L$AU:: JSR     PC,LAU                  ;                       1834
000004  104452                          TRAP    52
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0420
; Maximum stack depth per invocation:  2 words


;   1837  1
;   1838  1
```

```
;   1839  1
;   1840  2     BGNSRV (NXM_INT);
;   1841  2
;   1842  2     !++
;   1843  2     !
;   1844  2     !       GLOBAL LOCATION "INTERRUPT_FLG" IS SET TO TRUE WHICH INDICATES
;   1845  2     !       THE INITIALIZATION SEQUENCE INTERRUPT OCCURED.
;   1846  2     !
;   1847  2     !--
;   1848  2
;   1849  2        INTERRUPT_FLG     = %O'177777';
;   1850  2
;   1851  1     ENDSRV;


                                            .SBTTL  NXM.INT ADD UNIT SECTION
000000  012737  177777  000000G        NXM.INT::
                                            MOV     #-1,INTERRUPT.FLG           ;                   1849
000006  000002                              RTI                                 ;                   1840

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0430
; Maximum stack depth per invocation:  0 words
```

```
;   1852  1
;   1853  2     BGNSRV (QNA_INT);
;   1854  2
;   1855  2     !++
;   1856  2     !
;   1857  2     !       GLOBAL LOCATION "INTERRUPT_FLG" IS SET TO TRUE WHICH INDICATES
;   1858  2     !       THE INITIALIZATION SEQUENCE INTERRUPT OCCURED.
;   1859  2     !       in addition, interrupt causing bits in QNA csr are cleared
;   1860  2     !       (write 1 to clear)
;   1861  2     !
;   1862  2     !--
;   1863  2
;   1864  2        PUT_BIT [ CSR, XI, SET_IT ];          !wr 1 clr XI (RI & NXM by hrdwr design)
;   1865  2        INTERRUPT_FLG     = %O'177777';
;   1866  2
;   1867  1     ENDSRV;


                                        .SBTTL  QNA.INT ADD UNIT SECTION
000000  010046                         QNA.INT::
                                        MOV     R0,-(SP)                        ;                       1853
000002  013700  000000G                 MOV     REG.ADR,R0                      ;                       1864
000006  152760  000200  000016         BISB    #200,16(R0)
000014  012737  177777  000000G         MOV     #-1,INTERRUPT.FLG               ;                       1865
000022  012600                          MOV     (SP)+,R0                        ;                       1853
000024  000002                          RTI

; Routine Size:  11 words,      Routine Base:  AA$CODE$ + 0440
; Maximum stack depth per invocation:  2 words


;   1868  1
;   1869  1
;   1870  1     END
;   1871  0     ELUDOM


;                                       OTS external references
                                                .GLOBL  $SAVE4


;                                       PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;         $CODE$                          53       RO ,  I  ,  LCL,  REL,  CON
;         AA$CODE$                       155       RO ,  I  ,  LCL,  REL,  CON



;                                 Library Statistics
;
```

J7


```
;                                                  - Symbols      --        Pages       Processing
;       File                               Total     Loaded     Percent     Mapped      Time
;
;  DISK2:[SCODA.QNA.ZQNA]QNALIB.L13;2        224        51          22         14         00:00.1



;                                       COMMAND QUALIFIERS

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

; Size:          155 code + 53 data words
; Run Time:         00:10.0
; Elapsed Time:     00:11.2
; Lines/CPU Min:    11192
; Lexemes/CPU-Min: 78119
; Memory Used:  184 pages
; Compilation Complete
```

K7


```
;    0001  0     MODULE ZQNA3 (%TITLE 'CZQNAEO 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_RX_LPSTATUS       : NOVALUE,
;    1514  1        CHK_XMIT_STATUS       : NOVALUE,
;    1515  1        CLR_BUFFERS           : NOVALUE,
;    1516  1        CLR_DESCR             : NOVALUE,
;    1517  1        COMPARE_PACKETS       : NOVALUE,
;    1518  1        E1$REPORT             : NOVALUE,          ! PRINT EXTENDED ERROR MESSAGE
;    1519  1        ERROR$REPORT          : NOVALUE,          ! PRINT EXTENDED ERROR MESSAGE
;    1520  1        FORM_HEX_ADR          : NOVALUE,
;    1521  1        KBD_INT               : NOVALUE,
;    1522  1        NXM_INT               : L$ISR NOVALUE,    ! NXM INTERRUPT SERVICE ROUTINE
;    1523  1        QNA_INT               : L$ISR NOVALUE,    ! QNA INTERRUPT SERVICE ROUTINE
;    1524  1        PREP_FOR_SETUP        : NOVALUE,
;    1525  1        PWR_INT               : NOVALUE,
;    1526  1        RESET_DEQNA           : NOVALUE,
;    1527  1        SEND_ELOOP_PACKET     : NOVALUE,
;    1528  1        SEND_TEST_PACKET      : NOVALUE,
;    1529  1        INTR_TEST_PACKET      : NOVALUE,
;    1530  1        SET_XDESCR_LIST       : NOVALUE,
;    1531  1        SET_RDESCR_LIST       : NOVALUE,
;    1532  1        TURN_OFF_LED          : NOVALUE,
;    1533  1        VER_DESCR_STATUS      : NOVALUE,
;    1534  1        WAIT_FOR_TIMEOUT      : NOVALUE,
;    1535  1        WALKING_BIT           : NOVALUE,
;    1536  1        WRT_STATION_ADR       : NOVALUE,
;    1537  1        XMIT_AND_RCV_PACKET   : NOVALUE,
;    1538  1        XMIT_ILOOP_PACKET     : NOVALUE,
;    1539  1        XMIT_SETUP_PACKET     : NOVALUE,
;    1540  1        romchk                : novalue;     !does sumcheck for station addrs rom
;    1541  1
```

```
;    1542  1
;    1543  1
;    1544  1     EXTERNAL
;    1545  1
;    1546  1     !++
;    1547  1     !       COMMUNICATION AREA DECLARATIONS
;    1548  1     !--
;    1549  1
;    1550  1         RCV_D_LIST      : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1551  1         XMIT_D_LIST     : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1552  1         DESCR_LIST      : BLOCK  [ DESCR_SIZE, WORD] FIELD ( DL_FIELDS ),
;    1553  1         RCV_BUFFER      : VECTOR [ B_SIZE, BYTE ],
;    1554  1         XMIT_BUFFER     : VECTOR [ B_SIZE, BYTE ],
;    1555  1         DATA_BUFFER     : VECTOR [ BUF_SIZE, BYTE ],
;    1556  1         TARGET_ADR      : VECTOR [ T_SIZE, BYTE ],
;    1557  1         PHYS_ADR        : VECTOR [ 22, BYTE ],
;    1558  1         IOP_TABLE       : VECTOR [ 8, WORD ],
;    1559  1         RD13            : VECTOR [ 64, WORD ],
;    1560  1         TD13            : VECTOR [ 28, WORD ],
;    1561  1         TD16            : VECTOR [ 44, WORD ],
;    1562  1         BD_PROM_DESCR   : VECTOR [ BD_D_SIZE, WORD ],
;    1563  1         STATION_ADR     : VECTOR [ 4, WORD ],
;    1564  1         PTRN_TABLE      : VECTOR [ 8, BYTE ],
;    1565  1
;    1566  1     !++
;    1567  1     !   HARDWARE AND SOFTWARE P-TABLE STORAGE DECLARATIONS
;    1568  1     !
;    1569  1
;    1570  1         HWP_TABLE   : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;    1571  1         SWP_TABLE   : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    1572  1
;    1573  1         REG_ADR             : REF REG_STR FIELD ( IOP_FIELDS ),
;    1574  1         GET_ADR             : REF ADR_STR FIELD ( IOP_FIELDS ),
;    1575  1         IOP_DATA            : REF REG_STR FIELD ( IOP_FIELDS ),
;    1576  1
```

```
;  1577  1                                                ! (0=NONE, 1=L CLOCK,1=P_CLOCK)
;  1578  1     !++
;  1579  1     !       MISCELLANEOUS DATA DECLARATIONS
;  1580  1     !--
;  1581  1
;  1582  1         XBUF_LENGTH,        RBUF_LENGTH,    INTERRUPT_FLG,      COUNTER,
;  1583  1         SWP_BLOCK_MEM,      SWP_TOUT_VAL,   SWP_ILOOP,          SWP_TIMER,
;  1584  1         UP_COUNTER,         DOWN_COUNTER,   CHECKSUM,           ERR_NUMBER,
;  1585  1         XC_FLAG,            SWP_LBC,        SWP_NXM,
;  1586  1         ERR_COUNT,          ERR_FLAG,       CSR_WORD,           PRI00,
;  1587  1         PRI01,              PRI02,          PRI03,              PRI04,
;  1588  1         PRI05,              PRI06,          PRI07,              DEQNA_NO  : WORD,
;  1589  1
;  1590  1     !++
;  1591  1     !   TEMPORARY STORAGE DATA DECLARATIONS
;  1592  1     !--
;  1593  1
;  1594  1         P1,                 P2,             P3,                 P4,
;  1595  1         TMP_IOP_ADR,        TMP_REG_DATA,   TEMP1,              TEMP2,
;  1596  1         TEMP3,              TEMP4,          TEMP5,              TEMP6,
;  1597  1         TEMP7,              TEMP8,          TEMP9,              TADR1,
;  1598  1         TADR2                                                           : WORD,
;  1599  1         TBYTE1,             TBYTE2,         TBYTE3,             TBYTE4    : BYTE,
;  1600  1
```

```
;    1601  1
;    1602  1     !++
;    1603  1     !  ERROR MESSAGES DEFINED EXTERNALLY
;    1604  1     !--
;    1605  1
;    1606  1
;    1607  1         MSG00, MSG71, msg72, msg73, msg74, msg75, msg76,
;    1608  1         MSG01, MSG02, MSG03, MSG04, MSG05, MSG06, MSG07, MSG08, MSG09, MSG10,
;    1609  1         MSG11, MSG12, MSG13, MSG14, MSG15, MSG16, MSG17, MSG18, MSG19, MSG20,
;    1610  1         MSG21, MSG22, MSG23, MSG24, MSG25, MSG26, MSG27, MSG28, MSG29, MSG30,
;    1611  1         MSG31, MSG32, MSG33, MSG34, MSG35, MSG36, MSG37, MSG38, MSG39, MSG40,
;    1612  1         MSG41, MSG42, MSG43, MSG44, MSG45, MSG46, MSG47, MSG48, MSG49, MSG50,
;    1613  1         MSG51, MSG52, MSG53, MSG54, MSG55, MSG56, MSG57, MSG58, MSG59, MSG60,
;    1614  1         MSG61, MSG62, MSG63, MSG64, MSG65, MSG66, MSG67, MSG68, MSG69, MSG70;
;    1615  1
;    1616  1
```

```
;  1617  1     %SBTTL 'TEST 1   NON-EXISTANT I/O PAGE REGISTER TEST'
;  1618  1     !++
;  1619  1     !
;  1620  1     !  TEST 1:      NON-EXISTANT I/O PAGE REGISTER TEST
;  1621  1     !
;  1622  1     !  DESCRIPTION:
;  1623  1     !
;  1624  1     !       This test verifies that all the device registers residing in the
;  1625  1     !       I/O Page can be accessed without forcing a non-existant memory (NXM)
;  1626  1     !       interrupt. If the operator specifies loop on error, the program
;  1627  1     !       re-executes the code that detected the error until ^C is entered.
;  1628  1     !
;  1629  1     !       Hardware tested:         Q-Bus to DEQNA Slave Registers Interface
;  1630  1     !
;  1631  1     !       Processing:
;  1632  1     !
;  1633  1     !            BEGIN
;  1634  1     !               get ready for NXM interrupt
;  1635  1     !               REPEAT for every I/O page register
;  1636  1     !                  read I/O page register
;  1637  1     !                  IF NXM occured
;  1638  1     !                  THEN
;  1639  1     !                     print error message if not inhibited
;  1640  1     !                  ENDIF
;  1641  1     !               ENDREPEAT
;  1642  1     !
;  1643  1     !               write any data pattern into the first 2 I/O page
;  1644  1     !                 registers
;  1645  1     !               IF NXM occured
;  1646  1     !               THEN
;  1647  1     !                  print error message if not inhibited
;  1648  1     !               ENDIF
;  1649  1     !            END
;  1650  1     !--
;  1651  1     !
;  1652  1     !       test was modified to ensure that return PC after a trap to 4 was to
;  1653  1     !  a valid address. Previous code read the csr contents to a global location.
;  1654  1     !  The compiler generated the following assembly code : mov @addrs,temp1
;  1655  1     !  or 017737 OFFSET DESTADR . The pc was incremented by 2 after reading the
;  1656  1     !  instruction 017737, incremented again by 2 after reading the offset for the
;  1657  1     !  CSR. This left the PC pointing to the dest address stored in memory. If a bus
;  1658  1     !  timeout trap to 4 occurred, the return pc was still pointing to the dest
;  1659  1     !  address. When the RTI instruction in NXM_INT was executed, the next
;  1660  1     !  instruction to be executed was the dest address!! By making the dest location
;  1661  1     !  a local symbol, the resulting assembly code is mov @addrs,R3 of 017703 OFFSET
;  1662  1     !  now, the PC is incremented by 2 after reading the instruction 017703, and
;  1663  1     !  incremented again by 2 after reading the offset for the QNA CSR. The PC now
;  1664  1     !  points to the next valid instruction, because the dest address is internal
;  1665  1     !  R3, and its address does not have to be fetched from memory.
;  1666  1     !
```

```
;  1667  3     BGNTST;
;  1668  3     LOCAL
;  1669  3       LOCFLG;                 !used to force 0'177__ instr out of compiler
;  1670  3
;  1671  3
;  1672  3     SETVEC (4, NXM_INT, PRI07);                     ! SET UP FOR AN NXM INTERRUPT
;  1673  3     DELAY (M5_DELAY);                               ! DELAY 50 x 100 us = 5 ms
;  1674  3     INTERRUPT_FLG = CLEAR_FLG;                      ! CLEAR OUT NEX FLAG
;  1675  3
;  1676  3     TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;  1677  3     INCR INDEX FROM 0 TO 7 DO
;  1678  4       BEGIN
;  1679  6         BGNSUB;
;  1680  6           LOCFLG = ..TMP_IOP_ADR;
;  1681  6           DELAY(7);
;  1682  6           IF .INTERRUPT_FLG EQLU WORD_LIMIT         ! SEE IF WE GOT A NXM INTRT
;  1683  6             THEN
;  1684  7               BEGIN                                 ! ADDRESS NOT THERE
;  1685  7                 CLRVEC (4);                         !return vector to supervisc.
;  1686  7                 INTERRUPT_FLG = CLEAR_FLG;          ! CLEAR TRAP FLAG
;  1687  7                 PRINTB ( MSG59 );
;  1688  7                 PRINTB ( MSG70, .TMP_IOP_ADR );
;  1689  7                 ERRDF (0101, MSG00, E1$REPORT);     ! 'I/O PAGE REG. NOT PRESENT'
;  1690  7                 DOCLN;
;  1691  6               END;
;  1692  4         ENDSUB;
;  1693  4         TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;  1694  3       END;
;  1695  3
;  1696  3     TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;  1697  3     INCR INDEX FROM 0 TO 1 DO
;  1698  4       BEGIN
;  1699  6         BGNSUB;
;  1700  6           .TMP_IOP_ADR = %X'7F';                    ! WRITE FIRST 2 LOCATIONS
;  1701  6           DELAY(7);
;  1702  6           IF .INTERRUPT_FLG EQLU WORD_LIMIT         ! SEE IF WE GOT A NXM INTRT
;  1703  6             THEN
;  1704  7               BEGIN                                 ! ADDRESS NOT THERE
;  1705  7                 CLRVEC (4);                         !return vector to supervisor
;  1706  7                 INTERRUPT_FLG = CLEAR_FLG;          ! CLEAR TRAP FLAG
;  1707  7                 PRINTB ( MSG59 );
;  1708  7                 PRINTB ( MSG70, .TMP_IOP_ADR );
;  1709  7                 ERRDF (0102, MSG00, E1$REPORT);     ! 'I/O PAGE REG. NOT PRESENT'
;  1710  7                  DOCLN;
;  1711  6               END;
;  1712  4         ENDSUB;
;  1713  4         TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;  1714  3       END;
;  1715  3
;  1716  3     CLRVEC (4);                                     ! CLEAR INTERRUPT VECTOR
;  1717  3
;  1718  1     ENDTST;
```

```
                        .TITLE  ZQNA3 CZQNAE0 DEQNA FUNCTIONAL TEST
                        .IDENT  /V01.0/
                        .ENABL  AMA

                        .GLOBL  CHK.CSR.STATUS, CHK.RIXI.STATUS
                        .GLOBL  CHK.RCV.STATUS, CHK.RX.LPSTATUS
                        .GLOBL  CHK.XMIT.STATUS, CLR.BUFFERS, CLR.DESCR
                        .GLOBL  COMPARE.PACKETS, E1$REPORT, ERROR$REPORT
                        .GLOBL  FORM.HEX.ADR, KBD.INT, NXM.INT
                        .GLOBL  QNA.INT, PREP.FOR.SETUP, PWR.INT
                        .GLOBL  RESET.DEQNA, SEND.ELOOP.PACKET
                        .GLOBL  SEND.TEST.PACKET, INTR.TEST.PACKET
                        .GLOBL  SET.XDESCR.LIST, SET.RDESCR.LIST
                        .GLOBL  TURN.OFF.LED, VER.DESCR.STATUS
                        .GLOBL  WAIT.FOR.TIMEOUT, WALKING.BIT
                        .GLOBL  WRT.STATION.ADR, XMIT.AND.RCV.PACKET
                        .GLOBL  XMIT.ILOOP.PACKET, XMIT.SETUP.PACKET
                        .GLOBL  ROMCHK, RCV.D.LIST, XMIT.D.LIST
                        .GLOBL  DESCR.LIST, RCV.BUFFER, XMIT.BUFFER
                        .GLOBL  DATA.BUFFER, TARGET.ADR, PHYS.ADR
                        .GLOBL  IOP.TABLE, 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, SWP.NXM, 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, MSG71, MSG72, MSG73, MSG74
                        .GLOBL  MSG75, MSG76, MSG01, MSG02, MSG03
                        .GLOBL  MSG04, MSG05, MSG06, MSG07, MSG08
                        .GLOBL  MSG09, MSG10, MSG11, MSG12, MSG13
                        .GLOBL  MSG14, MSG15, MSG16, MSG17, MSG18
                        .GLOBL  MSG19, MSG20, MSG21, MSG22, MSG23
                        .GLOBL  MSG24, MSG25, MSG26, MSG27, MSG28
                        .GLOBL  MSG29, MSG30, MSG31, MSG32, MSG33
                        .GLOBL  MSG34, MSG35, MSG36, MSG37, MSG38
                        .GLOBL  MSG39, MSG40, MSG41, MSG42, MSG43
                        .GLOBL  MSG44, MSG45, MSG46, MSG47, MSG48
                        .GLOBL  MSG49, MSG50, MSG51, MSG52, MSG53
                        .GLOBL  MSG54, MSG55, MSG56, MSG57, MSG58
                        .GLOBL  MSG59, MSG60, MSG61, MSG62, MSG63
                        .GLOBL  MSG64, MSG65, MSG66, MSG67, MSG68
                        .GLOBL  MSG69, MSG70, L$DLY
```

F 8


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

000000  004137  000000G          $T1:   JSR     R1,$SAVE3                       ;                       1614
000004  005746                         TST     -(SP)
000006  012746  000000G                MOV     #PRIO7,-(SP)                    ;                       1672
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                          ; *,$$TMP2              1673
000034  001410                  1$:    BEQ     4$
000036  013700  000000G                MOV     L$DLY,R0                        ; *,$$TMP1
000042  001403                         BEQ     3$
000044  005066  000010          2$:    CLR     10(SP)                          ; $$TMP
000050  077003                         SOB     R0,2$                           ; $$TMP1,*
000052  005301                  3$:    DEC     R1                              ; $$TMP2
000054  000767                         BR      1$
000056  005037  000000G          4$:    CLR     INTERRUPT.FLG                   ;                       1674
000062  017737  000000G 000000G        MOV     @HWP.TABLE,TMP.IOP.ADR          ;                       1676
000070  012702  000010                 MOV     #10,R2                          ; *,INDEX               1677
000074  104402                  5$:    TRAP    2                               ;                       1678
000076  017703  000000G                MOV     @TMP.IOP.ADR,R3                 ; *,LOCFLG              1680
000102  012701  000007                 MOV     #7,R1                           ; *,$$TMP2              1681
000106  001410                  6$:    BEQ     9$
000110  013700  000000G                MOV     L$DLY,R0                        ; *,$$TMP1
000114  001403                         BEQ     8$
000116  005066  000010          7$:    CLR     10(SP)                          ; $$TMP
000122  077003                         SOB     R0,7$                           ; $$TMP1,*
000124  005301                  8$:    DEC     R1                              ; $$TMP2
000126  000767                         BR      6$
000130  023727  000000G 177777  9$:    CMP     INTERRUPT.FLG,#-1               ;                       1682
000136  001032                         BNE     10$
000140  012700  000004                 MOV     #4,R0                           ;                       1685
000144  104436                         TRAP    36
000146  005037  000000G                CLR     INTERRUPT.FLG                   ;                       1686
000152  012716  000000G                MOV     #MSG59,(SP)                     ;                       1687
000156  012746  000001                 MOV     #1,-(SP)
000162  010600                         MOV     SP,R0                           ; SP,*
000164  104414                         TRAP    14
000166  013716  000000G                MOV     TMP.IOP.ADR,(SP)                ;                       1688
000172  012746  000000G                MOV     #MSG70,-(SP)
000176  012746  000002                 MOV     #2,-(SP)
000202  010600                         MOV     SP,R0                           ; SP,*
000204  104414                         TRAP    14
000206  104455                         TRAP    55                              ;                       1689
000210  000145                         .WORD   145
000212  000000G                        .WORD   MSG00
000214  000000G                        .WORD   E1$REPORT
000216  104444                         TRAP    44
000220  062706  000006                 ADD     #6,SP                           ;                       1684
000224  104467                  10$:   TRAP    67                              ;                       1691
```

```
000226  006000                                  ROR     R0
000230  103721                                  BLO     5$
000232  062737  000002  000000G                 ADD     #2,TMP.IOP.ADR          ;                               1693
000240  077263                                  SOB     R2,5$                   ; INDEX,*                       1677
000242  017737  000000G 000000G                 MOV     @HWP.TABLE,TMP.IOP.ADR  ;                               1696
000250  012702  000002                          MOV     #2,R2                   ; *,INDEX                       1697
000254  104402                          11$:    TRAP    2                       ;                               1698
000256  012777  000177  000000G                 MOV     #177,@TMP.IOP.ADR       ;                               1700
000264  012701  000007                          MOV     #7,R1                   ; *,$$TMP2                      1701
000270  001410                          12$:    BEQ     15$
000272  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000276  001403                                  BEQ     14$
000300  005066  000010                  13$:    CLR     10(SP)                  ; $$TMP
000304  077003                                  SOB     R0,13$                  ; $$TMP1,*
000306  005301                          14$:    DEC     R1                      ; $$TMP2
000310  000767                                  BR      12$
000312  023727  000000G 177777          15$:    CMP     INTERRUPT.FLG,#-1       ;                               1702
000320  001032                                  BNE     16$
000322  012700  000004                          MOV     #4,R0                   ;                               1705
000326  104436                                  TRAP    36
000330  005037  000000G                         CLR     INTERRUPT.FLG           ;                               1706
000334  012716  000000G                         MOV     #MSG59,(SP)             ;                               1707
000340  012746  000001                          MOV     #1,-(SP)
000344  010600                                  MOV     SP,R0                   ; SP,*
000346  104414                                  TRAP    14
000350  013716  000000G                         MOV     TMP.IOP.ADR,(SP)        ;                               1708
000354  012746  000000G                         MOV     #MSG70,-(SP)
000360  012746  000002                          MOV     #2,-(SP)
000364  010600                                  MOV     SP,R0                   ; SP,*
000366  104414                                  TRAP    14
000370  104455                                  TRAP    55                      ;                               1709
000372  000146                                  .WORD   146
000374  000000G                                 .WORD   MSG00
000376  000000G                                 .WORD   E1$REPORT
000400  104444                                  TRAP    44
000402  062706  000006                          ADD     #6,SP                   ;                               1704
000406  104467                          16$:    TRAP    67                      ;                               1711
000410  006000                                  ROR     R0
000412  103720                                  BLO     11$
000414  062737  000002  000000G                 ADD     #2,TMP.IOP.ADR          ;                               1713
000422  077264                                  SOB     R2,11$                  ; INDEX,*                       1697
000424  012700  000004                          MOV     #4,R0                   ;                               1716
000430  104436                                  TRAP    36
000432  062706  000012                          ADD     #12,SP                  ;                               1714
000436  000207                                  RTS     PC

; Routine Size:  144 words,     Routine Base:  AB$CODE$ + 0000
; Maximum stack depth per invocation:  14 words


                                        .SBTTL  T1 TEST 1 - NON-EXISTANT I/O PAGE REGISTER TEST
```

```
000000  004737  000000'                T1::
000000                                  1$:     JSR     PC,$T1                  ;                       1716
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 0440
; Maximum stack depth per invocation:  2 words


;   1719  1
;   1720  1
```

```
;    1721  1     %SBTTL 'TEST 2 - CSR STATIC BIT TEST'
;    1722  1     !++
;    1723  1     !
;    1724  1     !  TEST 2:       CSR STATIC BIT TEST
;    1725  1     !
;    1726  1     !  DESCRIPTION:
;    1727  1     !
;    1728  1     !        This test verifies that the CSR register static bits can be set
;    1729  1     !        and cleared as specified.  The host writes data patterns to this
;    1730  1     !        register and reads them back verifying  no static
;    1731  1     !        (stuck at 1 / stuck at 0) faults occur. If the operator specifies
;    1732  1     !        loop on error, the program re-executes the code that detected the
;    1733  1     !        error until tC is entered.
;    1734  1     !
;    1735  1     !        Hardware tested:                Q-Bus to DEQNA Slave Regs. Interface
;    1736  1     !
;    1737  1     !        Processing:
;    1738  1     !
;    1739  1     !             BEGIN
;    1740  1     !                check Software Reset ( SR ) bit in the CSR for stuck at 0
;    1741  1     !                   and 1
;    1742  1     !                IF error
;    1743  1     !                THEN
;    1744  1     !                   print error message if not inhibited
;    1745  1     !                ENDIF
;    1746  1     !                set static bits ( 0,3,8,9 ) and check for expected CSR status
;    1747  1     !                IF error
;    1748  1     !                THEN
;    1749  1     !                   print error message if not inhibited
;    1750  1     !                ENDIF
;    1751  1     !                clear static bits and check for expected CSR status
;    1752  1     !                IF error
;    1753  1     !                THEN
;    1754  1     !                   print error message if not inhibited
;    1755  1     !                ENDIF
;    1756  1     !                set static bits ( 0,3,8,9 ) and check for expected CSR status
;    1757  1     !                IF error
;    1758  1     !                THEN
;    1759  1     !                   print error message if not inhibited
;    1760  1     !                ENDIF
;    1761  1     !                reset DEQNA and check for expected CSR status
;    1762  1     !                IF error
;    1763  1     !                THEN
;    1764  1     !                   print error message if not inhibited
;    1765  1     !                ENDIF
;    1766  1     !             END
;    1767  1     !--
```

```
;  1768  3     BGNTST;
;  1769  3
;  1770  5     BGNSUB;
;  1771  5
;  1772  5       !++
;  1773  5       !  CHECK IF CSR STATIC BITS (BIT 0,3,8 AND 9) ARE NOT STUCK AT 0
;  1774  5       !--
;  1775  5
;  1776  5       RESET_DEQNA ( );
;  1777  5       PUT_BIT ( CSR, ALL_BITS, PATRN1 );
;  1778  5       DELAY ( TIME6_LIMIT );
;  1779  5       TEMP1 = GET_BIT [ CSR_ALL ] AND PATRN1;
;  1780  5       IF .TEMP1 NEQU PATRN1
;  1781  5         THEN
;  1782  6           BEGIN
;  1783  6             PRINTB ( MSG59 );
;  1784  6             PRINTB ( MSG60 );
;  1785  6             PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP1, PATRN1 );
;  1786  6             ERRDF ( 0201, MSG00, E1$REPORT );
;  1787  5           END;
;  1788  3     ENDSUB;
;  1789  3
;  1790  3     !++
;  1791  3     !  CHECK IF CSR STATIC BITS (BIT 0,3,8 AND 9) ARE NOT STUCK AT 1
;  1792  3     !--
;  1793  3
;  1794  5     BGNSUB;
;  1795  5       PUT_BIT ( CSR, ALL_BITS, ZERO );
;  1796  5       DELAY ( TIME6_LIMIT );
;  1797  5       TEMP2 = GET_BIT [ CSR_ALL ] AND PATRN1;
;  1798  5       IF .TEMP2 NEQU ZERO
;  1799  5         THEN
;  1800  6           BEGIN
;  1801  6             PRINTB ( MSG59 );
;  1802  6             PRINTB ( MSG61 );
;  1803  6             PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP2, ZERO );
;  1804  6             ERRDF ( 0202, MSG00, E1$REPORT );
;  1805  5           END;
;  1806  3     ENDSUB;
;  1807  3
;  1808  5     BGNSUB;
;  1809  5       PUT_BIT ( CSR, ALL_BITS, PATRN1 );
;  1810  5       RESET_DEQNA ( );
;  1811  5       TEMP3 = GET_BIT [ CSR_ALL ] AND PATRN1;
;  1812  5       IF .TEMP3 NEQU ZERO
;  1813  5         THEN
;  1814  6           BEGIN
;  1815  6             PRINTB ( MSG59 );
;  1816  6             PRINTB ( MSG62 );
;  1817  6             PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP3, ZERO );
;  1818  6             ERRDF ( 0203, MSG00, E1$REPORT );
;  1819  5           END;
;  1820  3     ENDSUB;
```

K8


```
;   1821  3
;   1822  1     ENDTST;


                                            .SBTTL  $T2 TEST 2 - CSR STATIC BIT TEST
000000  004137  000000G                $T2: JSR     R1,$SAVE2                      ;                          1718
000004  162706  000016                     SUB     #16,SP
000010  104402                         1$: TRAP    2                              ;                          1768
000012  004737  000000G                     JSR     PC,RESET.DEQNA                 ;                          1776
000016  013701  000000G                     MOV     REG.ADR,R1                     ;                          1777
000022  012761  001411  000016             MOV     #1411,16(R1)
000030  012702  000001                     MOV     #1,R2                          ; *,$$TMP2                 1778
000034  001410                         2$: BEQ     5$
000036  013700  000000G                     MOV     L$DLY,R0                       ; *,$$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           1779
000062  011637  000000G                     MOV     (SP),TEMP1                     ; TMP.LOCATION,*
000066  042737  176366  000000G             BIC     #176366,TEMP1
000074  023727  000000G 001411             CMP     TEMP1,#1411                    ;                          1780
000102  001444                             BEQ     6$
000104  012746  000000G                     MOV     #MSG59,-(SP)                   ;                          1783
000110  012746  000001                     MOV     #1,-(SP)
000114  010600                             MOV     SP,R0                          ; SP,*
000116  104414                             TRAP    14
000120  012716  000000G                     MOV     #MSG60,(SP)                    ;                          1784
000124  012746  000001                     MOV     #1,-(SP)
000130  010600                             MOV     SP,R0                          ; SP,*
000132  104414                             TRAP    14
000134  012716  001411                     MOV     #1411,(SP)                     ;                          1785
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                             ;                          1786
000202  000311                             .WORD   311
000204  000000G                            .WORD   MSG00
000206  000000G                            .WORD   E1$REPORT
000210  062706  000016                     ADD     #16,SP                         ;                          1782
000214  104467                         6$: TRAP    67                             ;                          1787
000216  006000                             ROR     R0
000220  103673                             BLO     1$
000222  104402                         7$: TRAP    2                              ;                          1788
000224  013701  000000G                     MOV     REG.ADR,R1                     ;                          1795
000230  005061  000016                     CLR     16(R1)
000234  012702  000001                     MOV     #1,R2                          ; *,$$TMP2                 1796
```

L8


```
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$:    MOV     16(R1),4(SP)            ; *,TMP.LOCATION                        1797
000270  016637  000004  000000G                 MOV     4(SP),TEMP2             ; TMP.LOCATION,*
000276  042737  176366  000000G                 BIC     #176366,TEMP2
000304  001443                                  BEQ     12$                     ;                                       1798
000306  012746  000000G                         MOV     #MSG59,-(SP)            ;                                       1801
000312  012746  000001                          MOV     #1,-(SP)
000316  010600                                  MOV     SP,R0                   ; SP,*
000320  104414                                  TRAP    14
000322  012716  000000G                         MOV     #MSG61,(SP)             ;                                       1802
000326  012746  000001                          MOV     #1,-(SP)
000332  010600                                  MOV     SP,R0                   ; SP,*
000334  104414                                  TRAP    14
000336  005016                                  CLR     (SP)                    ;                                       1803
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                      ;                                       1804
000402  000312                                  .WORD   312
000404  000000G                                 .WORD   MSG00
000406  000000G                                 .WORD   E1$REPORT
000410  062706  000016                          ADD     #16,SP                  ;                                       1800
000414  104467                          12$:    TRAP    67                      ;                                       1805
000416  006000                                  ROR     R0
000420  103700                                  BLO     7$
000422  104402                          13$:    TRAP    2                       ;                                       1806
000424  013700  000000G                         MOV     REG.ADR,R0              ;                                       1809
000430  012760  001411  000016                  MOV     #1411,16(R0)
000436  004737  000000G                         JSR     PC,RESET.DEQNA          ;                                       1810
000442  013700  000000G                         MOV     REG.ADR,R0              ;                                       1811
000446  016066  000016  000010                  MOV     16(R0),10(SP)           ; *,TMP.LOCATION
000454  016637  000010  000000G                 MOV     10(SP),TEMP3            ; TMP.LOCATION,*
000462  042737  176366  000000G                 BIC     #176366,TEMP3
000470  001443                                  BEQ     14$                     ;                                       1812
000472  012746  000000G                         MOV     #MSG59,-(SP)            ;                                       1815
000476  012746  000001                          MOV     #1,-(SP)
000502  010600                                  MOV     SP,R0                   ; SP,*
000504  104414                                  TRAP    14
000506  012716  000000G                         MOV     #MSG62,(SP)             ;                                       1816
000512  012746  000001                          MOV     #1,-(SP)
000516  010600                                  MOV     SP,R0                   ; SP,*
000520  104414                                  TRAP    14
```

M8


```
000522  005016                          CLR     (SP)                    ;                               1817
000524  013746  000000G                 MOV     TEMP3,-(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                      ;                               1818
000566  000313                          .WORD   313
000570  000000G                         .WORD   MSG00
000572  000000G                         .WORD   E1$REPORT
000574  062706  000016                  ADD     #16,SP                  ;                               1814
000600  104467                  14$:    TRAP    67                      ;                               1819
000602  006000                          ROR     R0
000604  103706                          BLO     13$
000606  062706  000016                  ADD     #16,SP                  ;                               1718
000612  000207                          RTS     PC

; Routine Size:  198 words,     Routine Base:  AB$CODE$ + 0454
; Maximum stack depth per invocation:  19 words



                                        .SBTTL  T2 TEST 2 - CSR STATIC BIT TEST
000000  004737  000454'          T2::
000000                          1$:     JSR     PC,$T2                  ;                               1820
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 1270
; Maximum stack depth per invocation:  2 words


;   1823  1
;   1824  1
```

```
;  1825  1     %SBTTL 'TEST 3   ETHERNET STATION ADDRESS VERIFY TEST'
;  1826  1     !++
;  1827  1     !
;  1828  1     !  TEST 3:      ETHERNET STATION ADDRESS VERIFY TEST
;  1829  1     !
;  1830  1     !  DESCRIPTION:
;  1831  1     !
;  1832  1     !       This test verifies that  the Ethernet Station Address PROM can be
;  1833  1     !       read and loaded to host memory correctly. Ethernet Station Address is
;  1834  1     !       verified and checksum is computed from PROM data read and this checksum
;  1835  1     !       is compared to the checksum stored in the Ethernet Station Address
;  1836  1     !       PROM. Ethernet Station Address is always printed out on the console in
;  1837  1     !       the Ethernet standard format. If the address is not proper, the error
;  1838  1     !       is recorded and an appropriate error message is printed out on the
;  1839  1     !       console. If the operator specifies loop on error, the program
;  1840  1     !       re-executes the code that detected the error until ^C is entered.
;  1841  1     !
;  1842  1     !       Hardware tested:        Station Address PROM
;  1843  1     !                               Q-Bus DMA Interface
;  1844  1     !       Processing:
;  1845  1     !
;  1846  1     !            BEGIN
;  1847  1     !
;  1848  1     !               read DEQNA Station Address PROM and checksum
;  1849  1     !               save copy of Station Address PROM in host memory
;  1850  1     !               print Station Address on the console in standard format
;  1851  1     !               compute Station Address ROM checksum
;  1852  1     !               IF checksum read not equal checksum computed
;  1853  1     !               THEN
;  1854  1     !                  print error message if not inhibited
;  1855  1     !               ENDIF
;  1856  1     !               IF Station Address
;  1857  1     !                     [all 0's]
;  1858  1     !                  OR [all 1's]:
;  1859  1     !                  OR [multicast bit set]:
;  1860  1     !               THEN
;  1861  1     !                  print error message if not inhibited
;  1862  1     !               ENDIF
;  1863  1     !
;  1864  1     !            END
;  1865  1     !--
```

```
;  1866  3     BGNTST;
;  1867  3
;  1868  5     BGNSUB;
;  1869  5       RESET_DEQNA ( );
;  1870  5       FORM_HEX_ADR ( PHA_INDEX );
;  1871  5
;  1872  5       !++
;  1873  5       !  COMPUTE EXPECTED CHECKSUM
;  1874  5       !--
;  1875  5
;  1876  5        romchk ( );                          !use macro routine to calculate sum
;  1877  5                                             !BLISS doesn't know how to do add carry
;  1878  5
;  1879  5     !  CHECKSUM = 0;
;  1880  5     !
;  1881  5     !  INCR INDEX FROM 0 TO 5 BY 2 DO
;  1882  5     !    BEGIN
;  1883  5     !      IF ( .CHECKSUM AND %O'100000' ) NEQU ZERO
;  1884  5     !        THEN
;  1885  5     !          BEGIN
;  1886  5     !            CHECKSUM = .CHECKSUM ^ 1;
;  1887  5     !            CHECKSUM = .CHECKSUM + 1;
;  1888  5     !          END
;  1889  5     !        ELSE
;  1890  5     !          CHECKSUM = .CHECKSUM ^ 1;
;  1891  5     !
;  1892  5     !      CHECKSUM = .CHECKSUM + .STATION_ADR [ .COUNTER ];
;  1893  5     !
;  1894  5     !      IF .CHECKSUM GTRU WORD_LIMIT
;  1895  5     !        THEN
;  1896  5     !          CHECKSUM = .CHECKSUM + 1;
;  1897  5     !
;  1898  5     !      COUNTER = .COUNTER + 1;
;  1899  5     !    END;
;  1900  5
;  1901  5       !++
;  1902  5       !  PRINT PHYSICAL STATION ADDRESS
;  1903  5       !--
;  1904  5
;  1905  5       PRINTB ( MSG01, .HWP_TABLE [ ADDR ] );
;  1906  5       PRINTB ( PHYS_ADR );
;  1907  5
;  1908  5       !++
;  1909  5       !  READ ACTUAL CHECKSUM FROM DEQNA STATION ADDRESS PROM AND COMPARE IT TO
;  1910  5       !  THE EXPECTED CHECKSUM COMPUTED ABOVE.
;  1911  5       !--
;  1912  5
;  1913  5       PUT_BIT ( CSR, LB, EXT_LOOPBACK );
;  1914  5       DELAY ( 5 );
;  1915  5       TEMP1  = .REG_ADR [ 1, ALL_BITS ];
;  1916  5       TEMP1  = .TEMP1 ^ 8;
;  1917  5       TEMP2 =  .REG_ADR [ 0, ALL_BITS ];
;  1918  5       STATION_ADR [ CHSUM ] = .TEMP1 OR ( .TEMP2 AND %O'000377' );
```

C9


```
;   1919  5       PUT_BIT ( CSR, LB, ZERO );
;   1920  5       IF .CHECKSUM NEQU .STATION_ADR [ CHSUM ]
;   1921  5         THEN
;   1922  6           BEGIN
;   1923  6             PRINTB ( MSG59 );
;   1924  6             PRINTB ( MSG63, .CHECKSUM, .STATION_ADR [ CHSUM ] );
;   1925  6             ERRDF  (0301, MSG00, E1$REPORT);
;   1926  5           END;
;   1927  3     ENDSUB;
;   1928  3
;   1929  3     TEMP3 = ZERO;
;   1930  3     TEMP4 = ZERO;
;   1931  3     INCR INDEX FROM 0 TO 2 DO
;   1932  4       BEGIN
;   1933  4         TEMP3 = .TEMP3 + .STATION_ADR [ .INDEX ];
;   1934  4         IF  STATION_ADR [ .INDEX ] EQLU #X'FFFF'
;   1935  4           THEN
;   1936  4             TEMP4 = .TEMP4 + 1;
;   1937  3       END;
;   1938  3
;   1939  4     IF    ( .TEMP3 EQLU ZERO )
;   1940  4        OR ( .TEMP4 GTRU ZERO )
;   1941  4        OR (( .STATION_ADR [ ZERO ] AND #X'0100' ) EQLU #X'0100' )
;   1942  3           THEN
;   1943  4             BEGIN
;   1944  4               PRINTB ( MSG59 );
;   1945  4               PRINTB ( MSG64 );
;   1946  4               PRINTB ( PHYS_ADR );
;   1947  4               ERRDF  ( 0302, MSG00, E1$REPORT);
;   1948  3           END;
;   1949  3
;   1950  1     ENDTST;


                                        .SBTTL  $T3 TEST 3 - ETHERNET STATION ADDRESS VERIFY TEST
000000  004137  000000G          $T3:   JSR     R1,$SAVE2                       ;                       1822
000004  162706  000006                 SUB     #6,SP
000010  104402                  1$:    TRAP    2                               ;                       1866
000012  004737  000000G                JSR     PC,RESET.DEQNA                  ;                       1869
000016  012746  000023                 MOV     #23,-(SP)                       ;                       1870
000022  004737  000000G                JSR     PC,FORM.HEX.ADR
000026  004737  000000G                JSR     PC,ROMCHK                       ;                       1876
000032  017716  000000G                MOV     @HWP.TABLE,(SP)                 ;                       1905
000036  012746  000000G                MOV     #MSG01,-(SP)
000042  012746  000002                 MOV     #2,-(SP)
000046  010600                        MOV     SP,R0                           ; SP,*
000050  104414                        TRAP    14
000052  012716  000000G                MOV     #PHYS.ADR,(SP)                  ;                       1906
000056  012746  000001                 MOV     #1,-(SP)
000062  010600                        MOV     SP,R0                           ; SP,*
000064  104414                        TRAP    14
000066  013701  000000G                MOV     REG.ADR,R1                      ;                       1913
000072  052761  001400  000016        BIS     #1400,16(R1)
```

```
000100  012702  000005                        MOV     #5,R2                           ; *,$$TMP2                      1914
000104  001410                          2$:    BEQ     5$
000106  013700  000000G                        MOV     L$DLY,R0                        ; *,$$TMP1
000112  001403                                 BEQ     4$
000114  005066  000014                  3$:    CLR     14(SP)                          ; $$TMP
000120  077003                                 SOB     R0,3$                           ; $$TMP1,*
000122  005302                          4$:    DEC     R2                              ; $$TMP2
000124  000767                                 BR      2$
000126  016166  000002  000010          5$:    MOV     2(R1),10(SP)                    ; *,TMP.LOCATION                1915
000134  016600  000010                        MOV     10(SP),R0                       ; TEMP1,*                       1916
000140  072027  000010                        ASH     #10,R0
000144  010037  000000G                        MOV     R0,TEMP1
000150  011166  000012                        MOV     (R1),12(SP)                     ; *,TMP.LOCATION                1917
000154  011137  000000G                        MOV     (R1),TEMP2                      ; TMP.LOCATION,*
000160  005037  000006G                        CLR     STATION.ADR+6                   ;                               1918
000164  111137  000006G                        MOVB    (R1),STATION.ADR+6              ; TEMP2,*
000170  050037  000006G                        BIS     R0,STATION.ADR+6                ; TEMP1,*
000174  042761  001400  000016                BIC     #1400,16(R1)                    ;                               1919
000202  023737  000000G 000006G               CMP     CHECKSUM,STATION.ADR+6          ;                               1920
000210  001426                                 BEQ     6$
000212  012716  000000G                        MOV     #MSG59,(SP)                     ;                               1923
000216  012746  000001                        MOV     #1,-(SP)
000222  010600                                 MOV     SP,R0                           ; SP,*
000224  104414                                 TRAP    14
000226  013716  000006G                        MOV     STATION.ADR+6,(SP)              ;                               1924
000232  013746  000000G                        MOV     CHECKSUM,-(SP)
000236  012746  000000G                        MOV     #MSG63,-(SP)
000242  012746  000003                        MOV     #3,-(SP)
000246  010600                                 MOV     SP,R0                           ; SP,*
000250  104414                                 TRAP    14
000252  104455                                 TRAP    55                              ;                               1925
000254  000455                                 .WORD   455
000256  000000G                                .WORD   MSG00
000260  000000G                                .WORD   E1$REPORT
000262  062706  000010                        ADD     #10,SP                          ;                               1922
000266  062706  000010                  6$:    ADD     #10,SP                          ;                               1866
000272  104467                                 TRAP    67                              ;                               1926
000274  006000                                 ROR     R0
000276  103644                                 BLO     1$
000300  005037  000000G                        CLR     TEMP3                           ;                               1929
000304  005037  000000G                        CLR     TEMP4                           ;                               1930
000310  005000                                 CLR     R0                              ; INDEX                         1931
000312  066037  000000G 000000G       7$:    ADD     STATION.ADR(R0),TEMP3           ; *(INDEX),*                    1933
000320  026027  000000G 177777               CMP     STATION.ADR(R0),#-1             ; *(INDEX),*                    1934
000326  001002                                 BNE     8$
000330  005237  000000G                        INC     TEMP4                           ;                               1936
000334  062700  000002                  8$:    ADD     #2,R0                           ; *,INDEX                       1931
000340  020027  000004                        CMP     R0,#4                           ; INDEX,*
000344  003762                                 BLE     7$
000346  005737  000000G                        TST     TEMP3                           ;                               1939
000352  001407                                 BEQ     9$
000354  005737  000000G                        TST     TEMP4                           ;                               1940
000360  001004                                 BNE     9$
```

E9


```
000362  032737  000400  000000G                 BIT     #400,STATION.ADR                ;                                       1941
000370  001430                                  BEQ     10$
000372  012746  000000G                 9$:     MOV     #MSG59,-(SP)                    ;                                       1944
000376  012746  000001                          MOV     #1,-(SP)
000402  010600                                  MOV     SP,R0                           ; SP,*
000404  104414                                  TRAP    14
000406  012716  000000G                         MOV     #MSG64,(SP)                     ;                                       1945
000412  012746  000001                          MOV     #1,-(SP)
000416  010600                                  MOV     SP,R0                           ; SP,*
000420  104414                                  TRAP    14
000422  012716  000000G                         MOV     #PHYS.ADR,(SP)                  ;                                       1946
000426  012746  000001                          MOV     #1,-(SP)
000432  010600                                  MOV     SP,R0                           ; SP,*
000434  104414                                  TRAP    14
000436  104455                                  TRAP    55                              ;                                       1947
000440  000456                                  .WORD   456
000442  000000G                                 .WORD   MSG00
000444  000000G                                 .WORD   E1$REPORT
000446  062706  000010                          ADD     #10,SP                          ;                                       1943
000452  062706  000006                 10$:     ADD     #6,SP                           ;                                       1822
000456  000207                                  RTS     PC

; Routine Size:  152 words,     Routine Base:  AB$CODE$ + 1304
; Maximum stack depth per invocation:  16 words



                                                .SBTTL  T3 TEST 3 - ETHERNET STATION ADDRESS VERIFY TEST
000000  004737  001304'                 T3::
000000                                  1$:     JSR     PC,$T3                          ;                                       1948
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 1764
; Maximum stack depth per invocation:  2 words


;   1951  1
;   1952  1
```

F9


```
;  1953  1     %SBTTL 'TEST 4   INTERRUPT VECTOR ADDRESS TEST'
;  1954  1     !++
;  1955  1     !
;  1956  1     !  TEST 4:      INTERRUPT VECTOR ADDRESS TEST
;  1957  1     !
;  1958  1     !  DESCRIPTION:
;  1959  1     !
;  1960  1     !       This test verifies that all bits of the vector address register
;  1961  1     !       can be set and cleared as specified.  The host writes data patterns
;  1962  1     !       to this register and reads them back verifying no static
;  1963  1     !       (stuck at 1 / stuck at 0) faults occur. If the operator specifies
;  1964  1     !       loop on error, the program re-executes the code that detected the
;  1965  1     !       error until ^C is entered.
;  1966  1     !
;  1967  1     !       NOTE:  Only bits 9:2 of the Interrupt Vector Address Register are
;  1968  1     !              valid, rest read as 0.
;  1969  1     !
;  1970  1     !       The following BINARY data patterns are used:
;  1971  1     !
;  1972  1     !                       00000000        11111111
;  1973  1     !                       10101010        01010101
;  1974  1     !                       11001100        00110011
;  1975  1     !                       11110000        00001111
;  1976  1     !                       walking 1's, 1 propagating thru Vector Address Reg.
;  1977  1     !                       walking 0's, 0 propagating thru Vector Address Reg.
;  1978  1     !
;  1979  1     !       Hardware tested:        Device Vector Address Register
;  1980  1     !                               Slave Interface Registers
;  1981  1     !
;  1982  1     !       Processing:
;  1983  1     !
;  1984  1     !            BEGIN
;  1985  1     !
;  1986  1     !               reset device
;  1987  1     !               REPEAT for each pattern
;  1988  1     !                  write pattern to Vector Address Register ( bits 9:2 )
;  1989  1     !                  read pattern from Vector Address Register ( bits 9:2 )
;  1990  1     !                  compare write pattern to read pattern (less noise bits)
;  1991  1     !                  IF not equal
;  1992  1     !                  THEN
;  1993  1     !                     print error message if not inhibited
;  1994  1     !                  ENDIF
;  1995  1     !
;  1996  1     !               ENDREPEAT
;  1997  1     !            END
;  1998  1     !--
```

```
;  1999  3     BGNTST;
;  2000  3
;  2001  3     RESET_DEQNA ( );
;  2002  3
;  2003  3     !++
;  2004  3     !  WRITE ALTERNATING 0'S AND 1'S TO INTERRUPT VECTOR ADDRESS REGISTER
;  2005  3     !  IN THE I/O PAGE, THEN READ AND COMPARE TO THE WRITE PATTERN
;  2006  3     !--
;  2007  3
;  2008  3     INCR INDEX FROM 0 TO 7 DO
;  2009  4       BEGIN
;  2010  4         TBYTE1 = .PTRN_TABLE [ .INDEX ];
;  2011  6         BGNSUB;
;  2012  6           PUT_BIT [ INT_VEC, VEC_ADR, .TBYTE1 ];
;  2013  6           IF GET_BIT [ INT_VEC, VEC_ADR ] NEQU .TBYTE1
;  2014  6             THEN
;  2015  7               BEGIN
;  2016  7                 PRINTB ( MSG59 );
;  2017  7                 PRINTB ( MSG65 );
;  2018  7                 PRINTB ( MSG30, .GET_ADR [ VEC_ALL ], GET_BIT [ INT_VEC, VEC_ADR ], .TBYTE1 );
;  2019  7                 ERRDF  ( 0401, MSG00, E1$REPORT );
;  2020  6               END;
;  2021  4         ENDSUB;
;  2022  3       END;
;  2023  3     !++
;  2024  3     !  WRITE WALKING 1 PATTERN INTO THE INTERRUPT VECTOR ADDRESS IN THE I/O PAGE
;  2025  3     !  REGISTER THEN READ AND COMPARE TO THE WRITE PATTERN
;  2026  3     !--
;  2027  3
;  2028  3     TEMP1 = %B'00000001';
;  2029  3
;  2030  3     INCR INDEX FROM 0 TO 7 DO
;  2031  4       BEGIN
;  2032  6         BGNSUB;
;  2033  6         PUT_BIT [ INT_VEC, VEC_ADR, .TEMP1 ];
;  2034  6         IF GET_BIT [ INT_VEC, VEC_ADR ] NEQU .TEMP1
;  2035  6           THEN
;  2036  7             BEGIN
;  2037  7                 PRINTB ( MSG59 );
;  2038  7                 PRINTB ( MSG65 );
;  2039  7                 PRINTB ( MSG30, .GET_ADR [ VEC_ALL ], GET_BIT [ INT_VEC, VEC_ADR ], .TEMP1 );
;  2040  7                 ERRDF  ( 0402, MSG00, E1$REPORT );
;  2041  6             END;
;  2042  6         TEMP1 = .TEMP1 + 1;
;  2043  4         ENDSUB;
;  2044  3       END;
;  2045  3
;  2046  3     !++
;  2047  3     !  WRITE WALKING 0 PATTERN INTO THE INTERRUPT VECTOR ADDRESS IN THE I/O PAGE
;  2048  3     !  REGISTER THEN READ AND COMPARE TO THE WRITE PATTERN
;  2049  3     !--
;  2050  3
;  2051  3     TEMP1 = %B'11111110';
```

H9


```
;    2052  3
;    2053  3     INCR INDEX FROM 0 TO 7 DO
;    2054  4       BEGIN
;    2055  6         BGNSUB;
;    2056  6         PUT_BIT [ INT_VEC, VEC_ADR, .TEMP1 ];
;    2057  6         IF GET_BIT [ INT_VEC, VEC_ADR ] NEQU .TEMP1
;    2058  6           THEN
;    2059  7             BEGIN
;    2060  7                 PRINTB ( MSG59 );
;    2061  7                 PRINTB ( MSG65 );
;    2062  7                 PRINTB ( MSG30, .GET_ADR [ VEC_ALL ], GET_BIT [ INT_VEC, VEC_ADR ], .TEMP1 );
;    2063  7                 ERRDF  ( 0403, MSG00, E1$REPORT );
;    2064  6             END;
;    2065  6
;    2066  6         TEMP1 = (( .TEMP1 + 1 ) + 1 ) AND %O'000377' ;
;    2067  4         ENDSUB;
;    2068  3       END;
;    2069  3
;    2070  1     ENDTST;


                                                .SBTTL  $T4 TEST 4 - INTERRUPT VECTOR ADDRESS TEST
000000  004137  000000G                 $T4:    JSR     R1,$SAVE2                       ;                               1950
000004  162706  000022                          SUB     #22,SP
000010  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                               2001
000014  005001                                  CLR     R1                              ; INDEX                         2008
000016  116137  000000G 000000G         1$:     MOVB    PTRN.TABLE(R1),TBYTE1           ; *(INDEX),*                    2010
000024  105037  000001G                         CLRB    TBYTE1+1
000030  104402                          2$:     TRAP    2
000032  013700  000000G                         MOV     REG.ADR,R0                      ;                               2012
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                2013
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)                    ;                               2016
000116  012746  000001                         MOV     #1,-(SP)
000122  010600                                  MOV     SP,R0                           ; SP,*
000124  104414                                  TRAP    14
000126  012716  000000G                         MOV     #MSG65,(SP)                     ;                               2017
000132  012746  000001                         MOV     #1,-(SP)
000136  010600                                  MOV     SP,R0                           ; SP,*
000140  104414                                  TRAP    14
000142  013716  000000G                         MOV     TBYTE1,(SP)                     ;                               2018
```

```
000146  013700  000000G                        MOV     REG.ADR,R0
000152  016066  000014  000010                MOV     14(R0),10(SP)           ; *,TMP.LOCATION
000160  016600  000010                        MOV     10(SP),R0               ; TMP.LOCATION,*
000164  006200                                ASR     R0
000166  006200                                ASR     R0
000170  042700  177400                        BIC     #177400,R0
000174  010046                                MOV     R0,-(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                      ;                               2019
000234  000621                                .WORD   621
000236  000000G                               .WORD   MSG00
000240  000000G                               .WORD   E1$REPORT
000242  062706  000016                        ADD     #16,SP                  ;                               2015
000246  104467                        3$:     TRAP    67                      ;                               2020
000250  006000                                ROR     R0
000252  103666                                BLO     2$
000254  005201                                INC     R1                      ; INDEX                         2008
000256  020127  000007                        CMP     R1,#7                   ; INDEX,*
000262  003655                                BLE     1$
000264  012737  000001  000000G               MOV     #1,TEMP1                ;                               2028
000272  012701  000010                        MOV     #10,R1                  ; *,INDEX                       2030
000276  104402                        4$:     TRAP    2                       ;                               2031
000300  013700  000000G                       MOV     REG.ADR,R0              ;                               2033
000304  013702  000000G                       MOV     TEMP1,R2
000310  006302                                ASL     R2
000312  006302                                ASL     R2
000314  042702  176003                        BIC     #176003,R2
000320  042760  001774  000014                BIC     #1774,14(R0)
000326  050260  000014                        BIS     R2,14(R0)
000332  016066  000014  000006                MOV     14(R0),6(SP)            ; *,TMP.LOCATION                2034
000340  013702  000000G                       MOV     TEMP1,R2
000344  016600  000006                        MOV     6(SP),R0                ; TMP.LOCATION,*
000350  006200                                ASR     R0
000352  006200                                ASR     R0
000354  042700  177400                        BIC     #177400,R0
000360  020002                                CMP     R0,R2
000362  001456                                BEQ     5$
000364  012746  000000G                       MOV     #MSG59,-(SP)            ;                               2037
000370  012746  000001                        MOV     #1,-(SP)
000374  010600                                MOV     SP,R0                   ; SP,*
000376  104414                                TRAP    14
000400  012716  000000G                       MOV     #MSG65,(SP)             ;                               2038
000404  012746  000001                        MOV     #1,-(SP)
000410  010600                                MOV     SP,R0                   ; SP,*
000412  104414                                TRAP    14
000414  013716  000000G                       MOV     TEMP1,(SP)              ;                               2039
000420  013700  000000G                       MOV     REG.ADR,R0
```

J9


```
000424  016066  000014  000016              MOV     14(R0),16(SP)           ; *,TMP.LOCATION
000432  016600  000016                      MOV     16(SP),R0               ; TMP.LOCATION,*
000436  006200                              ASR     R0
000440  006200                              ASR     R0
000442  042700  177400                      BIC     #177400,R0
000446  010046                              MOV     R0,-(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                      ;                                       2040
000506  000622                              .WORD   622
000510  000000G                             .WORD   MSG00
000512  000000G                             .WORD   E1$REPORT
000514  062706  000016                      ADD     #16,SP                  ;                                       2036
000520  006337  000000G             5$:     ASL     TEMP1                   ;                                       2042
000524  104467                              TRAP    67
000526  006000                              ROR     R0
000530  103662                              BLO     4$
000532  005301                              DEC     R1                      ; INDEX                                 2030
000534  001260                              BNE     4$
000536  012737  000376  000000G             MOV     #376,TEMP1              ;                                       2051
000544  012701  000010                      MOV     #10,R1                  ; *,INDEX                               2053
000550  104402                      6$:     TRAP    2                       ;                                       2054
000552  013700  000000G                     MOV     REG.ADR,R0              ;                                       2056
000556  013702  000000G                     MOV     TEMP1,R2
000562  006302                              ASL     R2
000564  006302                              ASL     R2
000566  042702  176003                      BIC     #176003,R2
000572  042760  001774  000014              BIC     #1774,14(R0)
000600  050260  000014                      BIS     R2,14(R0)
000604  016066  000014  000014              MOV     14(R0),14(SP)           ; *,TMP.LOCATION                        2057
000612  013702  000000G                     MOV     TEMP1,R2
000616  016600  000014                      MOV     14(SP),R0               ; TMP.LOCATION,*
000622  006200                              ASR     R0
000624  006200                              ASR     R0
000626  042700  177400                      BIC     #177400,R0
000632  020002                              CMP     R0,R2
000634  001456                              BEQ     7$
000636  012746  000000G                     MOV     #MSG59,-(SP)            ;                                       2060
000642  012746  000001                      MOV     #1,-(SP)
000646  010600                              MOV     SP,R0                   ; SP,*
000650  104414                              TRAP    14
000652  012716  000000G                     MOV     #MSG65,(SP)             ;                                       2061
000656  012746  000001                      MOV     #1,-(SP)
000662  010600                              MOV     SP,R0                   ; SP,*
000664  104414                              TRAP    14
000666  013716  000000G                     MOV     TEMP1,(SP)              ;                                       2062
000672  013700  000000G                     MOV     REG.ADR,R0
000676  016066  000014  000024              MOV     14(R0),24(SP)           ; *,TMP.LOCATION
```

K9


```
000704  016600  000024                          MOV     24(SP),R0                       ; TMP.LOCATION,*
000710  006200                                  ASR     R0
000712  006200                                  ASR     R0
000714  042700  177400                          BIC     #177400,R0
000720  010046                                  MOV     R0, (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                              ;                                       2063
000760  000623                                  .WORD   623
000762  000000G                                 .WORD   MSG00
000764  000000G                                 .WORD   E1$REPORT
000766  062706  000016                          ADD     #16,SP                          ;                                       2059
000772  013700  000000G                 7$:     MOV     TEMP1,R0                        ;                                       2066
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                                 2053
001022  001252                                  BNE     6$
001024  062706  000022                          ADD     #22,SP                          ;                                       1950
001030  000207                                  RTS     PC

; Routine Size:  269 words,     Routine Base:  AB$CODE$ + 2000
; Maximum stack depth per invocation:  21 words



                                                .SBTTL  T4 TEST 4 - INTERRUPT VECTOR ADDRESS TEST
000000  004737  002000'                  T4::
000000                                  1$:     JSR     PC,$T4                          ;                                       2068
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 3032
; Maximum stack depth per invocation:  2 words


;   2071  1
```

```
;  2072  1     %SBTTL 'TEST 5   BOOT/DIAGNOSTIC PROM CHECKSUM TEST'
;  2073  1     !++
;  2074  1     !
;  2075  1     !  TEST 5:      BOOT/DIAGNOSTIC PROM CHECKSUM TEST
;  2076  1     !
;  2077  1     !  DESCRIPTION:
;  2078  1     !
;  2079  1     !       This test verifies that the contents of the on-board ROM
;  2080  1     !       (Boot/Diagnostic ROM) can be loaded to the host memory correctly.
;  2081  1     !       Checksum is generated from the ROM data read and this checksum is
;  2082  1     !       compared to the checksum stored in the last word location of the
;  2083  1     !       on-board ROM. If the operator specifies loop on error, the program
;  2084  1     !       re-executes the code that detected the  error until ^C is entered.
;  2085  1     !
;  2086  1     !
;  2087  1     !       Hardware tested:        Q-Bus to DMA interface
;  2088  1     !                               I8051 microprocessor
;  2089  1     !                               I8051 ROM
;  2090  1     !                               CSR register
;  2091  1     !                               Receive FIFO
;  2092  1     !       Processing:
;  2093  1     !
;  2094  1     !            BEGIN
;  2095  1     !               reset device
;  2096  1     !               setup Receive Descriptor List(s)
;  2097  1     !               set Boot/Diagnostic ROM and External loopback bits
;  2098  1     !                   This moves ROM boot code into receive FIFO
;  2099  1     !               wait 10 msec. or until RL ( bit 5 in CSR ) = 0
;  2100  1     !               check CSR status ( bit 5 ) and RCV Descriptor List status
;  2101  1     !               IF error
;  2102  1     !               THEN
;  2103  1     !                  print error message if not inhibited
;  2104  1     !               ENDIF
;  2105  1     !               clear Boot/Diagnostic ROM bit in CSR
;  2106  1     !                    This moves contents of FIFO to host memory
;  2107  1     !               wait 10 msec. or until RCV Descriptor status changed
;  2108  1     !               IF change in status
;  2109  1     !               THEN
;  2110  1     !                  print error message if not inhibited
;  2111  1     !               ENDIF
;  2112  1     !               compute ROM checksum and compare to checksum read from ROM
;  2113  1     !               IF not equal
;  2114  1     !               THEN
;  2115  1     !                  print error message if not inhibited
;  2116  1     !               ENDIF
;  2117  1     !            END
;  2118  1     !--
```

M9


```
;    2119  3     BGNTST;
;    2120  3
;    2121  3     RESET_DEQNA ( );
;    2122  3     CLR_BUFFERS ( 2 * K );
;    2123  3
;    2124  3     !++
;    2125  3     !   COPY BOOT/DIAGNOSTIC PROM DESCRIPTOR LIST INTO WORK AREA
;    2126  3     !-
;    2127  3
;    2128  3     INCR INDEX FROM 0 TO BD_D_SIZE - 1 DO
;    2129  3       DESCR_LIST [ .INDEX, W_LEN ] = .BD_PROM_DESCR [ .INDEX ];
;    2130  3
;    2131  3     .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;    2132  3     .IOP_TABLE [ RHI_ADR ] = 0;
;    2133  3
;    2134  3     PUT_BIT ( CSR, LB, EXT_LOOPBACK );
;    2135  3     PUT_BIT ( CSR, BD, SET_IT );
;    2136  3
;    2137  3     DELAY ( K );                                    !give time for rcv list invalid
;    2138  3     INCR INDEX FROM 0 TO TIME3_LIMIT DO             !to set
;    2139  3       IF GET_BIT [ CSR, RL ] EQLU ZERO
;    2140  3          THEN
;    2141  4            BEGIN
;    2142  4              TEMP1 = .INDEX;
;    2143  4              EXITLOOP;
;    2144  4            END
;    2145  3          ELSE
;    2146  3            IF .INDEX EQLU TIME3_LIMIT
;    2147  3               THEN
;    2148  4                 BEGIN
;    2149  4                   PRINTB ( MSG59 );
;    2150  4                   PRINTB ( MSG66, GET_BIT [ CSR_ALL ] );
;    2151  4                   ERRDF ( 0501, MSG00, ERROR$REPORT );
;    2152  3                 END;
;    2153  3
;    2154  3     VER_DESCR_STATUS ( );
;    2155  3
;    2156  3     !++
;    2157  3     !  FINISH BOOT/DIAGNOSTIC PROM UPLOAD
;    2158  3     !--
;    2159  3
;    2160  3     PUT_BIT ( CSR, BD, CLR_IT );
;    2161  3     DELAY ( K );
;    2162  3
;    2163  3     !++
;    2164  3     !  CHECK IF RECEIVE STATUS CHANGED
;    2165  3     !--
;    2166  3
;    2167  3     VER_DESCR_STATUS ( );
;    2168  3
;    2169  3     RESET_DEQNA ( );
;    2170  3
;    2171  3     TEMP3 = 0;
```

```
;    2172  3     TEMP3 = .DATA_BUFFER [ CHSUM_OFFSET + 1 ];
;    2173  3     TEMP3 = ( .TEMP3 ^ 8 ) AND %X'FF00';
;    2174  3     TEMP3 = .DATA_BUFFER [ CHSUM_OFFSET ] + .TEMP3;
;    2175  3
;    2176  3     TEMP2 = .DATA_BUFFER [ .TEMP3 + 1 ];
;    2177  3     TEMP2 = ( .TEMP2 ^ 8 ) AND %X'FF00';
;    2178  3     TEMP2 = .DATA_BUFFER [ .TEMP3 ] + .TEMP2;
;    2179  3
;    2180  3     COUNTER  = 0;
;    2181  3     CHECKSUM = 0;
;    2182  3
;    2183  3     INCR INDEX FROM 0 TO PROM_SIZE - 2 DO
;    2184  3       IF .COUNTER EQLU .TEMP3
;    2185  3         THEN
;    2186  3           COUNTER = .COUNTER + 2
;    2187  3         ELSE
;    2188  4           BEGIN
;    2189  4             CHECKSUM = .CHECKSUM + ( .DATA_BUFFER [ .COUNTER ] AND %X'FF' );
;    2190  4             COUNTER = .COUNTER + 1;
;    2191  3           END;
;    2192  3
;    2193  4     IF ( .TEMP2 EQLU ZERO ) OR ( .TEMP2 NEQU .CHECKSUM )
;    2194  3       THEN
;    2195  4         BEGIN
;    2196  4           CSR_WORD = GET_BIT ( CSR_ALL );
;    2197  4           PRINTB ( MSG59 );
;    2198  4           PRINTB ( MSG67, .TEMP3, .CHECKSUM, .TEMP2 );
;    2199  4           ERRDF  ( 0502, MSG00, E1$REPORT);
;    2200  3         END;
;    2201  3
;    2202  1     ENDTST;


                                        .SBTTL  $T5 TEST 5 - BOOT/DIAGNOSTIC PROM CHECKSUM TEST
000000  004137  000000G          $T5:   JSR     R1,$SAVE3                       ;                                   2070
000004  162706  000010                 SUB     #10,SP
000010  004737  000000G                JSR     PC,RESET.DEQNA                  ;                                   2121
000014  012746  004000                 MOV     #4000,-(SP)                     ;                                   2122
000020  004737  000000G                JSR     PC,CLR.BUFFERS
000024  005000                         CLR     R0                              ; INDEX                             2128
000026  016060  000000G 000000G  1$:    MOV     BD.PROM.DESCR(R0),DESCR.LIST(R0); *(INDEX),*(INDEX)                 2129
000034  062700  000002                 ADD     #2,R0                           ; *,INDEX                           2128
000040  020027  000036                 CMP     R0,#36                          ; INDEX,*
000044  003770                         BLE     1$
000046  012777  000000G 000004G        MOV     #RCV.D.LIST,@IOP.TABLE+4        ;                                   2131
000054  005077  000006G                CLR     @IOP.TABLE+6                    ;                                   2132
000060  013700  000000G                MOV     REG.ADR,R0                      ;                                   2134
000064  052760  001410  000016         BIS     #1410,16(R0)                    ;                                   2135
000072  012701  002000                 MOV     #2000,R1                        ; *,$$TMP2                          2137
000076  001410                  2$:    BEQ     5$
000100  013700  000000G                MOV     L$DLY,R0                        ; *,$$TMP1
000104  001403                         BEQ     4$
000106  005066  000010           3$:    CLR     10(SP)                          ; $$TMP
```

B10


```
000112  077003                          SOB     R0,3$                   ; $$TMP1,*
000114  005301                  4$:     DEC     R1                      ; $$TMP2
000116  000767                          BR      2$
000120  005001                  5$:     CLR     R1                      ; INDEX                         2138
000122  013700  000000G         6$:     MOV     REG.ADR,R0              ;                               2139
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,*                       2142
000150  000440                          BR      9$                      ;                               2141
000152  020127  002000          7$:     CMP     R1,#2000                ; INDEX,*                       2146
000156  001031                          BNE     8$
000160  012716  000000G                 MOV     #MSG59,(SP)             ;                               2149
000164  012746  000001                  MOV     #1,-(SP)
000170  010600                          MOV     SP,R0                   ; SP,*
000172  104414                          TRAP    14
000174  013700  000000G                 MOV     REG.ADR,R0              ;                               2150
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                      ;                               2151
000230  000765                          .WORD   765
000232  000000G                         .WORD   MSG00
000234  000000G                         .WORD   ERROR$REPORT
000236  062706  000006                  ADD     #6,SP                   ;                               2148
000242  005201                  8$:     INC     R1                      ; INDEX                         2138
000244  020127  002000                  CMP     R1,#2000                ; INDEX,*
000250  003724                          BLE     6$
000252  004737  000000G         9$:     JSR     PC,VER.DESCR.STATUS     ;                               2154
000256  013700  000000G                 MOV     REG.ADR,R0              ;                               2160
000262  142760  000010  000016          BICB    #10,16(R0)
000270  012701  002000                  MOV     #2000,R1                ; *,$$TMP2                      2161
000274  001410                  10$:    BEQ     13$
000276  013700  000000G                 MOV     L$DLY,R0                ; *,$$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     ;                               2167
000322  004737  000000G                 JSR     PC,RESET.DEQNA          ;                               2169
000326  005037  000000G                 CLR     TEMP3                   ;                               2172
000332  113737  000007G 000000G         MOVB    DATA.BUFFER+7,TEMP3
000340  013700  000000G                 MOV     TEMP3,R0                ;                               2173
000344  072027  000010                  ASH     #10,R0
000350  010037  000000G                 MOV     R0,TEMP3
000354  042737  000377  000000G         BIC     #377,TEMP3
000362  005000                          CLR     R0                      ;                               2174
000364  153700  000006G                 BISB    DATA.BUFFER+6,R0
000370  060037  000000G                 ADD     R0,TEMP3
```

C10


```
000374  013701  000000G                        MOV     TEMP3,R1                ;                                       2176
000400  116137  000001G 000000G                MOVB    DATA.BUFFER+1(R1),TEMP2
000406  105037  000001G                        CLRB    TEMP2+1
000412  013700  000000G                        MOV     TEMP2,R0                ;                                       2177
000416  072027  000010                         ASH     #10,R0
000422  010037  000000G                        MOV     R0,TEMP2
000426  042737  000377  000000G                BIC     #377,TEMP2
000434  005000                                 CLR     R0                      ;                                       2178
000436  156100  000000G                        BISB    DATA.BUFFER(R1),R0
000442  060037  000000G                        ADD     R0,TEMP2
000446  005037  000000G                        CLR     COUNTER                 ;                                       2180
000452  005037  000000G                        CLR     CHECKSUM                ;                                       2181
000456  012702  007777                         MOV     #7777,R2                ; *,INDEX                               2183
000462  013700  000000G             14$:       MOV     COUNTER,R0              ;                                       2184
000466  020001                                 CMP     R0,R1
000470  001004                                 BNE     15$
000472  062737  000002  000000G                ADD     #2,COUNTER              ;                                       2186
000500  000407                                 BR      16$                     ;                                       2184
000502  005003                         15$:    CLR     R3                      ;                                       2189
000504  156003  000000G                        BISB    DATA.BUFFER(R0),R3
000510  060337  000000G                        ADD     R3,CHECKSUM
000514  005237  000000G                        INC     COUNTER                 ;                                       2190
000520  077220                         16$:    SOB     R2,14$                  ; INDEX,*                               2183
000522  013700  000000G                        MOV     TEMP2,R0                ;                                       2193
000526  001403                                 BEQ     17$
000530  020037  000000G                        CMP     R0,CHECKSUM
000534  001440                                 BEQ     18$
000536  013700  000000G             17$:       MOV     REG.ADR,R0              ;                                       2196
000542  016066  000016  000006                MOV     16(R0),6(SP)            ; *,TMP.LOCATION
000550  016637  000006  000000G                MOV     6(SP),CSR.WORD          ; TMP.LOCATION,*
000556  012716  000000G                        MOV     #MSG59,(SP)             ;                                       2197
000562  012746  000001                         MOV     #1,-(SP)
000566  010600                                 MOV     SP,R0                   ; SP,*
000570  104414                                 TRAP    14
000572  013716  000000G                        MOV     TEMP2,(SP)              ;                                       2198
000576  013746  000000G                        MOV     CHECKSUM,-(SP)
000602  013746  000000G                        MOV     TEMP3,-(SP)
000606  012746  000000G                        MOV     #MSG67,-(SP)
000612  012746  000004                         MOV     #4,-(SP)
000616  010600                                 MOV     SP,R0                   ; SP,*
000620  104414                                 TRAP    14
000622  104455                                 TRAP    55                      ;                                       2199
000624  000766                                 .WORD   766
000626  000000G                                .WORD   MSG00
000630  000000G                                .WORD   E1$REPORT
000632  062706  000012                         ADD     #12,SP                  ;                                       2195
000636  062706  000012             18$:       ADD     #12,SP                  ;                                       2070
000642  000207                                 RTS     PC

; Routine Size:  210 words,     Routine Base:  AB$CODE$ + 3046
; Maximum stack depth per invocation:  16 words
```

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

000000  004737  003046'                T5::
000000                                  1$:     JSR     PC,$T5                          ;                       2200
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 3712
; Maximum stack depth per invocation:  2 words


;   2203  1
```

E10


```
;    2204  1     %SBTTL 'TEST 6   INTERRUPT SANITY TEST'
;    2205  1     !++
;    2206  1     !
;    2207  1     !  TEST 6:      INTERRUPT SANITY TEST
;    2208  1     !
;    2209  1     !  DESCRIPTION:
;    2210  1     !
;    2211  1     !       This test verifies that DEQNA interrupts the processor only at
;    2212  1     !       the expected level ( 4 ) and not any other level. If the operator
;    2213  1     !       specifies loop on error, the program re-executes the code that
;    2214  1     !       detected the error until ^C is entered.
;    2215  1     !
;    2216  1     !       Hardware tested:        Q-Bus to QTDC interface
;    2217  1     !                               CSR register
;    2218  1     !                               Q-Bus timeout logic
;    2219  1     !                               QTDC interrupt logic
;    2220  1     !       Processing:
;    2221  1     !
;    2222  1     !            BEGIN
;    2223  1     !               reset device
;    2224  1     !               set-up for TX Done interrupt
;    2225  1     !               REPEAT for each processor priority level
;    2226  1     !                  enable device interrupt (set CSR bit 6)
;    2227  1     !                  cause TX Done interrupt
;    2228  1     !                  check for expected CSR status
;    2229  1     !                  IF error
;    2230  1     !                  THEN
;    2231  1     !                     print error message if not inhibited
;    2232  1     !                  ENDIF
;    2233  1     !               ENDREPEAT
;    2234  1     !            END
;    2235  1     !--
```

```
;  2236  1
;  2237  3     BGNTST;
;  2238  3
;  2239  3     RESET_DEQNA ( );
;  2240  3     SETVEC ( .HWP_TABLE [ VEC ], QNA_INT, PRI07 );  ! SET UP FOR a tx done INTERRUPT
;  2241  3     .IOP_TABLE [ INT_VEC ] = .HWP_TABLE [ VEC ];
;  2242  3     TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;  2243  3     COUNTER = 0;
;  2244  3
;  2245  3     INCR PRIORITY FROM PRI00 TO PRI07 BY %0'40' DO
;  2246  4       BEGIN
;  2247  4         SETPRI ( .PRIORITY );                       ! SET PROCESSOR PRI LEVEL
;  2248  6         BGNSUB;
;  2249  6           PUT_BIT ( CSR, IE, SET_IT );              ! ENABLE INTERRUPTS
;  2250  6           DELAY ( 5 );                              !
;  2251  6           INTERRUPT_FLG = CLEAR_FLG;
;  2252  6
;  2253  6               INTR_TEST_PACKET ( );                 ! this should cause xmit intr
;  2254  6               DELAY ( 400 );
;  2255  6
;  2256  6           GETPRI ( TEMP1 );
;  2257  6           TEMP1 = .TEMP1 + ( - 5 );
;  2258  6
;  2259  6           IF .INTERRUPT_FLG EQLU WORD_LIMIT
;  2260  6             THEN                                    ! INTERRUPT SHOULD NOT OCCUR
;  2261  6               IF .PRIORITY GTRU PRI03
;  2262  6                 THEN
;  2263  7                   BEGIN
;  2264  7                     PRINTB ( MSG59 );
;  2265  7                     PRINTB ( msg73, .TEMP1 );         !report unexpected interrupt
;  2266  7                     ERRDF  ( 0601, MSG00, E1$REPORT );
;  2267  6                   END;
;  2268  6
;  2269  6           IF .INTERRUPT_FLG EQLU ZERO
;  2270  6             THEN                                    ! INTERRUPT SHOULD OCCUR
;  2271  6               IF .PRIORITY LEQU PRI03
;  2272  6                     THEN
;  2273  7                       BEGIN
;  2274  7                         PRINTB ( MSG59 );
;  2275  7                         PRINTB ( msg72, .TEMP1 );          !report device failed to
;  2276  7                         ERRDF  ( 0602, MSG00, ERROR$REPORT ); !interrupt
;  2277  6                       END;
;  2278  6           RESET_DEQNA ( );
;  2279  4         ENDSUB;
;  2280  4         COUNTER = .COUNTER + 1;
;  2281  3       END;
;  2282  3
;  2283  3     SETPRI ( PRI03 );                               ! SET PROCESSOR PRI LEVEL
;  2284  3
;  2285  1     ENDTST;

                                        .SBTTL  $T6 TEST 6 - INTERRUPT SANITY TEST
```

G10


```
000000  004137  000000G                 $T6:    JSR     R1,$SAVE2                       ;                               2202
000004  005746                                  TST     -(SP)
000006  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                               2239
000012  012746  000000G                         MOV     #PRI07,-(SP)                    ;                               2240
000016  012746  000000G                         MOV     #QNA.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                    ;                               2241
000044  016077  000002  000014G                 MOV     2(R0),@IOP.TABLE+14
000052  017737  000000G 000000G                 MOV     @HWP.TABLE,TMP.IOP.ADR          ;                               2242
000060  005037  000000G                         CLR     COUNTER                         ;                               2243
000064  012702  000000G                         MOV     #PRI00,R2                       ; *,PRIORITY                    2245
000070  000545                                  BR      13$
000072  010200                          1$:     MOV     R2,R0                           ; PRIORITY,*                    2247
000074  104441                                  TRAP    41
000076  104402                          2$:     TRAP    2
000100  013700  000000G                         MOV     REG.ADR,R0                      ;                               2249
000104  152760  000100  000016                 BISB    #100,16(R0)
000112  012701  000005                          MOV     #5,R1                           ; *,$$TMP2                      2250
000116  001410                          3$:     BEQ     6$
000120  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000124  001403                                  BEQ     5$
000126  005066  000010                  4$:     CLR     10(SP)                          ; $$TMP
000132  077003                                  SOB     R0,4$                           ; $$TMP1,*
000134  005301                          5$:     DEC     R1                              ; $$TMP2
000136  000767                                  BR      3$
000140  005037  000000G                 6$:     CLR     INTERRUPT.FLG                   ;                               2251
000144  004737  000000G                         JSR     PC,INTR.TEST.PACKET             ;                               2253
000150  012701  000620                          MOV     #620,R1                         ; *,$$TMP2                      2254
000154  001410                          7$:     BEQ     10$
000156  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000162  001403                                  BEQ     9$
000164  005066  000010                  8$:     CLR     10(SP)                          ; $$TMP
000170  077003                                  SOB     R0,8$                           ; $$TMP1,*
000172  005301                          9$:     DEC     R1                              ; $$TMP2
000174  000767                                  BR      7$
000176  104440                          10$:    TRAP    40                              ;                               2256
000200  072027  177773                          ASH     #-5,R0                          ;                               2257
000204  010037  000000G                         MOV     R0,TEMP1
000210  023727  000000G 177777                 CMP     INTERRUPT.FLG,#-1               ;                               2259
000216  001027                                  BNE     11$
000220  020227  000000G                         CMP     R2,#PRI03                       ; PRIORITY,*                    2261
000224  101424                                  BLOS    11$
000226  012716  000000G                         MOV     #MSG59,(SP)                     ;                               2264
000232  012746  000001                          MOV     #1,-(SP)
000236  010600                                  MOV     SP,R0                           ; SP,*
000240  104414                                  TRAP    14
000242  013716  000000G                         MOV     TEMP1,(SP)                      ;                               2265
000246  012746  000000G                         MOV     #MSG73,-(SP)
000252  012746  000002                          MOV     #2,-(SP)
000256  010600                                  MOV     SP,R0                           ; SP,*
```

```
000260  104414                                  TRAP    14
000262  104455                                  TRAP    55                      ;                                       2266
000264  001131                                  .WORD   1131
000266  000000G                                 .WORD   MSG00
000270  000000G                                 .WORD   E1$REPORT
000272  062706  000006                          ADD     #6,SP                   ;                                       2263
000276  005737  000000G               11$:      TST     INTERRUPT.FLG           ;                                       2269
000302  001027                                  BNE     12$
000304  020227  000000G                         CMP     R2,#PRI03               ; PRIORITY,*                            2271
000310  101024                                  BHI     12$
000312  012716  000000G                         MOV     #MSG59,(SP)             ;                                       2274
000316  012746  000001                          MOV     #1,-(SP)
000322  010600                                  MOV     SP,R0                   ; SP,*
000324  104414                                  TRAP    14
000326  013716  000000G                         MOV     TEMP1,(SP)              ;                                       2275
000332  012746  000000G                         MOV     #MSG72,-(SP)
000336  012746  000002                          MOV     #2,-(SP)
000342  010600                                  MOV     SP,R0                   ; SP,*
000344  104414                                  TRAP    14
000346  104455                                  TRAP    55                      ;                                       2276
000350  001132                                  .WORD   1132
000352  000000G                                 .WORD   MSG00
000354  000000G                                 .WORD   ERROR$REPORT
000356  062706  000006                          ADD     #6,SP                   ;                                       2273
000362  004737  000000G               12$:      JSR     PC,RESET.DEQNA          ;                                       2278
000366  104467                                  TRAP    67
000370  006000                                  ROR     R0
000372  103641                                  BLO     2$
000374  005237  000000G                         INC     COUNTER                 ;                                       2280
000400  062702  000040                          ADD     #40,R2                  ; *,PRIORITY                            2245
000404  020227  000000G               13$:      CMP     R2,#PRI07               ; PRIORITY,*
000410  003630                                  BLE     1$
000412  012700  000000G                         MOV     #PRI03,R0               ;                                       2283
000416  104441                                  TRAP    41
000420  062706  000012                          ADD     #12,SP                  ;                                       2202
000424  000207                                  RTS     PC

; Routine Size:  139 words,     Routine Base:  AB$CODE$ + 3726
; Maximum stack depth per invocation:  13 words
```

```
                                        .SBTTL  T6 TEST 6 - INTERRUPT SANITY TEST

000000  004737  003726'                T6::
000000                                  1$:     JSR     PC,$T6                          ;                               2283
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 4354
; Maximum stack depth per invocation:  2 words


;   2286  1
```

J10


```
;  2287  1     %SBTTL 'TEST 7 - ETHERNET CARRIER SENSE TEST'
;  2288  1     !++
;  2289  1     !
;  2290  1     !  TEST 7:      ETHERNET CARRIER SENSE TEST
;  2291  1     !
;  2292  1     !  DESCRIPTION:
;  2293  1     !
;  2294  1     !       This test verifies that the DEQNA can transmit external loopback
;  2295  1     !       packets and if not faulty FRU is can be found by executing this
;  2296  1     !       by implementing the instructions printed on the operator's console.
;  2297  1     !
;  2298  1     !       In order to run this test successfully the operator has to make
;  2299  1     !       sure that DEQNA is connected to the transceiver. If the operator
;  2300  1     !       specifies loop on error, the program re-executes the code that detected
;  2301  1     !       the error until ^C is entered.
;  2302  1     !
;  2303  1     !       Hardware tested:        Carrier Sense circuitry
;  2304  1     !                               Encode/Decode ( ED ) chip
;  2305  1     !
;  2306  1     !       Processing:
;  2307  1     !
;  2308  1     !            BEGIN
;  2309  1     !               reset device
;  2310  1     !               select external loopback mode
;  2311  1     !               check external hardware
;  2312  1     !               IF bad hardware
;  2313  1     !               THEN
;  2314  1     !                  print error message if not inhibited
;  2315  1     !               ENDIF
;  2316  1     !               read CSR
;  2317  1     !               IF Ethernet Carrier Sense bit ( bit 13 ) = 1
;  2318  1     !               THEN
;  2319  1     !                  print error message if not inhibited
;  2320  1     !               ENDIF
;  2321  1     !               transmit longest unchained loopback packet ( ETHERNET format )
;  2322  1     !               read CSR while transmitting loopback packet
;  2323  1     !               IF Ethernet Carrier Sense bit (bit 13) = 0
;  2324  1     !               THEN
;  2325  1     !                  print error message if not inhibited
;  2326  1     !               ELSE
;  2327  1     !                  wait until Carrer Sense bit goes to 0
;  2328  1     !               ENDIF
;  2329  1     !               read CSR
;  2330  1     !               IF Ethernet Carrier Sense bit (bit 13) = 1
;  2331  1     !               THEN
;  2332  1     !                  print error message if not inhibited
;  2333  1     !               ENDIF
;  2334  1     !            END
;  2335  1     !--
```

```
;  2336  3     BGNTST;
;  2337  3
;  2338  3     IF .SWP_ILOOP
;  2339  3       THEN
;  2340  4         BEGIN
;  2341  4           RESET_DEQNA ( );
;  2342  5           IF ( NOT GET_BIT [ CSR, XC ] ) AND ( .SWP_LBC EQLU ZERO )
;  2343  4             THEN
;  2344  5               BEGIN
;  2345  5                           PRINTB ( MSG59 );
;  2346  5                           PRINTB ( MSG47 );
;  2347  5                           ERRDF  ( 0701, MSG00, E1$REPORT );
;  2348  5                 EXIT_TST;
;  2349  4               END;
;  2350  4
;  2351  4           !++
;  2352  4           !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM IF EXECUTING
;  2353  4           !  TESTS IN EXTERNAL LOOPBACK MODE.
;  2354  4           !--
;  2355  4
;  2356  4           RESET_DEQNA ( );
;  2357  4           PREP_FOR_SETUP ( );
;  2358  4           INCR INDEX1 FROM 1 TO 14 DO
;  2359  4             WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  2360  4
;  2361  6           BGNSUB;
;  2362  6             XMIT_SETUP_PACKET ( N_MODE );
;  2363  4           ENDSUB;
;  2364  4
;  2365  4           ERR_FLAG = ZERO;
;  2366  4           INCR INDEX2 FROM 0 TO 19 DO
;  2367  5             BEGIN
;  2368  5               SEND_TEST_PACKET ( );
;  2369  5               DELAY ( 200 );                          !changed from 100, failed heavy
;  2370  5               CSR_WORD = GET_BIT ( CSR_ALL );         !network traffic
;  2371  5               IF ( .CSR_WORD AND %O'100220' ) NEQU %O'100220' !RI,XI,XL bits in csr0
;  2372  5                THEN
;  2373  6                 BEGIN
;  2374  6                    PRINTB ( MSG59 );
;  2375  6                    PRINTB ( MSG74 );
;  2376  6                    PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .CSR_WORD, %o'100220' );
;  2377  6                    ERRDF  ( 0702, MSG00, ERROR$REPORT );
;  2378  6                 EXIT_TST;
;  2379  5                 end;
;  2380  4             END;
;  2381  4
;  2382  4             XC_FLAG   = ZERO;
;  2383  4             ERR_COUNT = ZERO;
;  2384  4
;  2385  6           BGNSUB;
;  2386  6             INCR INDEX2 FROM 0 TO TIME1_LIMIT DO    !if wire errors, retry 128 times
;  2387  7               BEGIN
;  2388  7                 RESET_DEQNA ( );
```

```
;  2389  7                TEMP5 = .INDEX2;
;  2390  7
;  2391  7                !++
;  2392  7                !  CHECK ETHERNET CARRIER SENSE BIT ( CA - BIT 13 ) IN THE CSR. CA SHOULD BE
;  2393  7                !  SET TO '1' WHILE THE DEQNA IS TRANSMITTING. IF CA ISN'T SET TO '1' WITHIN
;  2394  7                !  THE EXPECTED TIME LIMIT, ERROR MESSAGE IS PRINTED OUT.
;  2395  7                ! -
;  2396  7
;  2397  7                SEND_TEST_PACKET ( );
;  2398  7
;  2399  7                INCR INDEX FROM 0 TO TIME1_LIMIT DO
;  2400  7                  IF GET_BIT [ CSR, CA ] EQLU ONE
;  2401  7                    THEN
;  2402  8                      BEGIN
;  2403  8                        TEMP2 = GET_BIT [ CSR_ALL ];
;  2404  8                        EXITLOOP;
;  2405  8                      END
;  2406  7                    ELSE
;  2407  7                      IF .INDEX EQLU TIME1_LIMIT
;  2408  7                        THEN
;  2409  8                          BEGIN
;  2410  8                            PRINTB ( MSG59 );
;  2411  8                            PRINTB ( MSG19, GET_BIT [ CSR_ALL ] );
;  2412  8                            ERRDF ( 0703, MSG00, ERROR$REPORT );
;  2413  7                          END;
;  2414  7
;  2415  7                !++
;  2416  7                !  NOW CHECK IF THE CA BIT RESETS TO '0' WHEN THE DEQNA COMPLETES TRANSMITTING
;  2417  7                !  LOOPBACK PACKET. PRINT ERROR MESSAGE IF LOOPBACK PACKET TRANSMISSION
;  2418  7                !  EXCEEDS SELECTED TIME LIMIT.
;  2419  7                ! -
;  2420  7
;  2421  7                INCR INDEX FROM 0 TO TIME2_LIMIT DO
;  2422  7                  IF GET_BIT [ CSR, CA ] EQLU ZERO
;  2423  7                    THEN
;  2424  8                      BEGIN
;  2425  8                        TEMP3 = GET_BIT [ CSR_ALL ];
;  2426  8                        EXITLOOP;
;  2427  8                      END
;  2428  7                    ELSE
;  2429  7                      IF .INDEX EQLU TIME2_LIMIT
;  2430  7                        THEN
;  2431  8                          BEGIN
;  2432  8                            PRINTB ( MSG59 );
;  2433  8                            PRINTB ( MSG20, GET_BIT [ CSR_ALL ] );
;  2434  8                            ERRDF ( 0704, MSG00, ERROR$REPORT );
;  2435  7                          END;
;  2436  7
;  2437  7                !++
;  2438  7                !  CHECK RECEIVE INTERRUPT REQUEST BIT ( RI - BIT 15 ) TO VERIFY THAT DEQNA
;  2439  7                !  ACTUALLY TRANSMITTED LOOPBACK PACKET
;  2440  7                !--
;  2441  7
```

```
;   2442  7                 DELAY ( 50 );
;   2443  7
;   2444  7                 IF GET_BIT [ CSR, RI ] EQLU ONE
;   2445  7                   THEN
;   2446  8                     BEGIN
;   2447  8                       TEMP4 = GET_BIT [ CSR_ALL ];
;   2448  8                       EXITLOOP;
;   2449  7                     END;
;   2450  6               END;
;   2451  6
;   2452  6               IF .TEMP5 EQLU TIME1_LIMIT
;   2453  6                 THEN
;   2454  7                   BEGIN
;   2455  7                     PRINTB ( MSG59 );
;   2456  7                     PRINTB ( MSG21, GET_BIT [ CSR_ALL ] );
;   2457  7                     ERRDF  ( 0705, MSG00, ERROR$REPORT );
;   2458  6                   END;
;   2459  6
;   2460  6
;   2461  7               IF ( .XMIT_D_LIST [ ERRSU ] EQLU 1 ) AND ( .XMIT_D_LIST [ ABORT ] EQLU 1 )
;   2462  6                 THEN
;   2463  7                   BEGIN
;   2464  7                     PRINTB ( MSG59 );
;   2465  7                     PRINTB ( MSG71 );
;   2466  7                     ERRDF  ( 0706, MSG00, ERROR$REPORT );
;   2467  6                   END;
;   2468  6
;   2469  6               !++
;   2470  6               !  COMPARE STATUS REGISTERS TO EXPECTED VALUES
;   2471  6               !--
;   2472  6
;   2473  6                                                          !177377 masks our FAIL
;   2474  6               CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK      );  ! 0'100220', 0'100220'
;   2475  6               XMIT_D_LIST [ STWD1 ] = .XMIT_D_LIST [ STWD1 ] AND %O'177377';
;   2476  6               CHK_XMIT_STATUS ( XFLG_STATUS, XWD11_STATUS );  ! 0'140000', 0'000000'
;   2477  6               CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS  );  ! 0'140000', 0'020000'
;   2478  6
;   2479  6               IF .XMIT_D_LIST [ TDR ] EQLU ZERO
;   2480  6                 THEN
;   2481  7                   BEGIN
;   2482  7                     PRINTB ( MSG59 );
;   2483  7                     PRINTB ( MSG58 );
;   2484  7                     ERRDF  ( 0707, MSG00, ERROR$REPORT );
;   2485  6                   END;
;   2486  6
;   2487  4           ENDSUB;
;   2488  3         END;
;   2489  1     ENDTST;


                                        .SBTTL  $T7 TEST 7 - ETHERNET CARRIER SENSE TEST
000000  004137  000000G          $T7:   JSR     R1,$SAVE2                        ;                2285
000004  162706  000032                 SUB     #32,SP
```

N10


```
000010  032737  000001  000000G                 BIT     #1,SWP.ILOOP                    ;                               2338
000016  001576                                  BEQ     9$
000020  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                               2341
000024  013700  000000G                         MOV     REG.ADR,R0                      ;                               2342
000030  016016  000016                          MOV     16(R0),(SP)                     ; *,TMP.LOCATION
000034  032716  010000                          BIT     #10000,(SP)                     ; *,TMP.LOCATION
000040  001027                                  BNE     1$
000042  005737  000000G                         TST     SWP.LBC
000046  001024                                  BNE     1$
000050  012746  000000G                         MOV     #MSG59,-(SP)                    ;                               2345
000054  012746  000001                          MOV     #1,-(SP)
000060  010600                                  MOV     SP,R0                           ; SP,*
000062  104414                                  TRAP    14
000064  012716  000000G                         MOV     #MSG47,(SP)                     ;                               2346
000070  012746  000001                          MOV     #1,-(SP)
000074  010600                                  MOV     SP,R0                           ; SP,*
000076  104414                                  TRAP    14
000100  104455                                  TRAP    55                              ;                               2347
000102  001275                                  .WORD   1275
000104  000000G                                 .WORD   MSG00
000106  000000C                                 .WORD   E1$REPORT
000110  104463                                  TRAP    63
000112  062706  000006                          ADD     #6,SP
000116  000536                                  BR      9$                              ;                               2344
000120  004737  000000G                 1$:     JSR     PC,RESET.DEQNA                  ;                               2356
000124  004737  000000G                         JSR     PC,PREP.FOR.SETUP               ;                               2357
000130  012701  000001                          MOV     #1,R1                           ; *,INDEX1                      2358
000134  010146                          2$:     MOV     R1,-(SP)                        ; INDEX1,*                      2359
000136  012746  000023                          MOV     #23,-(SP)
000142  004737  000000G                         JSR     PC,WRT.STATION.ADR
000146  022626                                  CMP     (SP)+,(SP)+
000150  005201                                  INC     R1                              ; INDEX1                        2358
000152  020127  000016                          CMP     R1,#16                          ; INDEX1,*
000156  003766                                  BLE     2$
000160  104402                          3$:     TRAP    2                               ;                               2359
000162  012746  000200                          MOV     #200,-(SP)                      ;                               2362
000166  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000172  005726                                  TST     (SP)+                           ;                               2359
000174  104467                                  TRAP    67                              ;                               2362
000176  006000                                  ROR     R0
000200  103767                                  BLO     3$
000202  005037  000000G                         CLR     ERR.FLAG                        ;                               2365
000206  012702  000024                          MOV     #24,R2                          ; *,INDEX2                      2366
000212  004737  000000G                 4$:     JSR     PC,SEND.TEST.PACKET             ;                               2368
000216  012701  000310                          MOV     #310,R1                         ; *,$$TMP2                      2369
000222  001410                          5$:     BEQ     8$
000224  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000230  001403                                  BEQ     7$
000232  005066  000030                  6$:     CLR     30(SP)                          ; $$TMP
000236  077003                                  SOB     R0,6$                           ; $$TMP1,*
000240  005301                          7$:     DEC     R1                              ; $$TMP2
000242  000767                                  BR      5$
000244  013700  000000G                 8$:     MOV     REG.ADR,R0                      ;                               2370
```

B11


```
000250  016066  000016  000002                MOV     16(R0),2(SP)                    ; *,TMP.LOCATION
000256  016637  000002  000000G               MOV     2(SP),CSR.WORD                  ; TMP.LOCATION,*
000264  016600  000002                        MOV     2(SP),R0                        ; CSR.WORD,*                    2371
000270  042700  077557                        BIC     #77557,R0
000274  020027  100220                        CMP     R0,#-77560
000300  001447                                BEQ     10$
000302  012746  000000G                       MOV     #MSG59,-(SP)                    ;                               2374
000306  012746  000001                        MOV     #1,-(SP)
000312  010600                                MOV     SP,R0                           ; SP,*
000314  104414                                TRAP    14
000316  012716  000000G                       MOV     #MSG74,(SP)                     ;                               2375
000322  012746  000001                        MOV     #1,-(SP)
000326  010600                                MOV     SP,R0                           ; SP,*
000330  104414                                TRAP    14
000332  012716  100220                        MOV     #-77560,(SP)                    ;                               2376
000336  013746  000000G                       MOV     CSR.WORD,-(SP)
000342  013766  000000G 000014                MOV     GET.ADR,14(SP)                  ; *,TMP.LOCATION
000350  062766  000016  000014                ADD     #16,14(SP)                      ; *,TMP.LOCATION
000356  016646  000014                        MOV     14(SP),-(SP)                    ; TMP.LOCATION,*
000362  012746  000000G                       MOV     #MSG30,-(SP)
000366  012746  000004                        MOV     #4,-(SP)
000372  010600                                MOV     SP,R0                           ; SP,*
000374  104414                                TRAP    14
000376  104455                                TRAP    55                              ;                               2377
000400  001276                                .WORD   1276
000402  000000G                               .WORD   MSG00
000404  000000G                               .WORD   ERROR$REPORT
000406  104463                                TRAP    63
000410  062706  000016                        ADD     #16,SP
000414  000137  005760'              9$:     JMP     30$                             ;                               2373
000420  005302                      10$:     DEC     R2                              ; INDEX2                        2366
000422  001273                              BNE     4$
000424  005037  000000G                       CLR     XC.FLAG                         ;                               2382
000430  005037  000000G                       CLR     ERR.COUNT                       ;                               2383
000434  104402                      11$:     TRAP    2
000436  005002                              CLR     R2                              ; INDEX2                        2386
000440  004737  000000G             12$:     JSR     PC,RESET.DEQNA                  ;                               2388
000444  010237  000000G                       MOV     R2,TEMP5                        ; INDEX2,*                      2389
000450  004737  000000G                       JSR     PC,SEND.TEST.PACKET             ;                               2397
000454  005001                              CLR     R1                              ; INDEX                         2399
000456  013700  000000G             13$:     MOV     REG.ADR,R0                      ;                               2400
000462  016066  000016  000006              MOV     16(R0),6(SP)                    ; *,TMP.LOCATION
000470  032766  020000  000006              BIT     #20000,6(SP)                    ; *,TMP.LOCATION
000476  001407                              BEQ     14$
000500  016666  000006  000010              MOV     6(SP),10(SP)                    ; *,TMP.LOCATION                2403
000506  016637  000010  000000G             MOV     10(SP),TEMP2                    ; TMP.LOCATION,*
000514  000440                              BR      16$                             ;                               2402
000516  020127  000200             14$:     CMP     R1,#200                         ; INDEX,*                       2407
000522  001031                              BNE     15$
000524  012746  000000G                       MOV     #MSG59,-(SP)                    ;                               2410
000530  012746  000001                        MOV     #1,-(SP)
000534  010600                                MOV     SP,R0                           ; SP,*
000536  104414                                TRAP    14
```

C 1


```
000540  013700  000000G                        MOV     REG.ADR,R0                      ;                               2411
000544  016066  000016  000016                 MOV     16(R0),16(SP)                   ; *,TMP.LOCATION
000552  016616  000016                         MOV     16(SP),(SP)                     ; TMP.LOCATION,*
000556  012746  000000G                        MOV     #MSG19,-(SP)
000562  012746  000002                         MOV     #2,-(SP)
000566  010600                                 MOV     SP,R0                           ; SP,*
000570  104414                                 TRAP    14
000572  104455                                 TRAP    55                              ;                               2412
000574  001277                                 .WORD   1277
000576  000000G                                .WORD   MSG00
000600  000000G                                .WORD   ERROR$REPORT
000602  062706  000010                         ADD     #10,SP                          ;                               2409
000606  005201                         15$:    INC     R1                              ; INDEX                         2399
000610  020127  000200                         CMP     R1,#200                         ; INDEX,*
000614  003720                                 BLE     13$
000616  005001                         16$:    CLR     R1                              ; INDEX                         2421
000620  013700  000000G                17$:    MOV     REG.ADR,R0                      ;                               2422
000624  016066  000016  000014                 MOV     16(R0),14(SP)                   ; *,TMP.LOCATION
000632  032766  020000  000014                 BIT     #20000,14(SP)                   ; *,TMP.LOCATION
000640  001007                                 BNE     18$
000642  016666  000014  000016                 MOV     14(SP),16(SP)                   ; *,TMP.LOCATION                2425
000650  016637  000016  000000G                MOV     16(SP),TEMP3                    ; TMP.LOCATION,*
000656  000440                                 BR      20$                             ;                               2424
000660  020127  002000                 18$:    CMP     R1,#2000                        ; INDEX,*                       2429
000664  001031                                 BNE     19$
000666  012746  000000G                        MOV     #MSG59,-(SP)                    ;                               2432
000672  012746  000001                         MOV     #1,-(SP)
000676  010600                                 MOV     SP,R0                           ; SP,*
000700  104414                                 TRAP    14
000702  013700  000000G                        MOV     REG.ADR,R0                      ;                               2433
000706  016066  000016  000024                 MOV     16(R0),24(SP)                   ; *,TMP.LOCATION
000714  016616  000024                         MOV     24(SP),(SP)                     ; TMP.LOCATION,*
000720  012746  000000G                        MOV     #MSG20,-(SP)
000724  012746  000002                         MOV     #2,-(SP)
000730  010600                                 MOV     SP,R0                           ; SP,*
000732  104414                                 TRAP    14
000734  104455                                 TRAP    55                              ;                               2434
000736  001300                                 .WORD   1300
000740  000000G                                .WORD   MSG00
000742  000000G                                .WORD   ERROR$REPORT
000744  062706  000010                         ADD     #10,SP                          ;                               2431
000750  005201                         19$:    INC     R1                              ; INDEX                         2421
000752  020127  002000                         CMP     R1,#2000                        ; INDEX,*
000756  003720                                 BLE     17$
000760  012701  000062                 20$:    MOV     #62,R1                          ; *,$$TMP2                      2442
000764  001410                         21$:    BEQ     24$
000766  013700  000000G                        MOV     L$DLY,R0                        ; *,$$TMP1
000772  001403                                 BEQ     23$
000774  005066  000030                 22$:    CLR     30(SP)                          ; $$TMP
001000  077003                                 SOB     R0,22$                          ; $$TMP1,*
001002  005301                         23$:    DEC     R1                              ; $$TMP2
001004  000767                                 BR      21$
001006  013700  000000G                24$:    MOV     REG.ADR,R0                      ;                               2444
```

```
001012  016066  000016  000022              MOV     16(R0),22(SP)           ; *,TMP.LOCATION
001020  100007                              BPL     25$
001022  016666  000022  000024              MOV     22(SP),24(SP)           ; *,TMP.LOCATION                2447
001030  016637  000024  000000G             MOV     24(SP),TEMP4            ; TMP.LOCATION,*
001036  000406                              BR      26$                     ;                               2446
001040  005202                      25$:    INC     R2                      ; INDEX2                        2386
001042  020227  000200                      CMP     R2,#200                 ; INDEX2,*
001046  003002                              BGT     26$
001050  000137  005030'                      JMP     12$
001054  023727  000000G 000200      26$:    CMP     TEMP5,#200              ;                               2452
001062  001031                              BNE     27$
001064  012746  000000G                     MOV     #MSG59,-(SP)            ;                               2455
001070  012746  000001                      MOV     #1,-(SP)
001074  010600                              MOV     SP,R0                   ; SP,*
001076  104414                              TRAP    14
001100  013700  000000G                     MOV     REG.ADR,R0              ;                               2456
001104  016066  000016  000032              MOV     16(R0),32(SP)           ; *,TMP.LOCATION
001112  016616  000032                      MOV     32(SP),(SP)             ; TMP.LOCATION,*
001116  012746  000000G                     MOV     #MSG21,-(SP)
001122  012746  000002                      MOV     #2,-(SP)
001126  010600                              MOV     SP,R0                   ; SP,*
001130  104414                              TRAP    14
001132  104455                              TRAP    55                      ;                               2457
001134  001301                              .WORD   1301
001136  000000G                             .WORD   MSG00
001140  000000G                             .WORD   ERROR$REPORT
001142  062706  000010                      ADD     #10,SP                  ;                               2454
001146  032737  040000  000010G     27$:    BIT     #40000,XMIT.D.LIST+10   ;                               2461
001154  001426                              BEQ     28$
001156  032737  001000  000010G             BIT     #1000,XMIT.D.LIST+10
001164  001422                              BEQ     28$
001166  012746  000000G                     MOV     #MSG59,-(SP)            ;                               2464
001172  012746  000001                      MOV     #1,-(SP)
001176  010600                              MOV     SP,R0                   ; SP,*
001200  104414                              TRAP    14
001202  012716  000000G                     MOV     #MSG71,(SP)             ;                               2465
001206  012746  000001                      MOV     #1,-(SP)
001212  010600                              MOV     SP,R0                   ; SP,*
001214  104414                              TRAP    14
001216  104455                              TRAP    55                      ;                               2466
001220  001302                              .WORD   1302
001222  000000G                             .WORD   MSG00
001224  000000G                             .WORD   ERROR$REPORT
001226  062706  000006                      ADD     #6,SP                   ;                               2463
001232  012746  100220              28$:    MOV     #-77560,-(SP)           ;                               2474
001236  011646                              MOV     (SP),-(SP)
001240  004737  000000G                     JSR     PC,CHK.CSR.STATUS
001244  042737  000400  000010G             BIC     #400,XMIT.D.LIST+10     ;                               2475
001252  012716  140000                      MOV     #-40000,(SP)            ;                               2476
001256  005046                              CLR     -(SP)
001260  004737  000000G                     JSR     PC,CHK.XMIT.STATUS
001264  012716  140000                      MOV     #-40000,(SP)            ;                               2477
001270  012746  020000                      MOV     #20000,-(SP)
```

E11


```
001274  004737  000000G                         JSR     PC,CHK.RCV.STATUS
001300  032737  037777  000012G                 BIT     #37777,XMIT.D.LIST+12           ;                       2479
001306  001021                                   BNE     29$
001310  012716  000000G                         MOV     #MSG59,(SP)                     ;                       2482
001314  012746  000001                           MOV     #1, (SP)
001320  010600                                   MOV     SP,R0                           ; SP,*
001322  104414                                   TRAP    14
001324  012716  000000G                         MOV     #MSG58,(SP)                     ;                       2483
001330  012746  000001                           MOV     #1,-(SP)
001334  010600                                   MOV     SP,R0                           ; SP,*
001336  104414                                   TRAP    14
001340  104455                                   TRAP    55                              ;                       2484
001342  001303                                   .WORD   1303
001344  000000G                                  .WORD   MSG00
001346  000000G                                  .WORD   ERROR$REPORT
001350  022626                                   CMP     (SP)+,(SP)+                     ;                       2481
001352  062706  000010                 29$:      ADD     #10,SP                          ;                       2383
001356  104467                                   TRAP    o7                              ;                       2485
001360  006000                                   ROR     R0
001362  103002                                   BHIS    30$
001364  000137  005024'                         JMP     11$
001370  062706  000032                 30$:      ADD     #32,SP                          ;                       2285
001374  000207                                   RTS     PC

; Routine Size:  383 words,     Routine Base:  AB$CODE$ + 4370
; Maximum stack depth per invocation:  25 words




                                                 .SBTTL  T7 TEST 7 - ETHERNET CARRIER SENSE TEST
000C00  004737  004370'                 T7::
000000                                   1$:     JSR     PC,$T7                          ;                       2488
000004  104466                                   TRAP    66
000006  006000                                   ROR     R0
000010  103773                                   BLO     1$
000012  000207                                   RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 5766
; Maximum stack depth per invocation:  2 words


;   2490  1
;   2491  1
```

```
;  2492  1     %SBTTL 'TEST 8   STATION ADDRESS RAM TEST'
;  2493  1     !++
;  2494  1     !
;  2495  1     !  TEST 8:      STATION ADDRESS RAM TEST
;  2496  1     !
;  2497  1     !  DESCRIPTION:
;  2498  1     !
;  2499  1     !       This test verifies that Station Address RAM has no static faults.
;  2500  1     !       The host writes and then reads data patterns to all of the
;  2501  1     !       addressable RAM ( 128 decimal bytes ). The data is checked to see
;  2502  1     !       that the data pattern received is the same as the data pattern
;  2503  1     !       transmited. This test continues until all the data patterns are
;  2504  1     !       exhausted. If the operator specifies loop on error, the program
;  2505  1     !       re-executes the code that detected the error until ^C is entered.
;  2506  1     !
;  2507  1     !       The following BINARY patterns are used:
;  2508  1     !
;  2509  1     !                       11111111        00000000
;  2510  1     !                       10101010        01010101
;  2511  1     !                       11001100        00110011
;  2512  1     !                       11110000        00001111
;  2513  1     !                       marching 1's, propagating 1's through the RAM
;  2514  1     !                       marching 0's, propagating 0's through the RAM
;  2515  1     !
;  2516  1     !       Hardware tested:        Station Address RAM
;  2517  1     !                               Q-Bus to QTDC interface
;  2518  1     !                               CSR register - Receiver Enable (bit 0)
;  2519  1     !                               Portion of Receive and Transmit FIFO
;  2520  1     !       Processing:
;  2521  1     !
;  2522  1     !            BEGIN
;  2523  1     !               reset device
;  2524  1     !               select Setup mode
;  2525  1     !               REPEAT for each pattern
;  2526  1     !                  load transmit packet with data pattern
;  2527  1     !                  transmit loopback packet (fill all of the RAM)
;  2528  1     !                  receive packet
;  2529  1     !                  check for expected loopback status
;  2530  1     !                  IF error
;  2531  1     !                  THEN
;  2532  1     !                     print error message if not inhibited
;  2533  1     !                  ENDIF
;  2534  1     !                  call compare_packets
;  2535  1     !               ENDREPEAT
;  2536  1     !            END
;  2537  1     !--
```

```
;    2538  3     BGNTST;
;    2539  3
;    2540  3           RESET_DEQNA ( );
;    2541  3
;    2542  3           DECR INDEX1 FROM 7 TO 0 DO
;    2543  4             BEGIN
;    2544  4               INCR INDEX2 FROM 0 TO 127 DO
;    2545  4                 XMIT_BUFFER [ .INDEX2 ] = .PTRN_TABLE [ .INDEX1 ];
;    2546  4
;    2547  6               BGNSUB;
;    2548  6               XMIT_SETUP_PACKET ( N_MODE );
;    2549  4               ENDSUB;
;    2550  3             END;
;    2551  3
;    2552  3     !        TEMP3 = ( N_MODE * 8 ) - 1;
;    2553  3     !        INCR INDEX1 FROM 0 TO .TEMP3 DO
;    2554  3     !          BEGIN
;    2555  3     !            P1 = ZERO;
;    2556  3     !            P2 = .INDEX1;
;    2557  3     !            WALKING_BIT ( );
;    2558  3     !            P1 = N_MODE;
;    2559  3     !            XMIT_SETUP_PACKET ( );
;    2560  3     !
;    2561  3     !            INCR INDEX FROM 0 TO .P3 DO
;    2562  3     !              XMIT_BUFFER [ .INDEX ] = ( - .XMIT_BUFFER [ .INDEX ] ) - 1;
;    2563  3     !            P1 = N_MODE;
;    2564  3     !            XMIT_SETUP_PACKET ( );
;    2565  3     !          END;
;    2566  3
;    2567  3           INCR INDEX1 FROM 0 TO N_MODE - 1 DO
;    2568  4             BEGIN
;    2569  4               INCR INDEX FROM 0 TO N_MODE - 1 DO
;    2570  4                 XMIT_BUFFER [ .INDEX ] = ZERO;
;    2571  4               XMIT_BUFFER [  .INDEX1 ] = %X'FF';
;    2572  4
;    2573  6               BGNSUB;
;    2574  6               XMIT_SETUP_PACKET ( N_MODE );
;    2575  4               ENDSUB;
;    2576  4
;    2577  4               INCR INDEX FROM 0 TO .P3 DO
;    2578  4                 XMIT_BUFFER [ .INDEX ] = ( - .XMIT_BUFFER [ .INDEX ] ) - 1;
;    2579  4
;    2580  6               BGNSUB;
;    2581  6               XMIT_SETUP_PACKET ( N_MODE );
;    2582  4               ENDSUB;
;    2583  4
;    2584  3             END;
;    2585  1     ENDTST;


                                        .SBTTL  $T8 TEST 8 - STATION ADDRESS RAM TEST
000000  004137  000000G          $T8:   JSR     R1,$SAVE3                       ;  2489
000004  004737  000000G                 JSR     PC,RESET.DEQNA                  ;  2540
```

```
000010  012701  000007                          MOV     #7,R1                           ; *,INDEX1                      2542
000014  005000                          1$:     CLR     R0                              ; INDEX2                        2544
000016  116160  0C0000G 000000G         2$:     MOVB    PTRN.TABLE(R1),XMIT.BUFFER(R0)  ; *(INDEX1),*(INDEX2)           2545
000024  005200                                  INC     R0                              ; INDEX2                        2544
000026  020027  000177                          CMP     R0,#177                         ; INDEX2,*
000032  003771                                  BLE     2$
000034  104402                          3$:     TRAP    2                               ;                               2545
000036  012746  000200                          MOV     #200,-(SP)                      ;                               2548
000042  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000046  005726                                  TST     (SP)+                           ;                               2545
000050  104467                                  TRAP    67                              ;                               2548
000052  006000                                  ROR     R0
000054  103767                                  BLO     3$
000056  005301                                  DEC     R1                              ; INDEX1                        2542
000060  002355                                  BGE     1$
000062  005001                                  CLR     R1                              ; INDEX1                        2567
000064  005000                          4$:     CLR     R0                              ; INDEX                         2569
000066  105060  000000G                 5$:     CLRB    XMIT.BUFFER(R0)                 ; *(INDEX)                      2570
000072  005200                                  INC     R0                              ; INDEX                         2569
000074  020027  000177                          CMP     R0,#177                         ; INDEX,*
000100  003772                                  BLE     5$
000102  112761  000377  000000G                 MOVB    #377,XMIT.BUFFER(R1)            ; *,*(INDEX1)                   2571
000110  104402                          6$:     TRAP    2
000112  012746  000200                          MOV     #200,-(SP)                      ;                               2574
000116  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000122  005726                                  TST     (SP)+                           ;                               2571
000124  104467                                  TRAP    67                              ;                               2574
000126  006000                                  ROR     R0
000130  103767                                  BLO     6$
000132  005000                                  CLR     R0                              ; INDEX                         2577
000134  000411                                  BR      8$
000136  012702  177777                  7$:     MOV     #-1,R2                          ;                               2578
000142  005003                                  CLR     R3
000144  156003  000000G                         BISB    XMIT.BUFFER(R0),R3              ; *(INDEX),*
000150  160302                                  SUB     R3,R2
000152  110260  000000G                         MOVB    R2,XMIT.BUFFER(R0)              ; *,*(INDEX)
000156  005200                                  INC     R0                              ; INDEX                         2577
000160  020037  000000G                 8$:     CMP     R0,P3                           ; INDEX,*
000164  003764                                  BLE     7$
000166  104402                          9$:     TRAP    2                               ;                               2578
000170  012746  000200                          MOV     #200,-(SP)                      ;                               2581
000174  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000200  005726                                  TST     (SP)+                           ;                               2578
000202  104467                                  TRAP    67                              ;                               2581
000204  006000                                  ROR     R0
000206  103767                                  BLO     9$
000210  005201                                  INC     R1                              ; INDEX1                        2567
000212  020127  000177                          CMP     R1,#177                         ; INDEX1,*
000216  003722                                  BLE     4$
000220  000207                                  RTS     PC                              ;                               2489

; Routine Size:  73 words,      Routine Base:  AB$CODE$ + 6002
; Maximum stack depth per invocation:  6 words
```

```
                                        .SBTTL  T8 TEST 8 - STATION ADDRESS RAM TEST
000000  004737  006002'            T8::
000000                              1$:     JSR     PC,$T8                          ;                       2584
000004  104466                              TRAP    66
000006  006000                              ROR     R0
000010  103773                              BLO     1$
000012  000207                              RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 6224
; Maximum stack depth per invocation:  2 words


;   2586  1
```

```
;  2587  1     %SBTTL 'TEST 9   PROMISCUOUS STATION ADDRESS TEST'
;  2588  1     !++
;  2589  1     !
;  2590  1     !  TEST 9:      PROMISCUOUS STATION ADDRESS TEST
;  2591  1     !
;  2592  1     !  DESCRIPTION:
;  2593  1     !
;  2594  1     !       This test verifies that DEQNA promiscuous addressing mode functions
;  2595  1     !       as specified. Bit patterns and addresses in and out of the range of
;  2596  1     !       setup addresses are used to assure that there is true promiscuity.
;  2597  1     !       If the operator specifies loop on error, the program re-executes the
;  2598  1     !       code that detected the error until ^C is entered.
;  2599  1     !
;  2600  1     !       Hardware tested:        Promiscuous addressing mode logic
;  2601  1     !
;  2602  1     !       Set of Target Addresses in HEXADECIMAL:
;  2603  1     !
;  2604  1     !               00-00-00-00-00-00
;  2605  1     !               AA-AA-AA AA-AA-AA
;  2606  1     !               55-55-55-55-55-55
;  2607  1     !               FF-FF-FF-FF-FF-FF
;  2608  1     !               Walking 1, shifting 1 across the Target Station Address
;  2609  1     !               Walking 0, shifting 0 across the Target Station Address
;  2610  1     !
;  2611  1     !       Processing:
;  2612  1     !
;  2613  1     !            BEGIN
;  2614  1     !               reset device
;  2615  1     !               select internal loopback mode
;  2616  1     !               set mode to Setup
;  2617  1     !               set 'promiscuous' addressing mode bit
;  2618  1     !               REPEAT for each Target Address
;  2619  1     !                  load Target Address of the packet
;  2620  1     !                  disable receiver
;  2621  1     !                  transmit loopback packet
;  2622  1     !                  enable receiver
;  2623  1     !                  check for expected loopback status
;  2624  1     !                  IF error
;  2625  1     !                  THEN
;  2626  1     !                     print error message if not inhibited
;  2627  1     !                  ENDIF
;  2628  1     !                  call compare_packets
;  2629  1     !               ENDREPEAT
;  2630  1     !             END
;  2631  1     !--
```

```
;  2632  3     BGNTST;
;  2633  3
;  2634  3           !++
;  2635  3           !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM IF EXECUTING
;  2636  3           !  TESTS IN EXTERNAL LOOPBACK MODE.
;  2637  3           !--
;  2638  3
;  2639  3           RESET_DEQNA ( );
;  2640  3           PREP_FOR_SETUP ( );
;  2641  3           INCR INDEX1 FROM 1 TO 14 DO
;  2642  3             WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  2643  3
;  2644  5           BGNSUB;
;  2645  5             XMIT_SETUP_PACKET ( P_MODE );
;  2646  3           ENDSUB;
;  2647  3
;  2648  3           !++
;  2649  3           !  NOW LOOPBACK 6 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;  2650  3           !--
;  2651  3
;  2652  3           RBUF_LENGTH = 6;
;  2653  3           XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  2654  3
;  2655  3           INCR INDEX1 FROM 0 TO 99 DO
;  2656  4             BEGIN
;  2657  4               SELECTONE .INDEX1 OF
;  2658  4                 SET
;  2659  4                 [  0 TO 3  ]:
;  2660  4                              WRT_STATION_ADR ( ZERO, .INDEX1 );
;  2661  4                 [  4 TO 51 ]:
;  2662  4                              WALKING_BIT ( ZERO, .INDEX1 - 4, 5 );
;  2663  4                 [ 52 TO 99 ]:
;  2664  4                              WALKING_BIT ( ONE, .INDEX1 - 52, 5 );
;  2665  4                 TES;
;  2666  4
;  2667  4               WRT_STATION_ADR ( ZERO, ZERO );
;  2668  4
;  2669  6               BGNSUB;
;  2670  6                 XMIT_ILOOP_PACKET ( ZERO );
;  2671  4               ENDSUB;
;  2672  4
;  2673  3             END;
;  2674  3
;  2675  3           INCR INDEX FROM 0 TO 5 DO
;  2676  3             TARGET_ADR [ .INDEX ] = ZERO;
;  2677  1     ENDTST;


                                        .SBTTL  $T9 TEST 9 - PROMISCUOUS STATION ADDRESS TEST
000000  010146                  $T9:    MOV     R1,-(SP)                ;                       2585
000002  004737  000000G                 JSR     PC,RESET.DEQNA          ;                       2639
000006  004737  000000G                 JSR     PC,PREP.FOR.SETUP       ;                       2640
000012  012701  000001                  MOV     #1,R1                   ; *.INDEX1              2641
```

```
000016  010146                          1$:     MOV     R1,-(SP)                ; INDEX1,*                      2642
000020  012746  000023                          MOV     #23,-(SP)
000024  004737  000000G                        JSR     PC,WRT.STATION.ADR
000030  022626                                  CMP     (SP)+,(SP)+
000032  005201                                  INC     R1                      ; INDEX1                        2641
000034  020127  000016                          CMP     R1,#16                  ; INDEX1,*
000040  003766                                  BLE     1$
000042  104402                          2$:     TRAP    2                       ;                               2642
000044  012746  000202                          MOV     #202,-(SP)              ;                               2645
000050  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000054  005726                                  TST     (SP)+                   ;                               2642
000056  104467                                  TRAP    67                      ;                               2645
000060  006000                                  ROR     R0
000062  103767                                  BLO     2$
000064  012737  000006  000000G                MOV     #6,RBUF.LENGTH          ;                               2652
000072  012700  000006                          MOV     #6,R0                   ;                               2653
000076  006200                                  ASR     R0
000100  005400                                  NEG     R0
000102  010037  000000G                        MOV     R0,XBUF.LENGTH
000106  005001                                  CLR     R1                      ; INDEX1                        2655
000110  005701                          3$:     TST     R1                      ; INDEX1                        2659
000112  002411                                  BLT     4$
000114  020127  000003                          CMP     R1,#3                   ; INDEX1,*
000120  003006                                  BGT     4$
000122  005046                                  CLR     -(SP)                   ;                               2660
000124  010146                                  MOV     R1,-(SP)                ; INDEX1,*
000126  004737  000000G                        JSR     PC,WRT.STATION.ADR
000132  022626                                  CMP     (SP)+,(SP)+
000134  000434                                  BR      7$                      ;                               2657
000136  020127  000004                  4$:     CMP     R1,#4                   ; INDEX1,*                      2661
000142  002410                                  BLT     5$
000144  020127  000063                          CMP     R1,#63                  ; INDEX1,*
000150  003005                                  BGT     5$
000152  005046                                  CLR     -(SP)                   ;                               2662
000154  010146                                  MOV     R1,-(SP)                ; INDEX1,*
000156  162716  000004                          SUB     #4,(SP)
000162  000413                                  BR      6$
000164  020127  000064                  5$:     CMP     R1,#64                  ; INDEX1,*                      2663
000170  002416                                  BLT     7$
000172  020127  000143                          CMP     R1,#143                 ; INDEX1,*
000176  003013                                  BGT     7$
000200  012746  000001                          MOV     #1,-(SP)                ;                               2664
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)                   ;                               2667
000230  005046                                  CLR     -(SP)
000232  004737  000000G                        JSR     PC,WRT.STATION.ADR
000236  104402                          8$:     TRAP    2
000240  005016                                  CLR     (SP)                    ;                               2670
000242  004737  000000G                        JSR     PC,XMIT.ILOOP.PACKET
```

M11


```
000246  104467                                  TRAP    67
000250  006000                                  ROR     R0
000252  103771                                  BLO     8$
000254  022626                                  CMP     (SP)+,(SP)+                     ;                               2656
000256  005201                                  INC     R1                              ; INDEX1                        2655
000260  020127  000143                          CMP     R1,#143                         ; INDEX1,*
000264  003711                                  BLE     3$
000266  005000                                  CLR     R0                              ; INDEX                         2675
000270  105060  000000G                 9$:     CLRB    TARGET.ADR(R0)                  ; *(INDEX)                      2676
000274  005200                                  INC     R0                              ; INDEX                         2675
000276  020027  000005                          CMP     R0,#5                           ; INDEX,*
000302  003772                                  BLF     9$
000304  012601                                  MOV     (SP)+,R1                        ;                               2585
000306  000207                                  RTS     PC

; Routine Size:  100 words,     Routine Base:  AB$CODE$ + 6240
; Maximum stack depth per invocation:  5 words




                                        .SBTTL  T9 TEST 9 - PROMISCUOUS STATION ADDRESS TEST
000000  004737  006240'                 T9::
000000                                  1$:     JSR     PC,$T9                          ;                               2676
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 6550
; Maximum stack depth per invocation:  2 words


;   2678  1
```

```
;   2679  1     %SBTTL 'TEST 10   TRANSMIT AND RECEIVE FIFO MEMORY TEST'
;   2680  1     !++
;   2681  1     !
;   2682  1     !  TEST 10:     TRANSMIT AND RECEIVE FIFO MEMORY TEST
;   2683  1     !
;   2684  1     !  DESCRIPTION:
;   2685  1     !
;   2686  1     !       This test verifies that link memory (receive FIFO and transmit
;   2687  1     !       buffer) has no static faults. The host writes and then reads
;   2688  1     !       a sequence of data patterns to the link memory. The data is then
;   2689  1     !       checked to see that the data pattern received is the same as the
;   2690  1     !       data pattern transmitted. This test continues until all the data
;   2691  1     !       patterns are exhausted. If the operator specifies loop on error, the
;   2692  1     !       program re-executes the code that detected the error until ^C is
;   2693  1     !       entered.
;   2694  1     !
;   2695  1     !       Hardware tested:        Transmit buffer address logic
;   2696  1     !                               Transmit buffer memory ( first 1512 bytes )
;   2697  1     !                               Receive FIFO address logic
;   2698  1     !                               Receive FIFO memory ( first 1512 bytes )
;   2699  1     !
;   2700  1     !       The following BINARY patterns are used:
;   2701  1     !
;   2702  1     !                       11111111        00000000
;   2703  1     !                       10101010        01010101
;   2704  1     !                       11001100        00110011
;   2705  1     !                       11110000        00001111
;   2706  1     !
;   2707  1     !       Processing:
;   2708  1     !
;   2709  1     !            BEGIN
;   2710  1     !               reset device
;   2711  1     !               select internal/extended loopback mode
;   2712  1     !               REPEAT for each pattern
;   2713  1     !                  write link memory with pattern - transmit loopback packet
;   2714  1     !                  read link memory with pattern - receive loopback packet
;   2715  1     !                  check for expected loopback status
;   2716  1     !                  IF error
;   2717  1     !                  THEN
;   2718  1     !                     print error message if not inhibited
;   2719  1     !                  ENDIF
;   2720  1     !                  call compare_packets
;   2721  1     !               ENDREPEAT
;   2722  1     !            END
;   2723  1     !--
```

```
;  2724  3      BGNTST;
;  2725  3
;  2726  3      !++
;  2727  3      !  LOOPBACK 1514 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;  2728  3      !--
;  2729  3
;  2730  3      RBUF_LENGTH = LONGEST_PACKET;
;  2731  3      XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  2732  3
;  2733  3      INCR INDEX FROM 0 TO 7 DO
;  2734  4        BEGIN
;  2735  4          RESET_DEQNA ( );
;  2736  4          TEMP1 = 0;
;  2737  4          INCR INDEX1 FROM 0 TO 189 DO
;  2738  4            INCR INDEX2 FROM 0 TO 7 DO
;  2739  5              BEGIN
;  2740  5                XMIT_BUFFER [ .TEMP1 ] = .PTRN_TABLE [ .INDEX2 ];
;  2741  5                TEMP1 = .TEMP1 + 1;
;  2742  4              END;
;  2743  4
;  2744  4          !++
;  2745  4          !  ROTATE PATTERN TABLE
;  2746  4          !--
;  2747  4
;  2748  4          TEMP2 = .PTRN_TABLE [ 0 ];
;  2749  4          INCR INDEX3 FROM 0 TO 6 DO
;  2750  4            PTRN_TABLE [ .INDEX3 ] = .PTRN_TABLE [ .INDEX3 + 1 ];
;  2751  4          PTRN_TABLE [ 7 ] = .TEMP2;
;  2752  4
;  2753  6          BGNSUB;
;  2754  6            SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  2755  6            SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  2756  6            SEND_ELOOP_PACKET ( ZERO );
;  2757  6            COMPARE_PACKETS ( );
;  2758  4          ENDSUB;
;  2759  4
;  2760  3        END;
;  2761  3
;  2762  3      !  INCR INDEX1 FROM 0 TO LONGEST_PACKET   1 DO
;  2763  3      !    BEGIN
;  2764  3      !      INCR INDEX FROM 0 TO LONGEST_PACKET - 1 DO
;  2765  3      !        XMIT_BUFFER [ .INDEX ] = ZERO;
;  2766  3      !      XMIT_BUFFER [  .INDEX1 ] = %X'FF';
;  2767  3      !
;  2768  3      !      BGNSUB;
;  2769  3      !        SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  2770  3      !        SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  2771  3      !        SEND_ELOOP_PACKET ( ZERO );
;  2772  3      !        COMPARE_PACKETS ( );
;  2773  3      !      ENDSUB;
;  2774  3      !
;  2775  3      !      INCR INDEX FROM 0 TO .P3 DO
;  2776  3      !        XMIT_BUFFER [ .INDEX ] = ( - .XMIT_BUFFER [ .INDEX ] ) - 1;
```

C12


```
;     2777  3      !
;     2778  3      !        BGNSUB;
;     2779  3      !          SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;     2780  3      !          SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;     2781  3      !          SEND_ELOOP_PACKET ( ZERO );
;     2782  3      !          COMPARE_PACKETS ( );
;     2783  3      !        ENDSUB;
;     2784  3      !
;     2785  3      !      END;
;     2786  3
;     2787  1      ENDTST;


                                          .SBTTL   $T10 TEST 10 - TRANSMIT AND RECEIVE FIFO MEMORY TEST
000000  004137  000000G           $T10:   JSR      R1,$SAVE3                              ;                        2677
000004  012737  002752  000000G           MOV      #2752,RBUF.LENGTH                      ;                        2730
000012  012700  002752                    MOV      #2752,R0                               ;                        2731
000016  006200                            ASR      R0
000020  005400                            NEG      R0
000022  010037  000000G                   MOV      R0,XBUF.LENGTH
000026  012703  000010                    MOV      #10,R3                                 ; *,INDEX                2733
000032  004737  000000G           1$:     JSR      PC,RESET.DEQNA                         ;                        2735
000036  005037  000000G                   CLR      TEMP1                                  ;                        2736
000042  012702  000276                    MOV      #276,R2                                ; *,INDEX1               2737
000046  005000                    2$:     CLR      R0                                     ; INDEX2                 2738
000050  013701  000000G           3$:     MOV      TEMP1,R1                               ;                        2740
000054  116061  000000G 000000G           MOVB     PTRN.TABLE(R0),XMIT.BUFFER(R1)         ; *(INDEX2),*
000062  005237  000000G                   INC      TEMP1                                  ;                        2741
000066  005200                            INC      R0                                     ; INDEX2                 2738
000070  020027  000007                    CMP      R0,#7                                  ; INDEX2,*
000074  003765                            BLE      3$
000076  077215                            SOB      R2,2$                                  ; INDEX1,*               2737
000100  005037  000000G                   CLR      TEMP2                                  ;                        2748
000104  113737  000000G 000000G           MOVB     PTRN.TABLE,TEMP2
000112  005000                            CLR      R0                                     ; INDEX3                 2749
000114  116060  000001G 000000G   4$:     MOVB     PTRN.TABLE+1(R0),PTRN.TABLE(R0)        ; *(INDEX3),*(INDEX3)    2750
000122  005200                            INC      R0                                     ; INDEX3                 2749
000124  020027  000006                    CMP      R0,#6                                  ; INDEX3,*
000130  003771                            BLE      4$
000132  113737  000000G 000007G           MOVB     TEMP2,PTRN.TABLE+7                     ;                        2751
000140  104402                    5$:     TRAP     2
000142  013746  000000G                   MOV      XBUF.LENGTH,-(SP)                      ;                        2754
000146  012746  120000                    MOV      #-60000,-(SP)
000152  004737  000000G                   JSR      PC,SET.RDESCR.LIST
000156  013716  000000G                   MOV      XBUF.LENGTH,(SP)                       ;                        2755
000162  012746  120000                    MOV      #-60000,-(SP)
000166  004737  000000G                   JSR      PC,SET.XDESCR.LIST
000172  005016                            CLR      (SP)                                   ;                        2756
000174  004737  000000G                   JSR      PC,SEND.ELOOP.PACKET
000200  004737  000000G                   JSR      PC,COMPARE.PACKETS                     ;                        2757
000204  062706  000006                    ADD      #6,SP                                  ;                        2751
000210  104467                            TRAP     67                                     ;                        2757
000212  006000                            ROR      R0
```

D12


```
000214  103751                          BLO     5$
000216  077373                          SOB     R3,1$                   ; INDEX,*                       2735
000220  000207                          RTS     PC                      ;                               2677

; Routine Size:  73 words,      Routine Base:  AB$CODE$ + 6564
; Maximum stack depth per invocation:  8 words



                                        .SBTTL  T10 TEST 10 - TRANSMIT AND RECEIVE FIFO MEMORY TEST
000000  004737  006564'          T10::
000000                           1$:    JSR     PC,$T10                 ;                               2760
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 7006
; Maximum stack depth per invocation:  2 words


;   2788  1
```

```
;  2789  1     %SBTTL 'TEST 11 - PACKET LENGTH TEST'
;  2790  1     !++
;  2791  1     !
;  2792  1     !  TEST 11:     PACKET LENGTH TEST
;  2793  1     !
;  2794  1     !  DESCRIPTION:
;  2795  1     !
;  2796  1     !       This test verifies that DEQNA can transmit and receive variable
;  2797  1     !       length packets ( equal to or greater than 60 bytes and equal to or
;  2798  1     !       less than 1514 bytes without the CRC ) without losing any data
;  2799  1     !       in the process. This test also verifies that the 9th bit of the
;  2800  1     !       FIFO memory is not static (stuck at 1/stuck at 0). If the operator
;  2801  1     !       specifies loop on error, the program re-executes the code that
;  2802  1     !       detected the error until ^C is entered.
;  2803  1     !
;  2804  1     !       Hardware tested:        Transmit and Receive RAM
;  2805  1     !
;  2806  1     !       Processing:
;  2807  1     !
;  2808  1     !            BEGIN
;  2809  1     !               reset device
;  2810  1     !               select internal/extended loopback mode
;  2811  1     !               set down_count to max. packet length
;  2812  1     !               set up_count to min. packet length
;  2813  1     !               REPEAT until down_count = min. packet length
;  2814  1     !                  transmit loopback packet (packet length = down_count)
;  2815  1     !                  check for expected loopback status and packet length
;  2816  1     !                  IF error
;  2817  1     !                  THEN
;  2818  1     !                       print error message if not inhibited
;  2819  1     !                  ENDIF
;  2820  1     !                  call compare_packets
;  2821  1     !                  transmit loopback packet (packet length = up_count)
;  2822  1     !                  check for expected loopback status and packet length
;  2823  1     !                  IF error
;  2824  1     !                  THEN
;  2825  1     !                     print error message if not inhibited
;  2826  1     !                  ENDIF
;  2827  1     !                  call compare_packets
;  2828  1     !                  decrement down_count by 2
;  2829  1     !                  increment up_count by 2
;  2830  1     !               ENDREPEAT
;  2831  1     !            END
;  2832  1     !--
```

```
;  2833  3     BGNTST;
;  2834  3
;  2835  3     !++
;  2836  3     !  LOOPBACK PACKETS OF INCREASING AND DECREASING LENGTH THEN CHECK IF PROPERLY
;  2837  3     !  RECEIVED
;  2838  3     !-
;  2839  3
;  2840  3     COUNTER      = ZERO;
;  2841  3     UP_COUNTER   = SHORTEST_PACKET;
;  2842  3     DOWN_COUNTER = LONGEST_PACKET;
;  2843  3
;  2844  3     INCR INDEX1 FROM SHORTEST_PACKET TO MAX_LENGTH BY STEP1 DO
;  2845  4       BEGIN
;  2846  4         RESET_DEQNA ( );
;  2847  4         IF .COUNTER EQLU ZERO
;  2848  4           THEN
;  2849  5             BEGIN
;  2850  5               RBUF_LENGTH = .UP_COUNTER;
;  2851  5               XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  2852  5               INCR INDEX FROM 0 TO .UP_COUNTER - 1 DO
;  2853  5                 XMIT_BUFFER [ .INDEX ] = %B'01010101';
;  2854  5               INCR INDEX FROM .UP_COUNTER TO MAX_LENGTH - 1 DO
;  2855  5                 XMIT_BUFFER [ .INDEX ] = ZERO;
;  2856  5               UP_COUNTER = .UP_COUNTER + STEP1;
;  2857  5               COUNTER = ONE;
;  2858  5             END
;  2859  4           ELSE
;  2860  5             BEGIN
;  2861  5               RBUF_LENGTH = .DOWN_COUNTER;
;  2862  5               XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  2863  5               INCR INDEX FROM 0 TO .DOWN_COUNTER - 1 DO
;  2864  5                 XMIT_BUFFER [ .INDEX ] = %B'10101010';
;  2865  5               INCR INDEX FROM .DOWN_COUNTER TO MAX_LENGTH - 1 DO
;  2866  5                 XMIT_BUFFER [ .INDEX ] = ZERO;
;  2867  5               DOWN_COUNTER = .DOWN_COUNTER - STEP1;
;  2868  5               COUNTER = ZERO;
;  2869  4             END;
;  2870  4
;  2871  6         BGNSUB;
;  2872  6           SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  2873  6           SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  2874  6           SEND_ELOOP_PACKET ( ZERO );
;  2875  6           COMPARE PACKETS ( );
;  2876  4         ENDSUB;
;  2877  4
;  2878  3       END;
;  2879  1     ENDTST;


                                        .SBTTL  $T11 TEST 11 - PACKET LENGTH TEST
000000  004137  000000G          $T11:  JSR     R1,$SAVE2                        ;  2787
000004  005037  000000G                 CLR     COUNTER                          ;  2840
000010  012737  000074  000000G        MOV     #74,UP.COUNTER                   ;  2841
```

G12


```
000016  012737  002752  000000G             MOV     #2752,DOWN.COUNTER      ;                                2842
000024  012 02  000074                      MOV     #74,R2                  ; *,INDEX1                       2844
000030  004 57  000000G             1$:     JSR     PC,RESET.DEQNA          ;                                2846
000034  005737  000000G                     TST     COUNTER                 ;                                2847
000040  001033                             BNE     6$
000042  013700  000000G                     MOV     UP.COUNTER,R0           ;                                2850
000046  010037  000000G                     MOV     R0,RBUF.LENGTH
000052  005001                             CLR     R1                      ; INDEX                          2852
000054  000404                             BR      3$
000056  112761  000125  000000G     2$:     MOVB    #125,XMIT.BUFFER(R1)    ; *,*(INDEX)                     2853
000064  005201                             INC     R1                      ; INDEX                          2852
000066  020100                     3$:     CMP     R1,R0                   ; INDEX,*
000070  002772                             BLT     2$
000072  005300                             DEC     R0                      ;                                2854
000074  000402                             BR      5$
000076  105060  000000G             4$:     CLRB    XMIT.BUFFER(R0)         ; *(INDEX)                       2855
000102  005200                     5$:     INC     R0                      ; INDEX                          2854
000104  020027  002775                     CMP     R0,#2775                ; INDEX,*
000110  003772                             BLE     4$
000112  062737  000002  000000G             ADD     #2,UP.COUNTER           ;                                2856
000120  012737  000001  000000G             MOV     #1,COUNTER              ;                                2857
000126  000431                             BR      11$                     ;                                2847
000130  013700  000000G             6$:     MOV     DOWN.COUNTER,R0         ;                                2861
000134  010037  000000G                     MOV     R0,RBUF.LENGTH
000140  005001                             CLR     R1                      ; INDEX                          2863
000142  000404                             BR      8$
000144  112761  000252  000000G     7$:     MOVB    #252,XMIT.BUFFER(R1)    ; *,*(INDEX)                     2864
000152  005201                             INC     R1                      ; INDEX                          2863
000154  020100                     8$:     CMP     R1,R0                   ; INDEX,*
000156  002772                             BLT     7$
000160  005300                             DEC     R0                      ;                                2865
000162  000402                             BR      10$
000164  105060  000000G             9$:     CLRB    XMIT.BUFFER(R0)         ; *(INDEX)                       2866
000170  005200                     10$:    INC     R0                      ; INDEX                          2865
000172  020027  002775                     CMP     R0,#2775                ; INDEX,*
000176  003772                             BLE     9$
000200  162737  000002  000000G             SUB     #2,DOWN.COUNTER         ;                                2867
000206  005037  000000G                     CLR     COUNTER                 ;                                2868
000212  013700  000000G             11$:    MOV     RBUF.LENGTH,R0          ;                                2851
000216  006200                             ASR     R0
000220  005400                             NEG     R0
000222  010037  000000G                     MOV     R0,XBUF.LENGTH
000226  104402                     12$:    TRAP    2                       ;                                2869
000230  013746  000000G                     MOV     XBUF.LENGTH,-(SP)       ;                                2872
000234  012746  120000                     MOV     #-60000,-(SP)
000240  004737  000000G                     JSR     PC,SET.RDESCR.LIST
000244  013716  000000G                     MOV     XBUF.LENGTH,(SP)        ;                                2873
000250  012746  120000                     MOV     #-60000,-(SP)
000254  004737  000000G                     JSR     PC,SET.XDESCR.LIST
000260  005016                             CLR     (SP)                    ;                                2874
000262  004737  000000G                     JSR     PC,SEND.ELOOP.PACKET
000266  004737  000000G                     JSR     PC,COMPARE.PACKETS      ;                                2875
000272  062706  000006                     ADD     #6,SP                   ;                                2869
```

```
000276  104467                          TRAP    67                              ;                                       2875
000300  006000                          ROR     R0
000302  103751                          BLO     12$
000304  062702  000002                  ADD     #2,R2                           ; *,INDEX1                              2844
000310  020227  002776                  CMP     R2,#2776                        ; INDEX1,*
000314  003645                          BLE     1$
000316  000207                          RTS     PC                              ;                                       2787

; Routine Size:  104 words,     Routine Base:  AB$CODE$ + 7022
; Maximum stack depth per invocation:  7 words



                                        .SBTTL  T11 TEST 11 - PACKET LENGTH TEST
000000  004737  007022'          T11::
000000                           1$:    JSR     PC,$T11                         ;                                       2878
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 7342
; Maximum stack depth per invocation:  2 words


;   2880  1
```

```
;    2881  1
;    2882  1     %SBTTL 'TEST 12 - NXM INTERRUPT TEST'
;    2883  1     !++
;    2884  1     !
;    2885  1     !  TEST 12:     NXM INTERRUPT TEST
;    2886  1     !
;    2887  1     !  DESCRIPTION:
;    2888  1     !
;    2889  1     !  NOTE: THIS IS IMPORTANT,,, PLEASE READ IT !!!!
;    2890  1     !
;    2891  1     !        This test is only run if selected by operator changing software
;    2892  1     !       defaults. This test should only be run if there is less than 4MB
;    2893  1     !       of memory in a '22 bit backplane system' or less than 256KB in an
;    2894  1     !       '18 bit backplane system'. This test sets up the DEQNA to transmit
;    2895  1     !       from a buffer at 17760000 (the most significant bits are truncated
;    2896  1     !       by the number of wires physically present in the backplane). If
;    2897  1     !       physical memory exists at that address, the test will fail. This
;    2898  1     !       test should not be selected if the maximum amount of memory is
;    2899  1     !       present in your system !!!!!
;    2900  1     !
;    2901  1     !
;    2902  1     !       This test verifies that Transmit and Receive List Invalid bits
;    2903  1     !       (CSR bits 4 and 5) can be set and reset as specified and that both,
;    2904  1     !       Transmit and Receive Descriptor List addresses in the I/O page have
;    2905  1     !       to be valid to succesfully loopback a packet.
;    2906  1     !
;    2907  1     !       After a software reset Transmit and Receive List Invalid bits are
;    2908  1     !       checked for their initial condition state (both set). Then these bits
;    2909  1     !       are cleared by writing Transmit and Receive Descriptor List addresses
;    2910  1     !       into Transmit and Receive Buffer Descriptor Registers.
;    2911  1     !
;    2912  1     !       First, valid loopback packet is sent to verify that UUT properly
;    2913  1     !       transmits and receives loopback packets. Then, a Non-Existant
;    2914  1     !       Memory  Access ( NI ) bit is forced to " 1 " each time an invalid
;    2915  1     !       loopback packet is sent.
;    2916  1     !
;    2917  1     !       If the operator specifies loop on error, the program re-executes the
;    2918  1     !       code that detected the error until ^C is entered.
;    2919  1     !
;    2920  1     !       Hardware tested:        Q-Bus to QTDC interface
;    2921  1     !                                       - Valid and invalid host memory
;    2922  1     !                                         address processing
;    2923  1     !                               CSR register - NXM access (bit 2)
;    2924  1     !                                       - Interrupt Enable (bit 6)
;    2925  1     !                                       - XMIT List Invalid (bit 4)
;    2926  1     !                                       - RCV  List Invalid (bit 5)
;    2927  1     !
;    2928  1     !       Use following Descriptor List and buffer addresses:
;    2929  1     !
;    2930  1     !
;    2931  1     !               TRANSMIT                                RECEIVE
;    2932  1     !               ********                                *******
;    2933  1     !
```

```
;   2934  1      !    DESCR LIST ADR     BUFFER ADR          DESCR LIST ADR     BUFFER ADR
;   2935  1      !    ------ ----- ---- ------ ---          -- -----------  -- -----------
;   2936  1      !
;   2937  1      !        VALID            VALID                VALID            VALID
;   2938  1      !        INVALID          DON'T CARE           DON'T CARE       DON'T CARE
;   2939  1      !        VALID            INVALID              DON'T CARE       DON'T CARE
;   2940  1      !        VALID            VALID                INVALID          DON'T CARE
;   2941  1      !        VALID            VALID                VALID            INVALID
;   2942  1      !
;   2943  1      !     ----- ---- --------- -------------- --------------------------------
```

K12


```
;    2944  1     !
;    2945  1     !       Processing:
;    2946  1     !
;    2947  1     !            BEGIN
;    2948  1     !               reset device ( disables device interrupt )
;    2949  1     !               select internal loopback mode
;    2950  1     !               read CSR
;    2951  1     !               IF XMIT and RCV List Invalid bits not = 1
;    2952  1     !               THEN
;    2953  1     !                  print error message if not inhibited
;    2954  1     !               ENDIF
;    2955  1     !               enable device interrupt (set CSR bit 6)
;    2956  1     !               transmit valid loopback packet
;    2957  1     !               check for expected loopback status
;    2958  1     !               IF error
;    2959  1     !               THEN
;    2960  1     !                  print error message if not inhibited
;    2961  1     !               ENDIF
;    2962  1     !               call compare_packets
;    2963  1     !               REPEAT for each set of addresses in the set
;    2964  1     !                  transmit invalid loopback packet
;    2965  1     !                  IF NXM interrupt didn't occured
;    2966  1     !                  THEN
;    2967  1     !                     print error message if not inhibited
;    2968  1     !                  ENDIF
;    2969  1     !                  check for expected loopback status
;    2970  1     !                  IF error
;    2971  1     !                  THEN
;    2972  1     !                     print error message if not inhibited
;    2973  1     !                  ENDIF
;    2974  1     !               ENDREPEAT
;    2975  1     !            END
;    2976  1     ! -
```

L 2


```
;  2977  3     BGNTST;
;  2978  3
;  2979  3     !++
;  2980  3     !  RESET DEQNA AND SELECT LOOPBACK MODE
;  2981  3     !--
;  2982  3
;  2983  3     RESET_DEQNA ( );
;  2984  3
;  2985  3     f .swp_nxm eqlu yes                    !only run test if operator selected
;  2986  3     then
;  2987  4     begin
;  2988  4
;  2989  4     SETVEC ( .HWP_TABLE [ VEC ], QNA_INT, PRI07 );  ! SET UP FOR QNA INTERRUPT
;  2990  4     PREP_FOR_SETUP ( );
;  2991  4     INCR INDEX FROM 1 TO 14 DO
;  2992  4       WRT_STATION_ADR ( .INDEX, PHA_INDEX );
;  2993  4
;  2994  6     BGNSUB;
;  2995  6       XMIT_SETUP_PACKET ( N_MODE );
;  2996  4     ENDSUB;
;  2997  4
;  2998  4     RBUF_LENGTH = 6;
;  2999  4     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  3000  4
;  3001  4     CLR_BUFFERS ( B_SIZE );
;  3002  4     ERR_NUMBER = ZERO;
;  3003  4
;  3004  4     !++
;  3005  4     !  LOOPBACK A PACKET, VALID DESCRIPTORS AND BUFFER ADDRESSES, THEN CHECK IF
;  3006  4     !  LOOPBACK PACKET WAS PROPERLY RECEIVED AND NI BIT IN CSR = 0
;  3007  4     !--
;  3008  4
;  3009  4     RESET_DEQNA ( );
;  3010  4     WRT_STATION_ADR ( ZERO, PHA_INDEX );
;  3011  4
;  3012  6     BGNSUB;
;  3013  6       XMIT_ILOOP_PACKET ( ZERO );
;  3014  6       IF GET_BIT ( CSR, NI )
;  3015  6         THEN
;  3016  7           BEGIN
;  3017  7             CSR_WORD = GET_BIT ( CSR_ALL );
;  3018  7             PRINTB ( MSG59 );
;  3019  7             PRINTB ( MSG29 );
;  3020  7             PRINTB ( MSG28 );
;  3021  7             ERRDF  ( 1201, MSG00, ERROR$REPORT );
;  3022  6           END;
;  3023  4     ENDSUB;
;  3024  4
;  3025  4     !++
;  3026  4     !  TRY TO LOOPBACK A PACKET WITH INVALID TRANSMIT DESCRIPTOR ADDRESS,
;  3027  4     !  THEN CHECK FOR NON-EXISTANT MEMORY INTERRUPT ( NI ) BIT IS SET TO 1
;  3028  4     !--
;  3029  4
```

M12


```
;    3030  6      BGNSUB;
;    3031  6        RESET_DEQNA ( );
;    3032  6        .IOP_TABLE [ XLO_ADR ] = NXM_LO_ADR;
;    3033  6        .IOP_TABLE [ XHI_ADR ] = NXM_HI_ADR;
;    3034  6        IF NOT GET_BIT ( CSR, NI )
;    3035  6          THEN
;    3036  6            IF ( .XMIT_D_LIST [ FLGWD ] AND XFLG_MASK ) NEQU XFLG_MASK
;    3037  6              THEN
;    3038  7                BEGIN
;    3039  7                  CSR_WORD = GET_BIT ( CSR_ALL );
;    3040  7                  PRINTB ( MSG59 );
;    3041  7                  PRINTB ( MSG29 );
;    3042  7                  PRINTB ( MSG27 );
;    3043  7                  ERRDF  ( 1202, MSG00, E1$REPORT );
;    3044  6                END;
;    3045  4      ENDSUB;
;    3046  4
;    3047  4      !++
;    3048  4      !  TRY TO LOOPBACK A PACKET WITH INVALID RECEIVE DESCRIPTOR ADDRESS,
;    3049  4      !  THEN CHECK IF NON-EXISTANT MEMORY INTERRUPT ( NI ) BIT IS SET TO 1
;    3050  4      !--
;    3051  4
;    3052  6      BGNSUB;
;    3053  6        RESET_DEQNA ( );
;    3054  6        WRT_STATION_ADR ( ZERO, PHA_INDEX );
;    3055  6
;    3056  6        .IOP_TABLE [ RLO_ADR ] = NXM_LO_ADR;
;    3057  6        .IOP_TABLE [ RHI_ADR ] = NXM_HI_ADR;
;    3058  6
;    3059  6        SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    3060  6        .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;    3061  6        .IOP_TABLE [ XHI_ADR ] = ZERO;
;    3062  6
;    3063  6        CHK_RIXI_STATUS ( ONE );
;    3064  6
;    3065  6        CHK_CSR_STATUS  ( %O'000220', %O'000220' );
;    3066  6        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );        ! O'140000', O'000400'
;    3067  6
;    3068  6        .IOP_TABLE [ CSR ] = EENABLE;
;    3069  6
;    3070  6        DELAY ( 20 );
;    3071  6        IF NOT GET_BIT ( CSR, NI )
;    3072  6          THEN
;    3073  6            IF ( .RCV_D_LIST [ FLGWD ] AND RFLG_MASK ) NEQU RFLG_MASK
;    3074  6              THEN
;    3075  7                BEGIN
;    3076  7                  .IOP_TABLE [ CSR ] = DISABLE;
;    3077  7                  CSR_WORD = GET_BIT ( CSR_ALL );
;    3078  7                  PRINTB ( MSG59 );
;    3079  7                  PRINTB ( MSG29 );
;    3080  7                  PRINTB ( MSG27 );
;    3081  7                  ERRDF  ( 1203, MSG00, E1$REPORT );
;    3082  6                END;
```

```
;    3083  6        .IOP_TABLE [ CSR ] = DISABLE;
;    3084  4      ENDSUB;
;    3085  4
;    3086  4      !++
;    3087  4      !  TRY TO LOOPBACK A PACKET WITH INVALID TRANSMIT BUFFER ADDRESS,
;    3088  4      !  THEN CHECK IF NON-EXISTANT MEMORY INTERRUPT ( NI ) BIT IS SET TO 1
;    3089  4      !
;    3090  4
;    3091  6      BGNSUB;
;    3092  6        RESET_DEQNA ( );
;    3093  6        SET_XDESCR_LIST ( .XBUF_LENGTH, VENXM );
;    3094  6        XMIT_D_LIST [ LOADR ] = NXM_LO_ADR;
;    3095  6        .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;    3096  6        .IOP_TABLE [ XHI_ADR ] = ZERO;
;    3097  6        DELAY ( 20 );
;    3098  6        IF NOT GET_BIT ( CSR, NI )
;    3099  6          THEN
;    3100  6            IF ( .XMIT_D_LIST [ FLGWD ] AND XFLG_MASK ) NEQU XFLG_MASK
;    3101  6              THEN
;    3102  7                BEGIN
;    3103  7                  CSR_WORD = GET_BIT ( CSR_ALL );
;    3104  7                  PRINTB ( MSG59 );
;    3105  7                  PRINTB ( MSG29 );
;    3106  7                  PRINTB ( MSG27 );
;    3107  7                  ERRDF  ( 1204, MSG00, ERROR$REPORT );
;    3108  6                END;
;    3109  4      ENDSUB;
;    3110  4
;    3111  4      !++
;    3112  4      !  TRY TO LOOPBACK A PACKET WITH INVALID RECEIVE BUFFER ADDRESS,
;    3113  4      !  THEN CHECK IF NON-EXISTANT MEMORY INTERRUPT ( NI ) BIT IS SET TO 1
;    3114  4      !--
;    3115  4
;    3116  6      BGNSUB;
;    3117  6        RESET_DEQNA ( );
;    3118  6
;    3119  6        SET_RDESCR_LIST ( .XBUF_LENGTH, VENXM );
;    3120  6        RCV_D_LIST [ LOADR ] = NXM_LO_ADR;
;    3121  6        .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;    3122  6        .IOP_TABLE [ RHI_ADR ] = ZERO;
;    3123  6
;    3124  6        SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    3125  6        .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;    3126  6        .IOP_TABLE [ XHI_ADR ] = ZERO;
;    3127  6
;    3128  6        CHK_RIXI_STATUS ( ONE );
;    3129  6
;    3130  6        CHK_CSR_STATUS  ( %O'000220', %O'000220' );
;    3131  6        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );        ! O'140000', O'000400'
;    3132  6
;    3133  6        .IOP_TABLE [ CSR ] = EENABLE;
;    3134  6
;    3135  6        DELAY ( 20 );
```

```
;  3136  6        IF NOT GET_BIT ( CSR, NI )
;  3137  6          THEN
;  3138  6            IF ( .RCV_D_LIST [ FLGWD ] AND RFLG_MASK ) NEQU RFLG_MASK
;  3139  6              THEN
;  3140  7                BEGIN
;  3141  7                  CSR_WORD = GET_BIT ( CSR_ALL );
;  3142  7                  .IOP_TABLE [ CSR ] = DISABLE;
;  3143  7                  PRINTB ( MSG59 );
;  3144  7                  PRINTB ( MSG29 );
;  3145  7                  PRINTB ( MSG27 );
;  3146  7                  ERRDF  ( 1205, MSG00, ERROR$REPORT );
;  3147  6                END;
;  3148  6        .IOP_TABLE [ CSR ] = DISABLE;
;  3149  4      ENDSUB;
;  3150  4
;  3151  3       end;
;  3152  3
;  3153  1      ENDTST;


                                                .SBTTL  $T12 TEST 12 - NXM INTERRUPT TEST
000000  010146                          $T12:   MOV     R1,-(SP)                        ;                               2879
000002  162706  000026                          SUB     #26,SP
000006  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                               2983
000012  023727  000000G 000001                  CMP     SWP.NXM,#1                      ;                               2985
000020  001402                                  BEQ     1$
000022  000137  011234'                         JMP     26$
000026  012746  000000G                 1$:     MOV     #PRI07,-(SP)                    ;                               2989
000032  012746  000000G                         MOV     #QNA.INT,-(SP)
000036  013700  000000G                         MOV     HWP.TABLE,R0
000042  016046  000002                          MOV     2(R0),-(SP)
000046  012746  000003                          MOV     #3,-(SP)
000052  104437                                  TRAP    37
000054  004737  000000G                         JSR     PC,PREP.FOR.SETUP               ;                               2990
000060  012701  000001                          MOV     #1,R1                           ; *,INDEX                       2991
000064  010116                          2$:     MOV     R1,(SP)                         ; INDEX,*                       2992
000066  012746  000023                          MOV     #23,-(SP)
000072  004737  000000G                         JSR     PC,WRT.STATION.ADR
000076  005726                                  TST     (SP)+
000100  005201                                  INC     R1                              ; INDEX                         2991
000102  020127  000016                          CMP     R1,#16                          ; INDEX,*
000106  003766                                  BLE     2$
000110  104402                          3$:     TRAP    2                               ;                               2992
000112  012716  000200                          MOV     #200,(SP)                       ;                               2995
000116  004737  000000G                         JSR     PC,XMIT.SETUP.PACKET
000122  104467                                  TRAP    67
000124  006000                                  ROR     R0
000126  103770                                  BLO     3$
000130  012737  000006  000000G                 MOV     #6,RBUF.LENGTH                  ;                               2998
000136  012700  000006                          MOV     #6,R0                           ;                               2999
000142  006200                                  ASR     R0
000144  005400                                  NEG     R0
000146  010037  000000G                         MOV     R0,XBUF.LENGTH
```

C13


```
000152  012716  004000                          MOV     #4000,(SP)              ;                                       3001
000156  004737  000000G                         JSR     PC,CLR.BUFFERS
000162  005037  000000G                         CLR     ERR.NUMBER              ;                                       3002
000166  004737  000000G                         JSR     PC,RESET.DEQNA          ;                                       3009
000172  005016                                  CLR     (SP)                    ;                                       3010
000174  012746  000023                          MOV     #23,-(SP)
000200  004737  000000G                         JSR     PC,WRT.STATION.ADR
000204  104402                          4$:     TRAP    2
000206  005016                                  CLR     (SP)                    ;                                       3013
000210  004737  000000G                         JSR     PC,XMIT.ILOOP.PACKET
000214  013700  000000G                         MOV     REG.ADR,R0              ;                                       3014
000220  016066  000016  000012                  MOV     16(R0),12(SP)           ; *,TMP.LOCATION
000226  032766  000004  000012                  BIT     #4,12(SP)               ; *,TMP.LOCATION
000234  001436                                  BEQ     5$
000236  016666  000012  000014                  MOV     12(SP),14(SP)           ; *,TMP.LOCATION                        3017
000244  016637  000014  000000G                 MOV     14(SP),CSR.WORD         ; TMP.LOCATION,*
000252  012716  000000G                         MOV     #MSG59,(SP)             ;                                       3018
000256  012746  000001                          MOV     #1,-(SP)
000262  010600                                  MOV     SP,R0                   ; SP,*
000264  104414                                  TRAP    14
000266  012716  000000G                         MOV     #MSG29,(SP)             ;                                       3019
000272  012746  000001                          MOV     #1,-(SP)
000276  010600                                  MOV     SP,R0                   ; SP,*
000300  104414                                  TRAP    14
000302  012716  000000G                         MOV     #MSG28,(SP)             ;                                       3020
000306  012746  000001                          MOV     #1,-(SP)
000312  010600                                  MOV     SP,R0                   ; SP,*
000314  104414                                  TRAP    14
000316  104455                                  TRAP    55                      ;                                       3021
000320  002261                                  .WORD   2261
000322  000000G                                 .WORD   MSG00
000324  000000G                                 .WORD   ERROR$REPORT
000326  062706  000006                          ADD     #6,SP                   ;                                       3016
000332  104467                          5$:     TRAP    67                      ;                                       3022
000334  006000                                  ROR     R0
000336  103722                                  BLO     4$
000340  104402                          6$:     TRAP    2                       ;                                       3023
000342  004737  000000G                         JSR     PC,RESET.DEQNA          ;                                       3031
000346  012777  160000  000010G                 MOV     #-20000,@IOP.TABLE+10   ;                                       3032
000354  012777  000077  000012G                 MOV     #77,@IOP.TABLE+12       ;                                       3033
000362  013700  000000G                         MOV     REG.ADR,R0              ;                                       3034
000366  016066  000016  000016                  MOV     16(R0),16(SP)           ; *,TMP.LOCATION
000374  032766  000004  000016                  BIT     #4,16(SP)               ; *,TMP.LOCATION
000402  001045                                  BNE     7$
000404  013701  000000G                         MOV     XMIT.D.LIST,R1          ;                                       3036
000410  042701  037777                          BIC     #37777,R1
000414  020127  140000                          CMP     R1,#-40000
000420  001436                                  BEQ     7$
000422  016666  000016  000020                  MOV     16(SP),20(SP)           ; *,TMP.LOCATION                        3039
000430  016637  000020  000000G                 MOV     20(SP),CSR.WORD         ; TMP.LOCATION,*
000436  012716  000000G                         MOV     #MSG59,(SP)             ;                                       3040
000442  012746  000001                          MOV     #1,-(SP)
000446  010600                                  MOV     SP,R0                   ; SP,*
```

D13


```
000450  104414                                  TRAP    14
000452  012716  000000G                         MOV     #MSG29,(SP)             ;                                       3041
000456  012746  000001                          MOV     #1,-(SP)
000462  010600                                  MOV     SP,R0                   ; SP,*
000464  104414                                  TRAP    14
000466  012716  000000G                         MOV     #MSG27,(SP)             ;                                       3042
000472  012746  000001                          MOV     #1,-(SP)
000476  010600                                  MOV     SP,R0                   ; SP,*
000500  104414                                  TRAP    14
000502  104455                                  TRAP    55                      ;                                       3043
000504  002262                                  .WORD   2262
000506  000000G                                 .WORD   MSG00
000510  000000G                                 .WORD   E1$REPORT
000512  062706  000006                          ADD     #6,SP                   ;                                       3038
000516  104467                          7$:     TRAP    67                      ;                                       3044
000520  006000                                  ROR     R0
000522  103706                                  BLO     6$
000524  104402                          8$:     TRAP    2                       ;                                       3045
000526  004737  000000G                         JSR     PC,RESET.DEQNA          ;                                       3053
000532  005016                                  CLR     (SP)                    ;                                       3054
000534  012746  000023                          MOV     #23,-(SP)
000540  004737  000000G                         JSR     PC,WRT.STATION.ADR
000544  012777  160000  000004G                 MOV     #-20000,@IOP.TABLE+4    ;                                       3056
000552  012777  000077  000006G                 MOV     #77,@IOP.TABLE+6        ;                                       3057
000560  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                                       3059
000564  012746  120000                          MOV     #-60000,-(SP)
000570  004737  000000G                         JSR     PC,SET.XDESCR.LIST
000574  012777  000000G 000010G                 MOV     #XMIT.D.LIST,@IOP.TABLE+10  ;                                   3060
000602  005077  000012G                         CLR     @IOP.TABLE+12           ;                                       3061
000606  012716  000001                          MOV     #1,(SP)                 ;                                       3063
000612  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
000616  012716  000220                          MOV     #220,(SP)               ;                                       3065
000622  011646                                  MOV     (SP),-(SP)
000624  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000630  012716  140000                          MOV     #-40000,(SP)            ;                                       3066
000634  012746  000400                          MOV     #400,-(SP)
000640  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
000644  012777  000001  000016G                 MOV     #1,@IOP.TABLE+16        ;                                       3068
000652  012701  000024                          MOV     #24,R1                  ; *,$$TMP2                              3070
000656  001410                          9$:     BEQ     12$
000660  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000664  001403                                  BEQ     11$
000666  005066  000046                  10$:    CLR     46(SP)                  ; $$TMP
000672  077003                                  SOB     R0,10$                  ; $$TMP1,*
000674  005301                          11$:    DEC     R1                      ; $$TMP2
000676  000767                                  BR      9$
000700  013700  000000G                 12$:    MOV     REG.ADR,R0              ;                                       3071
000704  016066  000016  000032                  MOV     16(R0),32(SP)           ; *,TMP.LOCATION
000712  032766  000004  000032                  BIT     #4,32(SP)               ; *,TMP.LOCATION
000720  001047                                  BNE     13$
000722  013701  000000G                         MOV     RCV.D.LIST,R1           ;                                       3073
000726  042701  037777                          BIC     #37777,R1
000732  020127  140000                          CMP     R1,#-40000
```

E13


```
000736  001440                              BEQ     13$
000740  005077  000016G                     CLR     @IOP.TABLE+16                   ;                            3076
000744  016666  000032  000034              MOV     32(SP),34(SP)                   ; *,TMP.LOCATION             3077
000752  016637  000034  000000G             MOV     34(SP),CSR.WORD                 ; TMP.LOCATION,*
000760  012716  000000G                     MOV     #MSG59,(SP)                     ;                            3078
000764  012746  000001                      MOV     #1,-(SP)
000770  010600                              MOV     SP,R0                           ; SP,*
000772  104414                              TRAP    14
000774  012716  000000G                     MOV     #MSG29,(SP)                     ;                            3079
001000  [illegible]16  000001               MOV     #1,-(SP)
001004  01[illegible]00                     MOV     SP,R0                           ; SP,*
001006  104414                              TRAP    14
001010  012716  000000G                     MOV     #MSG27,(SP)                     ;                            3080
001014  012746  000001                      MOV     #1,-(SP)
001020  010600                              MOV     SP,R0                           ; SP,*
001022  104414                              TRAP    14
001024  104455                              TRAP    55                              ;                            3081
001026  002263                              .WORD   2263
001030  000000G                             .WORD   MSG00
001032  000000G                             .WORD   E1$REPORT
001034  062706  000006                      ADD     #6,SP                           ;                            3075
001040  005077  000016G             13$:    CLR     @IOP.TABLE+16                   ;                            3083
001044  062706  000010                      ADD     #10,SP                          ;                            3045
001050  104467                              TRAP    67                              ;                            3083
001052  006000                              ROR     R0
001054  103623                              BLO     8$
001056  104402                      14$:    TRAP    2                               ;                            3084
001060  004737  000000G                     JSR     PC,RESET.DEQNA                  ;                            3092
001064  013716  000000G                     MOV     XBUF.LENGTH,(SP)                ;                            3093
001070  012746  120077                      MOV     #-57701,-(SP)
001074  004737  000000G                     JSR     PC,SET.XDESCR.LIST
001100  012737  160000  000004G             MOV     #-20000,XMIT.D.LIST+4           ;                            3094
001106  012777  000000G 000010G             MOV     #XMIT.D.LIST,@IOP.TABLE+10      ;                            3095
001114  005077  000012G                     CLR     @IOP.TABLE+12                   ;                            3096
001120  012701  000024                      MOV     #24,R1                          ; *,$$TMP2                   3097
001124  001410                      15$:    BEQ     18$
001126  013700  000000G                     MOV     L$DLY,R0                        ; *,$$TMP1
001132  001403                              BEQ     17$
001134  005066  000040              16$:    CLR     40(SP)                          ; $$TMP
001140  077003                              SOB     R0,16$                          ; $$TMP1,*
001142  005301                      17$:    DEC     R1                              ; $$TMP2
001144  000767                              BR      15$
001146  013700  000000G             18$:    MOV     REG.ADR,R0                      ;                            3098
001152  016066  000016  000030              MOV     16(R0),30(SP)                   ; *,TMP.LOCATION
001160  032766  000004  000030              BIT     #4,30(SP)                       ; *,TMP.LOCATION
001166  001045                              BNE     19$
001170  013701  000000G                     MOV     XMIT.D.LIST,R1                  ;                            3100
001174  042701  037777                      BIC     #37777,R1
001200  020127  140000                      CMP     R1,#-40000
001204  001436                              BEQ     19$
001206  016666  000030  000032              MOV     30(SP),32(SP)                   ; *,TMP.LOCATION             3103
001214  016637  000032  000000G             MOV     32(SP),CSR.WORD                 ; TMP.LOCATION,*
001222  012716  000000G                     MOV     #MSG59,(SP)                     ;                            3104
```

F13


```
001226  012746  000001                          MOV     #1,-(SP)
001232  010600                                  MOV     SP,R0                   ; SP,*
001234  104414                                  TRAP    14
001236  012716  000000G                         MOV     #MSG29,(SP)             ;                       3105
001242  012746  000001                          MOV     #1,-(SP)
001246  010600                                  MOV     SP,R0                   ; SP,*
001250  104414                                  TRAP    14
001252  012716  000000G                         MOV     #MSG27,(SP)             ;                       3106
001256  012746  000001                          MOV     #1,-(SP)
001262  010600                                  MOV     SP,R0                   ; SP,*
001264  104414                                  TRAP    14
001266  104455                                  TRAP    55                      ;                       3107
001270  002264                                  .WORD   2264
001272  000000G                                 .WORD   MSG00
001274  000000G                                 .WORD   ERROR$REPORT
001276  062706  000006                          ADD     #6,SP                   ;                       3102
001302  005726                          19$:    TST     (SP)+                   ;                       3084
001304  104467                                  TRAP    67                      ;                       3108
001306  006000                                  ROR     R0
001310  103662                                  BLO     14$
001312  104402                          20$:    TRAP    2                       ;                       3109
001314  004737  000000G                         JSR     PC,RESET.DEQNA          ;                       3117
001320  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                       3119
001324  012746  120077                          MOV     #-57701,-(SP)
001330  004737  000000G                         JSR     PC,SET.RDESCR.LIST
001334  012737  160000  000004G                 MOV     #-20000,RCV.D.LIST+4    ;                       3120
001342  012777  000000G 000004G                 MOV     #RCV.D.LIST,@IOP.TABLE+4 ;                      3121
001350  005077  000006G                         CLR     @IOP.TABLE+6            ;                       3122
001354  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                       3124
001360  012746  120000                          MOV     #-60000,-(SP)
001364  004737  000000G                         JSR     PC,SET.XDESCR.LIST
001370  012777  000000G 000010G                 MOV     #XMIT.D.LIST,@IOP.TABLE+10 ;                    3125
001376  005077  000012G                         CLR     @IOP.TABLE+12           ;                       3126
001402  012716  000001                          MOV     #1,(SP)                 ;                       3128
001406  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
001412  012716  000220                          MOV     #220,(SP)               ;                       3130
001416  011646                                  MOV     (SP),-(SP)
001420  004737  000000G                         JSR     PC,CHK.CSR.STATUS
001424  012716  140000                          MOV     #-40000,(SP)            ;                       3131
001430  012746  000400                          MOV     #400,-(SP)
001434  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
001440  012777  000001  000016G                 MOV     #1,@IOP.TABLE+16        ;                       3133
001446  012701  000024                          MOV     #24,R1                  ; *,$$TMP2              3135
001452  001410                          21$:    BEQ     24$
001454  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
001460  001403                                  BEQ     23$
001462  005066  000046                  22$:    CLR     46(SP)                  ; $$TMP
001466  077003                                  SOB     R0,22$                  ; $$TMP1,*
001470  005301                          23$:    DEC     R1                      ; $$TMP2
001472  000767                                  BR      21$
001474  013700  000000G                 24$:    MOV     REG.ADR,R0              ;                       3136
001500  016066  000016  000042                  MOV     16(R0),42(SP)           ; *,TMP.LOCATION
001506  032766  000004  000042                  BIT     #4,42(SP)               ; *,TMP.LOCATION
```

G13


```
001514  001047                                  BNE     25$
001516  013701  000000G                         MOV     RCV.D.LIST,R1           ;                                       3138
001522  042701  037777                          BIC     #37777,R1
001526  020127  140000                          CMP     R1,#-40000
001532  001440                                  BEQ     25$
001534  016666  000042  000044                  MOV     42(SP),44(SP)           ; *,TMP.LOCATION                        3141
001542  016637  000044  000000G                 MOV     44(SP),CSR.WORD         ; TMP.LOCATION,*
001550  005077  000016G                         CLR     @IOP.TABLE+16           ;                                       3142
001554  012716  000000G                         MOV     #MSG59,(SP)             ;                                       3143
001560  012746  000001                          MOV     #1,-(SP)
001564  010600                                  MOV     SP,R0                   ; SP,*
001566  104414                                  TRAP    14
001570  012716  000000G                         MOV     #MSG29,(SP)             ;                                       3144
001574  012746  000001                          MOV     #1,-(SP)
001600  010600                                  MOV     SP,R0                   ; SP,*
001602  104414                                  TRAP    14
001604  012716  000000G                         MOV     #MSG27,(SP)             ;                                       3145
001610  012746  000001                          MOV     #1,-(SP)
001614  010600                                  MOV     SP,R0                   ; SP,*
001616  104414                                  TRAP    14
001620  104455                                  TRAP    55                      ;                                       3146
001622  002265                                  .WORD   2265
001624  000000G                                 .WORD   MSG00
001626  000000G                                 .WORD   ERROR$REPORT
001630  062706  000006                          ADD     #6,SP                   ;                                       3140
001634  005077  000016G         25$:            CLR     @IOP.TABLE+16           ;                                       3148
001640  062706  000010                          ADD     #10,SP                  ;                                       3109
001644  104467                                  TRAP    67                      ;                                       3148
001646  006000                                  ROR     R0
001650  103620                                  BLO     20$
001652  062706  000012                          ADD     #12,SP                  ;                                       2987
001656  062706  000026         26$:            ADD     #26,SP                  ;                                       2879
001662  012601                                  MOV     (SP)+,R1
001664  000207                                  RTS     PC

; Routine Size:  475 words,     Routine Base:  AB$CODE$ + 7356
; Maximum stack depth per invocation:  26 words
```

H13


```
                                                .SBTTL  T12 TEST 12   NXM INTERRUPT TEST

000000  004737  007356'                    T12::
000000                                      1$:     JSR     PC,$T12                         ;                                       3151
000004  104466                                      TRAP    66
000006  006000                                      ROR     R0
000010  103773                                      BLO     1$
000012  000207                                      RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 11244
; Maximum stack depth per invocation:  2 words


;   3154  1
;   3155  1
```

```
;  3156  1
;  3157  1     %SBTTL 'TEST 13 - MULTIPLE AND CHAINED PACKET TEST'
;  3158  1     !++
;  3159  1     !
;  3160  1     !  TEST 13:     MULTIPLE AND CHAINED PACKET TEST
;  3161  1     !
;  3162  1     !  DESCRIPTION:
;  3163  1     !
;  3164  1     !       This test verifies that the DEQNA can transmit and receive multiple,
;  3165  1     !       linked and chained loopback packets.
;  3166  1     !
;  3167  1     !       If the operator specifies loop on error, the program re-executes the
;  3168  1     !       code that detected the  error until ^C is entered.
;  3169  1     !
;  3170  1     !       Hardware tested:
;  3171  1     !
;  3172  1     !       Processing:
;  3173  1     !
;  3174  1     !            BEGIN
;  3175  1     !               reset device
;  3176  1     !               select internal/extended loopback mode
;  3177  1     !               transmit simple loopback packet
;  3178  1     !               check for expected loopback status
;  3179  1     !               IF error
;  3180  1     !               THEN
;  3181  1     !                  print error message if not inhibited
;  3182  1     !               ENDIF
;  3183  1     !               call compare_packets
;  3184  1     !
;  3185  1     !               transmit multiple, linked and chaind loopback packet
;  3186  1     !               check for expected loopback status
;  3187  1     !               IF error
;  3188  1     !               THEN
;  3189  1     !                  print error message if not inhibited
;  3190  1     !               ENDIF
;  3191  1     !               call compare packets
;  3192  1     !            END
;  3193  1     !-
```

```
;  3194  3     BGNTST;
;  3195  3
;  3196  3     RBUF_LENGTH = 64;
;  3197  3     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  3198  3
;  3199  3     !++
;  3200  3     !  LOOPBACK UNCHAINED PACKET, THEN CHECK IF IT WAS PROPERLY RECEIVED
;  3201  3     !--
;  3202  3
;  3203  3     RESET_DEQNA ( );
;  3204  3     INCR INDEX FROM 0 TO 63 DO
;  3205  3       XMIT_BUFFER [ .INDEX ] = .INDEX;
;  3206  3
;  3207  5     BGNSUB;
;  3208  5       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  3209  5       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  3210  5       SEND_ELOOP_PACKET ( ZERO );
;  3211  5       COMPARE_PACKETS ( );
;  3212  3     ENDSUB;
;  3213  3
;  3214  3     RESET_DEQNA ( );
;  3215  3     CLR_BUFFERS ( 512 );
;  3216  3     INCR INDEX FROM 0 TO 383 DO
;  3217  3       XMIT_BUFFER [ .INDEX ] = .INDEX;
;  3218  3
;  3219  3
;  3220  5     BGNSUB;
;  3221  5       INCR INDEX FROM 0 TO 63 DO
;  3222  5         RCV_D_LIST [ .INDEX, W_LEN ] = .RD13 [ .INDEX ];
;  3223  5       INCR INDEX FROM 0 TO 31 DO
;  3224  5         XMIT_D_LIST [ .INDEX, W_LEN ] = .TD13 [ .INDEX ];
;  3225  5
;  3226  5       XMIT_D_LIST [  7, W_LEN ] = VE;                 !modify what came from td13
;  3227  5       XMIT_D_LIST [ 13, W_LEN ] = E;                  !this was here, comments added E
;  3228  5
;  3229  5       PUT_BIT [ CSR, LB, INX_LOOPBACK ];
;  3230  5       XMIT_AND_RCV_PACKET ( );
;  3231  5       CHK_RIXI_STATUS ( ZERO );
;  3232  5       CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );       ! 0'100220', 0'100220'
;  3233  5
;  3234  5       XMIT_D_LIST [  7, W_LEN ] = V;                  !this changes already used
;  3235  5       XMIT_D_LIST [ 12, W_LEN ] = NEWB;               !tx desc entries ???
;  3236  5       XMIT_D_LIST [ 13, W_LEN ] = V;                  !comments added rev E
;  3237  5
;  3238  5       .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST + 24;      !this forces the qna to do
;  3239  5       .IOP_TABLE [ XHI_ADR ] = ZERO;                  !another xmit ??!!
;  3240  5
;  3241  5       CHK_RIXI_STATUS ( ZERO );
;  3242  5       CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );       ! 0'100220', 0'100220'
;  3243  5
;  3244  5       !++
;  3245  5       !  CHECK IF RECEIVE BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;  3246  5       !--
```

K 13


```
;    3247  5
;    3248  5        INCR INDEX FROM 0 TO 53 DO
;    3249  5          IF .RCV_D_LIST [ .INDEX, W_LEN ] NEQU .RD13 [ .INDEX ]
;    3250  5            AND ( .RCV_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;    3251  5            AND .RCV_D_LIST [ .INDEX, W_LEN ] NEQU %O'020600'
;    3252  5              THEN
;    3253  6                BEGIN
;    3254  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;    3255  6                  PRINTB ( MSG59 );
;    3256  6                  PRINTB ( MSG48 );
;    3257  6                  PRINTB ( msg75, RCV_D_LIST, .INDEX, .RCV_D_LIST [ .INDEX, W_LEN ] );
;    3258  6                  ERRDF  ( 1301, MSG00, ERROR$REPORT );
;    3259  5                END;
;    3260  5
;    3261  5        !++
;    3262  5        ! CHECK IF TRANSMIT BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;    3263  5        !--
;    3264  5
;    3265  5        INCR INDEX FROM 0 TO 23 DO
;    3266  5          IF .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU .TD13 [ .INDEX ]
;    3267  5            AND ( .XMIT_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;    3268  5            AND .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU %O'020414'
;    3269  5            AND .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU %O'004140'
;    3270  5              THEN
;    3271  6                BEGIN
;    3272  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;    3273  6                  PRINTB ( MSG59 );
;    3274  6                  PRINTB ( MSG49 );
;    3275  6                  PRINTB ( msg76, XMIT_D_LIST, .INDEX, .XMIT_D_LIST [ .INDEX, W_LEN ] );
;    3276  6                  ERRDF  ( 1302, MSG00, ERROR$REPORT );
;    3277  5                END;
;    3278  5
;    3279  5        INCR INDEX FROM 0 TO 5 DO
;    3280  6          BEGIN
;    3281  6            XMIT_D_LIST [ .INDEX, W_LEN ] = .XMIT_D_LIST [ .INDEX + 24, W_LEN ];
;    3282  6            RCV_D_LIST  [ .INDEX, W_LEN ] = .RCV_D_LIST [ .INDEX + 54, W_LEN ];
;    3283  5          END;
;    3284  5
;    3285  5        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );  ! 0'140000', 0'000400'
;    3286  5        CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS  );  ! 0'140000', 0'020000'
;    3287  5
;    3288  5        INCR INDEX FROM 0 TO 383 DO
;    3289  5          IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;    3290  5            THEN
;    3291  6              BEGIN
;    3292  6                CSR_WORD = GET_BIT ( CSR_ALL );
;    3293  6                PRINTB ( MSG59 );
;    3294  6                PRINTB ( MSG51 );
;    3295  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;    3296  6                ERRDF  ( 1303, MSG00, ERROR$REPORT );
;    3297  5              END;
;    3298  3    ENDSUB;
;    3299  3
```

L13


```
;   3300  1     ENDTST;


                                                .SBTTL  $T13 TEST 13 - MULTIPLE AND CHAINED PACKET TEST
000000  004137  000000G                  $T13:   JSR     R1,$SAVE3                       ;                                       3153
000004  162706  000006                          SUB     #6,SP
000010  012737  000100  000000G                 MOV     #100,RBUF.LENGTH                ;                                       3196
000016  012700  000100                          MOV     #100,R0                         ;                                       3197
000022  006200                                  ASR     R0
000024  005400                                  NEG     R0
000026  010037  000000G                         MOV     R0,XBUF.LENGTH
000032  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                                       3203
000036  005000                                  CLR     R0                              ; INDEX                                 3204
000040  110060  000000G                  1$:     MOVB    R0,XMIT.BUFFER(R0)              ; INDEX,*(INDEX)                        3205
000044  005200                                  INC     R0                              ; INDEX                                 3204
000046  020027  000077                          CMP     R0,#77                          ; INDEX,*
000052  003772                                  BLE     1$
000054  104402                          2$:     TRAP    2                               ;                                       3205
000056  013746  000000G                         MOV     XBUF.LENGTH,-(SP)               ;                                       3208
000062  012746  120000                          MOV     #-60000,-(SP)
000066  004737  000000G                         JSR     PC,SET.RDESCR.LIST
000072  013716  000000G                         MOV     XBUF.LENGTH,(SP)                ;                                       3209
000076  012746  120000                          MOV     #-60000,-(SP)
000102  004737  000000G                         JSR     PC,SET.XDESCR.LIST
000106  005016                                  CLR     (SP)                            ;                                       3210
000110  004737  000000G                         JSR     PC,SEND.ELOOP.PACKET
000114  004737  000000G                         JSR     PC,COMPARE.PACKETS              ;                                       3211
000120  062706  000006                          ADD     #6,SP                           ;                                       3205
000124  104467                                  TRAP    67                              ;                                       3211
000126  006000                                  ROR     R0
000130  103751                                  BLO     2$
000132  004737  000000G                         JSR     PC,RESET.DEQNA                  ;                                       3214
000136  012746  001000                          MOV     #1000,-(SP)                     ;                                       3215
000142  004737  000000G                         JSR     PC,CLR.BUFFERS
000146  005000                                  CLR     R0                              ; INDEX                                 3216
000150  110060  000000G                  3$:     MOVB    R0,XMIT.BUFFER(R0)              ; INDEX,*(INDEX)                        3217
000154  005200                                  INC     R0                              ; INDEX                                 3216
000156  020027  000577                          CMP     R0,#577                         ; INDEX,*
000162  003772                                  BLE     3$
000164  104402                          4$:     TRAP    2                               ;                                       3217
000166  005000                                  CLR     R0                              ; INDEX                                 3221
000170  016060  000000G 000000G          5$:     MOV     RD13(R0),RCV.D.LIST(R0)         ; *(INDEX),*(INDEX)                     3222
000176  062700  000002                          ADD     #2,R0                           ; *,INDEX                               3221
000202  020027  000176                          CMP     R0,#176                         ; INDEX,*
000206  003770                                  BLE     5$
000210  005000                                  CLR     R0                              ; INDEX                                 3223
000212  016060  000000G 000000G          6$:     MOV     TD13(R0),XMIT.D.LIST(R0)        ; *(INDEX),*(INDEX)                     3224
000220  062700  000002                          ADD     #2,R0                           ; *,INDEX                               3223
000224  020027  000076                          CMP     R0,#76                          ; INDEX,*
000230  003770                                  BLE     6$
000232  012737  120000  000016G                 MOV     #-60000,XMIT.D.LIST+16          ;                                       3226
000240  012737  020000  000032G                 MOV     #20000,XMIT.D.LIST+32           ;                                       3227
000246  013700  000000G                         MOV     REG.ADR,R0                      ;                                       3229
```

M13


```
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          ;                       3230
000272  005016                                  CLR     (SP)                            ;                       3231
000274  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
000300  012716  100220                          MOV     # 77560,(SP)                    ;                       3232
000304  011646                                  MOV     (SP),-(SP)
000306  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000312  012737  100000  000016G                 MOV     #-100000,XMIT.D.LIST+16         ;                       3234
000320  012737  100000  000030G                 MOV     #-100000,XMIT.D.LIST+30         ;                       3235
000326  012737  100000  000032G                 MOV     #-100000,XMIT.D.LIST+32         ;                       3236
000334  012777  000030G 000010G                 MOV     #XMIT.D.LIST+30,@IOP.TABLE+10   ;                       3238
000342  005077  000012G                         CLR     @IOP.TABLE+12                   ;                       3239
000346  005016                                  CLR     (SP)                            ;                       3241
000350  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
000354  012716  100220                          MOV     #-77560,(SP)                    ;                       3242
000360  011646                                  MOV     (SP), (SP)
000362  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000366  0050U3                                  CLR     R3                              ; INDEX                 3248
000370  010301                          7$:     MOV     R3,R1                           ; INDEX,*               3249
000372  006301                                  ASL     R1
000374  016100  000000G                         MOV     RCV.D.LIST(R1),R0
000400  020061  000000G                         CMP     R0,RD13(R1)
000404  001456                                  BEQ     8$
000406  010002                                  MOV     R0,R2                           ;                       3250
000410  042702  037777                          BIC     #37777,R2
000414  020227  140000                          CMP     R2,#-40000
000420  001450                                  BEQ     8$
000422  020027  020600                          CMP     R0,#20600                       ;                       3251
000426  001445                                  BEQ     8$
000430  013700  000000G                         MOV     REG.ADR,R0                      ;                       3254
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)                     ;                       3255
000454  012746  000001                          MOV     #1,-(SP)
000460  010600                                  MOV     SP,R0                           ; SP,*
000462  104414                                  TRAP    14
000464  012716  000000G                         MOV     #MSG48,(SP)                     ;                       3256
000470  012746  000001                          MOV     #1,-(SP)
000474  010600                                  MOV     SP,R0                           ; SP,*
000476  104414                                  TRAP    14
000500  016116  000000G                         MOV     RCV.D.LIST(R1),(SP)             ;                       3257
000504  010346                                  MOV     R3,-(SP)                        ; INDEX,*
000506  012746  000000G                         MOV     #RCV.D.LIST,-(SP)
000512  012746  000000G                         MOV     #MSG75,-(SP)
000516  012746  000004                          MOV     #4,-(SP)
000522  010600                                  MOV     SP,R0                           ; SP,*
000524  104414                                  TRAP    14
000526  104455                                  TRAP    55                              ;                       3258
000530  002425                                  .WORD   2425
000532  000000G                                 .WORD   MSG00
000534  000000G                                 .WORD   ERROR$REPORT
000536  062706  000014                          ADD     #14,SP                          ;                       3253
```

N13


```
000542  005203                          8$:     INC     R3                              ; INDEX                         3248
000544  020327  000065                        CMP     R3,#65                          ; INDEX,*
000550  003707                                BLE     7$
000552  005003                                CLR     R3                              ; INDEX                         3265
000554  010301                          9$:     MOV     R3,R1                           ; INDEX,*                       3266
000556  006301                                ASL     R1
000560  016100  000000G                       MOV     XMIT.D.LIST(R1),R0
000564  020061  000000G                       CMP     R0,TD13(R1)
000570  001461                                BEQ     10$
000572  010002                                MOV     R0,R2                           ;                               3267
000574  042702  037777                        BIC     #37777,R2
000600  020227  140000                        CMP     R2,#-40000
000604  001453                                BEQ     10$
000606  020027  020414                        CMP     R0,#20414                       ;                               3268
000612  001450                                BEQ     10$
000614  020027  004140                        CMP     R0,#4140                        ;                               3269
000620  001445                                BEQ     10$
000622  013700  000000G                       MOV     REG.ADR,R0                      ;                               3272
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)                     ;                               3273
000646  012746  000001                        MOV     #1,-(SP)
000652  010600                                MOV     SP,R0                           ; SP,*
000654  104414                                TRAP    14
000656  012716  000000G                       MOV     #MSG49,(SP)                     ;                               3274
000662  012746  000001                        MOV     #1,-(SP)
000666  010600                                MOV     SP,R0                           ; SP,*
000670  104414                                TRAP    14
000672  016116  000000G                       MOV     XMIT.D.LIST(R1),(SP)            ;                               3275
000676  010346                                MOV     R3,-(SP)                        ; INDEX,*
000700  012746  000000G                       MOV     #XMIT.D.LIST,-(SP)
000704  012746  000000G                       MOV     #MSG76,-(SP)
000710  012746  000004                        MOV     #4,-(SP)
000714  010600                                MOV     SP,R0                           ; SP,*
000716  104414                                TRAP    14
000720  104455                                TRAP    55                              ;                               3276
000722  002426                                .WORD   2426
000724  000000G                               .WORD   MSG00
000726  000000G                               .WORD   ERROR$REPORT
000730  062706  000014                        ADD     #14,SP                          ;                               3271
000734  005203                          10$:    INC     R3                              ; INDEX                         3265
000736  020327  000027                        CMP     R3,#27                          ; INDEX,*
000742  003704                                BLE     9$
000744  005002                                CLR     R2                              ; INDEX                         3279
000746  010200                          11$:    MOV     R2,R0                           ; INDEX,*                       3281
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,*                       3282
000766  006301                                ASL     R1
000770  016160  000154G 000000G               MOV     RCV.D.LIST+154(R1),RCV.D.LIST(R0) ;
000776  005202                                INC     R2                              ; INDEX                         3279
```

```
001000  020227  000005                          CMP     R2,#5                           ; INDEX,*
001004  003760                                  BLE     11$
001006  012716  140000                          MOV     #-40000,(SP)                    ;                                       3285
001012  012746  000400                          MOV     #400,-(SP)
001016  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
001022  012716  140000                          MOV     #-40000,(SP)                    ;                                       3286
001026  012746  020000                          MOV     #20000,-(SP)
001032  004737  000000G                         JSR     PC,CHK.RCV.STATUS
001036  005001                                  CLR     R1                              ; INDEX                                 3288
001040  126161  000000G 000000G         12$:    CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                     3289
001046  001447                                  BEQ     13$
001050  013700  000000G                         MOV     REG.ADR,R0                      ;                                       3292
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)                     ;                                       3293
001074  012746  000001                          MOV     #1,-(SP)
001100  010600                                  MOV     SP,R0                           ; SP,*
001102  104414                                  TRAP    14
001104  012716  000000G                         MOV     #MSG51,(SP)                     ;                                       3294
001110  012746  000001                          MOV     #1,-(SP)
001114  010600                                  MOV     SP,R0                           ; SP,*
001116  104414                                  TRAP    14
001120  010116                                  MOV     R1,(SP)                         ; INDEX,*                               3295
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                              ;                                       3296
001154  002427                                  .WORD   2427
001156  000000G                                 .WORD   MSG00
001160  000000G                                 .WORD   ERROR$REPORT
001162  062706  000014                          ADD     #14,SP                          ;                                       3291
001166  005201                          13$:    INC     R1                              ; INDEX                                 3288
001170  020127  000577                          CMP     R1,#577                         ; INDEX,*
001174  003721                                  BLE     12$
001176  062706  000010                          ADD     #10,SP                          ;                                       3217
001202  104467                                  TRAP    67                              ;                                       3297
001204  006000                                  ROR     R0
001206  103002                                  BHIS    14$
001210  000137  011444'                         JMP     4$
001214  062706  000010                  14$:    ADD     #10,SP                          ;                                       3153
001220  000207                                  RTS     PC

; Routine Size:  329 words,     Routine Base:  AB$CODE$ + 11260
; Maximum stack depth per invocation:  20 words
```

C14


```
                                        .SBTTL  T13 TEST 13 - MULTIPLE AND CHAINED PACKET TEST
000000  004737  011260'                T13::
000000                                  1$:     JSR     PC,$T13                 ;                       3298
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 12502
; Maximum stack depth per invocation:  2 words


;   3301  1
```

```
;  3302  1     %SBTTL 'TEST 14   DMA TIMING TEST'
;  3303  1     !++
;  3304  1     !
;  3305  1     !  TEST 14:     DMA TIMING TEST
;  3306  1     !
;  3307  1     !  DESCRIPTION:
;  3308  1     !
;  3309  1     !       This test verifies that the DMA transfer completes within 'X' msec.
;  3310  1     !       Chained and linked 1514 byte loopback packet is used to accomplish
;  3311  1     !       this test. If the operator specifies loop on error, the program
;  3312  1     !       re-executes the code that detected the error until ^C is entered.
;  3313  1     !
;  3314  1     !       NOTE: An answer to the following software quastion
;  3315  1     !
;  3316  1     !             SYSTEM HAS BLOCK MODE MEMORY (L)?
;  3317  1     !
;  3318  1     !             determines the value for 'X'.
;  3319  1     !
;  3320  1     !       Hardware tested:   Internal/Extended loopback
;  3321  1     !                          Transmit status - last descriptor in chain (bit 15)
;  3322  1     !                          Receive  status - last descriptor in chain (bit 15)
;  3323  1     !                                          - error summary (bit 14)
;  3324  1     !       Processing:
;  3325  1     !
;  3326  1     !            BEGIN
;  3327  1     !               reset device
;  3328  1     !               select internal/extended loopback mode
;  3329  1     !               set the timeout timer to 'X' msec
;  3330  1     !               transmit chained loopback packet
;  3331  1     !               start the timer
;  3332  1     !               IF timeout
;  3333  1     !               THEN
;  3334  1     !                  print error message if not inhibited
;  3335  1     !               ENDIF
;  3336  1     !               check for expected loopback status
;  3337  1     !               IF error
;  3338  1     !               THEN
;  3339  1     !                  print error message if not inhibited
;  3340  1     !               ENDIF
;  3341  1     !               call compare_packets
;  3342  1     !            END
;  3343  1     !--
```

```
;    3344  3      BGNTST;
;    3345  3
;    3346  3      RBUF_LENGTH = LEGAL_LENGTH;
;    3347  3      XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3348  3      INCR INDEX FROM 0 TO LEGAL_LENGTH - 1 DO
;    3349  3        XMIT_BUFFER [ .INDEX ] = .INDEX;
;    3350  3
;    3351  5      BGNSUB;
;    3352  5        RESET_DEQNA ( );
;    3353  5        INCR INDEX FROM 0 TO 63 DO
;    3354  5          RCV_D_LIST [ .INDEX, W_LEN ] = .RD13 [ .INDEX ];
;    3355  5        INCR INDEX FROM 0 TO 31 DO
;    3356  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .TD13 [ .INDEX ];
;    3357  5
;    3358  5        TEMP5 = .XMIT_D_LIST [ 27, W_LEN ];
;    3359  5        TEMP6 = .RCV_D_LIST  [ 51, W_LEN ];
;    3360  5        TEMP7 = .RCV_D_LIST  [ 56, W_LEN ];
;    3361  5
;    3362  5        XMIT_D_LIST [ 27, W_LEN ] = -628;
;    3363  5        RCV_D_LIST  [ 51, W_LEN ] = -625;
;    3364  5        RCV_D_LIST  [ 56, W_LEN ] = RCV_BUFFER + LEGAL_LENGTH - 2;
;    3365  5
;    3366  5        PUT_BIT [ CSR, LB, INX_LOOPBACK ];
;    3367  5        XMIT_AND_RCV_PACKET ( );
;    3368  5
;    3369  5        CHK_RIXI_STATUS ( ONE );
;    3370  5
;    3371  5        IF .SWP_BLOCK_MEM EQLU ONE
;    3372  5          THEN
;    3373  5            TEMP4 = %O'367'            ! ADDED 25% TO "305" TO GET "367". FIX FOR $$$
;    3374  5          ELSE                         ! CHANGE FROM 15 MHZ TO 18 MHZ CPU, BY HLM. $$$
;    3375  5            TEMP4 = 4 * %O'367';       ! $$$
;    3376  5
;    3377  5        IF .TEMP1 GTRU .TEMP4
;    3378  5          THEN
;    3379  6            BEGIN
;    3380  6              CSR_WORD = GET_BIT ( CSR_ALL );
;    3381  6              PRINTB ( MSG59 );
;    3382  6              PRINTB ( MSG52 );
;    3383  6              ERRDF  ( 1401, MSG00, ERROR$REPORT );
;    3384  5            END;
;    3385  5
;    3386  5        CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );     ! O'100220', O'100220'
;    3387  5
;    3388  5        XMIT_D_LIST [ 27, W_LEN ] = .TEMP5;
;    3389  5        PCV_D_LIST  [ 51, W_LEN ] = .TEMP6;
;    3390  5        RCV_D_LIST  [ 56, W_LEN ] = .TEMP7;
;    3391  5
;    3392  5        !++
;    3393  5        !  CHECK IF TRANSMIT BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;    3394  5        !--
;    3395  5        INCR INDEX FROM 0 TO 23 DO
;    3396  5          IF .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU .TD13 [ .INDEX ]
```

F14


```
;  3397  5          AND ( .XMIT_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;  3398  5            THEN
;  3399  6              BEGIN
;  3400  6                CSR_WORD = GET_BIT ( CSR_ALL );
;  3401  6                PRINTB ( MSG59 );
;  3402  6                PRINTB ( MSG49 );
;  3403  6                PRINTB ( msg76, XMIT_D_LIST, .INDEX, .XMIT_D_LIST [ .INDEX, W_LEN ] );
;  3404  6                ERRDF  ( 1402, MSG00, ERROR$REPORT );
;  3405  5              END;
;  3406  5
;  3407  5      !++
;  3408  5      !  CHECK IF RECEIVE BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;  3409  5      ! -
;  3410  5      INCR INDEX FROM 0 TO 53 DO
;  3411  5        IF .RCV_D_LIST [ .INDEX, W_LEN ] NEQU .RD13 [ .INDEX ]
;  3412  5          AND ( .RCV_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;  3413  5            THEN
;  3414  6              BEGIN
;  3415  6                CSR_WORD = GET_BIT ( CSR_ALL );
;  3416  6                PRINTB ( MSG59 );
;  3417  6                PRINTB ( MSG48 );
;  3418  6                PRINTB ( msg75, RCV_D_LIST, .INDEX, .RCV_D_LIST [ .INDEX, W_LEN ] );
;  3419  6                ERRDF  ( 1403, MSG00, ERROR$REPORT );
;  3420  5              END;
;  3421  5
;  3422  5      INCR INDEX FROM 0 TO 5 DO
;  3423  6        BEGIN
;  3424  6          TEMP1 = .INDEX + 24;
;  3425  6          TEMP2 = .INDEX + 54;
;  3426  6          XMIT_D_LIST [ .INDEX, W_LEN ] = .XMIT_D_LIST [ .TEMP1, W_LEN ];
;  3427  6          RCV_D_LIST  [ .INDEX, W_LEN ] = .RCV_D_LIST  [ .TEMP2, W_LEN ];
;  3428  5        END;
;  3429  5
;  3430  5      RBUF_LENGTH = 1514;
;  3431  5      CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );  ! 0'140000', 0'000400'
;  3432  5      CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS  );  ! 0'140000', 0'020000'
;  3433  5
;  3434  5      INCR INDEX FROM 0 TO LEGAL_LENGTH - 1 DO
;  3435  5        IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;  3436  5          THEN
;  3437  6            BEGIN
;  3438  6              CSR_WORD = GET_BIT ( CSR_ALL );
;  3439  6              PRINTB ( MSG59 );
;  3440  6              PRINTB ( MSG51 );
;  3441  6              PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;  3442  6              ERRDF  ( 1404, MSG00, ERROR$REPORT );
;  3443  5            END;
;  3444  3        ENDSUB;
;  3445  3
;  3446  1    ENDTST;

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

G14


```
000000  004137  000000G                  $T14:   JSR     R1,$SAVE2               ;                                       3300
000004  162706  000010                            SUB     #10,SP
000010  012737  002752  000000G                   MOV     #2752,RBUF.LENGTH       ;                                       3346
000016  012700  002752                            MOV     #2752,R0                ;                                       3347
000022  006200                                    ASR     R0
000024  005400                                    NEG     R0
000026  010037  000000G                           MOV     R0,XBUF.LENGTH
000032  005000                                    CLR     R0                      ; INDEX                                 3348
000034  110060  000000G                  1$:     MOVB    R0,XMIT.BUFFER(R0)      ; INDEX,*(INDEX)                        3349
000040  005200                                    INC     R0                      ; INDEX                                 3348
000042  020027  002751                            CMP     R0,#2751                ; INDEX,*
000046  003772                                    BLE     1$
000050  104402                          2$:       TRAP    2                       ;                                       3349
000052  004737  000000G                           JSR     PC,RESET.DEQNA          ;                                       3352
000056  005000                                    CLR     R0                      ; INDEX                                 3353
000060  016060  000000G 000000G          3$:     MOV     RD13(R0),RCV.D.LIST(R0) ; *(INDEX),*(INDEX)                     3354
000066  062700  000002                            ADD     #2,R0                   ; *,INDEX                               3353
000072  020027  000176                            CMP     R0,#176                 ; INDEX,*
000076  003770                                    BLE     3$
000100  005000                                    CLR     R0                      ; INDEX                                 3355
000102  016060  000000G 000000G          4$:     MOV     TD13(R0),XMIT.D.LIST(R0) ; *(INDEX),*(INDEX)                    3356
000110  062700  000002                            ADD     #2,R0                   ; *,INDEX                               3355
000114  020027  000076                            CMP     R0,#76                  ; INDEX,*
000120  003770                                    BLE     4$
000122  013737  000066G 000000G                   MOV     XMIT.D.LIST+66,TEMP5    ;                                       3358
000130  013737  000146G 000000G                   MOV     RCV.D.LIST+146,TEMP6    ;                                       3359
000136  013737  000160G 000000G                   MOV     RCV.D.LIST+160,TEMP7    ;                                       3360
000144  012737  176614  000066G                   MOV     #-1164,XMIT.D.LIST+66   ;                                       3362
000152  012737  176617  000146G                   MOV     #-1161,RCV.D.LIST+146   ;                                       3363
000160  012737  002750G 000160G                   MOV     #RCV.BUFFER+2750,RCV.D.LIST+160 ;                               3364
000166  013700  000000G                           MOV     REG.ADR,R0              ;                                       3366
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  ;                                       3367
000212  012746  000001                            MOV     #1,-(SP)                ;                                       3369
000216  004737  000000G                           JSR     PC,CHK.RIXI.STATUS
000222  023727  000000G 000001                    CMP     SWP.BLOCK.MEM,#1        ;                                       3371
000230  001004                                    BNE     5$
000232  012737  000367  000000G                   MOV     #367,TEMP4              ;                                       3373
000240  000403                                    BR      6$                      ;                                       3371
000242  012737  001734  000000G          5$:     MOV     #1734,TEMP4             ;                                       3375
000250  023737  000000G 000000G          6$:     CMP     TEMP1,TEMP4             ;                                       3377
000256  101431                                    BLOS    7$
000260  013700  000000G                           MOV     REG.ADR,R0              ;                                       3380
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)             ;                                       3381
000304  012746  000001                            MOV     #1,-(SP)
000310  010600                                    MOV     SP,R0                   ; SP,*
000312  104414                                    TRAP    14
000314  012716  000000G                           MOV     #MSG52,(SP)             ;                                       3382
000320  012746  000001                            MOV     #1,-(SP)
000324  010600                                    MOV     SP,R0                   ; SP,*
```

```
000326  104414                          TRAP    14
000330  104455                          TRAP    55                      ;                               3383
000332  002571                          .WORD   2571
000334  000000G                         .WORD   MSG00
000336  000000G                         .WORD   ERROR$REPORT
000340  022626                          CMP     (SP)+,(SP)+             ;                               3379
000342  012716  100220          7$:     MOV     #-77560,(SP)            ;                               3386
000346  011646                          MOV     (SP),-(SP)
000350  004737  000000G                 JSR     PC,CHK.CSR.STATUS
000354  013737  000000G 000066G         MOV     TEMP5,XMIT.D.LIST+66    ;                               3388
000362  013737  000000G 000146G         MOV     TEMP6,RCV.D.LIST+146    ;                               3389
000370  013737  000000G 000160G         MOV     TEMP7,RCV.D.LIST+160    ;                               3390
000376  005002                          CLR     R2                      ; INDEX                         3395
000400  010201          8$:             MOV     R2,R1                   ; INDEX,*                       3396
000402  006301                          ASL     R1
000404  026161  000000G 000000G         CMP     XMIT.D.LIST(R1),TD13(R1)
000412  001454                          BEQ     9$
000414  016100  000000G                 MOV     XMIT.D.LIST(R1),R0      ;                               3397
000420  042700  037777                  BIC     #37777,R0
000424  020027  140000                  CMP     R0,#-40000
000430  001445                          BEQ     9$
000432  013700  000000G                 MOV     REG.ADR,R0              ;                               3400
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)             ;                               3401
000456  012746  000001                  MOV     #1,-(SP)
000462  010600                          MOV     SP,R0                   ; SP,*
000464  104414                          TRAP    14
000466  012716  000000G                 MOV     #MSG49,(SP)             ;                               3402
000472  012746  000001                  MOV     #1,-(SP)
000476  010600                          MOV     SP,R0                   ; SP,*
000500  104414                          TRAP    14
000502  016116  000000G                 MOV     XMIT.D.LIST(R1),(SP)    ;                               3403
000506  010246                          MOV     R2,-(SP)                ; INDEX,*
000510  012746  000000G                 MOV     #XMIT.D.LIST,-(SP)
000514  012746  000000G                 MOV     #MSG76,-(SP)
000520  012746  000004                  MOV     #4,-(SP)
000524  010600                          MOV     SP,R0                   ; SP,*
000526  104414                          TRAP    14
000530  104455                          TRAP    55                      ;                               3404
000532  002572                          .WORD   2572
000534  000000G                         .WORD   MSG00
000536  000000G                         .WORD   ERROR$REPORT
000540  062706  000014                  ADD     #14,SP                  ;                               3399
000544  005202          9$:             INC     R2                      ; INDEX                         3395
000546  020227  000027                  CMP     R2,#27                  ; INDEX,*
000552  003712                          BLE     8$                                                      3410
000554  005002                          CLR     R2                      ; INDEX                         3411
000556  010201          10$:            MOV     R2,R1                   ; INDEX,*
000560  006301                          ASL     R1
000562  026161  000000G 000000G         CMP     RCV.D.LIST(R1),RD13(R1)
000570  001454                          BEQ     11$
000572  016100  000000G                 MOV     RCV.D.LIST(R1),R0       ;                               3412
```

```
000576  042700  037777                          BIC     #37777,R0
000602  020027  140000                          CMP     R0,#-40000
000606  001445                                  BEQ     11$
000610  013700  000000G                         MOV     REG.ADR,R0              ;                               3415
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)             ;                               3416
000634  012746  000001                          MOV     #1,-(SP)
000640  010600                                  MOV     SP,R0                   ; SP,*
000642  104414                                  TRAP    14
000644  012716  000000G                         MOV     #MSG48,(SP)             ;                               3417
000650  012746  000001                          MOV     #1,-(SP)
000654  010600                                  MOV     SP,R0                   ; SP,*
000656  104414                                  TRAP    14
000660  016116  000000G                         MOV     RCV.D.LIST(R1),(SP)     ;                               3418
000664  010246                                  MOV     R2,-(SP)                ; INDEX,*
000666  012746  000000G                         MOV     #RCV.D.LIST,-(SP)
000672  012746  000000G                         MOV     #MSG75,-(SP)
000676  012746  000004                          MOV     #4,-(SP)
000702  010600                                  MOV     SP,R0                   ; SP,*
000704  104414                                  TRAP    14
000706  104455                                  TRAP    55                      ;                               3419
000710  002573                                  .WORD   2573
000712  000000G                                 .WORD   MSG00
000714  000000G                                 .WORD   ERROR$REPORT
000716  062706  000014                          ADD     #14,SP                  ;                               3414
000722  005202                          11$:    INC     R2                      ; INDEX                         3410
000724  020227  000065                          CMP     R2,#65                  ; INDEX,*
000730  003712                                  BLE     10$
000732  005002                                  CLR     R2                      ; INDEX                         3422
000734  010237  000000G                 12$:    MOV     R2,TEMP1                ; INDEX,*                       3424
000740  062737  000030  000000G                 ADD     #30,TEMP1
000746  010237  000000G                         MOV     R2,TEMP2                ; INDEX,*                       3425
000752  062737  000066  000000G                 ADD     #66,TEMP2
000760  010200                                  MOV     R2,R0                   ; INDEX,*                       3426
000762  006300                                  ASL     R0
000764  013701  000000G                         MOV     TEMP1,R1
000770  006301                                  ASL     R1
000772  016160  000000G 000000G                 MOV     XMIT.D.LIST(R1),XMIT.D.LIST(R0)
001000  013701  000000G                         MOV     TEMP2,R1                ;                               3427
001004  006301                                  ASL     R1
001006  016160  000000G 000000G                 MOV     RCV.D.LIST(R1),RCV.D.LIST(R0)
001014  005202                                  INC     R2                      ; INDEX                         3422
001016  020227  000005                          CMP     R2,#5                   ; INDEX,*
001022  003744                                  BLE     12$
001024  012737  002752  000000G                 MOV     #2752,RBUF.LENGTH       ;                               3430
001032  012716  140000                          MOV     #-40000,(SP)            ;                               3431
001036  012746  000400                          MOV     #400,-(SP)
001042  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
001046  012716  140000                          MOV     #-40000,(SP)            ;                               3432
001052  012746  020000                          MOV     #20000,-(SP)
001056  004737  000000G                         JSR     PC,CHK.RCV.STATUS
001062  005001                                  CLR     R1                      ; INDEX                         3434
```

J14


```
001064  126161  000000G 000000G          13$:    CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                    3435
001072  001447                                   BEQ     14$
001074  013700  000000G                          MOV     REG.ADR,R0                      ;                               3438
001100  016066  000016  000016                   MOV     16(R0),16(SP)                   ; *,IMP.LOCATION
001106  016637  000016  000000G                  MOV     16(SP),CSR.WORD                 ; TMP.LOCATION,*
001114  012716  000000G                          MOV     #MSG59,(SP)                     ;                               3439
001120  012746  000001                           MOV     #1,-(SP)
001124  010600                                   MOV     SP,R0                           ; SP,*
001126  104414                                   TRAP    14
001130  012716  000000G                          MOV     #MSG51,(SP)                     ;                               3440
001134  012746  000001                           MOV     #1,-(SP)
001140  010600                                   MOV     SP,R0                           ; SP,*
001142  104414                                   TRAP    14
001144  010116                                   MOV     R1,(SP)                         ; INDEX,*                       3441
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  104411                                   TRAP    14
001176  104455                                   TRAP    55                              ;                               3442
001200  002574                                   .WORD   2574
001202  000000G                                  .WORD   MSG00
001204  000000G                                  .WORD   ERROR$REPORT
001206  062706  000014                           ADD     #14,SP                          ;                               3437
001212  005201                          14$:     INC     R1                              ; INDEX                         3434
001214  020127  002751                           CMP     R1,#2751                        ; INDEX,*
001220  003721                                   BLE     13$
001222  062706  000010                           ADD     #10,SP                          ;                               3349
001226  104467                                   TRAP    67                              ;                               3443
001230  006000                                   ROR     R0
001232  103002                                   BHIS    15$
001234  000137  012566'                          JMP     2$
001240  062706  000010                  15$:     ADD     #10,SP                          ;                               3300
001244  000207                                   RTS     PC

; Routine Size:  339 words,     Routine B       .$CODE$ + 12516
; Maximum stack depth per invocation:  1
```

K14


```
                                        .SBTTL  T14 TEST 14   DMA TIMING TEST

000000  004737  012516'                T14::
000000                                  1$:     JSR     PC,$T14                         ;                               3444
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 13764
; Maximum stack depth per invocation:  2 words


;   3447  1
```

L14


```
;    3448  1     %SBTTL 'TEST 15   LONG PACKET TEST'
;    3449  1     !++
;    3450  1     !
;    3451  1     !  TEST 15:     LONG PACKET TEST
;    3452  1     !
;    3453  1     !  DESCRIPTION:
;    3454  1     !
;    3455  1     !       This test verifies that DEQNA can detect long packets ( 1600 bytes
;    3456  1     !       or more with the CRC ) when transmitted in internal/extended
;    3457  1     !       loopback mode. If the operator specifies loop on error, the
;    3458  1     !       program re-executes the code that detected the error until ^C is
;    3459  1     !       entered.
;    3460  1     !
;    3461  1     !       Hardware tested:        RCV Status - error summary (long packet-bit 14)
;    3462  1     !
;    3463  1     !       Processing:
;    3464  1     !
;    3465  1     !            BEGIN
;    3466  1     !               reset device
;    3467  1     !               select internal/extended loopback mode
;    3468  1     !               transmit loopback packet (legal packet length)
;    3469  1     !               check for expected loopback status
;    3470  1     !               IF error
;    3471  1     !               THEN
;    3472  1     !                  print error message if not inhibited
;    3473  1     !               ENDIF
;    3474  1     !               call compare_packets
;    3475  1     !               transmit loopback packet ( packet length > legal max. )
;    3476  1     !               IF Error Summary bit ( Receice Status Word 1, bit 14 ) = 1
;    3477  1     !                  AND ( receive packet length is truncated )
;    3478  1     !               THEN
;    3479  1     !                  print error message if not inhibited
;    3480  1     !               ENDIF
;    3481  1     !            END
;    3482  1     !--
```

M14


```
;    3483  3     BGNTST;
;    3484  3
;    3485  3     !++
;    3486  3     !  LOOPBACK 1534 BYTE PACKET AND THEN CHECK IF PROPERLY RECEIVED.
;    3487  3     !  THIS IS THE LONGEST PACKET LENGTH WHICH DOESN'T SET 'LONGP' BIT IN
;    3488  3     !  THE RECEIVE STATUS WORD 1 ( BIT 14 ).
;    3489  3     !--
;    3490  3
;    3491  3     RBUF_LENGTH = 1534;
;    3492  3     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3493  3
;    3494  5     BGNSUB;
;    3495  5       RESET_DEQNA ( );
;    3496  5       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    3497  5       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    3498  5       SEND_ELOOP_PACKET ( ZERO );
;    3499  5       COMPARE_PACKETS ( );
;    3500  3     ENDSUB;
;    3501  3
;    3502  3     !++
;    3503  3     !  LOOPBACK 1536 BYTE PACKET AND THEN CHECK IF BITS 13 AND 14 ARE SET IN
;    3504  3     !--
;    3505  3
;    3506  3
;    3507  3     RBUF_LENGTH = 1536;
;    3508  3     XBUF_LENGTH =   ( .RBUF LENGTH + -1 );
;    3509  3
;    3510  5     BGNSUB;
;    3511  5       RESET_DEQNA ( );
;    3512  5       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    3513  5       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    3514  5       SEND_ELOOP_PACKET ( ONE );
;    3515  5       COMPARE_PACKETS ( );
;    3516  3     ENDSUB;
;    3517  3
;    3518  1     ENDTST;

                                        .SBTTL  $T15 TEST 15 - LONG PACKET TEST
000000  012737  002776  000000G  $T15:  MOV     #2776,RBUF.LENGTH               ;  3491
000006  012700  002776                  MOV     #2776,R0                        ;  3492
000012  006200                          ASR     R0
000014  005400                          NEG     R0
000016  010037  000000G                 MOV     R0,XBUF.LENGTH
000022  104402                  1$:     TRAP    2
000024  004737  000000G                 JSR     PC,RESET.DEQNA                  ;  3495
000030  013746  000000G                 MOV     XBUF.LENGTH,-(SP)               ;  3496
000034  012746  120000                  MOV     #-60000,-(SP)
000040  004737  000000G                 JSR     PC,SET.RDESCR.LIST
000044  013716  000000G                 MOV     XBUF.LENGTH,(SP)                ;  3497
000050  012746  120000                  MOV     #-60000,-(SP)
000054  004737  000000G                 JSR     PC,SET.XDESCR.LIST
000060  005016                          CLR     (SP)                            ;  3498
```

```
000062  004737  000000G                        JSR     PC,SEND.ELOOP.PACKET
000066  004737  000000G                        JSR     PC,COMPARE.PACKETS      ;                       3499
000072  062706  000006                         ADD     #6,SP                   ;                       3492
000076  104467                                 TRAP    67                      ;                       3499
000100  006000                                 ROR     R0
000102  103747                                 BLO     1$
000104  012737  003000  000000G                MOV     #3000,RBUF.LENGTH       ;                       3507
000112  012700  003000                         MOV     #3000,R0                ;                       3508
000116  006200                                 ASR     R0
000120  005400                                 NEG     R0
000122  010037  000000G                        MOV     R0,XBUF.LENGTH
000126  104402                         2$:     TRAP    2
000130  004737  000000G                        JSR     PC,RESET.DEQNA          ;                       3511
000134  013746  000000G                        MOV     XBUF.LENGTH,-(SP)       ;                       3512
000140  012746  120000                         MOV     #-60000,-(SP)
000144  004737  000000G                        JSR     PC,SET.RDESCR.LIST
000150  013716  000000G                        MOV     XBUF.LENGTH,(SP)        ;                       3513
000154  012746  120000                         MOV     #-60000,-(SP)
000160  004737  000000G                        JSR     PC,SET.XDESCR.LIST
000164  012716  000001                         MOV     #1,(SP)                 ;                       3514
000170  004737  000000G                        JSR     PC,SEND.ELOOP.PACKET
000174  004737  000000G                        JSR     PC,COMPARE.PACKETS      ;                       3515
000200  062706  000006                         ADD     #6,SP                   ;                       3508
000204  104467                                 TRAP    67                      ;                       3515
000206  006000                                 ROR     R0
000210  103746                                 BLO     2$
000212  000207                                 RTS     PC                      ;                       3446

; Routine Size:  70 words,      Routine Base:  AB$CODE$ + 14000
; Maximum stack depth per invocation:  4 words



                                       .SBTTL  T15 TEST 15 - LONG PACKET TEST
000000  004737  014000'                 T15::
000000                                 1$:     JSR     PC,$T15                 ;                       3516
000004  104466                                 TRAP    66
000006  006000                                 ROR     R0
000010  103773                                 BLO     1$
000012  000207                                 RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 14214
; Maximum stack depth per invocation:  2 words


;   3519  1
;   3520  1
```

```
;  3521  1     %SBTTL 'TEST 16 - ODD PACKET TEST'
;  3522  1     !++
;  3523  1     !
;  3524  1     !  TEST 16:     ODD PACKET TEST
;  3525  1     !
;  3526  1     !  DESCRIPTION:
;  3527  1     !
;  3528  1     !       This test verifies that DEQNA can transmit and receive odd length
;  3529  1     !       packets and packets starting and/or ending on odd addresses. Chained
;  3530  1     !       and unchained descriptor lists are used to verify this. If the operator
;  3531  1     !       specifies loop on error, the program re-executes the code that detected
;  3532  1     !       the error until ^C is entered.
;  3533  1     !
;  3534  1     !       Hardware tested:        CSR register - XMIT List Invalid (bit 4)
;  3535  1     !                                            - RCV  List Invalid (bit 5)
;  3536  1     !                               Transmit Descriptor bits
;  3537  1     !                                            - XMIT buffer ends on odd byte
;  3538  1     !                                            - XMIT buffer ends on even byte
;  3539  1     !
;  3540  1     !       Set of addresses and packet lengths:
;  3541  1     !
;  3542  1     !               PACKET ADDRESS              PACKET LENGTH
;  3543  1     !               ----------------            -------------
;  3544  1     !
;  3545  1     !               odd begin                       odd
;  3546  1     !               odd begin and end               even
;  3547  1     !               odd end                         odd
;  3548  1     !
;  3549  1     !       Processing:
;  3550  1     !
;  3551  1     !            BEGIN
;  3552  1     !               reset device
;  3553  1     !               REPEAT for internal and internal/extended loopback mode
;  3554  1     !                 REPEAT for each packet address and length from set
;  3555  1     !                   check for expected loopback status
;  3556  1     !                   IF error
;  3557  1     !                   THEN
;  3558  1     !                     print error message if not inhibited
;  3559  1     !                   ENDIF
;  3560  1     !                   call compare_packets
;  3561  1     !                 ENDREPEAT
;  3562  1     !               ENDREPEAT
;  3563  1     !            END
;  3564  1     !--
```

```
;  3565  3      BGNTST;
;  3566  3
;  3567  3      !++
;  3568  3      !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM
;  3569  3      !--
;  3570  3
;  3571  3      RESET_DEQNA ( );
;  3572  3      PREP_FOR_SETUP ( );
;  3573  3      INCR INDEX1 FROM 1 TO 14 DO
;  3574  3        WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  3575  3
;  3576  5      BGNSUB;
;  3577  5        XMIT_SETUP_PACKET ( P_MODE );
;  3578  3      ENDSUB;
;  3579  3
;  3580  3      RBUF_LENGTH = 6;
;  3581  3      XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  3582  3
;  3583  3      !++
;  3584  3      !  LOOPBACK A PACKET, THEN CHECK IF LOOPBACK PACKET WAS PROPERLY
;  3585  3      !  RECEIVED
;  3586  3      !--
;  3587  3
;  3588  3      CLR_BUFFERS ( 32 );
;  3589  3      CLR_DESCR ( );
;  3590  3      INCR INDEX FROM 0 TO 5 DO
;  3591  3        XMIT_BUFFER [ .INDEX ] = .INDEX;
;  3592  3
;  3593  5      BGNSUB;
;  3594  5        INCR INDEX FROM 0 TO 43 DO
;  3595  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .TD16 [ .INDEX ];
;  3596  5        SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  3597  5        PUT_BIT [ CSR, LB, INT_LOOPBACK ];
;  3598  5
;  3599  5        XMIT_AND_RCV_PACKET ( );
;  3600  5        CHK_RIXI_STATUS ( ONE );
;  3601  5        .IOP_TABLE [ CSR ] = ONE;
;  3602  5        CHK_RIXI_STATUS ( ZERO );
;  3603  5        .IOP_TABLE [ CSR ] = ZERO;
;  3604  5
;  3605  5        CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );     ! 0'100220', 0'100220'
;  3606  5
;  3607  5        !++
;  3608  5        .  CHECK IF TRANSMIT BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;  3609  5        !--
;  3610  5
;  3611  5        INCR INDEX FROM 0 TO 17 DO
;  3612  5          IF .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU .TD16 [ .INDEX ]
;  3613  5            AND ( .XMIT_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;  3614  5              THEN
;  3615  6                BEGIN
;  3616  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;  3617  6                  PRINTB ( MSG59 );
```

```
;    3618  6                PRINTB ( MSG49 );
;    3619  6                PRINTB ( msg76, XMIT_D_LIST, .INDEX, .XMIT_D_LIST [ .INDEX, W_LEN ] );
;    3620  6                ERRDF  ( 1602, MSG00, ERROR$REPORT );
;    3621  5              END;
;    3622  5
;    3623  5
;    3624  5        INCR INDEX FROM 0 TO 5 DO
;    3625  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .XMIT_D_LIST [ .INDEX + 18, W_LEN ];
;    3626  5
;    3627  5        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );  ! 0'140000', 0'000400'
;    3628  5        CHK_RCV_STATUS  ( RFLG_STATUS, RWD13_STATUS );  ! 0 140000', 0'000000'
;    3629  5
;    3630  5        INCR INDEX FROM 0 TO 5 DO
;    3631  5          IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;    3632  5            THEN
;    3633  6              BEGIN
;    3634  6                CSR_WORD = GET_BIT ( CSR_ALL );
;    3635  6                PRINTB ( MSG59 );
;    3636  6                PRINTB ( MSG51 );
;    3637  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;    3638  6                ERRDF  ( 1603, MSG00, ERROR$REPORT );
;    3639  5              END;
;    3640  3      ENDSUB;
;    3641  3
;    3642  3      RESET_DEQNA ( );
;    3643  3      CLR_BUFFERS ( 32 );
;    3644  3      RBUF_LENGTH = 16;
;    3645  3      XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3646  3      INCR INDEX FROM 0 TO 19 DO
;    3647  3        XMIT_BUFFER [ .INDEX ] = .INDEX;
;    3648  3
;    3649  5      BGNSUB;
;    3650  5        INCR INDEX FROM 0 TO 43 DO
;    3651  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .TD16 [ .INDEX ];
;    3652  5
;    3653  5        XMIT_D_LIST [ 19, W_LEN ] = V;
;    3654  5        XMIT_D_LIST [ 25, W_LEN ] = C;
;    3655  5
;    3656  5        SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    3657  5        PUT_BIT [ CSR, LB, INX_LOOPBACK ];
;    3658  5        XMIT_AND_RCV_PACKET ( );
;    3659  5        CHK_RIXI_STATUS ( ZERO );
;    3660  5
;    3661  5        CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );     ! 0'100220', 0'100220'
;    3662  5
;    3663  5        XMIT_D_LIST [ 19, W_LEN ] = VE;
;    3664  5        XMIT_D_LIST [ 25, W_LEN ] = E;
;    3665  5
;    3666  5        !++
;    3667  5        !  CHECK IF TRANSMIT BUFFER DESCRIPTOR LISTS PROPERLY VOLIDATED
;    3668  5        !--
;    3669  5
;    3670  5        INCR INDEX FROM 0 TO 35 DO
```

```
;    3671  5          IF .XMIT_D_LIST [ .INDEX, W_LEN ] NEQU .TD16 [ .INDEX ]
;    3672  5            AND ( .XMIT_D_LIST [ .INDEX, W_LEN ] AND %O'140000' ) NEQU %O'140000'
;    3673  5              THEN
;    3674  6                BEGIN
;    3675  6                  CSR_WORD = GET_BIT ( CSR_ALL );
;    3676  6                  PRINTB ( MSG59 );
;    3677  6                  PRINTB ( MSG49 );
;    3678  6                  PRINTB ( msg76, XMIT_D_LIST, .INDEX, .XMIT_D_LIST [ .INDEX, W_LEN ] );
;    3679  6                  ERRDF  ( 1604, MSG00, ERROR$REPORT );
;    3680  5                END;
;    3681  5
;    3682  5        INCR INDEX FROM 0 TO 5 DO
;    3683  5          XMIT_D_LIST [ .INDEX, W_LEN ] = .XMIT_D_LIST [ .INDEX + 36, W_LEN ];
;    3684  5
;    3685  5        CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );  ! O'140000', O'000400'
;    3686  5        CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS );   ! O'140000', O'020000'
;    3687  5
;    3688  5
;    3689  5        INCR INDEX FROM 0 TO 5 DO
;    3690  5          IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;    3691  5            THEN
;    3692  6              BEGIN
;    3693  6                CSR_WORD = GET_BIT ( CSR_ALL );
;    3694  6                PRINTB ( MSG59 );
;    3695  6                PRINTB ( MSG51 );
;    3696  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;    3697  6                ERRDF  ( 1605, MSG00, ERROR$REPORT );
;    3698  5              END;
;    3699  5
;    3700  5        INCR INDEX FROM 6 TO 9 DO
;    3701  5          IF .RCV_BUFFER [ .INDEX ] NEQU ZERO
;    3702  5            THEN
;    3703  6              BEGIN
;    3704  6                CSR_WORD = GET_BIT ( CSR_ALL );
;    3705  6                PRINTB ( MSG59 );
;    3706  6                PRINTB ( MSG51 );
;    3707  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], ZERO, .INDEX );
;    3708  6                ERRDF  ( 1606, MSG00, ERROR$REPORT );
;    3709  5              END;
;    3710  5
;    3711  5        INCR INDEX FROM 0 TO 5 DO
;    3712  5          IF .RCV_BUFFER [ .INDEX + 10 ] NEQU .TARGET_ADR [ .INDEX + 114 ]
;    3713  5            THEN
;    3714  6              BEGIN
;    3715  6                CSR_WORD = GET_BIT ( CSR_ALL );
;    3716  6                PRINTB ( MSG59 );
;    3717  6                PRINTB ( MSG51 );
;    3718  6                PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ], .INDEX );
;    3719  6                ERRDF  ( 1607, MSG00, ERROR$REPORT );
;    3720  5              END;
;    3721  3      ENDSUB;
;    3722  3
;    3723  1      ENDTST;
```

F15


```
                                                .SBTTL  $T16 TEST 16 - ODD PACKET TEST
000000  004137  000000G                 $T16:  JSR     R1,$SAVE2                ;                                3518
000004  162706  000014                        SUB     #14,SP
000010  004737  000000G                        JSR     PC,RESET.DEQNA           ;                                3571
000014  004737  000000G                        JSR     PC,PREP.FOR.SETUP        ;                                3572
000020  012701  000001                        MOV     #1,R1                    ; *,INDEX1                       3573
000024  010146                          1$:    MOV     R1,-(SP)                 ; INDEX1,*                       3574
000026  012746  000023                        MOV     #23,-(SP)
000032  004737  000000G                        JSR     PC,WRT.STATION.ADR
000036  022626                                CMP     (SP)+,(SP)+
000040  005201                                INC     R1                       ; INDEX1                         3573
000042  020127  000016                        CMP     R1,#16                   ; INDEX1,*
000046  003766                                BLE     1$
000050  104402                          2$:    TRAP    2                        ;                                3574
000052  012746  000202                        MOV     #202,-(SP)               ;                                3577
000056  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000062  005726                                TST     (SP)+                    ;                                3574
000064  104467                                TRAP    67                       ;                                3577
000066  006000                                ROR     R0
000070  103767                                BLO     2$
000072  012737  000006  000000G               MOV     #6,RBUF.LENGTH           ;                                3580
000100  012700  000006                        MOV     #6,R0                    ;                                3581
000104  006200                                ASR     R0
000106  005400                                NEG     R0
000110  010037  000000G                        MOV     R0,XBUF.LENGTH
000114  012746  000040                        MOV     #40,-(SP)                ;                                3588
000120  004737  000000G                        JSR     PC,CLR.BUFFERS
000124  004737  000000G                        JSR     PC,CLR.DESCR             ;                                3589
000130  005000                                CLR     R0                       ; INDEX                          3590
000132  110060  000000G                 3$:    MOVB    R0,XMIT.BUFFER(R0)       ; INDEX,*(INDEX)                 3591
000136  005200                                INC     R0                       ; INDEX                          3590
000140  020027  000005                        CMP     R0,#5                    ; INDEX,*
000144  003772                                BLE     3$
000146  104402                          4$:    TRAP    2                        ;                                3591
000150  005000                                CLR     R0                       ; INDEX                          3594
000152  016060  000000G 000000G          5$:    MOV     TD16(R0),XMIT.D.LIST(R0) ; *(INDEX),*(INDEX)              3595
000160  062700  000002                        ADD     #2,R0                    ; *,INDEX                        3594
000164  020027  000126                        CMP     R0,#126                  ; INDEX,*
000170  003770                                BLE     5$
000172  013716  000000G                        MOV     XBUF.LENGTH,(SP)         ;                                3596
000176  012746  120000                        MOV     #-60000,-(SP)
000202  004737  000000G                        JSR     PC,SET.RDESCR.LIST
000206  013700  000000G                        MOV     REG.ADR,R0               ;                                3597
000212  042760  001400  000016               BIC     #1400,16(R0)
000220  004737  000000G                        JSR     PC,XMIT.AND.RCV.PACKET   ;                                3599
000224  012716  000001                        MOV     #1,(SP)                  ;                                3600
000230  004737  000000G                        JSR     PC,CHK.RIXI.STATUS
000234  012777  000001  000016G               MOV     #1,@IOP.TABLE+16         ;                                3601
000242  005016                                CLR     (SP)                     ;                                3602
000244  004737  000000G                        JSR     PC,CHK.RIXI.STATUS
000250  005077  000016G                        CLR     @IOP.TABLE+16            ;                                3603
```

```
000254  012716  100220                          MOV     #-77560,(SP)                    ;                                       3605
000260  011646                                  MOV     (SP),-(SP)
000262  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000266  005002                                  CLR     R2                              ; INDEX                                 3611
000270  010201                          6$:     MOV     R2,R1                           ; INDEX,*                               3612
000272  006301                                  ASL     R1
000274  026161  000000G 000000G                 CMP     XMIT.D.LIST(R1),TD16(R1)
000302  001454                                  BEQ     7$
000304  016100  000000G                         MOV     XMIT.D.LIST(R1),R0              ;                                       3613
000310  042700  037777                          BIC     #37777,R0
000314  020027  140000                          CMP     R0,#-40000
000320  001445                                  BEQ     7$
000322  013700  000000G                         MOV     REG.ADR,R0                      ;                                       3616
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)                     ;                                       3617
000346  012746  000001                          MOV     #1,-(SP)
000352  010600                                  MOV     SP,R0                           ; SP,*
000354  104414                                  TRAP    14
000356  012716  000000G                         MOV     #MSG49,(SP)                     ;                                       3618
000362  012746  000001                          MOV     #1,-(SP)
000366  010600                                  MOV     SP,R0                           ; SP,*
000370  104414                                  TRAP    14
000372  016116  000000G                         MOV     XMIT.D.LIST(R1),(SP)            ;                                       3619
000376  010246                                  MOV     R2,-(SP)                        ; INDEX,*
000400  012746  000000G                         MOV     #XMIT.D.LIST,-(SP)
000404  012746  000000G                         MOV     #MSG76,-(SP)
000410  012746  000004                          MOV     #4,-(SP)
000414  010600                                  MOV     SP,R0                           ; SP,*
000416  104414                                  TRAP    14
000420  104455                                  TRAP    55                              ;                                       3620
000422  003102                                  .WORD   3102
000424  000000G                                 .WORD   MSG00
000426  000000G                                 .WORD   ERROR$REPORT
000430  062706  000014                          ADD     #14,SP                          ;                                       3615
000434  005202                          7$:     INC     R2                              ; INDEX                                 3611
000436  020227  000021                          CMP     R2,#21                          ; INDEX,*
000442  003712                                  BLE     6$
000444  005002                                  CLR     R2                              ; INDEX                                 3624
000446  010201                          8$:     MOV     R2,R1                           ; INDEX,*                               3625
000450  006301                                  ASL     R1
000452  010200                                  MOV     R2,R0                           ; INDEX,*
000454  006300                                  ASL     R0
000456  016061  000044G 000000G                 MOV     XMIT.D.LIST+44(R0),XMIT.D.LIST(R1) ;
000464  005202                                  INC     R2                              ; INDEX                                 3624
000466  020227  000005                          CMP     R2,#5                           ; INDEX,*
000472  003765                                  BLE     8$
000474  012716  140000                          MOV     #-40000,(SP)                    ;                                       3627
000500  012746  000400                          MOV     #400,-(SP)
000504  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
000510  012716  140000                          MOV     #-40000,(SP)                    ;                                       3628
000514  005046                                  CLR     -(SP)
000516  004737  000000G                         JSR     PC,CHK.RCV.STATUS
```

H15


```
000522  005001                                  CLR     R1                              ; INDEX                                 3630
000524  126161  000000G 000000G         9$:     CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                  3631
000532  001447                                  BEQ     10$
000534  013700  000000G                         MOV     REG.ADR,R0                      ;                                       3634
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)                     ;                                       3635
000560  012746  000001                          MOV     #1,-(SP)
000564  010600                                  MOV     SP,R0                           ; SP,*
000566  104414                                  TRAP    14
000570  012716  000000G                         MOV     #MSG51,(SP)                     ;                                       3636
000574  012746  000001                          MOV     #1,-(SP)
000600  010600                                  MOV     SP,R0                           ; SP,*
000602  104414                                  TRAP    14
000604  010116                                  MOV     R1,(SP)                         ; INDEX,*                               3637
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                              ;                                       3638
000640  003103                                  .WORD   3103
000642  000000G                                 .WORD   MSG00
000644  000000G                                 .WORD   ERROR$REPORT
000646  062706  000014                          ADD     #14,SP                          ;                                       3633
000652  005201                          10$:    INC     R1                              ; INDEX                                 3630
000654  020127  000005                          CMP     R1,#5                           ; INDEX,*
000660  003721                                  BLE     9$
000662  062706  000010                          ADD     #10,SP                          ;                                       3591
000666  104467                                  TRAP    67                              ;                                       3639
000670  006000                                  ROR     R0
000672  103002                                  BHIS    11$
000674  000137  014376'                         JMP     4$
000700  004737  000000G             11$:        JSR     PC,RESET.DEQNA                  ;                                       3642
000704  012716  000040                          MOV     #40,(SP)                        ;                                       3643
000710  004737  000000G                         JSR     PC,CLR.BUFFERS
000714  012737  000020  000000G                 MOV     #20,RBUF.LENGTH                 ;                                       3644
000722  012700  000020                          MOV     #20,R0                          ;                                       3645
000726  006200                                  ASR     R0
000730  005400                                  NEG     R0
000732  010037  000000G                         MOV     R0,XBUF.LENGTH
000736  005000                                  CLR     R0                              ; INDEX                                 3646
000740  110060  000000G             12$:        MOVB    R0,XMIT.BUFFER(R0)              ; INDEX,*(INDEX)                        3647
000744  005200                                  INC     R0                              ; INDEX                                 3646
000746  020027  000023                          CMP     R0,#23                          ; INDEX,*
000752  003772                                  BLE     12$
000754  104402                          13$:    TRAP    2                               ;                                       3647
000756  005000                                  CLR     R0                              ; INDEX                                 3650
000760  016060  000000G 000000G         14$:    MOV     TD16(R0),XMIT.D.LIST(R0)        ; *(INDEX),*(INDEX)                     3651
000766  062700  000002                          ADD     #2,R0                           ; *,INDEX                               3650
```

```
000772  020027  000126                          CMP     R0,#126                 ; INDEX,*
000776  003770                                  BLE     14$
001000  012737  100000  000046G                MOV     #-100000,XMIT.D.LIST+46 ;                       3653
001006  012737  040000  000062G                MOV     #40000,XMIT.D.LIST+62   ;                       3654
001014  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                       3656
001020  012746  120000                          MOV     #-60000,-(SP)
001024  004737  000000G                         JSR     PC,SET.RDESCR.LIST
001030  013700  000000G                         MOV     REG.ADR,R0              ;                       3657
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  ;                       3658
001054  005016                                  CLR     (SP)                    ;                       3659
001056  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
001062  012716  100220                          MOV     #-77560,(SP)            ;                       3661
001066  011646                                  MOV     (SP),-(SP)
001070  004737  000000G                         JSR     PC,CHK.CSR.STATUS
001074  012737  120000  000046G                MOV     #-60000,XMIT.D.LIST+46  ;                       3663
001102  012737  020000  000062G                MOV     #20000,XMIT.D.LIST+62   ;                       3664
001110  005002                                  CLR     R2                      ; INDEX                 3670
001112  010201                          15$:    MOV     R2,R1                   ; INDEX,*               3671
001114  006301                                  ASL     R1
001116  026161  000000G 000000G                CMP     XMIT.D.LIST(R1),TD16(R1)
001124  001454                                  BEQ     16$
001126  016100  000000G                         MOV     XMIT.D.LIST(R1),R0      ;                       3672
001132  042700  037777                          BIC     #37777,R0
001136  020027  140000                          CMP     R0,#-40000
001142  001445                                  BEQ     16$
001144  013700  000000G                         MOV     REG.ADR,R0              ;                       3675
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)             ;                       3676
001170  012746  000001                          MOV     #1,-(SP)
001174  010600                                  MOV     SP,R0                   ; SP,*
001176  104414                                  TRAP    14
001200  012716  000000G                         MOV     #MSG49,(SP)             ;                       3677
001204  012746  000001                          MOV     #1,-(SP)
001210  010600                                  MOV     SP,R0                   ; SP,*
001212  104414                                  TRAP    14
001214  016116  000000G                         MOV     XMIT.D.LIST(R1),(SP)    ;                       3678
001220  010246                                  MOV     R2,-(SP)                ; INDEX,*
001222  012746  000000G                         MOV     #XMIT.D.LIST,-(SP)
001226  012746  000000G                         MOV     #MSG76,-(SP)
001232  012746  000004                          MOV     #4,-(SP)
001236  010600                                  MOV     SP,R0                   ; SP,*
001240  104414                                  TRAP    14
001242  104455                                  TRAP    55                      ;                       3679
001244  003104                                  .WORD   3104
001246  000000G                                 .WORD   MSG00
001250  000000G                                 .WORD   ERROR$REPORT
001252  062706  000014                          ADD     #14,SP                  ;                       3674
001256  005202                          16$:    INC     R2                      ; INDEX                 3680
001260  020227  000043                          CMP     R2,#43                  ; INDEX,*
001264  003712                                  BLE     15$
```

J15


```
001266  005002                          CLR     R2                              ; INDEX                                 3682
001270  010201                  17$:    MOV     R2,R1                           ; INDEX,*                               3683
001272  006301                          ASL     R1
001274  010200                          MOV     R2,R0                           ; INDEX,*
001276  006300                          ASL     R0
001300  016061  000110G 000000G         MOV     XMIT.D.LIST+110(R0),XMIT.D.LIST(R1) ;
001306  005202                          INC     R2                              ; INDEX                                 3682
001310  020227  000005                  CMP     R2,#5                           ; INDEX,*
001314  003765                          BLE     17$
001316  012716  140000                  MOV     #-40000,(SP)                    ;                                       3685
001322  012746  000400                  MOV     #400,-(SP)
001326  004737  000000G                 JSR     PC,CHK.XMIT.STATUS
001332  012716  140000                  MOV     #-40000,(SP)                    ;                                       3686
001336  012746  020000                  MOV     #20000,-(SP)
001342  004737  000000G                 JSR     PC,CHK.RCV.STATUS
001346  005001                          CLR     R1                              ; INDEX                                 3689
001350  126161  000000G 000000G 18$:    CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                     3690
001356  001447                          BEQ     19$
001360  013700  000000G                 MOV     REG.ADR,R0                      ;                                       3693
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)                     ;                                       3694
001404  012746  000001                  MOV     #1,-(SP)
001410  010600                          MOV     SP,R0                           ; SP,*
001412  104414                          TRAP    14
001414  012716  000000G                 MOV     #MSG51,(SP)                     ;                                       3695
001420  012746  000001                  MOV     #1,-(SP)
001424  010600                          MOV     SP,R0                           ; SP,*
001426  104414                          TRAP    14
001430  010116                          MOV     R1,(SP)                         ; INDEX,*                               3696
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                              ;                                       3697
001464  003105                          .WORD   3105
001466  000000G                         .WORD   MSG00
001470  000000G                         .WORD   ERROR$REPORT
001472  062706  000014                  ADD     #14,SP                          ;                                       3692
001476  005201                  19$:    INC     R1                              ; INDEX                                 3689
001500  020127  000005                  CMP     R1,#5                           ; INDEX,*
001504  003721                          BLE     18$
001506  012701  000006                  MOV     #6,R1                           ; *,INDEX                               3700
001512  105761  000000G         20$:    TSTB    RCV.BUFFER(R1)                  ; *(INDEX)                              3701
001516  001445                          BEQ     21$
001520  013700  000000G                 MOV     REG.ADR,R0                      ;                                       3704
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)                     ;                                       3705
```

K15


```
001544  012746  000001                          MOV     #1,(SP)
001550  010600                                  MOV     SP,R0                           ; SP,*
001552  104414                                  TRAP    14
001554  012716  000000G                         MOV     #MSG51,(SP)                     ;                                       3706
001560  012746  000001                          MOV     #1,-(SP)
001564  010600                                  MOV     SP,R0                           ; SP,*
001566  104414                                  TRAP    14
001570  010116                                  MOV     R1,(SP)                         ; INDEX,*                               3707
001572  005046                                  CLR     -(SP)
001574  005046                                  CLR     -(SP)
001576  116116  000000G                         MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
001602  012746  000000G                         MOV     #MSG50,-(SP)
001606  012746  000004                          MOV     #4,-(SP)
001612  010600                                  MOV     SP,R0                           ; SP,*
001614  104414                                  TRAP    14
001616  104455                                  TRAP    55                              ;                                       3708
001620  003106                                  .WORD   3106
001622  000000G                                 .WORD   MSG00
001624  000000G                                 .WORD   ERROR$REPORT
001626  062706  000014                          ADD     #14,SP                          ;                                       3703
001632  005201                          21$:    INC     R1                              ; INDEX                                 3700
001634  020127  000011                          CMP     R1,#11                          ; INDEX,*
001640  003724                                  BLE     20$
001642  005001                                  CLR     R1                              ; INDEX                                 3711
001644  126161  000012G 000162G          22$:    CMPB    RCV.BUFFER+12(R1),TARGET.ADR+162(R1) ;
                                                                                        ; *(INDEX),*(INDEX)                     3712
001652  001447                                  BEQ     23$
001654  013700  000000G                         MOV     REG.ADR,R0                      ;                                       3715
001660  016066  000016  000024                  MOV     16(R0),24(SP)                   ; *,TMP.LOCATION
001666  016637  000024  000000G                 MOV     24(SP),CSR.WORD                 ; TMP.LOCATION,*
001674  012716  000000G                         MOV     #MSG59,(SP)                     ;                                       3716
001700  012746  000001                          MOV     #1,-(SP)
001704  010600                                  MOV     SP,R0                           ; SP,*
001706  104414                                  TRAP    14
001710  012716  000000G                         MOV     #MSG51,(SP)                     ;                                       3717
001714  012746  000001                          MOV     #1,-(SP)
001720  010600                                  MOV     SP,R0                           ; SP,*
001722  104414                                  TRAP    14
001724  010116                                  MOV     R1,(SP)                         ; INDEX,*                               3718
001726  005046                                  CLR     -(SP)
001730  116116  000000G                         MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
001734  005046                                  CLR     -(SP)
001736  116116  000000G                         MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
001742  012746  000000G                         MOV     #MSG50,-(SP)
001746  012746  000004                          MOV     #4,-(SP)
001752  010600                                  MOV     SP,R0                           ; SP,*
001754  104414                                  TRAP    14
001756  104455                                  TRAP    55                              ;                                       3719
001760  003107                                  .WORD   3107
001762  000000G                                 .WORD   MSG00
001764  000000G                                 .WORD   ERROR$REPORT
001766  062706  000014                          ADD     #14,SP                          ;                                       3714
001772  005201                          23$:    INC     R1                              ; INDEX                                 3711
```

L15


```
001774  020127  000005                          CMP     R1,#5                           ; INDEX,*
002000  003721                                  BLE     22$
002002  062706  000010                          ADD     #10,SP                          ;                                       3647
002006  104467                                  TRAP    67                              ;                                       3720
002010  006000                                  ROR     R0
002012  103002                                  BHIS    24$
002014  000137  015204'                        JMP     13$
002020  062706  000016                  24$:    ADD     #16,SP                          ;                                       3518
002024  000207                                  RTS     PC

; Routine Size:  523 words,     Routine Base:  AB$CODE$ + 14230
; Maximum stack depth per invocation:  22 words




                                              .SBTTL  T16 TEST 16 - ODD PACKET TEST
000000  004737  014230'                  T16::
000000                                    1$:     JSR     PC,$T16                         ;                                       3721
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 16256
; Maximum stack depth per invocation:  2 words


;   3724  1
```

M15


```
;  3725  1     %SBTTL 'TEST 17   STATION ADDRESS TEST'
;  3726  1     !++
;  3727  1     !
;  3728  1     !  TEST 17:     STATION ADDRESS TEST
;  3729  1     !
;  3730  1     !  DESCRIPTION:
;  3731  1     !
;  3732  1     !       This test verifies that DEQNA accepts only packets with legitimate
;  3733  1     !       'multicast' and 'non-multicast' addresses and discards those with
;  3734  1     !       illegitimate 'multicast' and 'non-multicast' addresses.
;  3735  1     !
;  3736  1     !       Station Address RAM is loaded with a set of Target Addresses and
;  3737  1     !       Mode bits. Target Addresses in and out of the set are used to
;  3738  1     !       loopback packets. If the operator specifies loop on error, the
;  3739  1     !       program re-executes the code that detected the error until ^C is
;  3740  1     !       entered.
;  3741  1     !
;  3742  1     !       Hardware tested:        Address Filter Circuitry
;  3743  1     !
;  3744  1     !       Set of 'multicast' addresses in HEXADECIMAL:
;  3745  1     !
;  3746  1     !               01-00-00 00-00-00
;  3747  1     !               AB-AA-AA-AA-AA-AA
;  3748  1     !               55-55-55-55-55-55
;  3749  1     !               FF-FF-FF-FF-FF-FF
;  3750  1     !               Walking 1
;  3751  1     !
;  3752  1     !       Processing:
;  3753  1     !
;  3754  1     !            BEGIN
;  3755  1     !               reset device
;  3756  1     !               select internal loopback mode
;  3757  1     !               set mode to Setup
;  3758  1     !               load Station Address RAM with 'multicast' addresses
;  3759  1     !               REPEAT for each complemented and uncomplemented 'multicast'
;  3760  1     !                      address in the set
;  3761  1     !                  load address
;  3762  1     !                  disable receiver
;  3763  1     !                  transmit loopback packet
;  3764  1     !                  enable receiver
;  3765  1     !                  check for expected loopback status
;  3766  1     !                  IF error
;  3767  1     !                  THEN
;  3768  1     !                     print error message if not inhibited
;  3769  1     !                  ENDIF
;  3770  1     !                  call compare_packets
;  3771  1     !               ENDREPEAT
;  3772  1     !            END
;  3773  1     !
```

```
;    3774  3     BGNTST;
;    3775  3
;    3776  3       !++
;    3777  3       !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM TO ALL MULTICAST
;    3778  3       !  MODE.
;    3779  3       !
;    3780  3
;    3781  3       RESET_DEQNA ( );
;    3782  3       PREP_FOR_SETUP ( );
;    3783  3       INCR INDEX1 FROM 6 TO 19 DO
;    3784  3         WRT_STATION_ADR ( .INDEX1 - 5, .INDEX1 );
;    3785  3
;    3786  5       BGNSUB;
;    3787  5         XMIT_SETUP_PACKET ( N_MODE );
;    3788  3       ENDSUB;
;    3789  3
;    3790  3       !++
;    3791  3       !  NOW LOOPBACK 6 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;    3792  3       !--
;    3793  3
;    3794  3       RBUF_LENGTH = 6;
;    3795  3       XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    3796  3
;    3797  3       INCR INDEX1 FROM 6 TO 19 DO
;    3798  4         BEGIN
;    3799  4           WRT_STATION_ADR ( ZERO, .INDEX1 );
;    3800  4
;    3801  6           BGNSUB;
;    3802  6             XMIT_ILOOP_PACKET ( ZERO );
;    3803  4           ENDSUB;
;    3804  4
;    3805  4           INCR INDEX2 FROM 0 TO 5 DO
;    3806  5             BEGIN
;    3807  5               XMIT_BUFFER [ .INDEX2 ] = ( -.XMIT_BUFFER [ .INDEX2 ] ) - 1;
;    3808  5               TARGET_ADR [ .INDEX2 ] = .XMIT_BUFFER [ .INDEX2 ];
;    3809  4             END;
;    3810  4
;    3811  6           BGNSUB;
;    3812  6             XMIT_ILOOP_PACKET ( ONE );
;    3813  4           ENDSUB;
;    3814  3         END;
;    3815  3
;    3816  3       TEMP4 = 14;
;    3817  3       INCR INDEX3 FROM 0 TO 3 DO
;    3818  4         BEGIN
;    3819  4           IF .INDEX3 EQLU 3
;    3820  4             THEN
;    3821  4               TEMP4 = 6;
;    3822  4           RESET_DEQNA ( );
;    3823  4           PREP_FOR_SETUP ( );
;    3824  4           INCR INDEX4 FROM 1 TO .TEMP4 DO
;    3825  5             BEGIN
;    3826  5               WALKING_BIT ( ZERO, .INDEX4 + ( .INDEX3 * 14 ) - 1, 5 );
```

B10


```
;     3827  5            WRT_STATION_ADR ( .INDEX4, ZERO );
;     3828  4          END;
;     3829  4
;     3830  6        BGNSUB;
;     3831  6          XMIT_SETUP_PACKET ( N_MODE );
;     3832  4        ENDSUB;
;     3833  4
;     3834  4        RBUF_LENGTH = 6;
;     3835  4        XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;     3836  4
;     3837  4        INCR INDEX4 FROM 1 TO .TEMP4 DO
;     3838  5          BEGIN
;     3839  5            WALKING_BIT ( ZERO, .INDEX4 + ( .INDEX3 * 14 ) - 1, 5 );
;     3840  5            WRT_STATION_ADR ( ZERO, ZERO );
;     3841  5
;     3842  7            BGNSUB;
;     3843  7              XMIT_ILOOP_PACKET ( ZERO );
;     3844  5            ENDSUB;
;     3845  4          END;
;     3846  4
;     3847  4        INCR INDEX2 FROM 0 TO 5 DO
;     3848  5          BEGIN
;     3849  5            XMIT_BUFFER [ .INDEX2 ] = ( -.XMIT_BUFFER [ .INDEX2 ] ) - 1;
;     3850  5            TARGET_ADR [ .INDEX2 ] = .XMIT_BUFFER [ .INDEX2 ];
;     3851  5
;     3852  7            BGNSUB;
;     3853  7              XMIT_ILOOP_PACKET ( ONE );
;     3854  5            ENDSUB;
;     3855  4          END;
;     3856  3      END;
;     3857  3
;     3858  3      INCR INDEX2 FROM 0 TO 5 DO
;     3859  3        TARGET_ADR [ .INDEX2 ] = ZERO;
;     3860  3
;     3861  1    ENDTST;


                                                    .SBTTL  $T17 TEST 17 - STATION ADDRESS TEST
000000  004137  000000G             $T17:   JSR     R1,$SAVE4                       ;                       3723
000004  004737  000000G                     JSR     PC,RESET.DEQNA                  ;                       3781
000010  004737  000000G                     JSR     PC,PREP.FOR.SETUP               ;                       3782
000014  012701  000006                      MOV     #6,R1                           ; *,INDEX1              3783
000020  010146                      1$:     MOV     R1,-(SP)                        ; INDEX1,*              3784
000022  162716  000005                      SUB     #5,(SP)
000026  010146                              MOV     R1,-(SP)                        ; INDEX1,*
000030  004737  000000G                     JSR     PC,WRT.STATION.ADR
000034  022626                              CMP     (SP)+,(SP)+
000036  005201                              INC     R1                              ; INDEX1                3783
000040  020127  000023                      CMP     R1,#23                          ; INDEX1,*
000044  003765                              BLE     1$
000046  104402                      2$:     TRAP    2                               ;                       3784
000050  012746  000200                      MOV     #200,-(SP)                      ;                       3787
000054  004737  000000G                     JSR     PC,XMIT.SETUP.PACKET
```

C16


```
000060  005726                          TST     (SP)+                           ;                               3784
000062  104467                          TRAP    67                              ;                               3787
000064  006000                          ROR     R0
000066  103767                          BLO     2$
000070  012737  000006  000000G         MOV     #6,RBUF.LENGTH                  ;                               3794
000076  012700  000006                  MOV     #6,R0                           ;                               3795
000102  006200                          ASR     R0
000104  005400                          NEG     R0
000106  010037  000000G                 MOV     R0,XBUF.LENGTH
000112  012702  000006                  MOV     #6,R2                           ; *,INDEX1                      3797
000116  005046                  3$:     CLR     -(SP)                           ;                               3799
000120  010246                          MOV     R2,-(SP)                        ; INDEX1,*
000122  004737  000000G                 JSR     PC,WRT.STATION.ADR
000126  104402                  4$:     TRAP    2
000130  005016                          CLR     (SP)                            ;                               3802
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                        3805
000146  012701  000000G         5$:     MOV     #XMIT.BUFFER,R1                 ;                               3807
000152  060001                          ADD     R0,R1                           ; INDEX2,*
000154  012703  177777                  MOV     #-1,R3
000160  005004                          CLR     R4
000162  151104                          BISB    (R1),R4
000164  160403                          SUB     R4,R3
000166  110311                          MOVB    R3,(R1)
000170  110360  000000G                 MOVB    R3,TARGET.ADR(R0)               ; *,*(INDEX2)                   3808
000174  005200                          INC     R0                              ; INDEX2                        3805
000176  020027  000005                  CMP     R0,#5                           ; INDEX2,*
000202  003761                          BLE     5$
000204  104402                  6$:     TRAP    2                               ;                               3809
000206  012716  000001                  MOV     #1,(SP)                         ;                               3812
000212  004737  000000G                 JSR     PC,XMIT.ILOOP.PACKET
000216  104467                          TRAP    67
000220  006000                          ROR     R0
000222  103770                          BLO     6$
000224  022626                          CMP     (SP)+,(SP)+                     ;                               3798
000226  005202                          INC     R2                              ; INDEX1                        3797
000230  020227  000023                  CMP     R2,#23                          ; INDEX1,*
000234  003730                          BLE     3$
000236  012737  000016  000000G         MOV     #16,TEMP4                       ;                               3816
000244  005004                          CLR     R4                              ; INDEX3                        3817
000246  022727  000000  000003         CMP     #0,#3                           ;                               3819
000254  001003                  7$:     BNE     8$
000256  012737  000006  000000G         MOV     #6,TEMP4                        ;                               3821
000264  004737  000000G         8$:     JSR     PC,RESET.DEQNA                  ;                               3822
000270  004737  000000G                 JSR     PC,PREP.FOR.SETUP               ;                               3823
000274  013702  000000G                 MOV     TEMP4,R2                        ;                               3824
000300  010401                          MOV     R4,R1                           ; INDEX3,*                      3826
000302  070127  000016                  MUL     #16,R1
000306  005003                          CLR     R3                              ; INDEX4                        3824
000310  000417                          BR      10$
```

D16


```
000312  005046                          9$:     CLR     -(SP)                   ;                               3826
000314  010100                                  MOV     R1,R0
000316  060300                                  ADD     R3,R0                   ; INDEX4,*
000320  010046                                  MOV     R0,-(SP)
000322  005316                                  DEC     (SP)
000324  012746  000005                          MOV     #5,-(SP)
000330  004737  000000G                        JSR     PC,WALKING.BIT
000334  010316                                  MOV     R3,(SP)                 ; INDEX4,*                      3827
000336  005046                                  CLR     -(SP)
000340  004737  000000G                        JSR     PC,WRT.STATION.ADR
000344  062706  000010                          ADD     #10,SP                  ;                               3825
000350  005203                          10$:    INC     R3                      ; INDEX4                        3824
000352  020302                                  CMP     R3,R2                   ; INDEX4,*
000354  003756                                  BLE     9$
000356  104402                          11$:    TRAP    2                       ;                               3828
000360  012746  000200                          MOV     #200,-(SP)              ;                               3831
000364  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000370  005726                                  TST     (SP)+                   ;                               3828
000372  104467                                  TRAP    67                      ;                               3831
000374  006000                                  ROR     R0
000376  103767                                  BLO     11$
000400  012737  000006  000000G                MOV     #6,RBUF.LENGTH          ;                               3834
000406  012700  000006                          MOV     #6,R0                   ;                               3835
000412  006200                                  ASR     R0
000414  005400                                  NEG     R0
000416  010037  000000G                        MOV     R0,XBUF.LENGTH
000422  013703  000000G                        MOV     TEMP4,R3                ;                               3837
000426  005002                                  CLR     R2                      ; INDEX4
000430  000426                                  BR      14$
000432  005046                          12$:    CLR     -(SP)                   ;                               3839
000434  010100                                  MOV     R1,R0
000436  060200                                  ADD     R2,R0                   ; INDEX4,*
000440  010046                                  MOV     R0,-(SP)
000442  005316                                  DEC     (SP)
000444  012746  000005                          MOV     #5,-(SP)
000450  004737  000000G                        JSR     PC,WALKING.BIT
000454  005016                                  CLR     (SP)                    ;                               3840
000456  005046                                  CLR     -(SP)
000460  004737  000000G                        JSR     PC,WRT.STATION.ADR
000464  104402                          13$:    TRAP    2
000466  005016                                  CLR     (SP)                    ;                               3843
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                  ;                               3838
000506  005202                          14$:    INC     R2                      ; INDEX4                        3837
000510  020203                                  CMP     R2,R3                   ; INDEX4,*
000512  003747                                  BLE     12$
000514  005001                                  CLR     R1                      ; INDEX2                        3847
000516  012700  000000G                15$:    MOV     #XMIT.BUFFER,R0         ;                               3849
000522  060100                                  ADD     R1,R0                   ; INDEX2,*
000524  012702  177777                          MOV     #-1,R2
```

E16


```
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)                   3850
000544  104402                  16$:    TRAP    2
000546  012746  000001                  MOV     #1,-(SP)                ;                               3853
000552  004737  000000G                 JSR     PC,XMIT.ILOOP.PACKET
000556  005726                          TST     (SP)+                   ;                               3850
000560  104467                          TRAP    67                      ;                               3853
000562  006000                          ROR     R0
000564  103767                          BLO     16$
000566  005201                          INC     R1                      ; INDEX2                        3847
000570  020127  000005                  CMP     R1,#5                   ; INDEX2,*
000574  003750                          BLE     15$
000576  005204                          INC     R4                      ; INDEX3                        3817
000600  020427  000003                  CMP     R4,#3                   ; INDEX3,*
000604  003623                          BLE     7$
000606  005000                          CLR     R0                      ; INDEX2                        3858
000610  105060  000000G                 CLRB    TARGET.ADR(R0)          ; *(INDEX2)                     3859
000614  005200                          INC     R0                      ; INDEX2                        3858
000616  020027  000005                  CMP     R0,#5                   ; INDEX2,*
000622  003772                          BLE     17$
000624  000207                          RTS     PC                      ;                               3723

; Routine Size:  203 words,     Routine Base:  AB$CODE$ + 16272
; Maximum stack depth per invocation:  11 words



                                        .SBTTL  T17 TEST 17 - STATION ADDRESS TEST
000000  004737  016272'          T17::
000000                          1$:     JSR     PC,$T17                 ;                               3859
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 17120
; Maximum stack depth per invocation:  2 words


;   3862  1
```

F16


```
;  3863  1     %SBTTL 'TEST 18   ALL MULTICAST STATION ADDRESS TEST'
;  3864  1     !++
;  3865  1     !
;  3866  1     !  TEST 18:     ALL MULTICAST STATION ADDRESS TEST
;  3867  1     !
;  3868  1     !  DESCRIPTION:
;  3869  1     !
;  3870  1     !       This test vefifies that DEQNA recognizes 'all multicast' addresses of
;  3871  1     !       the node and discards loopback packets with non-enabled addresses.
;  3872  1     !       If the operator specifies loop on error, the program re-executes the
;  3873  1     !       code that detected the error until ^C is entered.
;  3874  1     !
;  3875  1     !       Hardware tested:        All Multicast Addressing
;  3876  1     !                               I8051 Microprocessor
;  3877  1     !                               Address Filter Circuitry
;  3878  1     !
;  3879  1     !       Set of  'all multicast' addresses:
;  3880  1     !
;  3881  1     !               DEQNA Physical Addr     FF-FF-FF-FF-FF-FF
;  3882  1     !               AA-00-00-00-00-00       55-55-55-55-55-55
;  3883  1     !               AA-00-02-AA-AA-AA       AA-AA-AA-AA-AA-AA
;  3884  1     !               AA 00-05-55-55-55       01-00-00-00-00-00
;  3885  1     !               AA-00-04-FF-FF-FF       AB-AA-AA-AA-AA-AA
;  3886  1     !               AA-00-04-00-00-00       FF-00-01-02-03-04
;  3887  1     !               AA-00-04-18-81-18       00-F4-FA-44-44-55
;  3888  1     !
;  3889  1     !       Processing:
;  3890  1     !
;  3891  1     !            BEGIN
;  3892  1     !               reset device
;  3893  1     !               select internal loopback mode
;  3894  1     !               set mode to Setup
;  3895  1     !               load Station Address RAM with 'all multicast' addresses
;  3896  1     !               REPEAT for 'all multicast' addresses in and out of set
;  3897  1     !                  load 'all multicast' address of the packet
;  3898  1     !                  disable receiver
;  3899  1     !                  transmit loopback packet
;  3900  1     !                   enable receiver
;  3901  1     !                  check for expected loopback status
;  3902  1     !                  IF error
;  3903  1     !                  THEN
;  3904  1     !                     print error message if not inhibited
;  3905  1     !                  ENDIF
;  3906  1     !                  call compare_packets
;  3907  1     !               ENDREPEAT
;  3908  1     !            END
;  3909  1     !--
```

```
;  3910  3     BGNTST;
;  3911  3
;  3912  3       !++
;  3913  3       !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM IF EXECUTING
;  3914  3       !  TESTS IN EXTERNAL LOOPBACK MODE.
;  3915  3       !--
;  3916  3
;  3917  3       RESET_DEQNA ( );
;  3918  3       PREP_FOR_SETUP ( );
;  3919  3       INCR INDEX1 FROM 1 TO 13 DO
;  3920  3         WRT_STATION_ADR ( .INDEX1, .INDEX1 );
;  3921  3       WRT_STATION_ADR ( 14, PHA_INDEX );
;  3922  3
;  3923  5       BGNSUB;
;  3924  5         XMIT_SETUP_PACKET ( A_MODE );
;  3925  3       ENDSUB;
;  3926  3
;  3927  3       !++
;  3928  3       !  NOW LOOPBACK 6 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;  3929  3       !--
;  3930  3
;  3931  3       RBUF_LENGTH = 6;
;  3932  3       XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  3933  3
;  3934  3       INCR INDEX FROM 6 TO 19 DO
;  3935  4         BEGIN
;  3936  4           WRT_STATION_ADR ( ZERO, .INDEX );
;  3937  4
;  3938  6           BGNSUB;
;  3939  6             XMIT_ILOOP_PACKET ( ZERO );
;  3940  4           ENDSUB;
;  3941  4
;  3942  4           INCR INDEX2 FROM 0 TO 5 DO
;  3943  5             BEGIN
;  3944  5               XMIT_BUFFER [ .INDEX2 ] = ( -.XMIT_BUFFER [ .INDEX2 ] ) - 1;
;  3945  5               TARGET_ADR [ .INDEX2 ] = .XMIT_BUFFER [ .INDEX2 ];
;  3946  4             END;
;  3947  4
;  3948  4           XMIT_BUFFER [ ZERO ] = .XMIT_BUFFER [ ZERO ] AND %O'177774';
;  3949  4           TARGET_ADR [ ZERO ] = .XMIT_BUFFER [ ZERO ];
;  3950  4
;  3951  6           BGNSUB;
;  3952  6             XMIT_ILOOP_PACKET ( ONE );
;  3953  4           ENDSUB;
;  3954  4
;  3955  3         END;
;  3956  3
;  3957  3         INCR INDEX2 FROM 0 TO 5 DO
;  3958  3           TARGET_ADR [ .INDEX2 ] = ZERO;
;  3959  3
;  3960  1     ENDTST;
```

H16


```
                                                .SBTTL  $T18 TEST 18   ALL MULTICAST STATION ADDRESS TEST
000000  004137  000000G                $T18:   JSR     R1,$SAVE4                       ;                                       3861
000004  004737  000000G                        JSR     PC,RESET.DEQNA                  ;                                       3917
000010  004737  000000G                        JSR     PC,PREP.FOR.SETUP               ;                                       3918
000014  012701  000001                        MOV     #1,R1                           ; *,INDEX1                              3919
000020  010146                        1$:     MOV     R1,-(SP)                        ; INDEX1,*                              3920
000022  010146                                MOV     R1,-(SP)                        ; INDEX1,*
000024  004737  000000G                        JSR     PC,WRT.STATION.ADR
000030  022626                                CMP     (SP)+,(SP)+
000032  005201                                INC     R1                              ; INDEX1                                3919
000034  020127  000015                        CMP     R1,#15                          ; INDEX1,*
000040  003767                                BLE     1$
000042  012746  000016                        MOV     #16,-(SP)                       ;                                       3921
000046  012746  000023                        MOV     #23,-(SP)
000052  004737  000000G                        JSR     PC,WRT.STATION.ADR
000056  104402                        2$:     TRAP    2
000060  012716  000201                        MOV     #201,(SP)                       ;                                       3924
000064  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000070  104467                                TRAP    67
000072  006000                                ROR     R0
000074  103770                                BLO     2$
000076  012737  000006  000000G               MOV     #6,RBUF.LENGTH                  ;                                       3931
000104  012700  000006                        MOV     #6,R0                           ;                                       3932
000110  006200                                ASR     R0
000112  005400                                NEG     R0
000114  010037  000000G                        MOV     R0,XBUF.LENGTH
000120  012702  000006                        MOV     #6,R2                           ; *,INDEX                               3934
000124  005016                        3$:     CLR     (SP)                            ;                                       3936
000126  010246                                MOV     R2,-(SP)                        ; INDEX,*
000130  004737  000000G                        JSR     PC,WRT.STATION.ADR
000134  104402                        4$:     TRAP    2
000136  005016                                CLR     (SP)                            ;                                       3939
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                                3942
000154  012701  000000G                5$:     MOV     #XMIT.BUFFER,R1                 ;                                       3944
000160  060001                                ADD     R0,R1                           ; INDEX2,*
000162  012703  177777                        MOV     #-1,R3
000166  005004                                CLR     R4
000170  151104                                BISB    (R1),R4
000172  160403                                SUB     R4,R3
000174  110311                                MOVB    R3,(R1)                         ; *,*(INDEX2)                           3945
000176  110360  000000G                       MOVB    R3,TARGET.ADR(R0)               ; INDEX2                                3942
000202  005200                                INC     R0                              ; INDEX2,*
000204  020027  000005                        CMP     R0,#5
000210  003761                                BLE     5$                                                                      3948
000212  142737  000003  000000G               BICB    #3,XMIT.BUFFER                  ;                                       3949
000220  113737  000000G 000000G               MOVB    XMIT.BUFFER,TARGET.ADR          ;
000226  104402                        6$:     TRAP    2                                                                       3952
000230  012716  000001                        MOV     #1,(SP)                         ;
000234  004737  000000G                        JSR     PC,XMIT.ILOOP.PACKET
```

```
000240  104467                                  TRAP    67
000242  006000                                  ROR     R0
000244  103770                                  BLO     6$
000246  005726                                  TST     (SP)+                           ;                                       3935
000250  005202                                  INC     R2                              ; INDEX                                 3934
000252  020227  000023                          CMP     R2,#23                          ; INDEX,*
000256  003722                                  BLE     3$
000260  005000                                  CLR     R0                              ; INDEX2                                3957
000262  105060  000000G                 7$:     CLRB    TARGET.ADR(R0)                  ; *(INDEX2)                             3958
000266  005200                                  INC     R0                              ; INDEX2                                3957
000270  020027  000005                          CMP     R0,#5                           ; INDEX2,*
000274  003772                                  BLE     7$
000276  022626                                  CMP     (SP)+,(SP)+                     ;                                       3861
000300  000207                                  RTS     PC

; Routine Size:  97 words,      Routine Base:  AB$CODE$ + 17134
; Maximum stack depth per invocation:  10 words



                                                .SBTTL  T18 TEST 18 - ALL MULTICAST STATION ADDRESS TEST
000000  004737  017134'                  T18::
000000                                  1$:     JSR     PC,$T18                         ;                                       3958
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 17436
; Maximum stack depth per invocation:  2 words


;   3961  1
;   3962  1
```

J16


```
;  3963  1     %SBTTL 'TEST 19 - RUNT PACKET TEST'
;  3964  1     !++
;  3965  1     !
;  3966  1     !  TEST 19:     RUNT PACKET TEST
;  3967  1     !
;  3968  1     !  DESCRIPTION:
;  3969  1     !
;  3970  1     !       This test verifies that the DEQNA can detect runt packets in FIFO.
;  3971  1     !       If the operator specifies loop on error, the program re-executes the
;  3972  1     !       code that detected the error until ^C is entered.
;  3973  1     !
;  3974  1     !       Hardware tested:        EPP
;  3975  1     !                               Address Filter Circuitry
;  3976  1     !
;  3977  1     !       Station Address table:
;  3978  1     !
;  3979  1     !                       DEQNA Physical Addr
;  3980  1     !                       AA-00-00-00-00-00
;  3981  1     !                       AA-00-02-AA-AA-AA
;  3982  1     !                       AA-00-05-55-55-55
;  3983  1     !                       AA-00-04-FF-FF-FF
;  3984  1     !                       AA-00-04-00-00-00
;  3985  1     !                       AA 00-04-18-81-18
;  3986  1     !
;  3987  1     !       Processing:
;  3988  1     !
;  3989  1     !            BEGIN
;  3990  1     !               reset device
;  3991  1     !               select internal loopback mode
;  3992  1     !               load Station Address RAM with Station Addresses from table
;  3993  1     !               load packet with valid Station Address
;  3994  1     !               disable receiver
;  3995  1     !               transmit loopback packet
;  3996  1     !               enable receiver
;  3997  1     !               check for expected loopback status
;  3998  1     !               IF error
;  3999  1     !               THEN
;  4000  1     !                  print error message if not inhibited
;  4001  1     !               ENDIF
;  4002  1     !               load packet with invalid Station Address
;  4003  1     !               disable receiver
;  4004  1     !               transmit loopback packet
;  4005  1     !               enable receiver
;  4006  1     !               check for expected loopback status
;  4007  1     !               IF error
;  4008  1     !               THEN
;  4009  1     !                  print error message if not inhibited
;  4010  1     !               ENDIF
;  4011  1     !            END
;  4012  1     !--
```

K16


```
;  4013  3     BGNTST;
;  4014  3
;  4015  3       !++
;  4016  3       !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM IF EXECUTING
;  4017  3       !  TESTS IN EXTERNAL LOOPBACK MODE.
;  4018  3       !--
;  4019  3
;  4020  3       RESET_DEQNA ( );
;  4021  3       PREP_FOR_SETUP ( );
;  4022  3       INCR INDEX1 FROM 6 TO 19 DO
;  4023  3         WRT_STATION_ADR ( .INDEX1   5, PHA_INDEX );
;  4024  3
;  4025  5       BGNSUB;
;  4026  5             XMIT_SETUP_PACKET ( N_MODE );
;  4027  3       ENDSUB;
;  4028  3
;  4029  3       !++
;  4030  3       !  NOW LOOPBACK 6 BYTE PACKETS AND CHECK IF THEY ARE RECEIVED PROPERLY
;  4031  3       !--
;  4032  3
;  4033  3       RBUF_LENGTH = 6;
;  4034  3       XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;  4035  3
;  4036  3       WRT_STATION_ADR ( ZERO, PHA_INDEX );
;  4037  3
;  4038  5       BGNSUB;
;  4039  5         XMIT_ILOOP_PACKET ( ZERO );
;  4040  3       ENDSUB;
;  4041  3
;  4042  5       BGNSUB;
;  4043  5         WRT_STATION_ADR ( ZERO, 2 );
;  4044  5
;  4045  5         .IOP_TABLE [ CSR ] = ONE;
;  4046  5
;  4047  5         SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  4048  5         .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;  4049  5         .IOP_TABLE [ RHI_ADR ] = ZERO;
;  4050  5
;  4051  5         SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  4052  5         .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;  4053  5         .IOP_TABLE [ XHI_ADR ] = ZERO;
;  4054  5
;  4055  5         CHK_RIXI_STATUS ( ZERO );
;  4056  5         CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK      );      ! 0'100220', 0'100220'
;  4057  5         CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );      ! 0'140000', 0'000400'
;  4058  5         CHK_RCV_STATUS  ( RFLG_STATUS, RWD16_STATUS );      ! 0'140000', 0'044000'
;  4059  5
;  4060  5         .IOP_TABLE [ CSR ] = ZERO;
;  4061  3       ENDSUB;
;  4062  3
;  4063  1     ENDTST;
```

L16


```
                                                .SBTTL  $T19 TEST 19   RUNT PACKET TEST
000000  010146                          $T19:   MOV     R1,-(SP)                        ;                               3960
000002  004737  000000G                        JSR     PC,RESET.DEQNA                  ;                               4020
000006  004737  000000G                        JSR     PC,PREP.FOR.SETUP               ;                               4021
000012  012701  000006                         MOV     #6,R1                           ; *,INDEX1                      4022
000016  010146                          1$:     MOV     R1,-(SP)                        ; INDEX1,*                      4023
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                        4022
000040  020127  000023                         CMP     R1,#23                          ; INDEX1,*
000044  003764                                 BLE     1$
000046  104402                          2$:     TRAP    2                               ;                               4023
000050  012746  000200                         MOV     #200,-(SP)                      ;                               4026
000054  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000060  005726                                 TST     (SP)+                           ;                               4023
000062  104467                                 TRAP    67                              ;                               4026
000064  006000                                 ROR     R0
000066  103767                                 BLO     2$
000070  012737  000006  000000G                MOV     #6,RBUF.LENGTH                  ;                               4033
000076  012700  000006                         MOV     #6,R0                           ;                               4034
000102  006200                                 ASR     R0
000104  005400                                 NEG     R0
000106  010037  000000G                        MOV     R0,XBUF.LENGTH
000112  005046                                 CLR     -(SP)                           ;                               4036
000114  012746  000023                         MOV     #23,-(SP)
000120  004737  000000G                        JSR     PC,WRT.STATION.ADR
000124  104402                          3$:     TRAP    2
000126  005016                                 CLR     (SP)                            ;                               4039
000130  004737  000000G                        JSR     PC,XMIT.ILOOP.PACKET
000134  104467                                 TRAP    67
000136  006000                                 ROR     R0
000140  103771                                 BLO     3$
000142  104402                          4$:     TRAP    2                               ;                               4040
000144  005016                                 CLR     (SP)                            ;                               4043
000146  012746  000002                         MOV     #2,-(SP)
000152  004737  000000G                        JSR     PC,WRT.STATION.ADR
000156  012777  000001  000016G                MOV     #1,@IOP.TABLE+16                ;                               4045
000164  013716  000000G                        MOV     XBUF.LENGTH,(SP)                ;                               4047
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        ;                               4048
000206  005077  000006G                        CLR     @IOP.TABLE+6                    ;                               4049
000212  013716  000000G                        MOV     XBUF.LENGTH,(SP)                ;                               4051
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      ;                               4052
000234  005077  000012G                        CLR     @IOP.TABLE+12                   ;                               4053
000240  005016                                 CLR     (SP)                            ;                               4055
000242  004737  000000G                        JSR     PC,CHK.RIXI.STATUS
000246  012716  100220                         MOV     #-77560,(SP)                    ;                               4056
000252  011646                                 MOV     (SP),-(SP)
```

M16


```
000254  004737  0000000G                JSR     PC,CHK.CSR.STATUS
000260  012716  140000                  MOV     #-40000,(SP)            ;                4057
000264  012746  000400                  MOV     #400,-(SP)
000270  004737  0000000G                JSR     PC,CHK.XMIT.STATUS
000274  012716  140000                  MOV     #-40000,(SP)            ;                4058
000300  012746  044000                  MOV     #44000,-(SP)
000304  004737  0000000G                JSR     PC,CHK.RCV.STATUS
000310  005077  000016G                 CLR     @IOP.TABLE+16           ;                4060
000314  062706  000014                  ADD     #14,SP                  ;                4040
000320  104467                         TRAP    67                      ;                4060
000322  006000                         ROR     R0
000324  103706                         BLO     4$
000326  022626                         CMP     (SP)+,(SP)+             ;                3960
000330  012601                         MOV     (SP)+,R1
000332  000207                         RTS     PC

; Routine Size:  110 words,     Routine Base:  AB$CODE$ + 17452
; Maximum stack depth per invocation:  10 words



                                       .SBTTL  T19 TEST 19 - RUNT PACKET TEST
000000  004737  017452'                T19::
000000                                 1$:     JSR     PC,$T19                 ;                4061
000004  104466                         TRAP    66
000006  006000                         ROR     R0
000010  103773                         BLO     1$
000012  000207                         RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 20006
; Maximum stack depth per invocation:  2 words


;   4064  1
;   4065  1
```

```
;  4066  1     %SBTTL 'TEST 20   FIFO OVERFLOW TEST'
;  4067  1     !++
;  4068  1     !
;  4069  1     !  TEST 20:     FIFO OVERFLOW TEST
;  4070  1     !
;  4071  1     !  DESCRIPTION:
;  4072  1     !
;  4073  1     !       This test verifies that the Ethernet Protocol Processor can
;  4074  1     !       detect receive FIFO overflow condition. If the operator specifies
;  4075  1     !       loop on error, the program re-executes the code that detected the
;  4076  1     !       error until ^C is entered.
;  4077  1     !
;  4078  1     !       Hardware tested:  RCV Status wd 1 - error summary (bit 14),
;  4079  1     !                                           FIFO overflow (bit 0),
;  4080  1     !                                           Byte FIFO in the EDLC,
;  4081  1     !                                           and discard packet (bit 12)
;  4082  1     !       Processing:
;  4083  1     !
;  4084  1     !            BEGIN
;  4085  1     !               reset device
;  4086  1     !               select loopback mode
;  4087  1     !               enable receiver ( set CSR bit 0)
;  4088  1     !               transmit loopback packet
;  4089  1     !               transmit another loopback packet
;  4090  1     !               check for expected loopback status
;  4091  1     !               IF error
;  4092  1     !               THEN
;  4093  1     !                  print error message if not inhibited
;  4094  1     !               ENDIF
;  4095  1     !
;  4096  1     !               reset device
;  4097  1     !               transmit loopback packet
;  4098  1     !               transmit a packet
;  4099  1     !               setup Receive Descriptor List
;  4100  1     !               enable receiver (set CSR BIT 0)
;  4101  1     !               check for expected loopback status
;  4102  1     !               IF error
;  4103  1     !               THEN
;  4104  1     !                  print error message if not inhibited
;  4105  1     !               ENDIF
;  4106  1     !               turn of 3 LED's on the module
;  4107  1     !            END
;  4108  1     !--
```

```
;  4109  3     BGNTST;
;  4110  3
;  4111  3       !++
;  4112  3       !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM
;  4113  3       !--
;  4114  3
;  4115  3       RESET_DEQNA ( );
;  4116  3       PREP_FOR_SETUP ( );
;  4117  3       INCR INDEX1 FROM 1 TO 14 DO
;  4118  3         WRT_STATION_ADR ( .INDEX1, PHA INDEX );
;  4119  3
;  4120  5       BGNSUB;
;  4121  5         XMIT_SETUP_PACKET ( P_MODE );
;  4122  3       ENDSUB;
;  4123  3
;  4124  3       !++
;  4125  3       !  LOOPBACK 2 6-BYTE PACKETS IN INTERNAL LOOPBACK MODE CHECK IF PACKETS
;  4126  3       !  WERE RECEIVED PROPERLY, SHOULD TRANSMIT AND RECEIVE  PROPERLY.
;  4127  3       !--
;  4128  3
;  4129  3       RBUF_LENGTH = 6;
;  4130  3       XBUF_LE  TH = - ( .RBUF_LENGTH + -1 );
;  4131  3
;  4132  3       INCR INDEX FROM 2 TO 3 DO
;  4133  4         BEGIN
;  4134  4           WRT_STATION_ADR ( ZERO, .INDEX );
;  4135  4
;  4136  6           BGNSUB;
;  4137  6             XMIT_ILOOP_PACKET ( ZERO );
;  4138  4           ENDSUB;
;  4139  3         END;
;  4140  3
;  4141  3       !++
;  4142  3       !  FORCE RECEIVE FIFO OVERFLOW ( RCV STATUS WD 1 - BIT 0 ) BY TRANSMITTING
;  4143  3       !  2 ND 6-BYTE PACKET IN INTERNAL LOOPBACK MODE BEFORE RECEIVING FIRST PACKET
;  4144  3       !--
;  4145  3
;  4146  5       BGNSUB;
;  4147  5         .IOP_TABLE [ CSR ] = ZERO;
;  4148  5
;  4149  5         WRT_STATION_ADR ( ZERO, 2 );
;  4150  5
;  4151  5         SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  4152  5         .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;  4153  5         .IOP_TABLE [ XHI_ADR ] = ZERO;
;  4154  5
;  4155  5         CHK_RIXI_STATUS ( ONE );
;  4156  5         WRT_STATION_ADR ( ZERO, 3 );
;  4157  5
;  4158  5         SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  4159  5         .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;  4160  5         .IOP_TABLE [ XHI_ADR ] = ZERO;
;  4161  5
```

```
;   4162  5          SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;   4163  5          .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;   4164  5          .IOP_TABLE [ RHI_ADR ] = ZERO;
;   4165  5
;   4166  5          .IOP_TABLE [ CSR ] = ONE;
;   4167  5
;   4168  5          CHK_RIXI_STATUS ( ZERO );
;   4169  5          CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK      );      ! 0'100220', 0'100220'
;   4170  5          CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );      ! 0'140000', 0'000400'
;   4171  5          CHK_RX_LPSTATUS  ( RFLG_STATUS, RWD15_STATUS );     ! 0'140000', 0'000001'
;   4172  5
;   4173  5          .IOP_TABLE [ CSR ] = ZERO;
;   4174  3        ENDSUB;
;   4175  3
;   4176  3        RESET_DEQNA ( );
;   4177  3
;   4178  3        TURN_OFF_LED ( N_MODE );
;   4179  3        TURN_OFF_LED ( LED1 );
;   4180  3        TURN_OFF_LED ( LED2 );
;   4181  3        TURN_OFF_LED ( LED3 );
;   4182  3
;   4183  1      ENDTST;


                                        .SBTTL  $T20 TEST 20 - FIFO OVERFLOW TEST
000000  010146                 $T20:   MOV     R1,-(SP)                ;                                       4063
000002  004737  000000G                JSR     PC,RESET.DEQNA          ;                                       4115
000006  004737  000000G                JSR     PC,PREP.FOR.SETUP       ;                                       4116
000012  012701  000001                 MOV     #1,R1                   ; *,INDEX1                              4117
000016  010146                 1$:     MOV     R1,-(SP)                ; INDEX1,*                              4118
000020  012746  000023                 MOV     #23,-(SP)
000024  004737  000000G                JSR     PC,WRT.STATION.ADR
000030  022626                         CMP     (SP)+,(SP)+
000032  005201                         INC     R1                      ; INDEX1                                4117
000034  020127  000016                 CMP     R1,#16                  ; INDEX1,*
000040  003766                         BLE     1$
000042  104402                 2$:     TRAP    2                       ;                                       4118
000044  012746  000202                 MOV     #202,-(SP)              ;                                       4121
000050  004737  000000G                JSR     PC,XMIT.SETUP.PACKET
000054  005726                         TST     (SP)+                   ;                                       4118
000056  104467                         TRAP    67                      ;                                       4121
000060  006000                         ROR     R0
000062  103767                         BLO     2$
000064  012737  000006  000000G       MOV     #6,RBUF.LENGTH          ;                                       4129
000072  012700  000006                 MOV     #6,R0                   ;                                       4130
000076  006200                         ASR     R0
000100  005400                         NEG     R0
000102  010037  000000G                MOV     R0,XBUF.LENGTH
000106  012701  000002                 MOV     #2,R1                   ; *,INDEX                               4132
000112  005046                 3$:     CLR     -(SP)                   ;                                       4134
000114  010146                         MOV     R1,-(SP)                ; INDEX,*
000116  004737  000000G                JSR     PC,WRT.STATION.ADR
000122  104402                 4$:     TRAP    2
```

```
000124  005016                                  CLR     (SP)                    ;                               4137
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)+             ;                               4133
000142  005201                                  INC     R1                      ; INDEX                         4132
000144  020127  000003                         CMP     R1,#3                   ; INDEX,*
000150  003760                                  BLE     3$
000152  104402                          5$:     TRAP    2                       ;                               4139
000154  005077  000016G                         CLR     @IOP.TABLE+16           ;                               4147
000160  005046                                  CLR     -(SP)                   ;                               4149
000162  012746  000002                         MOV     #2,-(SP)
000166  004737  000000G                         JSR     PC,WRT.STATION.ADR
000172  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                               4151
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  ;                           4152
000214  005077  000012G                         CLR     @IOP.TABLE+12           ;                               4153
000220  012716  000001                         MOV     #1,(SP)                 ;                               4155
000224  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
000230  005016                                  CLR     (SP)                    ;                               4156
000232  012746  000003                         MOV     #3,-(SP)
000236  004737  000000G                         JSR     PC,WRT.STATION.ADR
000242  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                               4158
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  ;                           4159
000264  005077  000012G                         CLR     @IOP.TABLE+12           ;                               4160
000270  013716  000000G                         MOV     XBUF.LENGTH,(SP)        ;                               4162
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    ;                           4163
000312  005077  000006G                         CLR     @IOP.TABLE+6            ;                               4164
000316  012777  000001  000016G                 MOV     #1,@IOP.TABLE+16        ;                               4166
000324  005016                                  CLR     (SP)                    ;                               4168
000326  004737  000000G                         JSR     PC,CHK.RIXI.STATUS
000332  012716  100220                         MOV     #-77560,(SP)            ;                               4169
000336  011646                                  MOV     (SP),-(SP)
000340  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000344  012716  140000                         MOV     #-40000,(SP)            ;                               4170
000350  012746  000400                         MOV     #400,-(SP)
000354  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
000360  012716  140000                         MOV     #-40000,(SP)            ;                               4171
000364  012746  000001                         MOV     #1,-(SP)
000370  004737  000000G                         JSR     PC,CHK.RX.LPSTATUS
000374  005077  000016G                         CLR     @IOP.TABLE+16           ;                               4173
000400  062706  000022                         ADD     #22,SP                  ;                               4139
000404  104467                                  TRAP    67                      ;                               4173
000406  006000                                  ROR     R0
000410  103660                                  BLO     5$
000412  004737  000000G                         JSR     PC,RESET.DEQNA          ;                               4176
000416  012746  000200                         MOV     #200,-(SP)              ;                               4178
```

F1


```
000422  004737  000000G                        JSR     PC,TURN.OFF.LED
000426  012716  000204                         MOV     #204,(SP)               ;                       4179
000432  004737  000000G                        JSR     PC,TURN.OFF.LED
000436  012716  000210                         MOV     #210,(SP)               ;                       4180
000442  004737  000000G                        JSR     PC,TURN.OFF.LED
000446  012716  000214                         MOV     #214,(SP)               ;                       4181
000452  004737  000000G                        JSR     PC,TURN.OFF.LED
000456  005726                                 TST     (SP)+                   ;                       4063
000460  012601                                 MOV     (SP)+,R1
000462  000207                                 RTS     PC

; Routine Size:  154 words,     Routine Base:  AB$CODE$ + 20022
; Maximum stack depth per invocation:  11 words



                                               .SBTTL  T20 TEST 20 - FIFO OVERFLOW TEST
000000  004737  020022'                 T20::
000000                                   1$:   JSR     PC,$T20                 ;                       4181
000004  104466                                 TRAP    66
000006  006000                                 ROR     R0
000010  103773                                 BLO     1$
000012  000207                                 RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 20506
; Maximum stack depth per invocation:  2 words


;   4184  1
```

```
;  4185  1     %SBTTL 'TEST 21 - SANITY TIMER TEST'
;  4186  1     !++
;  4187  1     !
;  4188  1     !  TEST 21:     SANITY TIMER TEST
;  4189  1     !
;  4190  1     !  DESCRIPTION:
;  4191  1     !
;  4192  1     !       This test verifies that the Sanity Timer times out after a pre-set
;  4193  1     !       ( supplied by the operator ) timeout period. The Sanity Timer uses
;  4194  1     !       DCOK line on the Q-Bus to force the power_fail interrupt of the
;  4195  1     !        processor which in turn causes the processor to reboot itself.
;  4196  1     !
;  4197  1     !       Hardware tested:  Sanity Timer Logic
;  4198  1     !
;  4199  1     !       Processing:
;  4200  1     !
;  4201  1     !            BEGIN
;  4202  1     !               reset device
;  4203  1     !               store Console Terminal and Power_fail interrupt vectors
;  4204  1     !                 ( location 24 and 60 octal )
;  4205  1     !               enable Console Terminal interrupt
;  4206  1     !               arm for Power_fail interrupt
;  4207  1     !               inform the operator about the test procedure
;  4208  1     !               set the Sanity Timer to timeout value supplied by the
;  4209  1     !                 operator
;  4210  1     !               enable the Sanity Timer
;  4211  1     !               wait
;  4212  1     !               IF Power-fail interrupt occured
;  4213  1     !               THEN
;  4214  1     !                  print 'SANITY TIMER TIMED OUT AS EXPECTED'
;  4215  1     !               ELSE
;  4216  1     !                  force Console Terminal input interrupt by typing "Q"
;  4217  1     !                  print error message if not inhibited
;  4218  1     !               ENDIF
;  4219  1     !               disable Sanity Timer
;  4220  1     !               restore Console Terminal and Power_fail interrupt vectors
;  4221  1     !                 ( location 24 and 60 octal )
;  4222  1     !            END
;  4223  1     !--
```

```
;   4224  3     BGNTST;
;   4225  3
;   4226  3     IF .SWP_TIMER
;   4227  3       THEN
;   4228  4         BEGIN
;   4229  4
;   4230  4           !++
;   4231  4           !  RESET DEQNA AND INITIALIZE ETHERNET STATION ADDRESS RAM
;   4232  4           !--
;   4233  4
;   4234  4           RESET_DEQNA ( );
;   4235  4           !++
;   4236  4           !  SETUP FOR POWER FAIL AND CONSOLE TERMINAL INTERRUPTS
;   4237  4           !--
;   4238  4
;   4239  4           SETVEC ( PF_VEC_LOC, PWR_INT, PRIO7 );            ! POWER FAIL
;   4240  4           SETVEC ( KB_VEC_LOC, KBD_INT, PRIO5 );            ! CONSOLE TERMINAL
;   4241  4           SETPRI ( PRIOO );                                 ! SET PROCESSOR PRI LEVEL
;   4242  4           PREP_FOR_SETUP ( );
;   4243  4           INCR INDEX1 FROM 1 TO 14 DO
;   4244  4             WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;   4245  4
;   4246  6           BGNSUB;
;   4247  6             PUT_BIT [ CSR, SE, EENABLE ];
;   4248  6             XMIT_SETUP_PACKET ( %O'200' + ( .SWP_TOUT_VAL + 4 ) );
;   4249  6
;   4250  6             SELECTONE .SWP_TOUT_VAL OF
;   4251  6               SET
;   4252  6               [ 0,1 ]:
;   4253  7                       BEGIN
;   4254  7                         TEMP1 = 1;
;   4255  7                         PRINTB ( MSG32, .TEMP1 );
;   4256  6                       END;
;   4257  6               [ 2 ]:
;   4258  7                       BEGIN
;   4259  7                         TEMP1 = 4;
;   4260  7                         PRINTB ( MSG32, .TEMP1 );
;   4261  6                       END;
;   4262  6               [ 3 ]:
;   4263  7                       BEGIN
;   4264  7                         TEMP1 = 16;
;   4265  7                         PRINTB ( MSG32, .TEMP1 );
;   4266  6                       END;
;   4267  6               [ 4 ]:
;   4268  7                       BEGIN
;   4269  7                         TEMP1 = 1;
;   4270  7                         PRINTB ( MSG55, .TEMP1 );
;   4271  6                       END;
;   4272  6               [ 5 ]:
;   4273  7                       BEGIN
;   4274  7                         TEMP1 = 4;
;   4275  7                         PRINTB ( MSG55, .TEMP1 );
;   4276  6                       END;
```

```
;  6              [ 6 ]:
;  7                     BEGIN
;  7                       TEMP1   16;
;  7                       PRINTB ( MSG55, .TEMP1 );
;  6                     END;
;  6              [ 7 ]:
;  7                     BEGIN
;  7                       TEMP1 = 1;
;  7                       PRINTB ( MSG56, .TEMP1 );
;  6                     END;
;  6              TES;
;  6
;  6            PRINTB ( MSG57 );
;  6            INTERRUPT_FLG = -1;
;  6            WAIT_FOR_TIMEOUT ( );
;  6
;  6            !++
;  6            !  PUT DEQNA IN NORMAL MODE AND CHECK STATUS
;  6            !--
;  6
;  6            PUT_BIT [ CSR, SE, DISABLE ];
;  6            PREP_FOR_SETUP ( );
;  6            INCR INDEX1 FROM 1 TO 14 DO
;  6              WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;  6
;  8            BGNSEG;
;  8              XMIT_SETUP_PACKET ( N_MODE );
;  6            ENDSEG;
;  6
;  6            CLRVEC ( PF_VEC_LOC );
;  6            CLRVEC ( KB_VEC_LOC );
;  6
;  6            IF .INTERRUPT_FLG
;  6              THEN
;  7                BEGIN
;  8                  PRINTB ( MSG33 )
;  7                END
;  6              ELSE
;  7                BEGIN
;  7                  CSR_WORD = GET_BIT ( CSR_ALL );
;  7                  PRINTB ( MSG59 );
;  7                  PRINTB ( MSG34 );
;  7                  ERRDF  ( 2101, MSG00, ERROR$REPORT );
;  6                END;
;  4          ENDSUB;
;  3        END;
;  3
;  1    ENDTST;

                                        .SBTTL  $T21 TEST 21 - SANITY TIMER TEST
000000  010146                  $T21:   MOV     R1,-(SP)                ;       4183
000002  005746                          TST     -(SP)
```

```
000004 032737 000001 000000G         BIT   #1,SWP.TIMER          ;              4226
000012 001002                        BNE   1$
000014 000137 021522'                JMP   17$
000020 004737 000000G          1$:   JSR   PC,RESET.DEQNA        ;              4234
000024 012746 000000G                MOV   #PRI07,-(SP)          ;              4239
000030 012746 000000G                MOV   #PWR.INT,-(SP)
000034 012746 000024                 MOV   #24,-(SP)
000040 012746 000003                 MOV   #3,-(SP)
000044 104437                        TRAP  37
000046 012716 000000G                MOV   #PRI05,(SP)           ;              4240
000052 012746 000000G                MOV   #KBD.INT,-(SP)
000056 012746 000060                 MOV   #60,-(SP)
000062 012746 000003                 MOV   #3,-(SP)
000066 104437                        TRAP  37
000070 012700 000000G                MOV   #PRI00,R0             ;              4241
000074 104441                        TRAP  41
000076 004737 000000G                JSR   PC,PREP.FOR.SETUP     ;              4242
000102 012701 000001                 MOV   #1,R1                 ; *,INDEX1     4243
000106 010116                  2$:   MOV   R1,(SP)               ; INDEX1,*     4244
000110 012746 000023                 MOV   #23,-(SP)
000114 004737 000000G                JSR   PC,WRT.STATION.ADR
000120 005726                        TST   (SP)+
000122 005201                        INC   R1                    ; INDEX1       4243
000124 020127 000016                 CMP   R1,#16                ; INDEX1,*
000130 003766                        BLE   2$
000132 104402                  3$:   TRAP  2                     ;              4244
000134 013700 000000G                MOV   REG.ADR,R0            ;              4247
000140 052760 002000 000016          BIS   #2000,16(R0)
000146 013700 000000G                MOV   SWP.TOUT.VAL,R0       ;              4248
000152 072027 000004                 ASH   #4,R0
000156 010016                        MOV   R0,(SP)
000160 062716 000200                 ADD   #200,(SP)
000164 004737 000000G                JSR   PC,XMIT.SETUP.PACKET
000170 013701 000000G                MOV   SWP.TOUT.VAL,R1       ;              4250
000174 002417                        BLT   4$                    ;              4252
000176 020127 000001                 CMP   R1,#1
000202 003014                        BGT   4$
000204 012737 000001 000000G         MOV   #1,TEMP1              ;              4254
000212 012716 000001                 MOV   #1,(SP)               ;              4255
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$                   ;              4253
000234 020127 000002           4$:   CMP   R1,#2                 ;              4257
000240 001014                        BNE   5$
000242 012737 000004 000000G         MOV   #4,TEMP1              ;              4259
000250 012716 000004                 MOV   #4,(SP)               ;              4260
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$                   ;              4258
```

K1


```
000272  020127  000003                  5$:     CMP     R1,#3                           ;                       4262
000276  001014                                  BNE     6$
000300  012737  000020  000000G                MOV     #20,TEMP1                       ;                       4264
000306  012716  000020                          MOV     #20,(SP)                        ;                       4265
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$                             ;                       4263
000330  020127  000004                  6$:     CMP     R1,#4                           ;                       4267
000334  001014                                  BNE     7$
000336  012737  000001  000000G                MOV     #1,TEMP1                        ;                       4269
000344  012716  000001                          MOV     #1,(SP)                         ;                       4270
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$                             ;                       4268
000366  020127  000005                  7$:     CMP     R1,#5                           ;                       4272
000372  001014                                  BNE     8$
000374  012737  000004  000000G                MOV     #4,TEMP1                        ;                       4274
000402  012716  000004                          MOV     #4,(SP)                         ;                       4275
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$                             ;                       4273
000424  020127  000006                  8$:     CMP     R1,#6                           ;                       4277
000430  001014                                  BNE     9$
000432  012737  000020  000000G                MOV     #20,TEMP1                       ;                       4279
000440  012716  000020                          MOV     #20,(SP)                        ;                       4280
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$                             ;                       4278
000462  020127  000007                  9$:     CMP     R1,#7                           ;                       4282
000466  001014                                  BNE     11$
000470  012737  000001  000000G                MOV     #1,TEMP1                        ;                       4284
000476  012716  000001                          MOV     #1,(SP)                         ;                       4285
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$:    CMP     (SP)+,(SP)+                     ;                       4283
000520  012716  000000G          11$:    MOV     #MSG57,(SP)                     ;                       4289
000524  012746  000001                  MOV     #1,-(SP)
000530  010600                          MOV     SP,R0                           ; SP,*
000532  104414                          TRAP    14
000534  012737  177777  000000G        MOV     #-1,INTERRUPT.FLG               ;                       4290
000542  004737  000000G                JSR     PC,WAIT.FOR.TIMEOUT             ;                       4291
000546  013700  000000G                MOV     REG.ADR,R0                      ;                       4297
000552  042760  002000  000016        BIC     #2000,16(R0)
```

L1


```
000560  004737  000000G                        JSR     PC,PREP.FOR.SETUP               ;                               4298
000564  012701  000001                         MOV     #1,R1                           ; *,INDEX1                      4299
000570  010116                         12$:    MOV     R1,(SP)                         ; INDEX1,*                      4300
000572  012746  000023                         MOV     #23,-(SP)
000576  004737  000000G                        JSR     PC,WRT.STATION.ADR
000602  005726                                 TST     (SP)+
000604  005201                                 INC     R1                              ; INDEX1                        4299
000606  020127  000016                         CMP     R1,#16                          ; INDEX1,*
000612  003766                                 BLE     12$
000614  104404                         13$:    TRAP    4                               ;                               4300
000616  012716  000200                         MOV     #200,(SP)                       ;                               4303
000622  004737  000000G                        JSR     PC,XMIT.SETUP.PACKET
000626  104470                                 TRAP    70
000630  006000                                 ROR     R0
000632  103770                                 BLO     13$
000634  012700  000024                         MOV     #24,R0                          ;                               4306
000640  104436                                 TRAP    36
000642  012700  000060                         MOV     #60,R0                          ;                               4307
000646  104436                                 TRAP    36
000650  032737  000001  000000G                BIT     #1,INTERRUPT.FLG                ;                               4309
000656  001407                                 BEQ     14$
000660  012716  000000G                        MOV     #MSG33,(SP)                     ;                               4312
000664  012746  000001                         MOV     #1,-(SP)
000670  010600                                 MOV     SP,R0                           ; SP,*
000672  104414                                 TRAP    14
000674  000431                                 BR      15$                             ;                               4309
000676  013700  000000G                14$:    MOV     REG.ADR,R0                      ;                               4316
000702  016066  000016  000020                 MOV     16(R0),20(SP)                   ; *,TMP.LOCATION
000710  016637  000020  000000G                MOV     20(SP),CSR.WORD                 ; TMP.LOCATION,*
000716  012716  000000G                        MOV     #MSG59,(SP)                     ;                               4317
000722  012746  000001                         MOV     #1,-(SP)
000726  010600                                 MOV     SP,R0                           ; SP,*
000730  104414                                 TRAP    14
000732  012716  000000G                        MOV     #MSG34,(SP)                     ;                               4318
000736  012746  000001                         MOV     #1,-(SP)
000742  010600                                 MOV     SP,R0                           ; SP,*
000744  104414                                 TRAP    14
000746  104455                                 TRAP    55                              ;                               4319
000750  004065                                 .WORD   4065
000752  000000G                                .WORD   MSG00
000754  000000G                                .WORD   ERROR$REPORT
000756  005726                                 TST     (SP)+                           ;                               4315
000760  022626                         15$:    CMP     (SP)+,(SP)+                     ;                               4244
000762  104467                                 TRAP    67                              ;                               4320
000764  006000                                 ROR     R0
000766  103002                                 BHIS    16$
000770  000137  020654'                        JMP     3$
000774  062706  000016                 16$:    ADD     #16,SP                          ;                               4228
001000  005726                         17$:    TST     (SP)+                           ;                               4183
001002  012601                                 MOV     (SP)+,R1
001004  000207                                 RTS     PC

; Routine Size:  259 words,    Routine Base:  AB$CODE$ + 20522
```

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



                                        .SBTTL  T21 TEST 21   SANITY TIMER TEST
000000  004737  020522'                T21::
000000                                  1$:     JSR     PC,$T21                         ;                               4322
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 21530
; Maximum stack depth per invocation:  2 words


;   4325  1
;   4326  1
;   4327  1     END
;   4328  0     ELUDOM


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

.                                       PSECT SUMMARY
.
;       Psect Name                      Words     Attributes
;        AB$CODE$                        4530       RO ,  I  ,  LCL,  REL,  CON



;                                  Library Statistics
;
;                                            -------- Symbols -------        Pages       Processing
;       File                                 Total    Loaded   Percent      Mapped      Time
;
;  DISK2:[SCODA.QNA.ZQNA]QNALIB.L16;2          224       142        63        14          00:00.1




;                                       COMMAND QUALIFIERS

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

```
; Size:          4530 code + 0 data words
; Run Time:        01:21.3
; Elapsed Time:    01:28.1
; Lines/CPU Min:    3194
; Lexemes/CPU-Min: 36699
; Memory Used:  408 pages
; Compilation Complete
```

B.)


```
;    0001  0     MODULE ZQNA4 (%TITLE  CZQNAEO 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 CZQNAE0 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
```

F2


```
                                        .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
```

G2


```
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  000370                         BIC     #370,(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
```

H2


```
;   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$REPORT );
;  1687  2         END;
;  1688  2
;  1689  2     !++
;  1690  2     !
;  1691  2     !  CLEAR SOFTWARE RESET BIT IN THE CSR AND CHECK FOR EXPECTED STATUS
;  1692  2     !
;  1693  2     !--
;  1694  2
;  1695  2     PUT_BIT ( CSR, SR, CLR_IT );
;  1696  2     DELAY ( TIME6_LIMIT );
;  1697  2     TEMP2 = GET_BIT [ CSR_ALL ] AND CSR2_MASK;
;  1698  2     IF .TEMP2 NEQU CSR2_STATUS
;  1699  2       THEN
;  1700  3         BEGIN
;  1701  3           ERR_FLAG = ONE;
;  1702  3           CSR_WORD = GET_BIT [ CSR_ALL ];
;  1703  3           PRINTB ( MSG59 );
;  1704  3           PRINTB ( MSG31 );
;  1705  3           PRINTB ( MSG30, .GET_ADR [ CSR_ALL ], .TEMP1, CSR2_STATUS );
;  1706  3           ERRDF ( 0002, MSG00, E1$REPORT );
;  1707  2         END;
;  1708  2
;  1709  1     END;


                                        .GLOBL  L$DLY

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

<12


```
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  00C)00G                  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  013746  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   MSG00
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
```

M2


```
;    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_NO   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)
```

N2


```
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
```

B3


```
;   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
```

D3


```
;  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     !++
```

E3


```
;   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)
```

F3


```
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
```

G3


```
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
```

H3


```
;  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
```

J3


```
;  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;          ! 0'140000'
;  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 NXWD1_MASK      ! 0'157400'
;  1979  2       ELSE
;  1980  2         TEMP3 = .XMIT_D_LIST [ STWD1 ] AND X1_MASK;        ! 0'100000'
;  1981  2
;  1982  2     IF .TEMP3 NEQU .P2
```

K3


```
;   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  020377  000000G         BIC     #20377,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  01060^                                  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
```

M3


```
;  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;            ! O'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           ! O'174017'
;  2043  2       ELSE
;  2044  2         TEMP2 = .RCV_D_LIST [ STWD1 ] AND .P2;
;  2045  2
;  2046  2
;  2047  2     ! added for error bits qualified by bit 12 (discard)
```

N3


```
;   2048  2     TEMP3 = .TEMP2 AND %O'10000';                   !IS BIT 12 (DISCARD) SET
;   2049  2     IF .TEMP3 EQLU %O'0'
;   2050  2         THEN
;   2051  3           BEGIN
;   2052  3           TEMP3 = .TEMP2 AND %O'177400';            !MASK BITS QUAL BY BIT 12
;   2053  3           TEMP2 = .TEMP3;                           !PUT FINAL DATA IN TEMP2
;   2054  2           END;
;   2055  2     ! end additions
;   2056  2
;   2057  2
;   2058  2     IF .TEMP2 NEQU .P2
;   2059  2       THEN
;   2060  3         BEGIN
;   2061  3           ERR_FLAG = ONE;
;   2062  3           CSR_WORD = GET_BIT [ CSR ALL ];
;   2063  3           PRINTB ( MSG59 );
;   2064  3           PRINTB ( MSG16, .TEMP2, .P2 );
;   2065  3           ERRDF ( 0010, MSG00, ERROR$REPORT );
;   2066  2         END;
;   2067  2
;   2068  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  003774  000000G         BIC     #3774,TEMP2
000142  013737  000000G 000000G         MOV     TEMP2,TEMP3                     ;                               2048
```

B4


```
000150  042737  167777  000000G                 BIC     #167777,TEMP3
000156  001011                                  BNE     2$                      ;                               2049
000160  013737  000000G 000000G                 MOV     TEMP2,TEMP3             ;                               2052
000166  042737  000377  000000G                 BIC     #377,TEMP3
000174  013737  000000G 000000G                 MOV     TEMP3,TEMP2             ;                               2053
000202  023766  000000G 000006          2$:     CMP     TEMP2,6(SP)             ; *,P2                          2058
000210  001441                                  BEQ     3$
000212  012737  000001  000000G                 MOV     #1,ERR.FLAG             ;                               2061
000220  013700  000000G                         MOV     REG.ADR,R0              ;                               2062
000224  016066  000016  000002                  MOV     16(R0),2(SP)            ; *,TMP.LOCATION
000232  016637  000002  000000G                 MOV     2(SP),CSR.WORD          ; TMP.LOCATION,*
000240  012746  000000G                         MOV     #MSG59,-(SP)            ;                               2063
000244  012746  000001                          MOV     #1,-(SP)
000250  010600                                  MOV     SP,R0                   ; SP,*
000252  104414                                  TRAP    14
000254  016616  000012                          MOV     12(SP),(SP)             ; P2,*                          2064
000260  013746  000000G                         MOV     TEMP2,-(SP)
000264  012746  000000G                         MOV     #MSG16,-(SP)
000270  012746  000003                          MOV     #3,-(SP)
000274  010600                                  MOV     SP,R0                   ; SP,*
000276  104414                                  TRAP    14
000300  104455                                  TRAP    55                      ;                               2065
000302  000012                                  .WORD   12
000304  000000G                                 .WORD   MSG00
000306  000000'                                 .WORD   ERROR$REPORT
000310  062706  000012                          ADD     #12,SP                  ;                               2060
000314  022626                          3$:     CMP     (SP)+,(SP)+             ;                               1997
000316  000207                                  RTS     PC

; Routine Size:  104 words,    Routine Base:  AC$CODE$ + 2336
; Maximum stack depth per invocation:  9 words


;   2069  1
```

```
;  2070  1     %SBTTL 'GLOBAL ROUTINE   CHK_RX LPSTATUS ( P1, P2 )'
;  2071  1
;  2072  1     GLOBAL ROUTINE CHK_RX LPSTATUS ( P1, P2 ) : NOVALUE  =
;  2073  1
;  2074  1     !++
;  2075  1     !
;  2076  1     !  GLOBAL ROUTINE :     CHK_RX_LPSTATUS
;  2077  1     !
;  2078  1     !  DESCRIPTION:
;  2079  1     !
;  2080  1     !       This routine checks receive status words for expected status.
;  2081  1     !
;  2082  1     !  INPUT PARAMETERS:
;  2083  1     !
;  2084  1     !       P1      - expected RCV flag word
;  2085  1     !       P2        expected RCV status word 1
;  2086  1     !       TEST NO - test number in which error occurred.
;  2087  1     !
;  2088  1     !--
;  2089  1
;  2090  1     !       added because discard does not set in loop mode
;  2091  1     !
;  2092  1
;  2093  2     BEGIN
;  2094  2
;  2095  2     !++
;  2096  2     !  MASK OUT DON'T CARE BITS IN THE RCV FLAG WORD AND COMPARE TO EXPECTED
;  2097  2     !  RCV FLAG STATUS. IF STATUS NOT EQUAL THEN PRINT 'BAD RCV FLAG WORD
;  2098  2     !  STATUS'
;  2099  2     !--
;  2100  2
;  2101  2     TEMP1 = .RCV_D_LIST [ FLGWD ] AND RFLG_MASK;            ! O'140000'
;  2102  2
;  2103  2     IF .TEMP1 NEQU .P1
;  2104  2       THEN
;  2105  3         BEGIN
;  2106  3           ERR_FLAG = ONE;
;  2107  3           CSR_WORD = GET_BIT [ CSR_ALL ];
;  2108  3           PRINTB ( MSG59 );
;  2109  3           PRINTB ( MSG15, .TEMP1, RFLG_MASK );
;  2110  3           ERRDF ( 0009, MSG00, ERROR$REPORT );
;  2111  2         END;
;  2112  2
;  2113  2     !++
;  2114  2     !  MASK OUT DON'T CARE BITS IN THE RCV STATUS WD1 AND COMPARE TO EXPECTED
;  2115  2     !  RCV STATUS WD1. IF STATUS NOT EQUAL THEN PRINT 'BAD RCV STATUS WORD 1'
;  2116  2     !--
;  2117  2
;  2118  2     IF .RCV_D_LIST [ STWD1 ] GEQU ZERO
;  2119  2       THEN
;  2120  2         TEMP2 = .RCV_D_LIST [ STWD1 ] AND R2_MASK           ! O'174017'
;  2121  2       ELSE
;  2122  2         TEMP2 = .RCV_D_LIST [ STWD1 ] AND .P2;
```

```
;   2123  2
;   2124  2
;   2125  2     IF .TEMP2 NEQU .P2
;   2126  2       THEN
;   2127  3         BEGIN
;   2128  3           ERR_FLAG = ONE;
;   2129  3           CSR_WORD = GET_BIT [ CSR_ALL ];
;   2130  3           PRINTB ( MSG59 );
;   2131  3           PRINTB ( MSG16, .TEMP2, .P2 );
;   2132  3           ERRDF ( 0010, MSG00, ERROR$REPORT );
;   2133  2         END;
;   2134  2
;   2135  1     END;


                                        .SBTTL  CHK.RX.LPSTATUS GLOBAL ROUTINE - CHK_RX_LPSTATUS ( P1, P2 )
000000  024646                  CHK.RX.LPSTATUS::
                                        CMP     -(SP),-(SP)                     ;                               2072
000002  013737  000000G 000000G         MOV     RCV.D.LIST,TEMP1                ;                               2101
000010  042737  037777  000000G         BIC     #37777,TEMP1
000016  023766  000000G 000010          CMP     TEMP1,10(SP)                    ; *,P1                          2103
000024  001437                          BEQ     1$
000026  012737  000001  000000G         MOV     #1,ERR.FLAG                     ;                               2106
000034  013700  000000G                 MOV     REG.ADR,R0                      ;                               2107
000040  016016  000016                  MOV     16(R0),(SP)                     ; *,TMP.LOCATION
000044  011637  000000G                 MOV     (SP),CSR.WORD                   ; TMP.LOCATION,*
000050  012746  000000G                 MOV     #MSG59,-(SP)                    ;                               2108
000054  012746  000001                  MOV     #1,-(SP)
000060  010600                          MOV     SP,R0                           ; SP,*
000062  104414                          TRAP    14
000064  012716  140000                  MOV     #-40000,(SP)                    ;                               2109
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                              ;                               2110
000112  000011                          .WORD   11
000114  000000G                         .WORD   MSG00
000116  000000'                         .WORD   ERROR$REPORT
000120  062706  000012                  ADD     #12,SP                          ;                               2105
000124  013700  000010G         1$:     MOV     RCV.D.LIST+10,R0                ;                               2118
000130  010037  000000G                 MOV     R0,TEMP2                        ;                               2120
000134  042737  003774  000000G         BIC     #3774,TEMP2
000142  023766  000000G 000006          CMP     TEMP2,6(SP)                     ; *,P2                          2125
000150  001441                          BEQ     2$
000152  012737  000001  000000G         MOV     #1,ERR.FLAG                     ;                               2128
000160  013700  000000G                 MOV     REG.ADR,R0                      ;                               2129
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)                    ;                               2130
000204  012746  000001                  MOV     #1,-(SP)
000210  010600                          MOV     SP,R0                           ; SP,*
```

E4


```
000212  104414                          TRAP    14
000214  016616  000012                  MOV     12(SP),(SP)             ; P2,*                  2131
000220  013746  000000G                 MOV     TEMP2, (.P)
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                      ;                       2132
000242  000012                          .WORD   12
000244  000000G                         .WORD   MSG00
000246  000000'                         .WORD   ERROR$REPORT
000250  062706  000012                  ADD     #12,SP                  ;                       2127
000254  022626                  2$:     CMP     (SP)+,(SP)+             ;                       2072
000256  000207                          RTS     PC

; Routine Size:  88 words,      Routine Base:  AC$CODE$ + 2656
; Maximum stack depth per invocation:  9 words


;   2136  1
```

```
;  2137  1     %SBTTL 'GLOBAL ROUTINE   COMPARE_PACKETS ( )'
;  2138  1
;  2139  1     GLOBAL ROUTINE COMPARE_PACKETS : NOVALUE =
;  2140  1
;  2141  1     !++
;  2142  1     !
;  2143  1     !  GLOBAL ROUTINE :     COMPARE_PACKETS
;  2144  1     !
;  2145  1     !  DESCRIPTION:
;  2146  1     !
;  2147  1     !       This routine compares contents of transmit packet to the contents
;  2148  1     !       of receive packet and prints an error message if the don't compare.
;  2149  1     !--
;  2150  1
;  2151  2     BEGIN
;  2152  2
;  2153  2     !++
;  2154  2     !  GET RECEIVE BYTE LENGTH ( RBL ) FROM RCV DISCRIPTOR AND COMPUTE WORD
;  2155  2     !  LENGTH. THEN COMPARE ACTUAL TO EXPECTED RCV WORD LENGTH.
;  2156  2     !--
;  2157  2
;  2158  2     TEMP3 = 0;
;  2159  2
;  2160  2     IF GET_BIT [ CSR, LB ] GTRU ZERO
;  2161  2       THEN
;  2162  2         TEMP3 = .RCV_D_LIST [ STWD1 ] AND RHL_MASK;                  ! O'003400'
;  2163  2
;  2164  2     IF ( .CSR_WORD AND %O'01' ) EQLU ZERO
;  2165  2       THEN
;  2166  3         TEMP3 = .TEMP3 + ( .RCV D_LIST [ STWD2 ] AND RLL_MASK )     ! O'000377'
;  2167  2       ELSE
;  2168  2         TEMP3 = 6;
;  2169  2
;  2170  2     IF .TEMP3 NEQU .RBUF_LENGTH
;  2171  2       THEN
;  2172  3         BEGIN
;  2173  3           ERR_FLAG = ONE;
;  2174  3           CSR_WORD = GET_BIT [ CSR_ALL ];
;  2175  3           PRINTB ( MSG59 );
;  2176  3           PRINTB ( MSG17, .TEMP3, .RBUF_LENGTH );
;  2177  3           ERRDF ( 0011, MSG00, ERROR$REPORT );
;  2178  2         END;
;  2179  2
;  2180  2     INCR INDEX FROM 0 TO .TEMP3 - 1 DO
;  2181  3       BEGIN
;  2182  3         IF .RCV_D_LIST [ STWD1 ] EQLU NEWB
;  2183  3           THEN
;  2184  3             RCV_BUFFER [ .INDEX ] = ZERO;
;  2185  3
;  2186  3         IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;  2187  3           THEN
;  2188  3             IF .RCV_D LIST [ LONGP ] EQLU ONE
;  2189  3               THEN
```

```
;   2190  4                 BEGIN
;   2191  4                   TEMP5   .INDEX;
;   2192  4                   EXITLOOP;
;   2193  4                 END
;   2194  3               ELSE
;   2195  4                 BEGIN
;   2196  4                   ERR_FLAG = ONE;
;   2197  4                   CSR_WORD = GET_BIT [ CSR_ALL ];
;   2198  4                   PRINTB ( MSG59 );
;   2199  4                   PRINTB ( MSG51 );
;   2200  4                   PRINTB ( MSG50, .RCV_BUFFER [ .INDEX ], .XMIT_BUFFER [ .INDEX ],  .INDEX );
;   2201  4                   ERRDF ( 0012, MSG00, ERROR$REPORT );
;   2202  3                 END;
;   2203  2       END;
;   2204  1     END;


                                        .SBTTL  COMPARE.PACKETS GLOBAL ROUTINE - COMPARE_PACKETS ( )
000000  004137  000000G                COMPARE.PACKETS::
                                        JSR     R1,$SAVE2                       ;                                       2139
000004  024646                         CMP     -(SP),-(SP)
000006  005037  000000G                CLR     TEMP3                           ;                                       2158
000012  013700  000000G                MOV     REG.ADR,R0                      ;                                       2160
000016  016046  000016                 MOV     16(R0),-(SP)                    ; *,TMP.LOCATION
000022  032716  001400                 BIT     #1400,(SP)                      ; *,TMP.LOCATION
000026  001406                         BEQ     1$
000030  013737  000010G 000000G        MOV     RCV.D.LIST+10,TEMP3             ;                                       2162
000036  042737  174377  000000G        BIC     #174377,TEMP3
000044  032737  000001  000000G 1$:    BIT     #1,CSR.WORD                     ;                                       2164
000052  001006                         BNE     2$
000054  005001                         CLR     R1                              ;                                       2166
000056  153701  000012G                BISB    RCV.D.LIST+12,R1
000062  060137  000000G                ADD     R1,TEMP3
000066  000403                         BR      3$                              ;                                       2164
000070  012737  000006  000000G 2$:    MOV     #6,TEMP3                        ;                                       2168
000076  023737  000000G 000000G 3$:    CMP     TEMP3,RBUF.LENGTH               ;                                       2170
000104  001437                         BEQ     4$
000106  012737  000001  000000G        MOV     #1,ERR.FLAG                     ;                                       2173
000114  016066  000016  000002         MOV     16(R0),2(SP)                    ; *,TMP.LOCATION                        2174
000122  016637  000002  000000G        MOV     2(SP),CSR.WORD                  ; TMP.LOCATION,*
000130  012746  000000G                MOV     #MSG59,-(SP)                    ;                                       2175
000134  012746  000001                 MOV     #1,-(SP)
000140  010600                         MOV     SP,R0                           ; SP,*
000142  104414                         TRAP    14
000144  013716  000000G                MOV     RBUF.LENGTH,(SP)                ;                                       2176
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                              ;                                       2177
000172  000013                         .WORD   13
000174  000000G                        .WORD   MSG00
```

```
000176  000000'                                .WORD   ERROR$REPORT
000200  062706  000012                         ADD     #12,SP                          ;                                       2172
000204  013702  000000G                 4$:    MOV     TEMP3,R2                        ;                                       2180
000210  005001                                 CLR     R1                              ; INDEX
000212  000474                                 BR      9$
000214  023727  000010G 100000          5$:    CMP     RCV.D.LIST+10,#-100000          ;                                       2182
000222  001002                                 BNE     6$
000224  105061  000000G                        CLRB    RCV.BUFFER(R1)                  ; *(INDEX)                              2184
000230  126161  000000G 000000G         6$:    CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)                    2186
000236  001461                                 BEQ     8$
000240  032737  040000  000010G               BIT     #40000,RCV.D.LIST+10            ;                                       2188
000246  001403                                 BEQ     7$
000250  010137  000000G                        MOV     R1,TEMP5                        ; INDEX,*                               2191
000254  000455                                 BR      10$                             ;                                       2190
000256  012737  000001  000000G         7$:    MOV     #1,ERR.FLAG                     ;                                       2196
000264  013700  000000G                        MOV     REG.ADR,R0                      ;                                       2197
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)                    ;                                       2198
000310  012746  000001                         MOV     #1,-(SP)
000314  010600                                 MOV     SP,R0                           ; SP,*
000316  104414                                 TRAP    14
000320  012716  000000G                        MOV     #MSG51,(SP)                     ;                                       2199
000324  012746  000001                         MOV     #1,-(SP)
000330  010600                                 MOV     SP,R0                           ; SP,*
000332  104414                                 TRAP    14
000334  010116                                 MOV     R1,(SP)                         ; INDEX,*                               2200
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                              ;                                       2201
000370  000014                                 .WORD   14
000372  000000G                                .WORD   MSG00
000374  000000'                                .WORD   ERROR$REPORT
000376  062706  000016                         ADD     #16,SP                          ;                                       2195
000402  005201                         8$:     INC     R1                              ; INDEX                                 2180
000404  020102                         9$:     CMP     R1,R2                           ; INDEX,*
000406  002702                                 BLT     5$
000410  062706  000006                 10$:    ADD     #6,SP                           ;                                       2139
000414  000207                                 RTS     PC

; Routine Size:  135 words,    Routine Base:  AC$CODE$ + 3136
; Maximum stack depth per invocation:  15 words


;   2205  1
;   2206  1
```

```
;   2207  1     %SBTTL  GLOBAL ROUTINE   SET_RDESCR_LIST ( P1, P2)'
;   2208  1
;   2209  1     GLOBAL ROUTINE SET RDESCR_LIST ( P1, P2 ) : NOVALUE  =
;   2210  1
;   2211  1     !++
;   2212  1     !
;   2213  1     !  GLOBAL ROUTINE :     SET_RDESCR_LIST
;   2214  1     !
;   2215  1     !  DESCRIPTION:
;   2216  1     !
;   2217  1     !       This routine initializes receive descriptor list.
;   2218  1     !
;   2219  1     !  INPUT PARAMETERS:
;   2220  1     !
;   2221  1     !       P1 - expected Ethernet packet length in words
;   2222  1     !       P2   expected RCV Descriptor List settings
;   2223  1     !
;   2224  1     !--
;   2225  1
;   2226  2     BEGIN
;   2227  2
;   2228  2       RCV_D_LIST [ FLGWD ]  = NEWB;
;   2229  2       RCV_D_LIST [ DBITS ]  = .P2;
;   2230  2       RCV_D_LIST [ LOADR ]  = RCV_BUFFER;
;   2231  2       RCV_D_LIST [ TWDL  ]  = .P1;
;   2232  2       RCV_D_LIST [ STWD1 ]  = 0;
;   2233  2       RCV_D_LIST [ STWD2 ]  = 0;
;   2234  2       RCV_D_LIST [ DLINK ]  = V;
;   2235  2       RCV_D_LIST [ BSTAT ]  = E;
;   2236  2
;   2237  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             ;                               2228
000006  016637  000002  000002G         MOV     2(SP),RCV.D.LIST+2              ; P2,*                          2229
000014  012737  000000G 000004G         MOV     #RCV.BUFFER,RCV.D.LIST+4        ;                               2230
000022  016637  000004  000006G         MOV     4(SP),RCV.D.LIST+6              ; P1,*                          2231
000030  005037  000010G                 CLR     RCV.D.LIST+10                   ;                               2232
000034  005037  000012G                 CLR     RCV.D.LIST+12                   ;                               2233
000040  012737  100000  000014G         MOV     #-100000,RCV.D.LIST+14          ;                               2234
000046  012737  020000  000016G         MOV     #20000,RCV.D.LIST+16            ;                               2235
000054  000207                         RTS     PC                              ;                               2209

; Routine Size:  23 words,      Routine Base:  AC$CODE$ + 3554
; Maximum stack depth per invocation:  0 words


;   2238  1
```

J4


```
;   2239  1     %SBTTL 'GLOBAL ROUTINE   SET XDESCR_LIST ( P1, P2 )'
;   2240  1
;   2241  1     GLOBAL ROUTINE SET_XDESCR_LIST ( P1, P2 ) : NOVALUE  =
;   2242  1
;   2243  1     !++
;   2244  1     !
;   2245  1     !  GLOBAL ROUTINE :     SET_XDESCR_LIST
;   2246  1     !
;   2247  1     !  DESCRIPTION:
;   2248  1     !
;   2249  1     !       This routine initializes transmit descriptor list.
;   2250  1     !
;   2251  1     !  INPUT PARAMETERS:
;   2252  1     !
;   2253  1     !       P1 - expected Ethernet packet length in words
;   2254  1     !       P2   expected XMIT Descriptor List settings
;   2255  1     !
;   2256  1     !--
;   2257  1
;   2258  2     BEGIN
;   2259  2
;   2260  2       XMIT_D_LIST [ FLGWD ] = NEWB;
;   2261  2       XMIT_D_LIST [ DBITS ] = .P2;
;   2262  2       XMIT_D_LIST [ LOADR ] = XMIT_BUFFER;
;   2263  2       XMIT_D_LIST [ TWDL  ] = .P1;
;   2264  2       XMIT_D_LIST [ STWD1 ] = 0;
;   2265  2       XMIT_D_LIST [ STWD2 ] = 0;
;   2266  2       XMIT_D_LIST [ DLINK ] = V;
;   2267  2       XMIT_D_LIST [ BSTAT ] = E;
;   2268  2
;   2269  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            ;                               2260
000006  016637  000002  000002G         MOV     2(SP),XMIT.D.LIST+2             ; P2,*                          2261
000014  012737  000000G 000004G         MOV     #XMIT.BUFFER,XMIT.D.LIST+4      ;                               2262
000022  016637  000004  000006G         MOV     4(SP),XMIT.D.LIST+6             ; P1,*                          2263
000030  005037  000010G                 CLR     XMIT.D.LIST+10                  ;                               2264
000034  005037  000012G                 CLR     XMIT.D.LIST+12                  ;                               2265
000040  012737  100000  000014G         MOV     #-100000,XMIT.D.LIST+14         ;                               2266
000046  012737  020000  000016G         MOV     #20000,XMIT.D.LIST+16           ;                               2267
000054  000207                         RTS     PC                              ;                               2241

; Routine Size:  23 words,      Routine Base:  AC$CODE$ + 3632
; Maximum stack depth per invocation:  0 words


;   2270  1
```

```
;    2271  1     %SBTTL 'GLOBAL ROUTINE - WALKING_BIT ( P1, P2, P3 )'
;    2272  1
;    2273  1     GLOBAL ROUTINE WALKING_BIT ( P1, P2, P3 ) : NOVALUE  =
;    2274  1
;    2275  1     !++
;    2276  1     !
;    2277  1     !  GLOBAL ROUTINE :     WALKING_BIT
;    2278  1     !
;    2279  1     !  DESCRIPTION:
;    2280  1     !
;    2281  1     !       This routine sets bit to 0 or 1 in a specified bit position of the
;    2282  1     !       Ethernet Station Address. For example,
;    2283  1     !
;    2284  1     !       if
;    2285  1     !               .P1 = 0  and  .P2 = 15    .P3 = 5
;    2286  1     !       then
;    2287  1     !               Ethernet Station Address  =  FF-FF-FF-FF-7F-FF
;    2288  1     !
;    2289  1     !  INPUT PARAMETERS:
;    2290  1     !
;    2291  1     !       P1   bit ( 0 or 1 )
;    2292  1     !       P2   bit position from base address
;    2293  1     !       P3   # of bytes to be tested using this pattern
;    2294  1     !
;    2295  1     !--
;    2296  1
;    2297  2     BEGIN
;    2298  2
;    2299  2     SELECTONE .P2 OF
;    2300  2       SET
;    2301  2       [  0 TO  7 ]:
;    2302  2                     TEMP1 = 0;
;    2303  2       [  8 TO ( .P3 + 1 ) * 8 ]:
;    2304  2                     TEMP1 = .P2 / 8;
;    2305  2       TES;
;    2306  2
;    2307  2     TEMP2 = .P2 MOD 8;
;    2308  2
;    2309  2     IF .P1 EQLU ZERO
;    2310  2       THEN
;    2311  3         BEGIN
;    2312  3           TBYTE1 = %B'00000000';
;    2313  3           SELECTONE .TEMP2 OF
;    2314  3             SET
;    2315  3             [ 0 ]:  TBYTE3 = %O'001';
;    2316  3             [ 1 ]:  TBYTE3 = %O'002';
;    2317  3             [ 2 ]:  TBYTE3 = %O'004';
;    2318  3             [ 3 ]:  TBYTE3 = %O'010';
;    2319  3             [ 4 ]:  TBYTE3 = %O'020';
;    2320  3             [ 5 ]:  TBYTE3 = %O'040';
;    2321  3             [ 6 ]:  TBYTE3 = %O'100';
;    2322  3             [ 7 ]:  TBYTE3 = %O'200';
;    2323  3             TES;
```

L4


```
;   2324  3          END
;   2325  2        ELSE
;   2326  3          BEGIN
;   2327  3            TBYTE1 = %B'11111111';
;   2328  3            SELECTONE .TEMP2 OF
;   2329  3              SET
;   2330  3              [ 0 ]:  TBYTE3 = %O'376';
;   2331  3              [ 1 ]:  TBYTE3 = %O'375';
;   2332  3              [ 2 ]:  TBYTE3 = %O'373';
;   2333  3              [ 3 ]:  TBYTE3 = %O'367';
;   2334  3              [ 4 ]:  TBYTE3 = %O'357';
;   2335  3              [ 5 ]:  TBYTE3 = %O'337';
;   2336  3              [ 6 ]:  TBYTE3 = %O'277';
;   2337  3              [ 7 ]:  TBYTE3 = %O'177';
;   2338  3              TES;
;   2339  2          END;
;   2340  2
;   2341  2      INCR INDEX FROM 0 TO .P3 DO
;   2342  2          TARGET_ADR  [ .INDEX ] = .TBYTE1;
;   2343  2
;   2344  2      TEMP3 = .P3    .TEMP1;
;   2345  2      TARGET_ADR  [ .TEMP3 ] = .TBYTE3;
;   2346  2
;   2347  1      END;


                                        .SBTTL  WALKING.BIT GLOBAL ROUTINE - WALKING_BIT ( P1, P2, P3 )
000000  004137  000000G          WALKING.BIT::
                                        JSR     R1,$SAVE2                       ;                               2273
000004  016602  000012                  MOV     12(SP),R2                       ; P2,*                          2299
000010  002406                          BLT     1$                              ;                               2301
000012  020227  000007                  CMP     R2,#7
000016  003003                          BGT     1$
000020  005037  000000G                 CLR     TEMP1                           ;                               2302
000024  000421                          BR      2$                              ;                               2299
000026  020227  000010          1$:     CMP     R2,#10                          ;                               2303
000032  002416                          BLT     2$
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                           ;                               2304
000056  006700                          SXT     R0
000060  071027  000010                  DIV     #10,R0
000064  010037  000000G                 MOV     R0,TEMP1
000070  010201                  2$:     MOV     R2,R1                           ;                               2307
000072  006700                          SXT     R0
000074  071027  000010                  DIV     #10,R0
000100  010137  000000G                 MOV     R1,TEMP2
000104  010100                          MOV     R1,R0                           ; TEMP2,*                       2313
000106  005766  000014                  TST     14(SP)                          ; P1                            2309
000112  001071                          BNE     10$
```

M4


```
000114  005037  000000G                        CLR     TBYTE1                  ;                       2312
000120  005700                                  TST     R0                      ;                       2315
000122  001004                                  BNE     3$
000124  012737  000001  000000G                MOV     #1,TBYTE3
000132  000552                                  BR      18$                     ;                       2313
000134  020027  000001                3$:       CMP     R0,#1                   ;                       2316
000140  001004                                  BNE     4$
000142  012737  000002  000000G                MOV     #2,TBYTE3
000150  000543                                  BR      18$                     ;                       2313
000152  020027  000002                4$:       CMP     R0,#2                   ;                       2317
000156  001004                                  BNE     5$
000160  012737  000004  000000G                MOV     #4,TBYTE3
000166  000534                                  BR      18$                     ;                       2313
000170  020027  000003                5$:       CMP     R0,#3                   ;                       2318
000174  001004                                  BNE     6$
000176  012737  000010  000000G                MOV     #10,TBYTE3
000204  000525                                  BR      18$                     ;                       2313
000206  020027  000004                6$:       CMP     R0,#4                   ;                       2319
000212  001004                                  BNE     7$
000214  012737  000020  000000G                MOV     #20,TBYTE3
000222  000516                                  BR      18$                     ;                       2313
000224  020027  000005                7$:       CMP     R0,#5                   ;                       2320
000230  001004                                  BNE     8$
000232  012737  000040  000000G                MOV     #40,TBYTE3
000240  000507                                  BR      18$                     ;                       2313
000242  020027  000006                8$:       CMP     R0,#6                   ;                       2321
000246  001004                                  BNE     9$
000250  012737  000100  000000G                MOV     #100,TBYTE3
000256  000500                                  BR      18$                     ;                       2313
000260  020027  000007                9$:       CMP     R0,#7                   ;                       2322
000264  001075                                  BNE     18$
000266  012737  000200  000000G                MOV     #200,TBYTE3
000274  000471                                  BR      18$                     ;                       2309
000276  012737  000377  000000G       10$:      MOV     #377,TBYTE1             ;                       2327
000304  005700                                  TST     R0                      ;                       2330
000306  001004                                  BNE     11$
000310  012737  000376  000000G                MOV     #376,TBYTE3
000316  000460                                  BR      18$                     ;                       2328
000320  020027  000001                11$:      CMP     R0,#1                   ;                       2331
000324  001004                                  BNE     12$
000326  012737  000375  000000G                MOV     #375,TBYTE3
000334  000451                                  BR      18$                     ;                       2328
000336  020027  000002                12$:      CMP     R0,#2                   ;                       2332
000342  001004                                  BNE     13$
000344  012737  000373  000000G                MOV     #373,TBYTE3
000352  000442                                  BR      18$                     ;                       2328
000354  020027  000003                13$:      CMP     R0,#3                   ;                       2333
000360  001004                                  BNE     14$
000362  012737  000367  000000G                MOV     #367,TBYTE3
000370  000433                                  BR      18$                     ;                       2328
000372  020027  000004                14$:      CMP     R0,#4                   ;                       2334
000376  001004                                  BNE     15$
000400  012737  000357  000000G                MOV     #357,TBYTE3
```

```
000406  000424                                  BR      18$                             ;                               2328
000410  020027  000005                  15$:    CMP     R0,#5                           ;                               2335
000414  001004                                  BNE     16$
000416  012737  000337  000000G                MOV     #337,TBYTE3
000424  000415                                  BR      18$                             ;                               2328
000426  020027  000006                  16$:    CMP     R0,#6                           ;                               2336
000432  001004                                  BNE     17$
000434  012737  000277  000000G                MOV     #277,TBYTE3
000442  000406                                  BR      18$                             ;                               2328
000444  020027  000007                  17$:    CMP     R0,#7                           ;                               2337
000450  001003                                  BNE     18$
000452  012737  000177  000000G                MOV     #177,TBYTE3
000460  005000                          18$:    CLR     R0                              ; INDEX                         2341
000462  000404                                  BR      20$
000464  113760  000000G 000000G          19$:    MOVB    TBYTE1,TARGET.ADR(R0)           ; *,*(INDEX)                    2342
000472  005200                                  INC     R0                              ; INDEX                         2341
000474  020066  000010                  20$:    CMP     R0,10(SP)                       ; INDEX,P3
000500  003771                                  BLE     19$
000502  016637  000010  000000G                MOV     10(SP),TEMP3                    ; P3,*                          2344
000510  163737  000000G 000000G                SUB     TEMP1,TEMP3
000516  013700  000000G                        MOV     TEMP3,R0                        ;                               2345
000522  113760  000000G 000000G                MOVB    TBYTE3,TARGET.ADR(R0)
000530  000207                                  RTS     PC                              ;                               2273

; Routine Size:  173 words,     Routine Base:  AC$CODE$ + 3710
; Maximum stack depth per invocation:  4 words


;   2348  1
```

```
;  2349  1     %SBTTL 'GLOBAL ROUTINE - WRT_STATION_ADR ( P1, P2 )'
;  2350  1
;  2351  1     GLOBAL ROUTINE WRT_STATION_ADR ( P1, P2 ): NOVALUE  =
;  2352  1
;  2353  1     !++
;  2354  1     !
;  2355  1     !  GLOBAL ROUTINE :     WRT_STATION_ADR
;  2356  1     !
;  2357  1     !  DESCRIPTION:
;  2358  1     !
;  2359  1     !       This routine writes Station Address to XMIT_BUFFER.
;  2360  1     !
;  2361  1     !  INPUT PARAMETERS:
;  2362  1     !
;  2363  1     !       P1 - Ethernet Station Address index (1:14) in Station Address RAM
;  2364  1     !       P2 - Ethernet Station Address index ( 0:19 ) in the TARGET_ADR table
;  2365  1     !
;  2366  1     !--
;  2367  1
;  2368  2     BEGIN
;  2369  2
;  2370  2     TEMP1 = .P2 * 6;
;  2371  2
;  2372  2     SELECTONE .P1 OF
;  2373  2       SET
;  2374  2       [  0 TO  7 ]:
;  2375  2                       TEMP2 = .P1;
;  2376  2       [  8 TO 14 ]:
;  2377  2                       TEMP2 = .P1 + 57;
;  2378  2       TES;
;  2379  2
;  2380  2     IF .TEMP2 EQLU ZERO
;  2381  2       THEN
;  2382  2         INCR INDEX FROM 0 TO 5 DO
;  2383  3           BEGIN
;  2384  3             XMIT_BUFFER [ .INDEX ] = .TARGET_ADR [ .INDEX + .TEMP1 ];
;  2385  3           END
;  2386  2       ELSE
;  2387  2         INCR INDEX FROM 0 TO 5 DO
;  2388  3           BEGIN
;  2389  3             TEMP3 = .INDEX * 8 + .TEMP2;
;  2390  3             XMIT_BUFFER [ .TEMP3 ] = .TARGET_ADR [ .INDEX + .TEMP1 ];
;  2391  2           END;
;  2392  1     END;


                                        .SBTTL  WRT.STATION.ADR GLOBAL ROUTINE - WRT_STATION_ADR ( P1, P2 )
000000  004137  000000G          WRT.STATION.ADR::
                                        JSR     R1,$SAVE3                       ;                                       2351
000004  016601  000012                  MOV     12(SP),R1                       ; P2,*                                  2370
000010  070127  000006                  MUL     #6,R1
000014  010137  000000G                 MOV     R1,TEMP1
000020  016600  000014                  MOV     14(SP),R0                       ; P1,*                                  2372
```

C5


```
000024  002406                                  BLT     1$                                      ;                                       2374
000026  020027  000007                          CMP     R0,#7
000032  003003                                  BGT     1$
000034  010037  000000G                         MOV     R0,TEMP2                                ;                                       2375
000040  000413                                  BR      2$                                      ;                                       2372
000042  020027  000010              1$:         CMP     R0,#10                                  ;                                       2376
000046  002410                                  BLT     2$
000050  020027  000016                          CMP     R0,#16
000054  003005                                  BGT     2$
000056  010037  000000G                         MOV     R0,TEMP2                                ;                                       2377
000062  062737  000071  000000G                 ADD     #71,TEMP2
000070  013703  000000G              2$:         MOV     TEMP2,R3                                ;                                       2380
000074  001014                                  BNE     4$
000076  005000                                  CLR     R0                                      ; INDEX                                 2382
000100  010001                      3$:         MOV     R0,R1                                   ; INDEX,*                               2384
000102  063701  000000G                         ADD     TEMP1,R1
000106  116160  000000G 000000G                 MOVB    TARGET.ADR(R1),XMIT.BUFFER(R0)          ; *,*(INDEX)
000114  005200                                  INC     R0                                      ; INDEX                                 2382
000116  020027  000005                          CMP     R0,#5                                   ; INDEX,*
000122  003766                                  BLE     3$
000124  000207                                  RTS     PC                                      ;                                       2380
000126  005002                      4$:         CLR     R2                                      ; INDEX                                 2387
000130  010200                      5$:         MOV     R2,R0                                   ; INDEX,*                               2389
000132  072027  000003                          ASH     #3,R0
000136  060300                                  ADD     R3,R0
000140  010037  000000G                         MOV     R0,TEMP3
000144  010201                                  MOV     R2,R1                                   ; INDEX,*                               2390
000146  063701  000000G                         ADD     TEMP1,R1
000152  116160  000000G 000000G                 MOVB    TARGET.ADR(R1),XMIT.BUFFER(R0)
000160  005202                                  INC     R2                                      ; INDEX                                 2387
000162  020227  000005                          CMP     R2,#5                                   ; INDEX,*
000166  003760                                  BLE     5$
000170  000207                                  RTS     PC                                      ;                                       2351

; Routine Size:  61 words,      Routine Base:  AC$CODE$ + 4442
; Maximum stack depth per invocation:  5 words

;   2393  1
```

D5


```
;    2394  1     %SBTTL 'GLOBAL ROUTINE - PREP_FOR_SETUP ( ) '
;    2395  1
;    2396  1     GLOBAL ROUTINE PREP_FOR_SETUP : NOVALUE  =
;    2397  1
;    2398  1     !++
;    2399  1     !
;    2400  1     !  GLOBAL ROUTINE :     PREP_FOR_SETUP
;    2401  1     !
;    2402  1     !  DESCRIPTION:
;    2403  1     !
;    2404  1     !       This routine retrives Ethernet Station Address from the Ethernet's
;    2405  1     !       Station Address PROM, saves copy of Ethernet Station Address PROM
;    2406  1     !       in the TARGET_ADR vector, initializes transmit and receive buffers
;    2407  1     !       to zero and finally sets buffer length to select promiscuous mode.
;    2408  1     !
;    2409  1     !  INPUT PARAMETERS:
;    2410  1     !
;    2411  1     !       none
;    2412  1     !--
;    2413  1
;    2414  2     BEGIN
;    2415  2
;    2416  2      !++
;    2417  2      !  RETRIEVE ETHERNET PHYSICAL STATION ADDRESS AND SAVE A COPY OF IT IN THE
;    2418  2      !  'TARGET_ADR' VECTOR.
;    2419  2      !--
;    2420  2
;    2421  2      INCR INDEX FROM 0 TO 5 DO
;    2422  3        BEGIN
;    2423  3          TBYTE1 = .REG_ADR [ .INDEX, ST_ADDR ];
;    2424  3          TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX ] = .TBYTE1;
;    2425  2        END;
;    2426  2
;    2427  2       CLR_BUFFERS ( 256 );
;    2428  2
;    2429  1     END;


                                        .SBTTL  PREP.FOR.SETUP GLOBAL ROUTINE   PREP FOR_SETUP ( )
000000  010146                  PREP.FOR.SETUP::
                                        MOV     R1,-(SP)                        ;                               2396
000002  005746                         TST     -(SP)
000004  005001                         CLR     R1                              ; INDEX                         2421
000006  010100                  1$:     MOV     R1,R0                           ; INDEX,*                       2423
000010  006300                         ASL     R0
000012  063700  000000G                ADD     REG.ADR,R0
000016  011016                         MOV     (R0),(SP)                       ; *,TMP.LOCATION
000020  005037  000000G                CLR     TBYTE1
000024  111637  000000G                MOVB    (SP),TBYTE1
000030  111661  000162G                MOVB    (SP),TARGET.ADR+162(R1)         ; *,*(INDEX)                    2424
000034  005201                         INC     R1                              ; INDEX                         2421
000036  020127  000005                CMP     R1,#5                           ; INDEX,*
000042  003761                         BLE     1$
```

E5


```
000044  012746  000400                          MOV     #400, (SP)                      ;                                       2427
000050  004737  001234'                        JSR     PC,CLR.BUFFERS
000054  022626                                  CMP     (SP)+,(SP)+                     ;                                       2396
000056  012601                                  MOV     (SP)+,R1
000060  000207                                  RTS     PC

; Routine Size:  25 words,      Routine Base:  AC$CODE$ + 4634
; Maximum stack depth per invocation:  4 words


;   2430  1
;   2431  1
;   2432  1
```

```
;  2433  1     %SBTTL 'GLOBAL ROUTINE - FORM HEX_ADR ( P3 ) '
;  2434  1
;  2435  1     GLOBAL ROUTINE FORM_HEX_ADR ( P3 ) : NOVALUE  =
;  2436  1
;  2437  1     !++
;  2438  1     !
;  2439  1     !  GLOBAL ROUTINE :     FORM_HEX_ADR
;  2440  1     !
;  2441  1     !  DESCRIPTION:
;  2442  1     !
;  2443  1     !       This routine retrives Ethernet Station Address from the Ethernet's
;  2444  1     !       Station Address PROM, saves its copy in the TARGET_ADR vector.
;  2445  1     !
;  2446  1     !  INPUT PARAMETERS:
;  2447  1     !
;  2448  1     !       P3 - Index to Station Address in the TARGET_ADR vector
;  2449  1     !-
;  2450  1
;  2451  2     BEGIN
;  2452  2
;  2453  2       !++
;  2454  2       !  RETRIEVE ETHERNET PHYSICAL STATION ADDRESS AND SAVE A COPY OF IT IN THE
;  2455  2       !  'TARGET_ADR' AND 'STATION_ADR' VECTORS.
;  2456  2       !--
;  2457  2
;  2458  2       IF .P3 EQLU ZERO
;  2459  2         THEN
;  2460  2           TEMP5 = 0
;  2461  2         ELSE
;  2462  2           TEMP5 = .P3 * 6;
;  2463  2
;  2464  2       INCR INDEX5 FROM 0 TO 5 DO
;  2465  3         BEGIN
;  2466  3           TBYTE1 = .REG_ADR [ .INDEX5, ST_ADDR ];
;  2467  3           TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX5 ] = .TBYTE1;
;  2468  2         END;
;  2469  2
;  2470  2       COUNTER = ZERO;
;  2471  2
;  2472  2       INCR INDEX5 FROM 0 TO 5 BY 2 DO
;  2473  3         BEGIN
;  2474  3           TEMP1  = .TARGET_ADR [ .TEMP5 + .INDEX5 ];
;  2475  3           TEMP1  = .TEMP1 ^ 8;
;  2476  3           TEMP2 =  .TARGET_ADR [ .TEMP5 + .INDEX5 + 1 ];
;  2477  3           STATION_ADR [ .COUNTER ] = .TEMP1 OR ( .TEMP2 AND %O'000377' );
;  2478  3           COUNTER = .COUNTER + 1;
;  2479  2         END;
;  2480  2
;  2481  2       !++
;  2482  2       !  PRINT ETHERNET STATION ADDRESS ON THE CONSOLE
;  2483  2       !--
;  2484  2
;  2485  2       COUNTER  = 18;
```

```
;   2486  2       PHYS_ADR [ 0  ]   %C'%';
;   2487  2       PHYS_ADR [ 1  ] = %C'A';
;   2488  2       PHYS_ADR [ 19 ] = %C' ';
;   2489  2       PHYS_ADR [ 20 ] = %C'%';
;   2490  2       PHYS_ADR [ 21 ] = %C'N';
;   2491  2
;   2492  2       DECR INDEX1 FROM 2 TO 0 DO
;   2493  3         BEGIN
;   2494  3           TEMP3 = .STATION_ADR [ .INDEX1 ];
;   2495  3           INCR INDEX2 FROM 0 TO 1 DO
;   2496  4             BEGIN
;   2497  4               INCR INDEX3 FROM 0 TO 1 DO
;   2498  5                 BEGIN
;   2499  5                   TEMP1 = .TEMP3 AND %X'F';
;   2500  5                   IF .TEMP1 LEQU %DECIMAL'9'
;   2501  5                     THEN
;   2502  5                       TBYTE1 = %C'0' + .TEMP1
;   2503  5                     ELSE
;   2504  5                       TBYTE1 = %C'A' + ( .TEMP1 - %DECIMAL'10' );
;   2505  5                   PHYS_ADR [ .COUNTER ] = .TBYTE1;
;   2506  5                   COUNTER = .COUNTER - 1;
;   2507  5                   TEMP3 = .TEMP3 ^ ( -4 );
;   2508  4                 END;
;   2509  4
;   2510  4               IF .COUNTER GTRU 2
;   2511  4                 THEN
;   2512  4                   PHYS_ADR [ .COUNTER ] = %C'-';
;   2513  4
;   2514  4               COUNTER = .COUNTER - 1;
;   2515  4
;   2516  3             END;
;   2517  2         END;
;   2518  2
;   2519  1       END;


                                        .SBTTL  FORM.HEX.ADR GLOBAL ROUTINE - FORM_HEX_ADR ( P3 )
000000  004137  000000G       FORM.HEX.ADR::
                                        JSR     R1,$SAVE3                       ;                               2435
000004  005746                         TST     -(SP)
000006  016600  000014                  MOV     14(SP),R0                       ; P3,*                          2458
000012  001003                         BNE     1$
000014  005037  000000G                 CLR     TEMP5                           ;                               2460
000020  000405                         BR      2$                              ;                               2458
000022  010001              1$:         MOV     R0,R1                           ;                               2462
000024  070127  000006                  MUL     #6,R1
000030  010137  000000G                 MOV     R1,TEMP5
000034  005000              2$:         CLR     R0                              ; INDEX5                        2464
000036  010001              3$:         MOV     R0,R1                           ; INDEX5,*                      2466
000040  006301                         ASL     R1
000042  063701  000000G                 ADD     REG.ADR,R1
000046  011116                         MOV     (R1),(SP)                       ; *,TMP.LOCATION
000050  005037  000000G                 CLR     TBYTE1
```

H5


```
000054  111637  000000G                        MOVB    (SP),TBYTE1
000060  111660  000162G                        MOVB    (SP),TARGET.ADR+162(R0)         ; *,*(INDEX5)                   2467
000064  005200                                 INC     R0                              ; INDEX5                        2464
000066  020027  000005                         CMP     R0,#5                           ; INDEX5,*
000072  003761                                 BLE     3$
000074  005037  000000G                        CLR     COUNTER                         ;                               2470
000100  005002                                 CLR     R2                              ; INDEX5                        2472
000102  010201                          4$:    MOV     R2,R1                           ; INDEX5,*                      2474
000104  063701  000000G                        ADD     TEMP5,R1
000110  116137  000000G 000000G                MOVB    TARGET.ADR(R1),TEMP1
000116  105037  000001G                        CLRB    TEMP1+1
000122  013700  000000G                        MOV     TEMP1,R0                        ;                               2475
000126  072027  000010                         ASH     #10,R0
000132  010037  000000G                        MOV     R0,TEMP1
000136  116137  000001G 000000G                MOVB    TARGET.ADR+1(R1),TEMP2          ;                               2476
000144  105037  000001G                        CLRB    TEMP2+1
000150  013701  000000G                        MOV     COUNTER,R1                      ;                               2477
000154  006301                                 ASL     R1
000156  005000                                 CLR     R0
000160  153700  000000G                        BISB    TEMP2,R0
000164  053700  000000G                        BIS     TEMP1,R0
000170  010061  000000G                        MOV     R0,STATION.ADR(R1)
000174  005237  000000G                        INC     COUNTER                         ;                               2478
000200  062702  000002                         ADD     #2,R2                           ; *,INDEX5                      2472
000204  020227  000005                         CMP     R2,#5                           ; INDEX5,*
000210  003734                                 BLE     4$
000212  012737  000022  000000G                MOV     #22,COUNTER                     ;                               2485
000220  112737  000045  000000G                MOVB    #45,PHYS.ADR                    ;                               2486
000226  112737  000101  000001G                MOVB    #101,PHYS.ADR+1                 ;                               2487
000234  112737  000040  000023G                MOVB    #40,PHYS.ADR+23                 ;                               2488
000242  112737  000045  000024G                MOVB    #45,PHYS.ADR+24                 ;                               2489
000250  112737  000116  000025G                MOVB    #116,PHYS.ADR+25                ;                               2490
000256  012701  000004                         MOV     #4,R1                           ; *,INDEX1                      2492
000262  016137  000000G 000000G          5$:    MOV     STATION.ADR(R1),TEMP3           ; *(INDEX1),*                   2494
000270  012703  000002                         MOV     #2,R3                           ; *,INDEX2                      2495
000274  012702  000002                  6$:    MOV     #2,R2                           ; *,INDEX3                      2497
000300  013737  000000G 000000G          7$:    MOV     TEMP3,TEMP1                     ;                               2499
000306  042737  177760  000000G                BIC     #177760,TEMP1
000314  013700  000000G                        MOV     TEMP1,R0                        ;                               2500
000320  020027  000011                         CMP     R0,#11
000324  101006                                 BHI     8$
000326  010037  000000G                        MOV     R0,TBYTE1                       ;                               2502
000332  062737  000060  000000G                ADD     #60,TBYTE1
000340  000405                                 BR      9$                              ;                               2500
000342  010037  000000G                  8$:    MOV     R0,TBYTE1                       ;                               2504
000346  062737  000067  000000G                ADD     #67,TBYTE1
000354  013700  000000G                  9$:    MOV     COUNTER,R0                      ;                               2505
000360  113760  000000G 000000G                MOVB    TBYTE1,PHYS.ADR(R0)
000366  005337  000000G                        DEC     COUNTER                         ;                               2506
000372  013700  000000G                        MOV     TEMP3,R0                        ;                               2507
000376  072027  177774                         ASH     #-4,R0
000402  010037  000000G                        MOV     R0,TEMP3
000406  077244                                 SOB     R2,7$                           ; INDEX3,*                      2497
```

```
000410  013702  000000G                        MOV     COUNTER,R2                      ;                               2510
000414  020227  000002                         CMP     R2,#2
000420  101403                                 BLOS    10$
000422  112762  000055  000000G                MOVB    #55,PHYS.ADR(R2)                ;                               2512
000430  005337  000000G              10$:      DEC     COUNTER                         ;                               2514
000434  077361                                 SOB     R3,6$                           ; INDEX2,*                      2495
000436  162701  000002                         SUB     #2,R1                           ; *,INDEX1                      2492
000442  100307                                 BPL     5$
000444  005726                                 TST     (SP)+                           ;                               2435
000446  000207                                 RTS     PC

; Routine Size:  148 words,     Routine Base:  AC$CODE$ + 4716
; Maximum stack depth per invocation:  6 words


;   2520  1
;   2521  1
```

```
;    2522  1     %SBTTL 'GLOBAL ROUTINE   XMIT SETUP PACKET ( P1 )'
;    2523  1
;    2524  1     GLOBAL ROUTINE XMIT_SETUP_PACKET ( P1 ) : NOVALUE  =
;    2525  1
;    2526  1     !++
;    2527  1     !
;    2528  1     !  GLOBAL ROUTINE :     XMIT SETUP_PACKET
;    2529  1     !
;    2530  1     !  DESCRIPTION:
;    2531  1     !
;    2532  1     !       This routine initializes descriptor lists to transmit and receive
;    2533  1     !       unchained Setup loopback packet. After loopback packet has been
;    2534  1     !       received DEQNA CSR, transmit and receive status registers are
;    2535  1     !       checked for proper status. Finally, transmit and receive packets
;    2536  1     !       are compared to verify that they are identical.
;    2537  1     !
;    2538  1     !       XMIT_D_LIST [ 0 ] = NEWB            RCV_D_LIST [ 0 ] = NEWB
;    2539  1     !       XMIT_D_LIST [ 1 ] = VSE             RCV_D_LIST [ 1 ] = VE
;    2540  1     !       XMIT_D_LIST [ 2 ] = XMIT_BUFFER     RCV_D_LIST [ 2 ] = RCV_BUFFER
;    2541  1     !       XMIT_D_LIST [ 3 ] = .XBUF_LENGTH    RCV_D_LIST [ 3 ] = .XBUF_LENGTH
;    2542  1     !       XMIT_D_LIST [ 4 ] = 0               RCV_D_LIST [ 4 ] = 0
;    2543  1     !       XMIT_D_LIST [ 5 ] = 0               RCV_D_LIST [ 5 ] = 0
;    2544  1     !       XMIT_D_LIST [ 6 ] = V               RCV_D_LIST [ 6 ] = V
;    2545  1     !       XMIT_D_LIST [ 7 ] = E               RCV_D_LIST [ 7 ] = E
;    2546  1     !
;    2547  1     !
;    2548  1     !  INPUT PARAMETERS:
;    2549  1     !
;    2550  1     !       P1 - transmit buffer length in bytes
;    2551  1     !
;    2552  1     !--
;    2553  1
;    2554  2     BEGIN
;    2555  2
;    2556  2     CLR_DESCR ( );
;    2557  2     RBUF_LENGTH = .P1;
;    2558  2     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    2559  2     SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2560  2     SET_XDESCR_LIST ( .XBUF_LENGTH, VSE );
;    2561  2
;    2562  2     IF .P1 EQLU A_MODE
;    2563  2       THEN
;    2564  3         BEGIN
;    2565  3           XBUF_LENGTH = - ( ( .RBUF_LENGTH  + -1 ) + 1 );
;    2566  3           SET_XDESCR_LIST ( .XBUF_LENGTH, VSEL );
;    2567  3           SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2568  2         END;
;    2569  2
;    2570  2       XMIT_AND_RCV_PACKET ( );
;    2571  2
;    2572  2       !++
;    2573  2       !  COMPARE STATUS REGISTERS TO EXPECTED VALUES
;    2574  2       !--
```

K5


```
;   2575  2
;   2576  2       CHK_RIXI_STATUS ( ONE );
;   2577  2       CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );          ! 0'100220', 0'100220'
;   2578  2       CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS );      ! 0'140000', 0'020000'
;   2579  2
;   2580  2       TEMP1 = XWD12_STATUS;                              ! 0'000400'
;   2581  2       IF .XMIT_D_LIST [ STE16 ]
;   2582  2         THEN
;   2583  2           TEMP1 = %O'002400';
;   2584  2       CHK_XMIT_STATUS ( XFLG_STATUS, .TEMP1 );           ! 0'140000', ???????
;   2585  2
;   2586  2       COMPARE_PACKETS ( );
;   2587  2
;   2588  1     END;


                                        .SBTTL  XMIT.SETUP.PACKET GLOBAL ROUTINE   XMIT SETUP_PACKET ( P1 )
000000  004737  001206'            XMIT.SETUP.PACKET::
                                        JSR     PC,CLR.DESCR                    ;                                       2556
000004  016637  000002  000000G         MOV     2(SP),RBUF.LENGTH               ; P1,*                                  2557
000012  016600  000002                  MOV     2(SP),R0                        ; RBUF.LENGTH,*                         2558
000016  006200                          ASR     R0
000020  005400                          NEG     R0
000022  010037  000000G                 MOV     R0,XBUF.LENGTH
000026  010046                          MOV     R0,-(SP)                        ; XBUF.LENGTH,*                         2559
000030  012746  120000                  MOV     #-60000,-(SP)
000034  004737  003554'                 JSR     PC,SET.RDESCR.LIST
000040  013716  000000G                 MOV     XBUF.LENGTH,(SP)                ;                                       2560
000044  012746  130000                  MOV     #-50000,-(SP)
000050  004737  003632'                 JSR     PC,SET.XDESCR.LIST
000054  026627  000010  000201         CMP     10(SP),#201                     ; P1,*                                  2562
000062  001023                          BNE     1$
000064  013700  000000G                 MOV     RBUF.LENGTH,R0                  ;                                       2565
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,*                         2566
000104  012746  130200                  MOV     #-47600,-(SP)
000110  004737  003632'                 JSR     PC,SET.XDESCR.LIST
000114  013716  000000G                 MOV     XBUF.LENGTH,(SP)                ;                                       2567
000120  012746  120000                  MOV     #-60000,-(SP)
000124  004737  003554'                 JSR     PC,SET.RDESCR.LIST
000130  022626                          CMP     (SP)+,(SP)+                     ;                                       2564
000132  004737  000000V         1$:     JSR     PC,XMIT.AND.RCV.PACKET          ;                                       2570
000136  012716  000001                  MOV     #1,(SP)                         ;                                       2576
000142  004737  001262'                 JSR     PC,CHK.RIXI.STATUS
000146  012716  100220                  MOV     #-77560,(SP)                    ;                                       2577
000152  011646                          MOV     (SP),-(SP)
000154  004737  001646'                 JSR     PC,CHK.CSR.STATUS
000160  012716  140000                  MOV     #-40000,(SP)                    ;                                       2578
000164  012746  020000                  MOV     #20000,-(SP)
000170  004737  002336'                 JSR     PC,CHK.RCV.STATUS
```

L5


```
000174  012737  000400  000000G                 MOV     #400,TEMP1              ;                       2580
000202  032737  002000  000010G                 BIT     #2000,XMIT.D.LIST+10    ;                       2581
000210  001403                                  BEQ     2$
000212  012737  002400  000000G                 MOV     #2400,TEMP1             ;                       2583
000220  012716  140000                  2$:     MOV     #-40000,(SP)            ;                       2584
000224  013746  000000G                         MOV     TEMP1,-(SP)
000230  004737  002040'                         JSR     PC,CHK.XMIT.STATUS
000234  004737  003136'                         JSR     PC,COMPARE.PACKETS      ;                       2586
000240  062706  000014                          ADD     #14,SP                  ;                       2554
000244  000207                                  RTS     PC                      ;                       2524

; Routine Size:  83 words,      Routine Base:  AC$CODE$ + 5366
; Maximum stack depth per invocation:  7 words


;   2589  1
;   2590  1
```

```
;  2591  1     %SBTTL 'GLOBAL ROUTINE - SEND_ELOOP_PACKET ( P3 ) '
;  2592  1
;  2593  1     GLOBAL ROUTINE SEND_ELOOP_PACKET ( P3 ) : NOVALUE  =
;  2594  1
;  2595  1     !++
;  2596  1     !
;  2597  1     !  GLOBAL ROUTINE :     SEND_ELOOP_PACKET
;  2598  1     !
;  2599  1     !  DESCRIPTION:
;  2600  1     !
;  2601  1     !       This routine initializes transmit and receive descriptor lists and
;  2602  1     !       then initiates transmissin of a loopback packet. After
;  2603  1     !       loopback packet is received DEQNA CSR, transmit and receive status r
;  2604  1     !       egisters are checked for proper status. Finally, transmit and receive
;  2605  1     !       packets are compared to verify that they are identical.
;  2606  1     !
;  2607  1     !       XMIT_D_LIST [ 0 ] = NEWB            RCV_D_LIST [ 0 ] = NEWB
;  2608  1     !       XMIT_D_LIST [ 1 ] = VE              RCV_D_LIST [ 1 ] = VE
;  2609  1     !       XMIT_D_LIST [ 2 ] = XMIT_BUFFER     RCV_D_LIST [ 2 ] = RCV_BUFFER
;  2610  1     !       XMIT_D_LIST [ 3 ] = .XBUF_LENGTH    RCV_D_LIST [ 3 ] = .XBUF_LENGTH
;  2611  1     !       XMIT_D_LIST [ 4 ] = 0               RCV_D_LIST [ 4 ] = 0
;  2612  1     !       XMIT_D_LIST [ 5 ] = 0               RCV_D_LIST [ 5 ] = 0
;  2613  1     !       XMIT_D_LIST [ 6 ] = V               RCV_D_LIST [ 6 ] = V
;  2614  1     !       XMIT_D_LIST [ 7 ] = E               RCV_D_LIST [ 7 ] = E
;  2615  1     !
;  2616  1     !
;  2617  1     !  INPUT PARAMETERS:
;  2618  1     !
;  2619  1     !       P3 -
;  2620  1     !--
;  2621  1
;  2622  2     BEGIN
;  2623  2
;  2624  2       PUT_BIT ( CSR, LB, INX_LOOPBACK );
;  2625  2       XMIT_AND_RCV_PACKET ( );
;  2626  2
;  2627  2       !++
;  2628  2       !  COMPARE STATUS REGISTERS TO EXPECTED VALUES
;  2629  2       !--
;  2630  2
;  2631  2       CHK_RIXI_STATUS ( ZERO );
;  2632  2       CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );              ! 0'100220', 0'100220'
;  2633  2       CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );         ! 0'140000', 0'000400'
;  2634  2
;  2635  2       IF .P3 EQLU ZERO
;  2636  2         THEN
;  2637  3           BEGIN
;  2638  3             CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS );    ! 0'140000', 0'020000'
;  2639  3           END
;  2640  2         ELSE
;  2641  3           BEGIN
;  2642  3             TEMP1 = RWD14_STATUS;                            ! 0'060000'
;  2643  3             IF .RCV_D_LIST [ STWD1 ] AND %0'070001' EQLU %0'070001'
```

```
;   2644  3               THEN
;   2645  3                 TEMP1 = %O'070001';
;   2646  3             CHK RCV STATUS  ( RFLG_STATUS, .TEMP1 );        ! O'140000', ?????
;   2647  2         END;
;   2648  1     END;


                                                .SBTTL  SEND.ELOOP.PACKET GLOBAL ROUTINE - SEND ELOOP_PACKET ( P3 )
;  000000  013700  000000G                  SEND.ELOOP.PACKET::
                                                MOV     REG.ADR,R0                      ;                       2624
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          ;                       2625
000024  005046                                  CLR     -(SP)                           ;                       2631
000026  004737  001262'                         JSR     PC,CHK.RIXI.STATUS
000032  012716  100220                          MOV     #-77560,(SP)                    ;                       2632
000036  011646                                  MOV     (SP),-(SP)
000040  004737  001646'                         JSR     PC,CHK.CSR.STATUS
000044  012716  140000                          MOV     #-40000,(SP)                    ;                       2633
000050  012746  000400                          MOV     #400,-(SP)
000054  004737  002040'                         JSR     PC,CHK.XMIT.STATUS
000060  005766  000010                          TST     10(SP)                          ; P3                    2635
000064  001005                                  BNE     1$
000066  012716  140000                          MOV     #-40000,(SP)                    ;                       2638
000072  012746  020000                          MOV     #20000,-(SP)
000076  000416                                  BR      3$
000100  012737  060000  000000G         1$:     MOV     #60000,TEMP1                    ;                       2642
000106  032737  000001  000010G                 BIT     #1,RCV.D.LIST+10                ;                       2643
000114  001403                                  BEQ     2$
000116  012737  070001  000000G                 MOV     #70001,TEMP1                    ;                       2645
000124  012716  140000                  2$:     MOV     #-40000,(SP)                    ;                       2646
000130  013746  000000G                          MOV     TEMP1,-(SP)
000134  004737  002336'                 3$:     JSR     PC,CHK.RCV.STATUS
000140  062706  000010                          ADD     #10,SP                          ;                       2622
000144  000207                                  RTS     PC                              ;                       2593

; Routine Size:  51 words,      Routine Base:  AC$CODE$ + 5634
; Maximum stack depth per invocation:  5 words


;   2649  1
```

```
;  2650  1     %SBTTL 'GLOBAL ROUTINE   SEND_TEST_PACKET '
;  2651  1
;  2652  1     GLOBAL ROUTINE SEND_TEST PACKET : NOVALUE  =
;  2653  1
;  2654  1     !++
;  2655  1     !
;  2656  1     !  GLOBAL ROUTINE :     SEND_TEST_PACKET
;  2657  1     !
;  2658  1     !  DESCRIPTION:
;  2659  1     !
;  2660  1     !       This routine initializes transmit and receive descriptor lists and
;  2661  1     !       then initiates transmissin of an external loopback packet.
;  2662  1     !
;  2663  1     !       XMIT_D_LIST [ 0 ] = NEWB            RCV_D_LIST [ 0 ] = NEWB
;  2664  1     !       XMIT_D_LIST [ 1 ] = VE              RCV_D_LIST [ 1 ] = VE
;  2665  1     !       XMIT_D_LIST [ 2 ] = XMIT_BUFFER     RCV_D_LIST [ 2 ] = RCV_BUFFER
;  2666  1     !       XMIT_D_LIST [ 3 ] = .XBUF_LENGTH    RCV_D_LIST [ 3 ] = .XBUF_LENGTH
;  2667  1     !       XMIT_D_LIST [ 4 ] = 0               RCV_D_LIST [ 4 ] = 0
;  2668  1     !       XMIT_D_LIST [ 5 ] = 0               RCV_D_LIST [ 5 ] = 0
;  2669  1     !       XMIT_D_LIST [ 6 ] = V               RCV_D_LIST [ 6 ] = V
;  2670  1     !       XMIT_D_LIST [ 7 ] = E               RCV_D_LIST [ 7 ] = E
;  2671  1     !
;  2672  1     !
;  2673  1     !  INPUT PARAMETERS:
;  2674  1     !
;  2675  1     !       None
;  2676  1     !--
;  2677  1
;  2678  2     BEGIN
;  2679  2
;  2680  2       !++
;  2681  2       !  WRITE ETHERNET STATION ADDRESS AND DATA PATTERN INTO THE TRANSMIT BUFFER
;  2682  2       !--
;  2683  2
;  2684  2       RESET_DEQNA ( );
;  2685  2
;  2686  2       INCR INDEX FROM 0 TO 5 DO
;  2687  3         BEGIN
;  2688  3           XMIT_BUFFER [ .INDEX ]       = .TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX ];
;  2689  3           XMIT_BUFFER [ .INDEX + 6 ]   = .TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX ];
;  2690  2         END;
;  2691  2
;  2692  2       XMIT_BUFFER [ PKT_TYPE ]     = LPB_PKT;
;  2693  2       XMIT_BUFFER [ PKT_TYPE + 1 ] = SKIP_CNT;
;  2694  2       XMIT_BUFFER [ PKT_TYPE + 2 ] = RFC;
;  2695  2
;  2696  2       !++
;  2697  2       !  CONVERT SETUP PACKET SIZE FROM BYTE COUNT TO WORD COUNT AND SET UP
;  2698  2       !  DESCRIPTOR LISTS
;  2699  2       !--
;  2700  2
;  2701  2       RBUF_LENGTH = PKT_LENGTH + 14;
;  2702  2       XBUF_LENGTH =   ( .RBUF LENGTH ^ -1 );
```

C6


```
;   2703  2
;   2704  2       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;   2705  2       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;   2706  2
;   2707  2       !++
;   2708  2       ! SET DEQNA TO EXTERNAL LOOPBACK MODE AND SEND LOOPBACK PACKET
;   2709  2       !--
;   2710  2
;   2711  2       PUT_BIT ( CSR, LB, EXT_LOOPBACK );
;   2712  2       XMIT_AND_RCV_PACKET ( );
;   2713  2
;   2714  1     END;


                                        .SBTTL  SEND.TEST.PACKET GLOBAL ROUTINE - SEND_TEST_PACKET
000000  004737  000324'              SEND.TEST.PACKET::
                                        JSR     PC,RESET.DEQNA                  ;                               2684
000004  005000                          CLR     R0                              ; INDEX                         2686
000006  116060  000162G 000000G    1$: MOVB    TARGET.ADR+162(R0),XMIT.BUFFER(R0) ;
                                                                                ; *(INDEX),*(INDEX)             2688
000014  116060  000162G 000006G        MOVB    TARGET.ADR+162(R0),XMIT.BUFFER+6(R0) ;
                                                                                ; *(INDEX),*(INDEX)             2689
000022  005200                          INC     R0                              ; INDEX                         2686
000024  020027  000005                  CMP     R0,#5                           ; INDEX,*
000030  003766                          BLE     1$
000032  112737  000220  000014G        MOVB    #220,XMIT.BUFFER+14             ;                               2692
000040  105037  000015G                CLRB    XMIT.BUFFER+15                  ;                               2693
000044  112737  000001  000016G        MOVB    #1,XMIT.BUFFER+16               ;                               2694
000052  012737  002752  000000G        MOV     #2752,RBUF.LENGTH               ;                               2701
000060  012700  002752                  MOV     #2752,R0                        ;                               2702
000064  006200                          ASR     R0
000066  005400                          NEG     R0
000070  010037  000000G                MOV     R0,XBUF.LENGTH
000074  010046                          MOV     R0,-(SP)                        ; XBUF.LENGTH,*                 2704
000076  012746  120000                  MOV     #-60000,-(SP)
000102  004737  003554'                JSR     PC,SET.RDESCR.LIST
000106  013716  000000G                MOV     XBUF.LENGTH,(SP)                ;                               2705
000112  012746  120000                  MOV     #-60000,-(SP)
000116  004737  003632'                JSR     PC,SET.XDESCR.LIST
000122  013700  000000G                MOV     REG.ADR,R0                      ;                               2711
000126  052760  001400  000016        BIS     #1400,16(R0)
000134  004737  000000V                JSR     PC,XMIT.AND.RCV.PACKET          ;                               2712
000140  062706  000006                  ADD     #6,SP                           ;                               2678
000144  000207                          RTS     PC                              ;                               2652

; Routine Size:  51 words,      Routine Base:  AC$CODE$ + 6002
; Maximum stack depth per invocation:  4 words


;   2715  1
```

D6


```
;  2716  1     %SBTTL 'GLOBAL ROUTINE   INTR_TEST_PACKET '
;  2717  1
;  2718  1     GLOBAL ROUTINE INTR_TEST_PACKET : NOVALUE  =
;  2719  1
;  2720  1     !++
;  2721  1     !
;  2722  1     !  GLOBAL ROUTINE :     INTR TEST_PACKET
;  2723  1     !
;  2724  1     !  DESCRIPTION:
;  2725  1     !
;  2726  1     !       This routine initializes transmit and receive descriptor lists and
;  2727  1     !       then initiates transmissin of an external loopback packet. A reset
;  2728  1     !       is never done, so the state of IE does not change.
;  2729  1     !
;  2730  1     !       XMIT_D_LIST [ 0 ] = NEWB            RCV_D_LIST [ 0 ] = NEWB
;  2731  1     !       XMIT_D_LIST [ 1 ] = VE              RCV_D_LIST [ 1 ] = VE
;  2732  1     !       XMIT_D_LIST [ 2 ] = XMIT_BUFFER     RCV_D_LIST [ 2 ] = RCV_BUFFER
;  2733  1     !       XMIT_D_LIST [ 3 ] = .XBUF_LENGTH    RCV_D_LIST [ 3 ] = .XBUF_LENGTH
;  2734  1     !       XMIT_D_LIST [ 4 ] = 0               RCV_D_LIST [ 4 ] = 0
;  2735  1     !       XMIT_D_LIST [ 5 ] = 0               RCV_D_LIST [ 5 ] = 0
;  2736  1     !       XMIT_D_LIST [ 6 ] = V               RCV_D_LIST [ 6 ] = V
;  2737  1     !       XMIT_D_LIST [ 7 ] = E               RCV_D_LIST [ 7 ] = E
;  2738  1     !
;  2739  1     !
;  2740  1     !  INPUT PARAMETERS:
;  2741  1     !
;  2742  1     !       None
;  2743  1     !
;  2744  1
;  2745  2     BEGIN
;  2746  2
;  2747  2       !++
;  2748  2       !  WRITE ETHERNET STATION ADDRESS AND DATA PATTERN INTO THE TRANSMIT BUFFER
;  2749  2       !--
;  2750  2
;  2751  2
;  2752  2       INCR INDEX FROM 0 TO 5 DO
;  2753  3         BEGIN
;  2754  3           XMIT_BUFFER [ .INDEX ]       = .TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX ];
;  2755  3           XMIT_BUFFER [ .INDEX + 6 ]   = .TARGET_ADR [ ( PHA_INDEX * 6 ) + .INDEX ];
;  2756  2         END;
;  2757  2
;  2758  2       XMIT_BUFFER [ PKT_TYPE ]     = LPB_PKT;
;  2759  2       XMIT_BUFFER [ PKT_TYPE + 1 ] = SKIP_CNT;
;  2760  2       XMIT_BUFFER [ PKT_TYPE + 2 ] = RFC;
;  2761  2
;  2762  2       !++
;  2763  2       !  CONVERT SETUP PACKET SIZE FROM BYTE COUNT TO WORD COUNT AND SET UP
;  2764  2       !  DESCRIPTOR LISTS
;  2765  2       !--
;  2766  2
;  2767  2       RBUF_LENGTH = PKT_LENGTH + 14;
;  2768  2       XBUF_LENGTH = - ( .RBUF_LENGTH + 1 );
```

E6


```
;   2769  2
;   2770  2       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;   2771  2       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;   2772  2
;   2773  2       !++
;   2774  2       !  SET DEQNA TO EXTERNAL LOOPBACK MODE AND SEND LOOPBACK PACKET
;   2775  2       !--
;   2776  2
;   2777  2       PUT_BIT ( CSR, LB, EXT_LOOPBACK );          !this does a BIS, so IE stays
;   2778  2       XMIT_AND_RCV_PACKET ( );
;   2779  2
;   2780  1     END;


                                                .SBTTL  INTR.TEST.PACKET GLOBAL ROUTINE   INTR_TEST_PACKET
000000  005000                            INTR.TEST.PACKET::
                                                CLR     R0                              ; INDEX                         2752
000002  116060  000162G 000000G            1$:     MOVB    TARGET.ADR+162(R0),XMIT.BUFFER(R0) ;
                                                                                        ; *(INDEX),*(INDEX)             2754
000010  116060  000162G 000006G                    MOVB    TARGET.ADR+162(R0),XMIT.BUFFER+6(R0) ;
                                                                                        ; *(INDEX),*(INDEX)             2755
000016  005200                                      INC     R0                              ; INDEX                         2752
000020  020027  000005                              CMP     R0,#5                           ; INDEX,*
000024  003766                                      BLE     1$
000026  112737  000220  000014G                     MOVB    #220,XMIT.BUFFER+14             ;                               2758
000034  105037  000015G                             CLRB    XMIT.BUFFER+15                  ;                               2759
000040  112737  000001  000016G                     MOVB    #1,XMIT.BUFFER+16               ;                               2760
000046  012737  002752  000000G                     MOV     #2752,RBUF.LENGTH               ;                               2767
000054  012700  002752                              MOV     #2752,R0                        ;                               2768
000060  006200                                      ASR     R0
000062  005400                                      NEG     R0
000064  010037  000000G                             MOV     R0,XBUF.LENGTH
000070  010046                                      MOV     R0,-(SP)                        ; XBUF.LENGTH,*                 2770
000072  012746  120000                              MOV     #-60000,-(SP)
000076  004737  003554'                             JSR     PC,SET.RDESCR.LIST
000102  013716  000000G                             MOV     XBUF.LENGTH,(SP)                ;                               2771
000106  012746  120000                              MOV     #-60000,-(SP)
000112  004737  003632'                             JSR     PC,SET.XDESCR.LIST
000116  013700  000000G                             MOV     REG.ADR,R0                      ;                               2777
000122  052760  001400  000016                      BIS     #1400,16(R0)
000130  004737  000000V                             JSR     PC,XMIT.AND.RCV.PACKET          ;                               2778
000134  062706  000006                              ADD     #6,SP                           ;                               2745
000140  000207                                      RTS     PC                              ;                               2718

; Routine Size:  49 words,      Routine Base:  AC$CODE$ + 6150
; Maximum stack depth per invocation:  4 words


;   2781  1
;   2782  1
```

F6


```
;   2783  1     %SBTTL 'GLOBAL ROUTINE   XMIT_AND RCV_PACKET '
;   2784  1
;   2785  1     GLOBAL ROUTINE XMIT_AND RCV_PACKET : NOVALUE  =
;   2786  1
;   2787  1     !++
;   2788  1     !
;   2789  1     !  GLOBAL ROUTINE :     XMIT_AND_RCV_PACKET
;   2790  1     !
;   2791  1     !  DESCRIPTION:
;   2792  1     !
;   2793  1     !       This routine initiates transmit and receive operations.
;   2794  1     !
;   2795  1     !  INPUT PARAMETERS:
;   2796  1     !
;   2797  1     !
;   2798  1     !
;   2799  1     !
;   2800  1     !--
;   2801  1
;   2802  2     BEGIN
;   2803  2
;   2804  2       .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;   2805  2       .IOP_TABLE [ RHI_ADR ] = 0;
;   2806  2
;   2807  2       .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;   2808  2       .IOP_TABLE [ XHI_ADR ] = 0;
;   2809  2
;   2810  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        ;                       2804
000006  005077  000006G                 CLR     @IOP.TABLE+6                    ;                       2805
000012  012777  000000G 000010G         MOV     #XMIT.D.LIST,@IOP.TABLE+10      ;                       2807
000020  005077  000012G                 CLR     @IOP.TABLE+12                   ;                       2808
000024  000207                          RTS     PC                              ;                       2785

; Routine Size:  11 words,      Routine Base:  AC$CODE$ + 6312
; Maximum stack depth per invocation:  0 words


;   2811  1
;   2812  1
```

```
;  2813  1     %SBTTL 'GLOBAL ROUTINE - XMIT_ILOOP_PACKET ( P3 ) '
;  2814  1
;  2815  1     GLOBAL ROUTINE XMIT_ILOOP_PACKET ( P3 ) : NOVALUE  =
;  2816  1
;  2817  1     !++
;  2818  1     !
;  2819  1     !  GLOBAL ROUTINE :     XMIT_ILOOP_PACKET
;  2820  1     !
;  2821  1     !  DESCRIPTION:
;  2822  1     !
;  2823  1     !       This routine
;  2824  1     !
;  2825  1     !  INPUT PARAMETERS:
;  2826  1     !
;  2827  1     !       P3 - selector
;  2828  1     !
;  2829  1     !--
;  2830  1
;  2831  2     BEGIN
;  2832  2
;  2833  2       CLR_DESCR ( );
;  2834  2
;  2835  2       SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;  2836  2       SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;  2837  2
;  2838  2       XMIT_AND RCV_PACKET ( );
;  2839  2
;  2840  2       .IOP_TABLE [ CSR ] = EENABLE;
;  2841  2
;  2842  2       IF .P3 EQLU ONE
;  2843  2         THEN
;  2844  3           BEGIN
;  2845  3             CHK_RIXI_STATUS ( ONE );
;  2846  3             CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );       ! 0'100220', 0'100220'
;  2847  3             CHK_RCV_STATUS  ( RFLG_STATUS, RWD16_STATUS );  ! 0'140000', 0'044000'
;  2848  3           END
;  2849  2         ELSE
;  2850  3           BEGIN
;  2851  3             CHK_RIXI_STATUS ( ZERO );
;  2852  3             CHK_CSR_STATUS  ( CSR_STATUS, CSR_MASK );       ! 0'100220', 0'100220'
;  2853  3             CHK_RCV_STATUS  ( RFLG_STATUS, RWD13_STATUS );  ! 0'140000', 0'000000'
;  2854  2           END;
;  2855  2
;  2856  2       CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );        ! 0'140000', 0'000400'
;  2857  2       COMPARE_PACKETS ( );
;  2858  2       .IOP_TABLE [ CSR ] = DISABLE;
;  2859  2
;  2860  1     END;


                                        .SBTTL  XMIT.ILOOP.PACKET GLOBAL ROUTINE - XMIT_ILOOP_PACKET ( P3 )
000000  004737  001206'          XMIT.ILOOP.PACKET::
                                        JSR     PC,CLR.DESCR                    ;                                       2833
```

H6


```
000004  013746  000000G                        MOV     XBUF.LENGTH,-(SP)               ;                       2835
000010  012746  120000                          MOV     #-60000,-(SP)
000014  004737  003554'                        JSR     PC,SET.RDESCR.LIST
000020  013716  000000G                        MOV     XBUF.LENGTH,(SP)                ;                       2836
000024  012746  120000                          MOV     #-60000,-(SP)
000030  004737  003632'                        JSR     PC,SET.XDESCR.LIST
000034  004737  006312'                        JSR     PC,XMIT.AND.RCV.PACKET          ;                       2838
000040  012777  000001  000016G                MOV     #1,@IOP.TABLE+16                ;                       2840
000046  026627  000010  000001                CMP     10(SP),#1                       ; P3,*                  2842
000054  001016                                BNE     1$
000056  012716  000001                        MOV     #1,(SP)                         ;                       2845
000062  004737  001262'                        JSR     PC,CHK.RIXI.STATUS
000066  012716  100220                        MOV     #-77560,(SP)                    ;                       2846
000072  011646                                MOV     (SP),-(SP)
000074  004737  001646'                        JSR     PC,CHK.CSR.STATUS
000100  012716  140000                        MOV     #-40000,(SP)                    ;                       2847
000104  012746  044000                        MOV     #44000,-(SP)
000110  000413                                BR      2$
000112  005016                        1$:     CLR     (SP)                            ;                       2851
000114  004737  001262'                        JSR     PC,CHK.RIXI.STATUS
000120  012716  100220                        MOV     #-77560,(SP)                    ;                       2852
000124  011646                                MOV     (SP),-(SP)
000126  004737  001646'                        JSR     PC,CHK.CSR.STATUS
000132  012716  140000                        MOV     #-40000,(SP)                    ;                       2853
000136  005046                                CLR     -(SP)
000140  004737  002336'                2$:     JSR     PC,CHK.RCV.STATUS
000144  012716  140000                        MOV     #-40000,(SP)                    ;                       2856
000150  012746  000400                        MOV     #400,-(SP)
000154  004737  002040'                        JSR     PC,CHK.XMIT.STATUS
000160  004737  003136'                        JSR     PC,COMPARE.PACKETS              ;                       2857
000164  005077  000016G                        CLR     @IOP.TABLE+16                   ;                       2858
000170  062706  000014                        ADD     #14,SP                          ;                       2831
000174  000207                                RTS     PC                              ;                       2815

; Routine Size:  63 words,      Routine Base:  AC$CODE$ + 6340
; Maximum stack depth per invocation:  7 words


;   2861  1
;   2862  1
```

```
;   2863  1     %SBTTL 'GLOBAL ROUTINE   TURN OFF LED ( P1 )'
;   2864  1
;   2865  1     GLOBAL ROUTINE TURN_OFF_LED ( P1 ) : NOVALUE  =
;   2866  1
;   2867  1     !++
;   2868  1     !
;   2869  1     !  GLOBAL ROUTINE :     TURN OFF LED
;   2870  1     !
;   2871  1     !  DESCRIPTION:
;   2872  1     !
;   2873  1     !       This routine
;   2874  1     !
;   2875  1     !  INPUT PARAMETERS:
;   2876  1     !
;   2877  1     !       P1 -
;   2878  1     !
;   2879  1     !--
;   2880  1
;   2881  2     BEGIN
;   2882  2
;   2883  2     PREP_FOR_SETUP ( );
;   2884  2
;   2885  2     INCR INDEX1 FROM 1 TO 14 DO
;   2886  2       WRT_STATION_ADR ( .INDEX1, PHA_INDEX );
;   2887  2
;   2888  2     XMIT_SETUP_PACKET ( .P1 );
;   2889  2
;   2891  1     END;


                                        .SBTTL  TURN.OFF.LED GLOBAL ROUTINE - TURN_OFF_LED ( P1 )
000000  010146                 TURN.OFF.LED::
                                        MOV     R1,-(SP)                ;                               2865
000002  004737  004634'                 JSR     PC,PREP.FOR.SETUP       ;                               2883
000006  012701  000001                  MOV     #1,R1                   ; *,INDEX1                      2885
000012  010146                 1$:     MOV     R1,-(SP)                ; INDEX1,*                      2886
000014  012746  000023                  MOV     #23,-(SP)
000020  004737  004442'                 JSR     PC,WRT.STATION.ADR
000024  022626                          CMP     (SP)+,(SP)+
000026  005201                          INC     R1                      ; INDEX1                        2885
000030  020127  000016                  CMP     R1,#16                  ; INDEX1,*
000034  003766                          BLE     1$
000036  016646  000004                  MOV     4(SP),-(SP)             ; P1,*                          2888
000042  004737  005366'                 JSR     PC,XMIT.SETUP.PACKET
000046  005726                          TST     (SP)+                   ;                               2881
000050  012601                          MOV     (SP)+,R1                ;                               2865
000052  000207                          RTS     PC

; Routine Size:  22 words,      Routine Base:  AC$CODE$ + 6536
; Maximum stack depth per invocation:  4 words
```

```
;   2892  1
;   2893  1
;   2894  1     END
;   2895  0     ELUDOM


.                                       OTS external references
                                                .GLOBL  $SAVE3, $SAVE2


;                                       PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;        AC$CODE$                        1733       RO ,  I  ,  LCL,  REL,  CON


;                                 Library Statistics
;
;                                            -------- Symbols --------      Pages       Processing
;       File                                 Total    Loaded   Percent      Mapped      Time
;
;  DISK2:[SCODA.QNA.ZQNA]QNALIB.L16;2          224       134        59        14          00:0C.1




;                                       COMMAND QUALIFIERS

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

; Size:         1733 code + 0 data words
; Run Time:       00:35.3
; Elapsed Time:   00:37.8
; Lines/CPU Min:   4922
; Lexemes/CPU-Min: 35721
; Memory Used:  236 pages
; Compilation Complete
```

K6


```
;    0001  0     MODULE ZQNA5 (%TITLE 'CZQNAE0 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
```

L6


```
;   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 CZQNAE0 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
;
```

M6


```
;
;       File                                    Total   Symbols Loaded   Percent   Pages Mapped   Process ng T me
;
;  DISK2:[SCODA.QNA.ZQNA]QNALIB.L16;2             224              3         1             14          00:00.1
```

```
;                                       COMMAND QUALIFIERS

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

; S ze:          1 code + 7 data words
; Run Time:         00:04.3
; Elapsed Time:     00:05.3
; Lines/CPU Min:    21217
; Lexemes/CPU-Min:112903
; Memory Used:  102 pages
; Compilation Complete
```

N6


```
                                ;       Subroutine to calculate station address rom checksum
                                ;       This was necessary because the algorithm in the ETHERNET spec
                                ;       was written for a 32 bit data word. This routine uses a 16 bit
                                ;       word, but uses the PSW C bit to look for what would be bit 16
                                ;       getting set.
                                ;
                                ;
                                ;       INPUTS:: array STATION.ADR has 6 bytes of default physical address
                                ;
                                ;       OUTPUTS:: word CHECKSUM has the 16 bit checksum for the above address
                                ;
                                ;       TMPR1 is used as a counter variable
                                ;       R0 is saved, used, and restored
                                ;
                                ;       added Oct-1985 by Dave Scoda

000000                          romchk::
000000  010046                          mov     r0,-(sp)                ;save r0, for BLISS changes
000002  005067  000000G                 CLR     CHECKSUM                ;
000006  005067  000000G                 CLR     tmpr1                   ; INDEX
000012  016700  000000G         2$:     MOV     CHECKSUM,R0             ;
000016  006300                          ASL     R0
000020  032767  100000  000000G         BIT     #-100000,CHECKSUM       ;
000026  001405                          BEQ     3$
000030  010067  000000G                 MOV     R0,CHECKSUM             ;
000034  005267  000000G                 INC     CHECKSUM                ;
000040  000402                          BR      4$                      ;
000042  010067  000000G         3$:     MOV     R0,CHECKSUM             ;
000046  016700  000000G         4$:     MOV     COUNTER,R0              ;
000052  006300                          ASL     R0
000054  000241                          clc                             ;clear c before use
000056  066067  000000G 000000G         ADD     STATION.ADR(R0),CHECKSUM
000064  005567  000000G                 adc     checksum                ;fix algorithm
000070  005267  000000G                 INC     COUNTER                 ;
000074  062767  000002  000000G         ADD     #2,tmpr1                ; *,INDEX
000102  026727  000000G 000005          CMP     tmpr1,#5                ; INDEX,*
000110  003740                          BLE     2$                      ;
000112  012600                          mov     (sp)+,r0                ;restore r0
000114  000207                          rts     pc                      ;done
        000001                          .end
```

B7

```
.MAIN.  MACRO V05.03  Thursday 27 Mar 86 07:38  Page 1 1                                SEQ 286
Symbol table

CHECKS= ****** GX       COUNTE= ****** GX       ROMCHK  000000RG        STATIO= ****** GX       TMPR1 = ****** GX


. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000116    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 39 Words  ( 1 Pages)
Size of core pool: 19684 Words  ( 75 Pages)
Operating  system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:00:01.17
ZQNA6.OBJ,ZQNA6.LIS/ SP=ZQNA6
```

```
;  0001  0     ! update history:::
;  0002  0     !
;  0003  0     !       Dave Scoda      5-Nov-85        Changed R2_MASK from 174013 to 174003
;  0004  0     !                                       Changed RWD2_MASK from 177417 to 177407
;  0005  0     !                       5-Mar-86        Added to Software P table, swp_size
;  0006  0     !
;  0007  0     !++
;  0008  0     !
;  0009  0     !  DEFINE DATA STRUCTURES IN THIS SECTION
;  0010  0     !
;  0011  0     !-
;  0012  0     !
;  0013  0     STRUCTURE                                       ! DEFINE ACCESS ALGORITHM
;  0014  0        REG_STR [ O, P, S, E ]=
;  0015  1           BEGIN
;  0016  1              LOCAL TMP_LOCATION;
;  0017  1              TMP_LOCATION = .(REG_STR + %UPVAL * O) <O,%BPVAL,O>;
;  0018  1              TMP_LOCATION
;  0019  0           END < P, S, E >;
;  0020  0     !
;  0021  0     !
;  0022  0     STRUCTURE                                       ! DEFINE ACCESS ALGORITHM
;  0023  0        ADR_STR [ O, P, S, E ]=
;  0024  1           BEGIN
;  0025  1              LOCAL TMP_LOCATION;
;  0026  1              TMP_LOCATION = (ADR_STR + %UPVAL * O) <O,%BPVAL,O>;
;  0027  1              TMP_LOCATION
;  0028  0           END < P, S, E >;
;  0029  0     !
;  0030  0     STRUCTURE                                       ! DEFINE ACCESS ALGORITHM
;  0031  0        LBLOCK [ O, P, S, E, I ]=
;  0032  1           BEGIN
;  0033  1             CASE I FROM O TO 2 OF
;  0034  1                SET
;  0035  1                [ O ]:
;  0036  1                     ( LBLOCK + O * %UPVAL );
;  0037  1                [ 1 ]:
;  0038  1                     ( .LBLOCK + O * %UPVAL );
;  0039  1                [ 2 ]:
;  0040  1                     ( .LBLOCK + O * %UPVAL );
;  0041  1                TES;
;  0042  0            END < P, S, E >;
```

```
;    0043  0      !++
;    0044  0      !
;    0045  0      !   MACRO DEFINITIONS
;    0046  0      !
;    0047  0      ! -
;    0048  0
;    0049  0      MACRO
;    0050  0
; M  0051  0          TST_BIT ( ADDR, EXPECTED ) =
; M  0052  0              ( IF ( .ADDR AND EXPECTED ) EQLU EXPECTED
; M  0053  0                THEN
; M  0054  0                    TRUE
; M  0055  0                ELSE
;    0056  0                    FALSE )%,
;    0057  0
;    0058  0
; M  0059  0          PUT BIT ( OFFSET, POSITION, IMAGE ) =
; M  0060  0              BEGIN
; M  0061  0              ( .REG_ADR + %UPVAL * OFFSET )< %FIELDEXPAND ( POSITION ) > = IMAGE;
;    0062  0              END%,
;    0063  0
; M  0064  0          GET_STATION_ADR ( OFFSET, POSITION, IMAGE ) =
; M  0065  0              BEGIN
; M  0066  0              ( .STATION_ADR + OFFSET )< %FIELDEXPAND ( POSITION ) > = IMAGE;
;    0067  0              END%,
;    0068  0
;    0069  0
;    0070  0      !++
;    0071  0      !
;    0072  0      !  THIS MACRO GETS BITS SPECIFIED BY THE FIELD NAME " POSITION "
;    0073  0      !  AND MEMORY LOC SPECIFIED BY (.REG_ADR + %UPVAL * OFFSET)
;    0074  0      !
;    0075  0      !--
;    0076  0
; M  0077  0              GET BIT ( OFFSET, POSITION ) =
; M  0078  0
;    0079  0                      .REG_ADR [ OFFSET, POSITION ] %;
;    0080  0
```

```
;  0081  0
;  0082  0
;  0083  0     !++
;  0084  0     !
;  0085  0     !   PROGRAM LITERALS
;  0086  0     !
;  0087  0     !--
;  0088  0
;  0089  0     LITERAL
;  0090  0
;  0091  0        NO              = 0,            !
;  0092  0        YES             = 1,            !
;  0093  0        FALSE           = 0,            !
;  0094  0        TRUE            = 1,            !
;  0095  0        ZERO            = 0,            !
;  0096  0        ONE             = 1,            !
;  0097  0        DISABLE         = 0,            !
;  0098  0        EENABLE         = 1,            !
;  0099  0
;  0100  0        P_CLOCK         = 1,            !
;  0101  0        L_CLOCK         = 1,            !
;  0102  0        NO_CLOCK        = 0,            !
;  0103  0        CLEAR_FLG       = 0,            !
;  0104  0        SET_FLG         = 1,            !
;  0105  0        PWR_DELAY       = 10000,        !
;  0106  0        M1_DELAY        = 10,           !
;  0107  0        M2_DELAY        = 20,           !
;  0108  0        M3_DELAY        = 30,           !
;  0109  0        M4_DELAY        = 40,           !
;  0110  0        M5_DELAY        = 50,           !
;  0111  0
;  0112  0        K               = 1024,         !
;  0113  0        TIME1_LIMIT     = 128,          ! DELAY - LOOP ITERATION COUNT
;  0114  0        TIME2_LIMIT     = 1 * K,        ! DELAY - LOOP ITERATION COUNT
;  0115  0        TIME3_LIMIT     = 1 * K,        ! DELAY - LOOP ITERATION COUNT
;  0116  0        TIME4_LIMIT     = 512,          ! DELAY - LOOP ITERATION COUNT
;  0117  0        TIME5_LIMIT     = 16 * K,       ! DELAY - 16K LOOP ITERATION COUNT
;  0118  0        TIME6_LIMIT     = 1,            ! DELAY - LOOP ITERATION COUNT
;  0119  0        TIME7_LIMIT     = 10,           ! DELAY - LOOP ITERATION COUNT
;  0120  0        TIME8_LIMIT     = 50,           ! DELAY - LOOP ITERATION COUNT
;  0121  0        TIME9_LIMIT     = 100,          ! DELAY - LOOP ITERATION COUNT
;  0122  0
;  0123  0        STEP1           = 2,            !
;  0124  0
```

```
;   0125  0         RLO_ADR       - 2,          !
;   0126  0         RHI_ADR       - 3,          !
;   0127  0         XLO_ADR       - 4,          !
;   0128  0         XHI_ADR         5,          !
;   0129  0         IOP_LO_ADR    - 2,          !
;   0130  0         IOP_HI_ADR      3,          !
;   0131  0         IOP_SIZE      - %O'16',     ! I/O PAGE REGISTER SIZE
;   0132  0         IOP_ADR       - 0,          ! OFFSET TO DEVICE ADDRESS
;   0133  0         IOP_VEC       - 2,          ! OFFSET TO DEVICE VECTOR ADDRESS
;   0134  0         IOP_BRL       = 4,          ! OFFSET TO DEVICE BR LEVEL
;   0135  0         INT_VEC       = 6,          !
;   0136  0
;   0137  0         CSR           = 7,          !
;   0138  0         WORD_LIMIT    = %O'177777', !
;   0139  0
```

```
;  0140  0     !++
;  0141  0     !
;  0142  0     !  DESCRIPTOR LIST DEFINITIONS
;  0143  0     !
;  0144  0     ! -
;  0145  0
;  0146  0         D_FLAG_WD     = 0,            ! STATUS WORD 0, FLAG WORD
;  0147  0         D_DESCR_BITS  = 1,            !
;  0148  0         D_HI_ADR      = 1,            !
;  0149  0         D_LO_ADR      = 2,            !
;  0150  0         D_WD_COUNT    = 3,            !
;  0151  0         D_WD1_STATUS  = 4,            !
;  0152  0         D_WD2_STATUS  = 5,            !
;  0153  0
;  0154  0         D1_OFFSET     = 18,           !
;  0155  0         D2_OFFSET     = 36,           !
;  0156  0
;  0157  0         T_SIZE        = 120,
;  0158  0         DESCR_SIZE    = 128,
;  0159  0         D_SIZE        = DESCR_SIZE / 2,
;  0160  0         BD_D_SIZE     = 16,
;  0161  0         BUF_SIZE      = 4096,
;  0162  0         B_SIZE        = BUF_SIZE / 2,
;  0163  0         SETUB_SIZE    = 256,
;  0164  0         BYTE_COUNT    = - ( BUF_SIZE / 4 ),
;  0165  0         PROM_SIZE     = 4096,
;  0166  0         CHSUM_OFFSET  = 6,
;  0167  0
;  0168  0         SA_RBL        = %O'177775',          ! STATION ADR RCV BUF LENGTH - 3 WDS
;  0169  0
;  0170  0         PKT_LENGTH      = 1500,              ! PACKET LENGTH
;  0171  0         MAX_LENGTH      = 1534,              ! PACKET LENGTH
;  0172  0         LEGAL_LENGTH    = 1514,              ! LEGAL PACKET LENGTH
;  0173  0         ILLEGAL_LENGTH  = 1536,              ! ILLEGAL PACKET LENGTH
;  0174  0         LPB_PKT         = %O'0220',          ! LOOPBACK PACKET
;  0175  0         PKT_TYPE        = 12,                ! PACKET TYPE
;  0176  0         SKIP_CNT        = 0,                       !
;  0177  0         RFC             = 1,                       !
;  0178  0         PKT_DATA        = 15,                      !
;  0179  0         SHORTEST_PACKET = 60,                ! SHORTEST SETUP PACKET LENGTH
;  0180  0         LONGEST_PACKET  = 1514,              ! LONGEST SETUP PACKET LENGTH
;  0181  0         LSPL            = 1514,              ! LONGEST SETUP PACKET LENGTH
;  0182  0         PHA_INDEX       = 19,                ! PHYSICAL ADDRESS INDEX IN THE
;  0183  0                                              !  TARGET_ADR VECTOR
;  0184  0
;  0185  0         KB_VEC_LOC      = %O'000060',        ! INPUT CONSOLE TERMINAL VECTOR LOC
;  0186  0         PF_VEC_LOC      = %O'000024',        ! POWER FAIL VECTOR LOCATION
;  0187  0         CPU_LED         = %O'177524',        ! TURN OFF CPU LED LIT ON DCOK
;  0188  0         KB_ADDR         = %O'177560',        ! CONSOLE TERMINAL INPUT ADDRESS
;  0189  0         KB_ENABLE       = %O'000100',        ! ENABLE CONSOLE TERMINAL INPUT
;  0190  0
```

```
;  0191  0     !++
;  0192  0     !
;  0193  0     !  TRANSMIT, RECEIVE AND CSR STATUS AND MASK WORD DEFINITIONS
;  0194  0     !
;  0195  0     !--
;  0196  0
;  0197  0         CSR_STATUS      = %O'100220',       !
;  0198  0         CSR1_STATUS     = %O'000062',       !
;  0199  0         CSR2_STATUS     = %O'000060',       !
;  0200  0         CSR_MASK        = %O'100220',       !
;  0201  0         CSR1_MASK       = %O'010376',       !
;  0202  0         CSR2_MASK       = %O'167777',       ! TRANSCEIVER POWER ( XC - BIT 12 )
;  0203  0         CSR3_MASK       = %O'010000',       ! TRANSCEIVER POWER ( XC   BIT 12 )
;  0204  0
;  0205  0         PATRN1      = %O'001411',           ! CSR STATIC BITS
;  0206  0         PATRN2      = %O'001471',           ! CSR STATIC BITS
;  0207  0
;  0208  0         NXM_LO_ADR    = %O'160000',         ! NXM ADDRESS - LOW ORDER BITS
;  0209  0         NXM_HI_ADR    = %O'000077',         ! NXM ADDRESS - HIGH ORDER BITS
;  0210  0
;  0211  0         XFLG_MASK     = %O'140000',         ! TRANSMIT FLAG WORD MASK BITS
;  0212  0         X1_MASK       = %O'100000',         ! TRANSMIT STATUS WD 1 MASK BITS
;  0213  0         XWD1_MASK     = %O'157760',         ! TRANSMIT STATUS WD 1 MASK BITS
;  0214  0         nxwd1_mask    = %O'157400',         ! mask out retry count for busy net
;  0215  0         XWD2_MASK     = %O'037777',         ! TRANSMIT STATUS WD 2 MASK BITS
;  0216  0         XFLG_STATUS   = %O'140000',         ! EXPECTED TRANSMIT FLAG WORD
;  0217  0         XWD11_STATUS  = %O'000000',         !
;  0218  0         XWD12_STATUS  = %O'000400',         ! EXPECTED TRANSMIT STATUS WD 1
;  0219  0                                             !  BIT 8 IS SET IN INTERNAL LOOOPBACK MODES
;  0220  0                                             !  BIT 8 IS RESET IN EXTERNAL LOOOPBACK MODES
;  0221  0         XWD14_STATUS  = %O'047600',         ! EXPECTED TRANSMIT STATUS WD 1
;  0222  0
;  0223  0         RFLG_MASK     = %O'140000',         ! RECEIVE FLAG WORD MASK BITS
;  0224  0         R1_MASK       = %O'100000',         ! RECEIVE STATUS WD 1 MASK BITS
;  0225  0         R2_MASK       = %O'174003',         ! RECEIVE STATUS WD 1 MASK BITS ! N.M. CHANGED FROM 174017 TO 174013
;  0226  0         RWD1_MASK     = %O'140000',         ! RECEIVE STATUS WD 1 MASK BITS
;  0227  0         RWD2_MASK     = %O'177407',         ! RECEIVE STATUS WD 1 MASK BITS
;  0228  0         RWD1_STATUS   = %O'020000',         ! EXPECTED RECEIVE STATUS WD 1
;  0229  0         RWD11_STATUS  = %O'100000',         ! EXPECTED RECEIVE STATUS WD 1
;  0230  0         RWD12_STATUS  = %O'160000',         ! EXPECTED RECEIVE STATUS WD 1
;  0231  0         RWD13_STATUS  = %O'000000',         ! EXPECTED RECEIVE STATUS WD 1
;  0232  0         RWD14_STATUS  = %O'060000',         ! EXPECTED RECEIVE STATUS WD 1
;  0233  0         RWD15_STATUS  = %O'000001',         ! EXPECTED RECEIVE STATUS WD 1
;  0234  0         RWD16_STATUS  = %O'044000',         ! EXPECTED RECEIVE STATUS WD 1
;  0235  0
;  0236  0         RFLG_STATUS   = %O'140000',         ! EXPECTED RECEIVE FLAG WORD
;  0237  0
;  0238  0
;  0239  0         RHL_MASK      = %O'003400',         ! RCV HIGH ORDER LENGTH BITS
;  0240  0         RLL_MASK      = %O'000377',         ! RCV LOW ORDER LENGTH BITS
```

```
;  0241  0     !++
;  0242  0     !
;  0243  0     !  BUFFER DESCRIPTOR / CHAIN DESCRIPTOR BIT DEFINITIONS
;  0244  0     !
;  0245  0     !--
;  0246  0
;  0247  0         V               = %O'100000',           ! VALID ADDRESS IF 1
;  0248  0         C               = %O'040000',           ! CHAIN ADDRESS IF 1
;  0249  0         E               = %O'020000',           ! END OF MESSAGE IF 1
;  0250  0         S               = %O'010000',           ! SETUP MODE PACKET IF 1
;  0251  0
;  0252  0         NEWB            = %O'100000',           ! BUFFER NOT USED IF 1
;  0253  0         LASTD           = %O'100000',           ! LAST DESCRIPTOR IN CHAIN
;  0254  0         VE              = %O'120000',           !
;  0255  0         VL              = %O'100200',           !
;  0256  0         VH              = %O'100100',           !
;  0257  0         VC              = %O'140000',           !
;  0258  0         VHL             = %O'100300',           !
;  0259  0         VSE             = %O'130000',           !
;  0260  0         VSEL            = %O'130200',           !
;  0261  0         VENXM           = %O'120077',           !
;  0262  0
;  0263  0         XLRL_SET        = %B'11',               ! XMIT AND RCV LISTS INVALID
;  0264  0         ILEL_SET        = %B'11',               ! INTERNAL AND EXTERNAL LOOPBACK BITS
;  0265  0         ILEL_CLR        = %B'00',               ! INTERNAL AND EXTERNAL LOOPBACK BITS
;  0266  0
;  0267  0         INT_LOOPBACK    = %B'00',               ! INTERNAL LOOPBACK MODE
;  0268  0         INX_LOOPBACK    = %B'10',               ! INTERNAL/EXTENDED LOOPBACK MODE
;  0269  0         EXT_LOOPBACK    = %B'11',               ! EXTERNAL LOOPBACK MODE
;  0270  0
;  0271  0         N_MODE          = %O'000200',           ! ENABLE NORMAL MODE OF OPERATION
;  0272  0         P_MODE          = %O'000202',           ! ENABLE PROMISCUOUS MODE OF OPERATION
;  0273  0         A_MODE          = %O'000201',           ! ENABLE ALL MULTICAST MODE OF OPERATION
;  0274  0         LED1            = %O'000204',           ! TURN OFF LED 1
;  0275  0         LED2            = %O'000210',           ! TURN OFF LED 2
;  0276  0         LED3            = %O'000214',           ! TURN OFF LED 3
;  0277  0
```

```
;   0278  0     !++
;   0279  0     !  STATION ADDRESS CONSTANTS
;   0280  0     ! -
;   0281  0
;   0282  0        SADR1 = 0,                   ! HIGH STATION ADDRESS BITS
;   0283  0        SADR2 = 1,                   ! MIDDLE BITS
;   0284  0        SADR3 = 2,                   ! LOW STATION ADDRESS BITS
;   0285  0        CHSUM = 3,                   ! ACTUAL CHECKSUM INDEX
;   0286  0
;   0287  0     !++
;   0288  0     !   HARDWARE AND SOFTWARE P TABLE EQUATES
;   0289  0     !--
;   0290  0
;   0291  0         SWP_SIZE    = 6,            ! SOFTWARE P-TABLE SIZE ( WORDS )
;   0292  0         HWP_SIZE    - 2,            ! HARDWARE P-TABLE SIZE ( WORDS )
;   0293  0
;   0294  0
;   0295  0             SET_IT  = 1,
;   0296  0             CLR_IT  = 0;
;   0297  0
```

```
;  0298  0     !++
;  0299  0     !
;  0300  0     !   THE CONTROL AND STATUS REGISTER BIT DEFINITIONS
;  0301  0     !
;  0302  0     !--
;  0303  0
;  0304  0     FIELD
;  0305  0         IOP_FIELDS  =
;  0306  0             SET
;  0307  0             RE      = [ 0, 1, 0 ],  ! RECEIVER ENABLE          R/W ( ACTIVE HIGH )
;  0308  0             SR      = [ 1, 1, 0 ],  ! SOFTWARE RESET           R/W ( ACTIVE HIGH )
;  0309  0             NI      = [ 2, 1, 0 ],  ! NXM INTERRUPT            R   ( ACTIVE HIGH )
;  0310  0             BD      = [ 3, 1, 0 ],  ! BOOT/DIAGNOSTIC ROM      R/W ( ACTIVE HIGH )
;  0311  0             XL      = [ 4, 1, 0 ],  ! XMIT LIST INVALID        R   ( ACTIVE HIGH )
;  0312  0             RL      = [ 5, 1, 0 ],  ! RCV LIST INVALID         R   ( ACTIVE HIGH )
;  0313  0             IE      = [ 6, 1, 0 ],  ! INTERRUPT ENABLE         R/W ( ACTIVE HIGH )
;  0314  0             XI      = [ 7, 1, 0 ],  ! XMIT INTERRUPT REQUEST   R/W ( ACTIVE HIGH )
;  0315  0             IL      = [ 8, 1, 0 ],  ! INTERNAL LOOPBACK MODE   R/W ( ACTIVE LOW  )
;  0316  0             EL      = [ 9, 1, 0 ],  ! EXTERNAL LOOPBACK MODE   R/W ( ACTIVE HIGH )
;  0317  0             SE      = [10, 1, 0 ],  ! SANITY TIMER ENABLE      R/W ( ACTIVE HIGH )
;  0318  0             X1      = [11, 1, 0 ],  ! RESERVED, UNUSABLE
;  0319  0             XC      = [12, 1, 0 ],  ! TRANSCEIVER PWR          R   ( ACTIVE HIGH )
;  0320  0             CA      = [13, 1, 0 ],  ! CARRIER                  R   ( ACTIVE HIGH )
;  0321  0             X2      = [14, 1, 0 ],  ! RESERVED, UNUSABLE
;  0322  0             RI      = [15, 1, 0 ],  ! RCV INTERRUPT REQUEST    R/W ( ACTIVE HIGH )
;  0323  0
;  0324  0             LB      = [ 8, 2, 0 ],          ! LOOPBACK BITS
;  0325  0             XLRL    = [ 4, 2, 0 ],          ! XMIT AND RCV LISTS INVALID BITS
;  0326  0             ALL_BITS= [ 0,16, 0 ],          ! FETCH WHOLE WORD
;  0327  0
;  0328  0             LO_NIBBLE = [ 0,  0, 0 ],       !
;  0329  0             HI_NIBBLE = [ 0,  4, 0 ],       !
;  0330  0             LO_BYTE   = [ 0,  8, 0 ],       !
;  0331  0             HI_BYTE   = [ 0, 16, 0 ],       ! GET WORD, ALL BITS
;  0332  0             ST_ADDR   = [ 0,  8, 0 ],       ! STATION ADDRESS LOW BYTE
;  0333  0             ST_WORD   = [ 0, 16, 0 ],       ! GET WORD, ALL BITS
;  0334  0
;  0335  0             RCV_LO  = [ 2, 0, 16, 0 ],      ! RCV BUFFER DESCRIPTOR LIST LOW ADDRESS
;  0336  0             RCV_HI  = [ 3, 0, 8, 0 ],       ! RCV BUFFER DESCRIPTOR LIST HIGH ADDRESS
;  0337  0             XMIT_LO = [ 4, 0, 16, 0 ],      ! XMIT BUFFER DESCRIPTOR LIST LOW ADDRESS
;  0338  0             XMIT_HI = [ 5, 0, 8, 0 ],       ! XMIT BUFFER DESCRIPTOR LIST HIGH ADDRESS
;  0339  0             VEC_ADR = [ 2, 8, 0 ],          ! INTERRUPT VECTOR ADDRESS
;  0340  0             VEC_ALL = [ 6, 0, 16, 0 ],      ! INTERRUPT VECTOR ADDRESS
;  0341  0             CSR_ALL = [ 7, 0, 16, 0 ]       ! CONTROL AND STATUS REGISTER
;  0342  0             TES;
```

```
;  0343  0     !++
;  0344  0     !
;  0345  0     !  TRANSMIT AND RECEIVE DESCRIPTOR LIST FIELDS
;  0346  0     !
;  0347  0     !--
;  0348  0
;  0349  0     FIELD
;  0350  0         DL_FIELDS  =
;  0351  0             SET
;  0352  0             FLGWD   = [ 0, 0, 16, 0 ],      ! XMIT OF RCV FLAG WORD
;  0353  0
;  0354  0             DBITS   = [ 1, 0, 16, 0 ],      ! DESCRIPTOR BITS
;  0355  0             H_BIT   = [ 1,  6, 1, 0 ],      ! XMIT BUFFER BEGINS ON BYTE BOUNDARY
;  0356  0             L_BIT   = [ 1,  7, 1, 0 ],      ! XMIT BUFFER ENDS ON BYTE BOUNDARY
;  0357  0             S_BIT   = [ 1, 12, 1, 0 ],      ! SET-UP PACKET IF 1
;  0358  0             E_BIT   = [ 1, 13, 1, 0 ],      ! LAST DESCRIPTOR IN CHAIN ( END )
;  0359  0             C_BIT   = [ 1, 14, 1, 0 ],      ! DESCRIPTOR HAS CHAIN ADDRESS IF 1
;  0360  0             V_BIT   = [ 1, 15, 1, 0 ],      ! VALID ADDRESS IF 1
;  0361  0
;  0362  0             LOADR   = [ 2, 0, 16, 0 ],      ! LOW 16 BITS OF XMIT OR RCV BUFFER ADDRESS
;  0363  0
;  0364  0             TWDL    = [ 3, 0, 16, 0 ],      ! XMIT OR RCV PACKET WORD LENGTH
;  0365  0
;  0366  0             STWD1   = [ 4, 0, 16, 0 ],      ! XMIT OR RCV STATUS WORD 1
;  0367  0             OVF     = [ 4, 0, 1,  0 ],      ! FIFO BUFFER OVERFLOW
;  0368  0             ABORT   = [ 4,  9, 1, 0 ],      !
;  0369  0             STE16   = [ 4, 10, 1, 0 ],      ! SANITY TIMER ON AT POWER_UP
;  0370  0             NOCAR   = [ 4, 11, 1, 0 ],      ! NO CARRIER
;  0371  0             RUNT    = [ 4, 11, 1, 0 ],      ! RUNT PACKET IN FIFO
;  0372  0             ESETUP  = [ 4, 13, 1, 0 ],      ! CONTROL SET_UP OR LOOPBACK PACKET
;  0373  0             LONGP   = [ 4, 14, 1, 0 ],      ! LONG PACKET
;  0374  0             ERRSU   = [ 4, 14, 1, 0 ],      ! ERROR SUMMARY
;  0375  0             LSTD    = [ 4, 15, 1, 0 ],      ! LAST DESCRIPTOR LIST IN CHAIN
;  0376  0
;  0377  0             STWD2   = [ 5, 0, 16, 0 ],      ! XMIT OR RCV STATUS WORD 2
;  0378  0             TDR     = [ 5, 0, 14, 0 ],      !
;  0379  0             RBLL    = [ 5, 0,  8, 0 ],      ! RECEIVE BYTE LENGTH ( LOW 8 BITS )
;  0380  0
;  0381  0             DLINK   = [ 6, 0, 16, 0 ],      ! DESCRIPTOR LINK PRE-FILL STATUS WD
;  0382  0
;  0383  0             BSTAT   = [ 7, 0, 16, 0 ],      ! BUFFER STATE ! XMIT ODD/EVEN ! HIGH ORDER ADR
;  0384  0
;  0385  0             B_LEN   = [ 0, 8, 0 ],          !
;  0386  0             W_LEN   = [ 0,16, 0 ]           !
;  0387  0             TES;
```

M7


```
;    0388  0      !++
;    0389  0      !
;    0390  0      !  HARDWARE P TABLE FIELD DEFINITIONS
;    0391  0      !
;    0392  0      !--
;    0393  0
;    0394  0      FIELD
;    0395  0          HWP_FIELDS =
;    0396  0              SET
;    0397  0              ADDR    = [ 0, 0, 16, 0 ],      ! I/O PAGE BASE ADDRESS
;    0398  0              VEC     = [ 1, 0, 16, 0 ],      ! INTERRUPT VECTOR ADDRESS
;    0399  0              BRL     = [ 2, 0, 16, 0 ]       ! BR LEVEL
;    0400  0              TES;
;    0401  0
;    0402  0
;    0403  0      !++
;    0404  0      !
;    0405  0      !  SOFTWARE P-TABLE FIELD DEFINITIONS
;    0406  0      !
;    0407  0      !--
;    0408  0
;    0409  0      FIELD
;    0410  0          SWP_FIELDS =
;    0411  0              SET
;    0412  0              ERR_CNT = [0,0,16,0]            ! # OF ERRORS BEFORE DROPPING DEQNA
;    0413  0              TES;
;    0414  0
;    0415  0
```

```
;                                    COMMAND QUALIFIERS

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

; Run Time:        00:02.4
; Elapsed Time:    00:03.2
; Lines/CPU Min:   10331
; Lexemes/CPU-Min: 48697
; Memory Used:  46 pages
; Library Precompilation Complete
```

```
                                .TITLE CZQNAAO DEQNA FUNCTIONAL TEST
                                .IDENT /2.4/
000000                          .PSECT $XYZ$,RO,I,LCL,REL,CON
                                .REM _


                                        IDENTIFICATION
                                        -------- -----

                        PRODUCT CODE:   AC T614A MC

                        PRODUCT NAME:   CZQNAAO DEQNA FUNCTIONAL TEST

                        PRODUCT DATE:   10 OCT. 1983

                        MAINTAINER:     PSD DTAGNOSTIC ENGINEERING

                        AUTHOR:         S. MAZURCZY(


                                COPYRIGHT (C) 1984

                                DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754

                                THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                                COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                                ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                                MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                                EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                                TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                                REMAIN IN DEC.

                                THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                                AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                                CORPORATION.

                                DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OP  RELIABI'ITY OF ITS
                                SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
```

```
                        ABSTRACT:

                                Out-of-line routines to save and restore register contents.
                                Also, power fail and console input interrupt service routines.

                        ENVIRONMENT:

                                PDP-11 or Compatibility Mode of the VAX, EIS and NOEIS

                        DISCUSSION:

                                The routine $SAVEn is called at the beginning of a routine
                                which modifies the contents of registers 1 thru n.  The calling
                                sequence is the following:

                                        JSR  R1,$SAVEn
```

```
 62          000000                  R0=%0
 63          000001                  R1=%1
 64          000002                  R2=%2
 65          000003                  R3=%3
 66          000004                  R4=%4
 67          000005                  R5=%5
 68          000006                  SP=%6
 69          000007                  PC=%7
 70
 71 000000                           $SAVE2::
 72 000000  010246                           MOV     R2,-(SP)
 73 000002  010146                           MOV     R1,-(SP)
 74 000004  016601  000004                    MOV     4(SP),R1
 75 000010  004736                           JSR     PC,@(SP)+
 76 000012  000432                           BR      RE2
 77
 78 000014                           $SAVE3::
 79 000014  010246                           MOV     R2,-(SP)
 80 000016  010346                           MOV     R3,-(SP)
 81 000020  010146                           MOV     R1,-(SP)
 82 000022  016601  000006                    MOV     6(SP),R1
 83 000026  004736                           JSR     PC,@(SP)+
 84 000030  000422                           BR      RE3
 85
 86 000032                           $SAVE4::
 87 000032  010246                           MOV     R2,-(SP)
 88 000034  010346                           MOV     R3,-(SP)
 89 000036  010446                           MOV     R4,-(SP)
 90 000040  010146                           MOV     R1,-(SP)
 91 000042  016601  000010                    MOV     8.(SP),R1
 92 000046  004736                           JSR     PC,@(SP)+
 93 000050  000411                           BR      RE4
 94
 95 000052                           $SAVE5::
 96 000052  010246                           MOV     R2,-(SP)
 97 000054  010346                           MOV     R3,-(SP)
 98 000056  010446                           MOV     R4,-(SP)
 99 000060  010546                           MOV     R5,-(SP)
100 000062  010146                           MOV     R1,-(SP)
101 000064  016601  000012                    MOV     10.(SP),R1
102 000070  004736                           JSR     PC,@(SP)+
103 000072  012605                           MOV     (SP)+,R5
104 000074  012604                  RE4:     MOV     (SP)+,R4
105 000076  012603                  RE3:     MOV     (SP)+,R3
106 000100  012602                  RE2:     MOV     (SP)+,R2
107 000102  012601                           MOV     (SP)+,R1
108 000104  000207                           RTS     PC
```

D8


```
110
111 000106                          WAIT.F::
112 000106  052737  000100  177560          BIS     #100,@#177560           ; ENABLE CONSOLE INPUT INTERRUPT
113 000114  010667  000000G                 MOV     SP,TEMP6                ; SAVE COPY OF STACK POINTER ADDRESS
114 000120  000001                          WAIT                            ; WAIT FOR AN INTERRUPT
115 000122  000207                          RTS     PC
116
117 000124                          PWR.IN::
118 000124  000257                          CCC                             ; CLEAR PS CONDITION BITS ( 0 4 )
119 000126  012767  000001  000000G         MOV     #1,INTERR               ; SET IF SANITY TIMER TIMED OUT
120 000134  012737  000017  177524          MOV     #17,@#177524            ; TURN OFF CPU LED'S
121 000142  016706  000000G                 MOV     TEMP6,SP                ; RESORE STACK POINTER
122 000146  000240                          NOP
123 000150  000240                          NOP
124 000152  000240                          NOP
125 000154  000240                          NOP
126 000156  000240                          NOP
127 000160  000240                          NOP
128 000162  000240                          NOP
129 000164  000207                          RTS     PC
130 000166                          KBD.IN::
131 000166  012767  000000  000000G         MOV     #0,INTERR               ; SET IF INTERRUPTED FROM CONSOLE
132 000174  005067  000000G                 CLR     TEMP1                   ;
133 000200  005037  177560                  CLR     @#177560                ; DISABLE CONSOLE INTERRUPTS
134 000204  013767  177562  000000G         MOV     @#177562,TEMP1          ; SAVE CHARACTER
135 000212  000002                          RTI
136
137         000001                          .END
```

E8

```
CZQNFA0 DEQNA FUNCTIONAL TEST   MACRO V05.03  Thursday 27 Mar 86 07:38  Page 5 1                                    SEQ 302
Symbol table

INTERR= ****** GX       RE2     000100R     002 TEMP1 = ****** GX       $SAVE2  000000RG    002 $SAVE4  000032RG    002
KBD.IN  000166RG    002 RE3     000076R     002 TEMP6 = ****** GX       $SAVE3  000014RG    002 $SAVE5  000052RG    002
PWR.IN  000124RG    002 RE4     000074R     002 WAIT.F  000106RG    002

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
$XYZ$   000214    002   (RO,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 51 Words  ( 1 Pages)
Size of core pool: 19684 Words  ( 75 Pages)
Operating  system: RSX 11M/PLUS (Under VAX/VMS)

Elapsed time: 00:00:02.44
B16SAV.OBJ,B16SAV.LIS/ SP=SVC34/ML,B16SAV
```

F8

ZQNAE0.EXE;2 Memory allocation map TKB M42.00 Page 1 SEQ 307
27 MAR 86 07:38

Partition name : DUMMY
Identification : V01.0
Task UIC : [330,33]
Task attributes: HD
Total address windows: 1.
Task image size : 11360. words
Task address limits: 002000 056263
R-W disk blk limits: 000002 000056 000055 00045.

*** Root segment: ZQNA1

R/W mem limits: 002000 056263 054264 22708.
Disk blk limits: 000002 000056 000055 00045.

Memory allocation synopsis:

Section		Title	Ident	File
$CODE$:(RO,I,LCL,REL,CON)	002000 000416 00270.			
	002000 000244 00164.	ZQNA1	V01.0	ZQNA1.OBJ;2
	002244 000152 00106.	ZQNA2	V01.0	ZQNA2.OBJ;2
$GLOB$:(RW,D,LCL,REL,CON)	002416 012510 05448.			
	002416 012510 05448.	ZQNA1	V01.0	ZQNA1.OBJ;2
$PLIT$:(RO,D,LCL,REL,CON)	015126 007520 03920.			
	015126 007520 03920.	ZQNA1	V01.0	ZQNA1.OBJ;2
AA$COD:(RO,I,LCL,REL,CON)	024646 000466 00310.			
	024646 000466 00310.	ZQNA2	V01.0	ZQNA2.OBJ;2

```
AB$COD:(RO,I,LCL,REL,CON)     025334 021544 09060.
                              025334 021544 09060. ZQNA3  V01.0  ZQNA3.OBJ;2
AC$COD:(RO,I,LCL,REL,CON)     047100 006612 03466.
                              047100 006612 03466. ZQNA4  V01.0  ZQNA4.OBJ;2
. BLK.:(RW,I,LCL,REL,CON)     055712 000116 00078.
                              055712 000116 00078. .MAIN.        ZQNA6.OBJ;2
$XYZ$ :(RO,I,LCL,REL,CON)     056030 000234 00156.
                              056030 000214 00140. CZQNAA 2.4    B16SAV.OBJ;2
                              056244 000020 00016. ZQNA5  V01.0  ZQNA5.OBJ;2


Global symbols:

ADR     000020    BIT09  001000    BIT6    000100    CHK.XM 051140-R  EF.CON 000036    ERR.NU 015044-R  GP$7    002400 R
BD.PRO 014310-R   BIT1   000002    BIT7    000200    CLR.BU 050334-R  EF.NEW 000035    ETH.ST 014064-R  GP$8    002406-R
BIT0    000001    BIT10  002000    BIT8    000400    CLR.DE 050306-R  EF.PWR 000034    EVL     000004   HOE     100000
BIT00   000001    BIT11  004000    BIT9    001000    COMPAR 052236-R  EF.RES 000037    E1$REP 047404-R  HP.TAB 002210 R
BIT01   000002    BIT12  010000    BL$LAS 056244-R   COUNTE 015026-R  EF.STA 000040    FORM.H 054016-R  HWP.TA 015004 R
BIT02   000004    BIT13  020000    BOE     000400    CSR.WO 015040-R  ERRBLK 002204-R  GET.AD 015014-R  IBE     010000
BIT03   000010    BIT14  040000    BUF.LE 015036-R   DATA.B 003016-R  ERRMSG 002202-R  GP$1    002314-R  IDU     000040
BIT04   000020    BIT15  100000    CHECKS 015034-R   DEQNA. 015024-R  ERRNBR 002200-R  GP$2    002324-R  IER     020000
BIT05   000040    BIT2   000004    CHK.CS 050746-R   DESCR. 002416-R  ERROR$ 047100-R  GP$3    002340-R  INTERR 015022-R
```

H8

BIT06	000100	BIT3	000010	CHK.RC	051436 R	DFSTBL	002210-R	ERRTYP	002176-R	GP$4	002350-R	INTR.T	055250-R		
BIT07	000200	BIT4	000020	CHK.RI	050362 R	DOWN.C	015032-R	ERR.CO	015050-R	GP$5	002364-R	IOP.DA	015012-R		
BIT08	000400	BIT5	000040	CHK.RX	051756-R	D$PCNT	002122-R	ERR.FL	015046-R	GP$6	002372-R	IOP.TA	014044-R		
ISR	000100	L$HW	002210-R	MSG06	016220-R	MSG43	021572-R	PREP.F	053734-R	SET.XD	052732-R	T12	036600 R		
IXE	004000	L$HWLE	002206-R	MSG07	016312-R	MSG44	021656-R	PRI	002000	SP.TAB	002220-R	T13	040036-R		
KBD.IN	056216-R	L$ICP	002104-R	MSG08	016404-R	MSG45	021760-R	PRI00	000000	STATIO	014100-R	T14	041320-R		
LOE	040000	L$INIT	025210 R	MSG09	016476-R	MSG46	022036-R	PRI01	000040	SWP.BL	002230-R	T15	041550-R		
LOGUN	015124-R	L$LADP	002026 R	MSG10	016570-R	MSG47	022106-R	PRI02	000100	SWP.IL	002226-R	T16	043612-R		
LOT	000010	L$LAST	056250-R	MSG11	016652-R	MSG48	022172-R	PRI03	000140	SWP.LB	002222 R	T17	044454-R		
L$ACP	002110-R	L$LOAD	002100-R	MSG12	016736-R	MSG49	022230-R	PRI04	000200	SWP.NX	002232-R	T18	044772 R		
L$APT	002036-R	L$LUN	002074-R	MSG13	017002-R	MSG50	022266-R	PRI05	000240	SWP.TA	015006-R	T19	045342 R		
L$AU	025266 R	L$MREV	002050-R	MSG14	017066-R	MSG51	022350-R	PRI06	000300	SWP.TI	002220-R	T2	026624-R		
L$AUT	002070-R	L$NAME	002000-R	MSG15	017156-R	MSG52	022402-R	PRI07	000340	SWP.TO	002224-R	T20	046042-R		
L$AUTO	025222-R	L$NDHR	002334-R	MSG16	017240-R	MSG53	022446-R	PTRN.T	014110-R	TADR1	015120-R	T21	047064-R		
L$CCP	002106-R	L$NDHW	002214-R	MSG17	017326-R	MSG54	022476-R	PWR.IN	056154-R	TADR2	015122-R	T3	027320-R		
L$CLEA	025242-R	L$NDSF	002414-R	MSG18	017414-R	MSG55	022546-R	P1	015102-R	TARGET	014120 R	T4	030366-R		
L$CO	002032-R	L$NDSW	002234-R	MSG19	017440-R	MSG56	022610-R	P2	015104-R	TBYTE1	015114-R	T5	031246-R		

```
L$DEP0 002011-R  L$PRI0 002042-R  MSG20  017526 R  MSG57  022646 R  P3     015106-R  TBYTE2 015115-R  T6     031710-R
L$DESC 002262-R  L$PROT 002236 R  MSG21  017616 R  MSG58  022736-R  P4     015110-R  TBYTE3 015116-R  T7     033322-R
L$DESP 002076 R  L$PRT  002112-R  MSG22  017676-R  MSG59  023002-R  P5     015112-R  TBYTE4 015117-R  T8     033560-R
L$DEVP 002060-R  L$REPP 002062-R  MSG23  017762-R  MSG60  023114-R  QNA.IN 025306-R  TD13   014500-R  T9     034104-R
L$DISP 002124-R  L$REV  002010-R  MSG24  020040-R  MSG61  023156-R  QST01  015126-R  TD16   014350-R  UAM    000200
L$DLY  002116-R  L$RPT  024656-R  MSG25  020114-R  MSG62  023220-R  QST02  015156-R  TEMP1  015060-R  UP.COU 015030-R
L$DTP  002040-R  L$SFTL 002336-R  MSG26  020156-R  MSG63  023310-R  QST03  015206-R  TEMP2  015062-R  VER.DE 050106-R
L$DTYP 002034-R  L$SOFT 002340-R  MSG27  020220-R  MSG64  023404-R  QST04  015250-R  TEMP3  015064-R  WAIT.F 056136-R
L$DU   025254-R  L$SPC  002056-R  MSG28  020262-R  MSG65  023440-R  QST05  015312-R  TEMP4  015066 R  WALKIN 053010 R
L$DUT  002072-R  L$SPCP 002020-R  MSG29  020326-R  MSG66  023504-R  QST06  015354 R  TEMP5  015C'0 R  WRT.ST 053542 R
L$DVTY 002244-R  L$SPTP 002024-R  MSG30  020354-R  MSG67  023572-R  QST07  015416-R  TEMP6  015072-R  XBUF.L 015016 R
L$EF   002052-R  L$STA  002030 R  MSG31  020442-R  MSG68  023674-R  QST10  015460-R  TEMP7  015074-R  XC.FLA 015042 R
L$ENVI 002044-R  L$SW   002220-R  MSG32  020526-R  MSG69  023772-R  RBUF.L 015020-R  TEMP8  015076 R  XMIT.A 055412-R
L$ERRT 002176-R  L$SWLE 002216-R  MSG33  020570-R  MSG70  024072-R  RCV.BU 003016-R  TEMP9  015100-R  XMIT.B 007016-R
L$ETP  002102-R  L$TEST 002114-R  MSG34  020644-R  MSG71  024156-R  RCV.D. 002416-R  TMPR1  015052-R  XMIT.D 002616-R
L$EXP1 002046-R  L$TIML 002014-R  MSG35  020720-R  MSG72  024236-R  RD13   014604-R  TMP.IO 015054-R  XMIT.I 055440-R
L$EXP4 002064-R  L$UNIT 002012-R  MSG36  021016-R  MSG73  024324-R  REG.AD 015010-R  TMP.RE 015056-R  XMIT.S 054466-R
L$EXP5 002066-R  MSG00  015522-R  MSG37  021122-R  MSG74  024414-R  RESET. 047424-R  TURN.0 055636-R  $END.L 056262-R
```

J8


```
L$HARD 002314-R  MSG01  015560-R  MSG38  021214 R  MSG75  024466 R  ROMCHK 055712-R  T$FREE 056260-R  $SAVE2 056030 R

L$HIME 002120-R  MSG02  015642-R  MSG39  021274-R  MSG76  024556-R  SEND.E 054734-R  T$PTHV 000001    $SAVE3 056044 R

L$HPCP 002016-R  MSG03  015730-R  MSG40  021360-R  NXM.IN 025276-R  SEND.T 055102-R  T1     025774-R  $SAVE4 056062 R

L$HPTP 002022-R  MSG04  016034-R  MSG41  021450-R  PHYS.A 013016-R  SETUP. 013044-R  T10    034342 R  $SAVE5 056102-R

L$HRDL 002312-R  MSG05  016126-R  MSG42  021512 R  PNT    001000    SET.RD 052654-R  T11    034676-R

*** Task builder statistics:

    Total work file references: 89946.

    Work  file  reads: 0.
    Work  file writes: 0.

    Size of core pool: 23176. words (90. pages)
    Size of work file: 3584. words (14. pages)


    Elapsed time:00:00:14
```

SYMBOL	VALUE	REFERENCES...		
ADR	000020	# ZQNA1	# ZQNA2	
BD.PRO	014310-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$LAS  056244-R  # ZQNA5
BOE     000400    # ZQNA1   # ZQNA2
BUF.LE  015036-R  # ZQNA1
CHECKS  015034-R  # ZQNA1     ZQNA3     ZQNA4     .MAIN.
CHK.CS  050746-R    ZQNA3   # ZQNA4
CHK.RC  051436-R    ZQNA3   # ZQNA4
CHK.RI  050362-R    ZQNA3   # ZQNA4
CHK.RX  051756-R    ZQNA3   # ZQNA4
CHK.XM  051140-R    ZQNA3   # ZQNA4
CLR.BU  050334-R    ZQNA3   # ZQNA4
CLR.DE  050306-R    ZQNA3   # ZQNA4
COMPAR  052236-R    ZQNA3   # ZQNA4
COUNTE  015026-R  # ZQNA1     ZQNA3     ZQNA4     .MAIN.
CSR.WO  015040-R  # ZQNA1     ZQNA3     ZQNA4
DATA.B  003016-R  # ZQNA1     ZQNA3     ZQNA4
DEQNA.  015024-R  # ZQNA1     ZQNA3     ZQNA4
DESCR.  002416-R  # ZQNA1     ZQNA3     ZQNA4
DFSTBL  002210-R  # ZQNA1
DOWN.C  015032-R  # ZQNA1     ZQNA3     ZQNA4
D$PCNT  002122-R  # ZQNA1
EF.CON  000036    # ZQNA1   # ZQNA2
EF.NEW  000035    # ZQNA1   # ZQNA2
EF.PWR  000034    # ZQNA1   # ZQNA2
EF.RES  000037    # ZQNA1   # ZQNA2
```

ZQNAE0 CREATED BY TKB ON 27 MAR 86 AT 07:38 PAGE 3 SEQ 310

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...			
EF.STA	000040	# ZQNA1	# ZQNA2		
ERRBLK	002204-R	# ZQNA1			
ERRMSG	002202-R	# ZQNA1			
ERRNBR	002200-R	# ZQNA1			
ERROR$	047100-R	ZQNA3	# ZQNA4		
ERRTYP	002176-R	# ZQNA1			
ERR.CO	015050-R	# ZQNA1	ZQNA3	ZQNA4	
ERR.FL	015046-R	# ZQNA1	ZQNA3	ZQNA4	
ERR.NU	015044-R	# ZQNA1	ZQNA3	ZQNA4	
ETH.ST	014064-R	# ZQNA1			
EVL	000004	# ZQNA1	# ZQNA2		
E1$REP	047404-R	ZQNA3	# ZQNA4		
FORM.H	054016-R	ZQNA3	# ZQNA4		
GET.AD	015014-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
GP$1	002314-R	# ZQNA2			
GP$2	002324-R	# ZQNA2			
GP$3	002340-R	# ZQNA2			
GP$4	002350-R	# ZQNA2			
GP$5	002364-R	# ZQNA2			
GP$6	002372-R	# ZQNA2			
GP$7	002400-R	# ZQNA2			
GP$8	002406-R	# ZQNA2			
HOE	100000	# ZQNA1	# ZQNA2		
HP.TAB	002210-R	# ZQNA1			
HWP.TA	015004-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4

```
IBE      010000    # ZQNA1   # ZQNA2
IDU      000040    # ZQNA1   # ZQNA2
IER      020000    # ZQNA1   # ZQNA2
INTERR   015022-R    CZQNAA  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
INTR.T   055250-R    ZQNA3   # ZQNA4
IOP.DA   015012-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
IOP.TA   014044-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
ISR      000100    # ZQNA1   # ZQNA2
IXE      004000    # ZQNA1   # ZQNA2
KBD.IN   056216-R  # CZQNAA    ZQNA3
LOE      040000    # ZQNA1   # ZQNA2
LOGUN    015124-R  # ZQNA1     ZQNA2
LOT      000010    # ZQNA1   # ZQNA2
L$ACP    002110-R  # ZQNA1
L$APT    002036 R  # ZQNA1
L$AU     025266-R    ZQNA1   # ZQNA2
L$AUT    002070-R  # ZQNA1
L$AUTO   025222-R    ZQNA1   # ZQNA2
L$CCP    002106-R  # ZQNA1
L$CLEA   025242-R    ZQNA1   # ZQNA2
L$CO     002032-R  # ZQNA1
L$DEPO   002011-R  # ZQNA1
L$DESC   002262-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
```

B9

ZQNAE0 CREATED BY TKB ON 27 MAR 86 AT 07:38 PAGE 5 SEQ 312

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL VALUE REFERENCES...

L$DTP 002040 R # ZQNA1
L$DTYP 002034-R # ZQNA1
L$DU 025254-R ZQNA1 # ZQNA2
L$DUT 002072 R # ZQNA1
L$DVTY 002244-R ZQNA1 # ZQNA2
L$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$HARD 002314 R ZQNA1 # ZQNA2
L$HIME 002120-R # ZQNA1
L$HPCP 002016-R # ZQNA1
L$HPTP 002022-R # ZQNA1
L$HRDL 002312-R # ZQNA2
L$HW 002210-R # ZQNA1
L$HWLE 002206-R # ZQNA1
L$ICP 002104-R # ZQNA1
L$INIT 025210-R ZQNA1 # ZQNA2
L$LADP 002026-R # ZQNA1
L$LAST 056250-R ZQNA1 # ZQNA5
L$LOAD 002100-R # ZQNA1
L$LUN 002074 R # ZQNA1
L$MREV 002050 R # ZQNA1

```
L$NAME  002000 R  # ZQNA1
L$NDHR  002334 R  # ZQNA2
L$NDHW  002214 R  # ZQNA1
L$NDSF  002414 R  # ZQNA2
L$NDSW  002234 R  # ZQNA1
L$PRIO  002042 R  # ZQNA1
L$PROT  002236 R  # ZQNA1
L$PRT   002112 R  # ZQNA1
L$REPP  002062 R  # ZQNA1
L$REV   002010 R  # ZQNA1
L$RPT   024656 R    ZQNA1   # ZQNA2
L$SFTL  002336 R  # ZQNA2
L$SOFT  002340 R    ZQNA1   # ZQNA2
L$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   015522-R  # ZQNA1     ZQNA3     ZQNA4
MSG01   015560-R  # ZQNA1     ZQNA3     ZQNA4
MSG02   015642-R  # ZQNA1     ZQNA3     ZQNA4
MSG03   015730-R  # ZQNA1     ZQNA3     ZQNA4
```

D9

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...		
MSG04	016034-R	# ZQNA1	ZQNA3	ZQNA4
MSG05	016126-R	# ZQNA1	ZQNA3	ZQNA4
MSG06	016220 R	# ZQNA1	ZQNA3	ZQNA4
MSG07	016312 R	# ZQNA1	ZQNA3	ZQNA4
MSG08	016404 R	# ZQNA1	ZQNA3	ZQNA4
MSG09	016476 R	# ZQNA1	ZQNA3	ZQNA4
MSG10	016570 R	# ZQNA1	ZQNA3	ZQNA4
MSG11	016652 R	# ZQNA1	ZQNA3	ZQNA4
MSG12	016736 R	# ZQNA1	ZQNA3	ZQNA4
MSG13	017002 R	# ZQNA1	ZQNA3	ZQNA4
MSG14	017066 R	# ZQNA1	ZQNA3	ZQNA4
MSG15	017156 R	# ZQNA1	ZQNA3	ZQNA4
MSG16	017240-R	# ZQNA1	ZQNA3	ZQNA4
MSG17	017326-R	# ZQNA1	ZQNA3	ZQNA4
MSG18	017414 R	# ZQNA1	ZQNA3	ZQNA4
MSG19	017440-R	# ZQNA1	ZQNA3	ZQNA4
MSG20	017526-R	# ZQNA1	ZQNA3	ZQNA4
MSG21	017616-R	# ZQNA1	ZQNA3	ZQNA4
MSG22	017676-R	# ZQNA1	ZQNA3	ZQNA4
MSG23	017762-R	# ZQNA1	ZQNA3	ZQNA4
MSG24	020040-R	# ZQNA1	ZQNA3	ZQNA4
MSG25	020114-R	# ZQNA1	ZQNA3	ZQNA4
MSG26	020156 R	# ZQNA1	ZQNA3	ZQNA4
MSG27	020220-R	# ZQNA1	ZQNA3	ZQNA4
MSG28	020262-R	# ZQNA1	ZQNA3	ZQNA4
MSG29	020326 R	# ZQNA1	ZQNA3	ZQNA4

```
MSG30   020354-R  # ZQNA1    ZQNA3     ZQNA4
MSG31   020442 R  # ZQNA1    ZQNA3     ZQNA4
MSG32   020526-R  # ZQNA1    ZQNA3     ZQNA4
MSG33   020570-R  # ZQNA1    ZQNA3     ZQNA4
MSG34   020644-R  # ZQNA1    ZQNA3     ZQNA4
MSG35   020720 R  # ZQNA1    ZQNA3     ZQNA4
MSG36   021016 R  # ZQNA1    ZQNA3     ZQNA4
MSG37   021122-R  # ZQNA1    ZQNA3     ZQNA4
MSG38   021214-R  # ZQNA1    ZQNA3     ZQNA4
MSG39   021274-R  # ZQNA1    ZQNA3     ZQNA4
MSG40   021360 R  # ZQNA1    ZQNA3     ZQNA4
MSG41   021450 R  # ZQNA1    ZQNA3     ZQNA4
MSG42   021512 R  # ZQNA1    ZQNA3     ZQNA4
MSG43   021572-R  # ZQNA1    ZQNA3     ZQNA4
MSG44   021656 R  # ZQNA1    ZQNA3     ZQNA4
MSG45   021760-R  # ZQNA1    ZQNA3     ZQNA4
MSG46   022036 R  # ZQNA1    ZQNA3     ZQNA4
MSG47   022106-R  # ZQNA1    ZQNA3     ZQNA4
MSG48   022172-R  # ZQNA1    ZQNA3     ZQNA4
MSG49   022230 R  # ZQNA1    ZQNA3     ZQNA4
MSG50   022266-R  # ZQNA1    ZQNA3     ZQNA4
MSG51   022350-R  # ZQNA1    ZQNA3     ZQNA4
MSG52   022402-R  # ZQNA1    ZQNA3     ZQNA4
MSG53   022446-R  # ZQNA1    ZQNA3     ZQNA4
MSG54   022476-R  # ZQNA1    ZQNA2     ZQNA3     ZQNA4
MSG55   022546-R  # ZQNA1    ZQNA3     ZQNA4
```

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES..		
MSG56	022610 R	# ZQNA1	ZQNA3	ZQNA4
MSG57	022646 R	# ZQNA1	ZQNA3	ZQNA4
MSG58	022736 R	# ZQNA1	ZQNA3	ZQNA4
MSG59	023002-R	# ZQNA1	ZQNA3	ZQNA4
MSG60	023114 R	# ZQNA1	ZQNA3	ZQNA4
MSG61	023156-R	# ZQNA1	ZQNA3	ZQNA4
MSG62	023220-R	# ZQNA1	ZQNA3	ZQNA4
MSG63	023310-R	# ZQNA1	ZQNA3	ZQNA4
MSG64	023404-R	# ZQNA1	ZQNA3	ZQNA4
MSG65	023440-R	# ZQNA1	ZQNA3	ZQNA4
MSG66	023504 R	# ZQNA1	ZQNA3	ZQNA4
MSG67	023572-R	# ZQNA1	ZQNA3	ZQNA4
MSG68	023674-R	# ZQNA1	ZQNA3	ZQNA4
MSG69	023772-R	# ZQNA1	ZQNA3	ZQNA4
MSG70	024072-R	# ZQNA1	ZQNA3	ZQNA4
MSG71	024156-R	# ZQNA1	ZQNA3	
MSG72	024236-R	# ZQNA1	ZQNA3	
MSG73	024324-R	# ZQNA1	ZQNA3	
MSG74	024414-R	# ZQNA1	ZQNA3	
MSG75	024466-R	# ZQNA1	ZQNA3	
MSG76	024556-R	# ZQNA1	ZQNA3	
NXM.IN	025276-R	# ZQNA2	ZQNA3	
PHYS.A	013016-R	# ZQNA1	ZQNA3	ZQNA4
PNT	001000	# ZQNA1	# ZQNA2	
PREP.F	053734-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  014110-R  # ZQNA1     ZQNA3
PWR.IN  056154-R  # CZQNAA    ZQNA3
P1      015102-R  # ZQNA1     ZQNA3    ZQNA4
P2      015104-R  # ZQNA1     ZQNA3    ZQNA4
P3      015106-R  # ZQNA1     ZQNA3    ZQNA4
P4      015110-R  # ZQNA1     ZQNA3    ZQNA4
P5      015112-R  # ZQNA1
QNA.IN  025306-R  # ZQNA2     ZQNA3
QST01   015126-R  # ZQNA1     ZQNA2
QST02   015156-R  # ZQNA1     ZQNA2
QST03   015206-R  # ZQNA1     ZQNA2
QST04   015250-R  # ZQNA1     ZQNA2
QST05   015312-R  # ZQNA1     ZQNA2
QST06   015354-R  # ZQNA1     ZQNA2
QST07   015416-R  # ZQNA1     ZQNA2
QST10   015460-R  # ZQNA1     ZQNA2
RBUF.L  015020-R  # ZQNA1     ZQNA3    ZQNA4
RCV.BU  003016-R  # ZQNA1     ZQNA3    ZQNA4
```

ZQNAE0 CREATED BY TKB ON 27 MAR 86 AT 07:38 PAGE 11 SEQ 318

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...				
RCV.D.	002416 R	# ZQNA1	ZQNA3	ZQNA4		
RD13	014604 R	# ZQNA1	ZQNA3			
REG.AD	015010 R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
RESET.	047424-R	ZQNA2	ZQNA3	# ZQNA4		
ROMCHK	055712-R	ZQNA3	# .MAIN.			
SEND.E	054734-R	ZQNA3	# ZQNA4			
SEND.T	055102-R	ZQNA3	# ZQNA4			
SETUP.	013044 R	# ZQNA1	ZGNA4			
SET.RD	052654-R	ZQNA3	# ZQNA4			
SET.XD	052732-R	ZQNA3	# ZQNA4			
SP.TAB	002220-R	# ZQNA1				
STATIO	014100-R	# ZQNA1	ZQNA3	ZQNA4	.MAIN.	
SWP.BL	002230-R	# ZQNA1	ZQNA3	ZQNA4		
SWP.IL	002226-R	# ZQNA1	ZQNA3	ZQNA4		
SWP.LB	002222-R	# ZQNA1	ZQNA3			
SWP.NX	002232 R	# ZQNA1	ZQNA3			
SWP.TA	015006-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
SWP.TI	002220-R	# ZQNA1	ZQNA3	ZQNA4		
SWP.TO	002224 R	# ZQNA1	ZQNA3	ZQNA4		
TADR1	015120-R	# ZQNA1	ZQNA3	ZQNA4		
TADR2	015122 R	# ZQNA1	ZQNA3	ZQNA4		
TARGET	014120-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE1	015114-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE2	015115-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE3	015116-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE4	015117-R	# ZQNA1	ZQNA3	ZQNA4		

```
TD13    014500-R  # ZQNA1     ZQNA3
TD16    014350 R  # ZQNA1     ZQNA3
TEMP1   015060 R    CZQNAA  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TEMP2   015062 R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TEMP3   015064 R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TEMP4   015066-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TEMP5   015070-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TEMP6   015072-R    CZQNAA  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TEMP7   015074-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TEMP8   015076-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TEMP9   015100-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TMPR1   015052-R  # ZQNA1     .MAIN.
TMP.IO  015054-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TMP.RE  015056-R  # ZQNA1     ZQNA2     ZQNA3     ZQNA4
TURN.0  055636 R    ZQNA3   # ZQNA4
T$FREE  056260-R  # ZQNA5
T$PTHV  000001      ZQNA1   # ZQNA5
T1      025774-R    ZQNA1   # ZQNA3
T10     034342-R    ZQNA1   # ZQNA3
T11     034676-R    ZQNA1   # ZQNA3
T12     036600-R    ZQNA1   # ZQNA3
T13     040036-R    ZQNA1   # ZQNA3
T14     041320-R    ZQNA1   # ZQNA3
T15     041550-R    ZQNA1   # ZQNA3
T16     043612-R    ZQNA1   # ZQNA3
T17     044454-R    ZQNA1   # ZQNA3
```

J9

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...		
T18	044772-R	ZQNA1	# ZQNA3	
T19	045342-R	ZQNA1	# ZQNA3	
T2	026624-R	ZQNA1	# ZQNA3	
T20	046042 R	ZQNA1	# ZQNA3	
T21	047064-R	ZQNA1	# ZQNA3	
T3	027320-R	ZQNA1	# ZQNA3	
T4	030366-R	ZQNA1	# ZQNA3	
T5	031246-R	ZQNA1	# ZQNA3	
T6	031710 R	ZQNA1	# ZQNA3	
T7	033322-R	ZQNA1	# ZQNA3	
T8	033560-R	ZQNA1	# ZQNA3	
T9	03410^-R	ZQNA1	# ZQNA3	
UAM	000200	# ZQNA1	# ZQNA2	
UP.COU	015030-R	# ZQNA1	ZQNA3	ZQNA4
VER.DE	050106-R	ZQNA3	# ZQNA4	
WAIT.F	056136-R	# CZQNAA	ZQNA3	
WALKIN	053010-R	ZQNA3	# ZQNA4	
WRT.ST	053542-R	ZQNA3	# ZQNA4	
XBUF.L	015016 R	# ZQNA1	ZQNA3	ZQNA4
XC.FLA	015042-R	# ZQNA1	ZQNA3	
XMIT.A	055412 R	ZQNA3	# ZQNA4	
XMIT.B	007016-R	# ZQNA1	ZQNA3	ZQNA4
XMIT.D	002616-R	# ZQNA1	ZQNA3	ZQNA4
XMIT.I	055440-R	ZQNA3	# ZQNA4	
XMIT.S	054466-R	ZQNA3	# ZQNA4	
$END.L	056262-R	# ZQNA5		

K9


```
$SAVE2  056030-R  # CZQNAA    ZQNA3     ZQNA4
$SAVE3  056044-R  # CZQNAA    ZQNA3     ZQNA4
$SAVE4  056062-R  # CZQNAA    ZQNA2     ZQNA3
$SAVE5  056102-R  # CZQNAA
```