DEQNA

DEQNA FUNCTIONAL TEST
CZQNAAO

AH-T615A-MC
FICHE 1 OF 1

APR 1984
digital
Made In USA

```
;    0001  0     MODULE ZQNA1 (%TITLE 'CZQNAAO DEQNA FUNCTIONAL TEST'
;    0002  0                   IDENT = 'V01.0',
;    0003  0                   ADDRESSING_MODE(ABSOLUTE),
;    0004  0                   LANGUAGE(BLISS16)) =
;    0005  0     %SBTTL 'GLOBAL DEFINITION MODULE'
;    0006  0
;    0007  1     BEGIN
;    0008  1
; C  0009  1     %(
; C  0010  1                               IDENTIFICATION
; C  0011  1                               -----  --------
; C  0012  1
; C  0013  1             PRODUCT CODE:     AC-T614A-MC
; C  0014  1
; C  0015  1             PRODUCT NAME:     CZQNAAO DEQNA FUNCTIONAL TEST
; C  0016  1
; C  0017  1             PRODUCT DATE:     10 OCT. 1983
; C  0018  1
; C  0019  1             MAINTAINER:       PSD DIAGNOSTIC ENGINEERING
; C  0020  1
; C  0021  1             AUTHOR:           S. MAZURCZYK
; C  0022  1
; C  0023  1
; C  0024  1             COPYRIGHT (C) 1984
; C  0025  1
; C  0026  1             DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
; C  0027  1
; C  0028  1             THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
; C  0029  1             COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
; C  0030  1             ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
; C  0031  1             MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
; C  0032  1             EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
; C  0033  1             TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
; C  0034  1             REMAIN IN DEC.
; C  0035  1
; C  0036  1             THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
; C  0037  1             AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
; C  0038  1             CORPORATION.
; C  0039  1
; C  0040  1             DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
; C  0041  1             SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
; C  0042  1
; C  0043  1             THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:
; C  0044  1
; C  0045  1                     DIGITAL          PDP              UNIBUS           MASSBUS
; C  0046  1                     DEC              DECUS            DECTAPE
; C  0047  1
; C  0048  1
; C  0049  1
```

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

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

```
; C 0128  1          1.3  RELATED DOCUMENTS AND STANDARDS
; C 0129  1               --- ------------      ----- -
; C 0130  1
; C 0131  1          XXDP+ Supervisor/User's Manual   ( CHQUS ).
; C 0132  1
; C 0133  1
; C 0134  1          1.4  ASSUMPTIONS
; C 0135  1               -----------
; C 0136  1
; C 0137  1          It is assumed that the system has been tested without DEQNA and found
; C 0138  1          working before this diagnostic is run, or that DEQNA DEC/X11
; C 0139  1          Exerciser has dropped DEQNA option module when running system test.
; C 0140  1
; C 0141  1
; C 0142  1          2.0  OPERATING INSTRUCTIONS
; C 0143  1               ----------------------
; C 0144  1
; C 0145  1          This  section contains a brief description of the runtime services.
; C 0146  1          for detailed information, refer to the XXDP+ User's Manual (CHQUS).
; C 0147  1
; C 0148  1
; C 0149  1          2.1  COMMANDS
; C 0150  1               --------
; C 0151  1
; C 0152  1          There  are eleven legal commands for the diagnostic runtime services
; C 0153  1          (supervisor).  This  section lists  the commands  and gives  a  very
; C 0154  1          brief description of them.  The XXDP+ User's Manual has more details.
; C 0155  1
; C 0156  1                  COMMAND         EFFECT
; C 0157  1                  -------         --------------------------
; C 0158  1
; C 0159  1                  START           Start the diagnostic from an initial state
; C 0160  1                  R:START         Start the diagnostic without initializing
; C 0161  1                  CONTINUE        Continue at test that was interrupted (after ^C)
; C 0162  1                  PROCEED         Continue from an error halt
; C 0163  1                  EXIT            Return to XXDP+ monitor (XXDP+ operation only!)
; C 0164  1                  ADD             Activate a unit for testing (all units are
; C 0165  1                                  considered to be active at start time
; C 0166  1                  DROP            Deactivate a un't
; C 0167  1                  PRINT           Print statistical information (if implemented
; C 0168  1                                  by the diagnostic - section 4.0)
; C 0169  1                  DISPLAY         Type a list of all device information
; C 0170  1                  FLAGS           Type the state of all flags (see section 2.3)
; C 0171  1                  ZFLAGS          Clear all flags (see section 2.3)
; C 0172  1
; C 0173  1          A command can be recognized by the first three characters.
; C 0174  1          So you may,  for example, type  "STA" instead  of "START".
```

2.2 SWITCHES

There are several switches which are used to modify supervisor operation. These switches are appended to the legal commands. All of the legal switches are tabulated below with a brief description of each. In the descriptions below, a decimal number is designated by "DDDDD".

SWITCH	EFFECT
/TESTS:LIST	Execute only those tests specified in the list. List is a string of test numbers, for example - /TESTS:1:5:7-10. This list will cause tests 1,5,7,8,9,10 to be run. All other tests will not be run.
/PASS:DDDDD	Execute DDDDD passes (DDDDD = 1 to 64000)
/FLAGS:FLGS	Set specified flags. flags are described in section 2.3.
/EOP:DDDDD	Report end of pass message after every DDDDD passes only. (DDDDD = 1 to 64000)
/UNITS:LIST	TEST/ADD/DROP only those units specified in the list. List example - /UNITS:0:5:10-12 use units 0,5,10,11,12 (unit numbers = 0-63)

Example of switch usage:

```
START/TESTS:1-5/PASS:1000/EOP:100
```

The effect of this command will be:

1. Tests 1 through 5 will be executed.
2. All units will be tested 1000 times.
3. The end of pass messages will be printed after each 100 passes only.

A Switch can be recognized by the first three characters. You may, for example, type "/TES:1-5" instead of "/TESTS:1-5".

Below is a table that specifies which switches can be used by each command.

	TESTS	PASS	FLAGS	EOP	UNITS
START	X	X	X	X	X
RESTART	X	X	X	X	X
CONTINUE		X	X	X	
PROCEED			X		
DROP					X
ADD					X
PRINT					
DISPLAY					X

```
; C 0228  1          FLAGS
; C 0229  1          ZFLAGS
; C 0230  1          EXIT
```

```
;C 0231  1               2.3  FLAGS
;C 0232  1                    -----
;C 0233  1
;C 0234  1               Flags are used to set up certain operational parameters such as
;C 0235  1               looping  on error.  All flags are cleared at startup and remain
;C 0236  1               cleared  until explicitly  set  using the  flags switch.  Flags
;C 0237  1               are  also cleared  after  a start  command unless set using the
;C 0238  1               flag  switch.  The ZFLAGS  command  may also  be used  to clear
;C 0239  1               all flags.  with the exception of the START and ZFLAGS commands,
;C 0240  1               No  commands affect the state of the flags;  they remain set or
;C 0241  1               cleared as specified by the last flag switch.
;C 0242  1
;C 0243  1                       FLAG            EFFECT
;C 0244  1                       ----            ------------------------------
;C 0245  1
;C 0246  1                       HOE             Halt on error - control is returned to
;C 0247  1                                       runtime services command mode
;C 0248  1                       LOE             Loop on error
;C 0249  1                       IER*            Inhibit all error reports
;C 0250  1                       IBR*            Inhibit  all error reports except first
;C 0251  1                                       level  (first level contains error type,
;C 0252  1                                       number, PC, test and unit)
;C 0253  1                       IXR*            Inhibit extended error reports (those
;C 0254  1                                       called by PRINTX macro's)
;C 0255  1                       PRI             Direct messages to line printer
;C 0256  1                       PNT             Print test number as test executes
;C 0257  1                       BOE             "BELL" on error
;C 0258  1                       UAM             Unattended mode (no manual intervention)
;C 0259  1                       ISR             Inhibit statistical reports (does not apply
;C 0260  1                                       to diagnostics which do not support statis-
;C 0261  1                                       tical reporting)
;C 0262  1                       IDR             Inhibit program dropping of units
;C 0263  1                       ADR             Execute autodrop code
;C 0264  1                       LOT             Loop on test
;C 0265  1                       EVL             Execute evaluation (on diagnostics which
;C 0266  1                                       have evaluation support)
;C 0267  1
;C 0268  1                               *error messages are described in section 3.0
;C 0269  1
;C 0270  1               See the XXDP+ User's Manual for more details on flags.  You may
;C 0271  1               specify  more  than one flag with the flag switch.  For example,
;C 0272  1               to  cause  the program to  loop on error, inhibit error reports
;C 0273  1               and  type a "BELL"  on error,  you may use the following string:
;C 0274  1
;C 0275  1                       /FLAGS:LOE:IER:BOE
```

```
; C 0276  1               2.4  HARDWARE QUESTIONS
; C 0277  1                    --- --------------
; C 0278  1
; C 0279  1               When a diagnostic is started, the DRS prompts the user for hardware
; C 0280  1               information by displaying
; C 0281  1
; C 0282  1               "CHANGE HW (L) ?"
; C 0283  1
; C 0284  1               you must answer "Y" after a start command unless the hardware
; C 0285  1               information has been "preloaded" using the Setup Utility (see chapter
; C 0286  1               6 of the XXDP+ User's Manual). When you answer this question with a
; C 0287  1               "Y", the DRS asks for the number of units. You will then be asked the
; C 0288  1               following questions for each unit.
; C 0289  1
; C 0290  1               # OF DEVICES (D) ?
; C 0291  1
; C 0292  1               Answer with the number of units to be tested (no default). This answer
; C 0293  1               will determine how many times the following questions are asked.
; C 0294  1               One (1) device must be specified.
; C 0295  1
; C 0296  1               DEQNA I/O PAGE  ADR  (0) 174440 ?
; C 0297  1
; C 0298  1               Answer with the address of the I/O page register assigned for one
; C 0299  1               of the DEQNA devices. The I/O page addresses permited are: 174440
; C 0300  1               and 174460.
; C 0301  1
; C 0302  1               INTERRUPT VECTOR ADR (0) 700 ?
; C 0303  1
; C 0304  1               Answer with the interrupt vector address of the DEQNA module.
; C 0305  1               Interrupt vector address for device at I/O page address 174440 is 700
; C 0306  1               oct. and that for I/O page address of 174460 is 704 oct.
; C 0307  1
; C 0308  1
; C 0309  1
; C 0310  1               2.5  SOFTWARE QUESTIONS
; C 0311  1                    -- -- ------------
; C 0312  1
```

; C 0313 1
; C 0314 1
; C 0315 1
; C 0316 1
; C 0317 1
; C 0318 1
; C 0319 1
; C 0320 1
; C 0321 1
; C 0322 1
; C 0323 1
; C 0324 1
; C 0325 1
; C 0326 1
; C 0327 1
; C 0328 1
; C 0329 1
; C 0330 1
; C 0331 1
; C 0332 1
; C 0333 1
; C 0334 1
; C 0335 1
; C 0336 1
; C 0337 1
; C 0338 1

2.6 QUICK START UP PROCEDURE (XXDP+)

To start-up this program:

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

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

```
;  C 0339  1               3.0  ERROR INFORMATION
;  C 0340  1                    --- ------- -----
;  C 0341  1
;  C 0342  1
;  C 0343  1               TYPES OF ERROR MESSAGES
;  C 0344  1               ----------------------
;  C 0345  1
;  C 0346  1               There  are three  levels  of error  messages that may be issued by
;  C 0347  1               a diagnostic: general, basic and extended.  General error messages
;  C 0348  1               are always printed unless the IBE and/or IER flag is set. The general
;  C 0349  1               error message is of the form:
;  C 0350  1
;  C 0351  1                       NAME  ER_TYPE  ER_NO  UNIT_NO  TEST_NO  PC_ADDR
;  C 0352  1
;  C 0353  1               ,where;
;  C 0354  1                       NAME = Diagnostic name
;  C 0355  1                       ER_TYPE = Error type ( all errors are HARD )
;  C 0356  1                       ER_NO = Error number
;  C 0357  1                       UNIT_NO = 0
;  C 0358  1                       TEST_NO = Test and subtest where error occurred
;  C 0359  1                       PC_ADDR = Program Counter contents
;  C 0360  1
;  C 0361  1               Basic error messages are messages that contain some additional
;  C 0362  1               information about the error. These are always printed unless one or
;  C 0363  1               more of the DRS error flag(s) ( IBE, IXE, IER ) is set. These
;  C 0364  1               messages are printed after the associated general message.
;  C 0365  1
;  C 0366  1               Extended error messages contain supplementary error information such
;  C 0367  1               as register contents or good/bad data. These are always printed unless
;  C 0368  1               the IXE and/or IER flag is set. These messages are printed after the
;  C 0369  1               associated general error message and any associated basic error
;  C 0370  1               messages. A typical extended error message might have a following
;  C 0371  1               format:
;  C 0372  1
;  C 0373  1
;  C 0374  1                       TRANSMIT DESCRIPTOR LIST          RECEIVE DESCRIPTOR LIST
;  C 0375  1
;  C 0376  1                          Flag Word                         Flag Word
;  C 0377  1                          Low  Order Addr Bits              Low  Order Addr Bits
;  C 0378  1                          High Order Addr Bits              High Order Addr Bits
;  C 0379  1                          Packet Length (byte)              Packet Length (byte)
;  C 0380  1                          Status Word 1                     Status Word 1
;  C 0381  1                          Status Word 2                     Status Word 2
;  C 0382  1
```

```
; C 0383  1               SPECIFIC ERROR MESSAGES
; C 0384  1               -------- -----  -------
; C 0385  1
; C 0386  1               Error messages will be "defined as needed". The following are possible
; C 0387  1               error messages.
; C 0388  1
; C 0389  1
; C 0390  1               Device fatal error messages:
; C 0391  1
; C 0392  1               1) CSR REGISTER FAILED TO RESPOND
; C 0393  1               2) NO INTERRUPT FROM DEQNA
; C 0394  1
; C 0395  1
; C 0396  1
; C 0397  1               Return status messages.
; C 0398  1
; C 0399  1
; C 0400  1               1) TRANSMIT STATUS ERROR
; C 0401  1               2) RECEIVE STARUS ERROR
; C 0402  1               3) CSR STATUS ERROR
; C 0403  1
```

M1


```
;  C 0404  1               4.0  TEST SUMMARIES
;  C 0405  1               ----------- --
;  C 0406  1
;  C 0407  1
;  C 0408  1
;  C 0409  1               NON-EXISTANT I/O PAGE REGISTER TEST
;  C 0410  1               - --------------- ---------------
;  C 0411  1
;  C 0412  1               This test verifies that all the device registers residing in the
;  C 0413  1               I/O Page can be accessed without forcing a non-existant memory (NXM)
;  C 0414  1               interrupt. If the operator specifies loop on error, the program
;  C 0415  1               re-executes the code that detected the error until ^C is entered.
;  C 0416  1
;  C 0417  1               Hardware tested:          Q-Bus to DEQNA Slave Registers Interface
;  C 0418  1
;  C 0419  1
;  C 0420  1               Processing:
;  C 0421  1
;  C 0422  1                    BEGIN
;  C 0423  1                       get ready for NXM interrupt
;  C 0424  1                       REPEAT for every I/O page register
;  C 0425  1
;  C 0426  1                          read I/O page register
;  C 0427  1                          IF NXM occured
;  C 0428  1                          THEN
;  C 0429  1                             print error message if not inhibited
;  C 0430  1                          ENDIF
;  C 0431  1
;  C 0432  1                       ENDREPEAT
;  C 0433  1
;  C 0434  1                       write any data pattern into the first 2 I/O page
;  C 0435  1                         registers
;  C 0436  1                       read I/O page register
;  C 0437  1                       IF NXM occured
;  C 0438  1                       THEN
;  C 0439  1                          print error message if not inhibited
;  C 0440  1                       ENDIF
;  C 0441  1
;  C 0442  1                    END
;  C 0443  1
;  C 0444  1
```

```
;C 0445  1              CSR BIT TEST
;C 0446  1              ------------
;C 0447  1
;C 0448  1              This test verifies that the CSR register static bits can be set
;C 0449  1              and cleared as specified.  The host writes data patterns to this
;C 0450  1              register and reads them back verifying  no static
;C 0451  1              (stuck at 1 / stuck at 0) faults occur. If the operator specifies
;C 0452  1              loop on error, the program re-executes the code that detected the
;C 0453  1              error until tC is entered.
;C 0454  1
;C 0455  1
;C 0456  1              Hardware tested:                Q-Bus to DEQNA Slave Regs. Interface
;C 0457  1
;C 0458  1
;C 0459  1              Processing:
;C 0460  1
;C 0461  1                   BEGIN
;C 0462  1
;C 0463  1                      set Software Reset (SR) bit in CSR and check for
;C 0464  1                          expected CSR status
;C 0465  1                      IF error
;C 0466  1                      THEN
;C 0467  1                         print error message if not inhibited
;C 0468  1                      ENDIF
;C 0469  1
;C 0470  1                      clear SR bit in CSR and check for expected CSR status
;C 0471  1                      IF error
;C 0472  1                      THEN
;C 0473  1                         print error message if not inhibited
;C 0474  1                      ENDIF
;C 0475  1
;C 0476  1                      set static bits ( 0,3,8,9 ) and check for expected CSR status
;C 0477  1                      IF error
;C 0478  1                      THEN
;C 0479  1                         print error message if not inhibited
;C 0480  1                      ENDIF
;C 0481  1
;C 0482  1                      clear static bits and check for expected CSR status
;C 0483  1                      IF error
;C 0484  1                      THEN
;C 0485  1                         print error message if not inhibited
;C 0486  1                      ENDIF
;C 0487  1
;C 0488  1                   END
;C 0489  1
```

```
;C 0490  1          ETHERNET STATION ADDRESS VERIFY TEST
;C 0491  1          ----------------------------------
;C 0492  1
;C 0493  1          This test verifies that  the Ethernet Station Address PROM can be
;C 0494  1          read and loaded to host memory correctly. Ethernet Station Address is
;C 0495  1          verified and checksum is computed from PROM data read and this checksum
;C 0496  1          is compared to the checksum stored in the Ethernet Station Address
;C 0497  1          PROM. Ethernet Station Address is always printed out on the console in
;C 0498  1          the Ethernet standard format. If the address is not proper, the error
;C 0499  1          is recorded and an appropriate error message is printed out on the
;C 0500  1          console. If the operator specifies loop on error, the program
;C 0501  1          re-executes the code that detected the error until ^C is entered.
;C 0502  1
;C 0503  1
;C 0504  1          Hardware tested:        Station Address PROM
;C 0505  1                                  Q-Bus to DEQNA Slave Regs. Interface
;C 0506  1
;C 0507  1
;C 0508  1
;C 0509  1          Processing:
;C 0510  1
;C 0511  1               BEGIN
;C 0512  1
;C 0513  1                  reset device
;C 0514  1                  read DEQNA Station Address PROM and checksum
;C 0515  1                  save copy of Station Address PROM in host memory
;C 0516  1                  print Station Address on the console in standard format
;C 0517  1                  compute Station Address PROM checksum
;C 0518  1                  IF checksum read not equal checksum computed
;C 0519  1                  THEN
;C 0520  1                     print error message if not inhibited
;C 0521  1                  ENDIF
;C 0522  1                  IF Station Address
;C 0523  1                        [all 0's]
;C 0524  1                     OR [all 1's]:
;C 0525  1                     OR [not assigned to DEQNA space]:
;C 0526  1                     OR [multicast bit set]:
;C 0527  1                  THEN
;C 0528  1                     print error message if not inhibited
;C 0529  1                  ENDIF
;C 0530  1
;C 0531  1               END
```

```
;C 0532  1              INTERRUPT VECTOR ADDRESS TEST
;C 0533  1               ----------------------------
;C 0534  1
;C 0535  1              This test verifies that all bits of the vector address register
;C 0536  1              can be set and cleared as specified.  The host writes data patterns
;C 0537  1              to this register and reads them back verifying no static
;C 0538  1              (stuck at 1 / stuck at 0) faults occur. If the operator specifies
;C 0539  1              loop on error, the program re-executes the code that detected the
;C 0540  1              error until ^C is entered.
;C 0541  1
;C 0542  1              NOTE: Only bits 9:2 of the Interrupt Vector Address Register are
;C 0543  1                    valid, rest read as zero ( 0 ).
;C 0544  1
;C 0545  1              The following BINARY data patterns are used:
;C 0546  1
;C 0547  1                                    00000000        11111111
;C 0548  1                                    10101010        01010101
;C 0549  1                                    11001100        00110011
;C 0550  1                                    11110000        00001111
;C 0551  1                                    walking 1's, 1 propagating thru Vector Address Reg.
;C 0552  1                                    walking 0's, 0 propagating thru Vector Address Reg.
;C 0553  1
;C 0554  1              Hardware tested:              Device Vector Address Register
;C 0555  1                                            Slave Interface Registers
;C 0556  1
;C 0557  1              Processing:
;C 0558  1
;C 0559  1              BEGIN
;C 0560  1                 reset device
;C 0561  1                 REPEAT for each pattern
;C 0562  1
;C 0563  1                    write pattern to Vector Address Register ( bits 9:2 )
;C 0564  1                    read pattern from Vector Address Register ( bits 9:2 )
;C 0565  1                    compare write pattern to read pattern (less noise bits)
;C 0566  1                    IF not equal
;C 0567  1                    THEN
;C 0568  1                       print error message if not inhibited
;C 0569  1                    ENDIF
;C 0570  1
;C 0571  1                 ENDREPEAT
;C 0572  1              END
```

```
; C 0573  1               INTERRUPT SANITY TEST
; C 0574  1               -------------- ---
; C 0575  1
; C 0576  1               This test verifies that DEQNA interrupts the processor only at
; C 0577  1               the expected level ( 4 ) and not any other level. If the operator
; C 0578  1               specifies loop on error, the program re-executes the code that
; C 0579  1               detected the error until ^C is entered.
; C 0580  1
; C 0581  1
; C 0582  1               Hardware tested:        Q-Bus to QTDC interface
; C 0583  1                                       CSR register
; C 0584  1                                       Q-Bus timeout logic
; C 0585  1                                       QTDC interrupt logic
; C 0586  1
; C 0587  1
; C 0588  1               Processing:
; C 0589  1
; C 0590  1                    BEGIN
; C 0591  1                       reset device
; C 0592  1                       set-up for NXM interrupt
; C 0593  1                       REPEAT for each processor priority level
; C 0594  1
; C 0595  1                          enable device interrupt (set CSR bit 6)
; C 0596  1                          force NXM interrupt
; C 0597  1                          check for expected CSR status
; C 0598  1                          IF error
; C 0599  1                          THEN
; C 0600  1                             print error message if not inhibited
; C 0601  1                          ENDIF
; C 0602  1
; C 0603  1                       ENDREPEAT
; C 0604  1                    END
```

```
;C 0605  1               ETHERNET CARRIER SENSE TEST
;C 0606  1                ------------------------
;C 0607  1
;C 0608  1               This test verifies that the DEQNA can transmit loopback packets.
;C 0609  1               In order to run this test successfully the operator has to make
;C 0610  1               sure that DEQNA is connected to the transceiver. If the operator
;C 0611  1               specifies loop on error, the program re-executes the code that detected
;C 0612  1               the error until ^C is entered.
;C 0613  1
;C 0614  1
;C 0615  1               Hardware tested:        Carrier Sense circuitry
;C 0616  1                                       Encode/Decode ( ED ) chip
;C 0617  1
;C 0618  1
;C 0619  1               Processing:
;C 0620  1
;C 0621  1                    BEGIN
;C 0622  1
;C 0623  1                       reset device
;C 0624  1                       select internal/extended or external loopback mode
;C 0625  1                       enable interrupts ( set bit 6 )
;C 0626  1
;C 0627  1                       read CSR
;C 0628  1                       IF Ethernet Carrier Sense bit ( bit 13 ) = 1
;C 0629  1                       THEN
;C 0630  1                          print error message if not inhibited
;C 0631  1                       ENDIF
;C 0632  1
;C 0633  1                       transmit longest unchained loopback packet ( ETHERNET format )
;C 0634  1                       read CSR while transmitting loopback packet
;C 0635  1                       IF Ethernet Carrier Sense bit (bit 13) = 0
;C 0636  1                       THEN
;C 0637  1                          print error message if not inhibited
;C 0638  1                       ELSE
;C 0639  1                          wait until Carrer Sense bit goes to 0
;C 0640  1                       ENDIF
;C 0641  1
;C 0642  1                       read CSR
;C 0643  1                       IF Ethernet Carrier Sense bit (bit 13) = 1
;C 0644  1                       THEN
;C 0645  1                          print error message if not inhibited
;C 0646  1                       ENDIF
;C 0647  1
;C 0648  1                    END
;C 0649  1
```

```
;C 0650  1
;C 0651  1          STATION ADDRESS RAM TEST
;C 0652  1          ----- ------------ ---
;C 0653  1
;C 0654  1          This test verifies that Station Address RAM has no static faults.
;C 0655  1          The host writes and then reads data patterns to all of the
;C 0656  1          addressable RAM (8 * 6 * 14 = 572 bytes ). The data is checked to see
;C 0657  1          that the data pattern received is the same as the data pattern
;C 0658  1          transmited. This test continues until all the data patterns are
;C 0659  1          exhausted. If the operator specifies loop on error, the program
;C 0660  1          re-executes the code that detected the error until ^C is entered.
;C 0661  1
;C 0662  1
;C 0663  1          The following BINARY patterns are used:
;C 0664  1
;C 0665  1                              11111111        00000000
;C 0666  1                              10101010        01010101
;C 0667  1                              11001100        00110011
;C 0668  1                              11110000        00001111
;C 0669  1                              walking 1, shifting 1 across memory BYTE
;C 0670  1                              walking 0, shifting 0 across memory BYTE
;C 0671  1                              marching 1's, propagating 1's through the RAM
;C 0672  1                              marching 0's, propagating 0's through the RAM
;C 0673  1
;C 0674  1          Hardware tested:        Station Address RAM
;C 0675  1                                  Q-Bus to QTDC interface
;C 0676  1                                  CSR register - Receiver Enable (bit 0)
;C 0677  1                                  Portion of Receive and Transmit FIFO
;C 0678  1
;C 0679  1
;C 0680  1          Processing:
;C 0681  1
;C 0682  1               BEGIN
;C 0683  1                  reset device
;C 0684  1                  select Setup mode
;C 0685  1                  REPEAT for each pattern
;C 0686  1
;C 0687  1                     load transmit packet with data pattern
;C 0688  1                     transmit loopback packet (fill all of the RAM)
;C 0689  1                     receive packet
;C 0690  1                     check for expected loopback status
;C 0691  1                     IF error
;C 0692  1                     THEN
;C 0693  1                        print error message if not inhibited
;C 0694  1                     ENDIF
;C 0695  1                     call compare_packets
;C 0696  1
;C 0697  1                  ENDREPEAT
;C 0698  1               END
```

```
;  C 0699  1
;  C 0700  1              5.0  MAINTENANCE HISTORY
;  C 0701  1                   --- --- ---- --
;  C 0702  1
;  C 0703  1
;  C 0704  1                  Modified By:              Date:          Version:
;  C 0705  1                  ............              .....          ........
;  C 0706  1
;  C 0707  1
;  C 0708  1
;  C 0709  1
;  C 0710  1
;     0711  1      )%
;     0712  1
```

```
;    0713  1      %SBTTL 'PROGRAM HEADER'
;    0714  1      LIBRARY 'QNALIB';
;    0715  1      REQUIRE 'BLSMAC.REQ ;                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;    2205  1
;    2206  1
;    2207  1      !++
;    2208  1      !       DEFINE THE NUMBER OF TESTS IN THIS DIAGNOSTIC
;    2209  1      !--
;    2210  1
;    2211  1      PSECT
;    2212  1          CODE = AA$CODE$;
;    2213  1
;    2214  1      LITERAL
;    2215  1          DS$NBR_OF_TESTS = 7;
;    2216  1
;    2217  1      EQUALS;
;    2218  1
;    2219  1      POINTER (ALL);
;    2220  1
;    2221  1      !++
;    2222  1      !         THE PROGRAM HEADER IS THE INTERFACE BETWEEN
;    2223  1      !         THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
;    2224  1      !--
;    2225  1
;    2226  1      HEADER (%ASCII'CZQNA ',%ASCII'A',%ASCII'0', 120, 0, PRI00);
;    2227  1
;    2228  1
;    2229  1      !++
;    2230  1      !       NO POINTERS ARE OPTIONAL USING BLISS. MAKE SURE THE FOLLOWING
;    2231  1      !       SECTIONS OF CODE ARE IN PLACE (IN THE CORRECT SKELS),EVEN IF
;    2232  1      !       THE SECTIONS ARE BLANK.
;    2233  1      !
;    2234  1      !       ARGUMENT        FUNCTION
;    2235  1      !       --------        ------
;    2236  1      !       RPT             REPORT CODE
;    2237  1      !       SW              SOFTWARE TABLE
;    2238  1      !       SFT             SOFTWARE TABLE QUESTIONS
;    2239  1      !       AU              ADD CODE
;    2240  1      !       DU              DROP CODE
;    2241  1      !       TBL             ERROR TABLE
;    2242  1      !       SETUP           ASSEMBLED P-TABLES
;    2243  1      !
;    2244  1      !       CHANGE THE "HEADER" TO CONTAIN THE PROPER ARGUMENTS.
;    2245  1      !       ARGUMENTS ARE: NAME,REV,PATCH,LONGEST TEST TIME,TYPE
;    2246  1      !       WHERE "TYPE" = 0 FOR SEQUENTIAL DIAGNOSTIC AND =1
;    2247  1      !       FOR EXERCISER.  THERE IS ALSO AN OPTIONAL SIXTH ARGUMENT
;    2248  1      !       WHICH SPECIFIES THE PROCESSOR PRIORITY TO BE SET WHEN
;    2249  1      !       STARTING THE DIAGNOSTIC (DEFAULT IS 0).
;    2250  1      !--
;    2251  1
;    2252  1
```

```
;    2253  1     %SBTTL 'DISPATCH TABLE'
;    2254  1
;    2255  1     !++
;    2256  1     !       THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
;    2257  1     !       IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
;    2258  1     !--
;    2259  1
;    2260  1     DISPATCH (DS$NBR_OF_TESTS);
;    2261  1     ERRTBL;
;    2262  1
;    2263  1     !++
;    2264  1     !       CHANGE THE LITERAL DECLARATION OF DS$NBR_OF_TESTS TO BE
;    2265  1     !       THE NUMBER OF HARDWARE TESTS IN YOUR PROGRAM.
;    2266  1     !--
;    2267  1
;    2268  1
```

J2


```
;    2269  1     %SBTTL 'DEFAULT HARDWARE P-TABLE'
;    2270  1
;    2271  1     !++
;    2272  1     !       THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF THE
;    2273  1     !       TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE IS IDENTICAL TO
;    2274  1     !       THE STRUCTURE OF THE HARDWARE P-TABLES, AND IS USED AS A "TEMPLATE"
;    2275  1     !       FOR BUILDING THE P-TABLES.
;    2276  1     !
;    2277  1     !
;    2278  1     !       PLACE YOUR DEFAULT HARDWARE P-TABLE HERE.  THE VALUES AND
;    2279  1     !       SIZE WILL BE USED AS A "TEMPLATE" FOR CREATING ACTUAL P-TABLE
;    2280  1     !       ENTRIES AND THE DEFAULT VALUES IN THE OPERATOR DIALOGUE.
;    2281  1     !       THE ACTUAL P-TABLE BUILT AT RUNTIME IS STORED IN SUPERVISOR
;    2282  1     !       SPACE.
;    2283  1     !--
;    2284  1
;    2285  1     BGNHW (HWPTBL);
;    2286  1         GLOBAL
;    2287  1             DFSTBL   : BLOCK [HWP_SIZE,WORD] INITIAL (%O'174440',%O'700');
;    2288  1     ENDHW;
;    2289  1
;    2290  1
```

```
;    2291  1     %SBTTL 'SOFTWARE P-TABLE'
;    2292  1
;    2293  1     !++
;    2294  1     !       THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
;    2295  1     !       PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
;    2296  1     !       SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
;    2297  1     !       AT RUN TIME.
;    2298  1     !
;    2299  1     !
;    2300  1     !       PLACE YOUR SOFTWARE P-TABLE HERE,USING GLOBAL OR OWN DECLARATIONS
;    2301  1     !       THIS TABLE IS NOT OPTIONAL.  THIS TABLE, UNLIKE THE HARDWARE TABLE,
;    2302  1     !       WILL CONTAIN THE ACTUAL VALUES ENTERED BY THE OPERATOR.
;    2303  1     !--
;    2304  1
;    2305  1     BGNSW (SWPTBL);
;    2306  1     GLOBAL
;    2307  1             DFHTBL  : BLOCK [ SWP_SIZE, WORD ] INITIAL (%DECIMAL'1');
;    2308  1     ENDSW;
;    2309  1
;    2310  1
```

```
;   2311  1     %SBTTL 'PROTECTION TABLE'
;   2312  1
;   2313  1     !++
;   2314  1     !       THIS TABLE IS USED BY THE RUNTIME SERVICES TO PROTECT THE LOAD MEDIA.
;   2315  1     !--
;   2316  1
;   2317  1     BGNPROT (-1, -1, -1);
;   2318  1
;   2319  1     !++
;   2320  1     !       1ST ARG =       OFFSET INTO P-TABLE FOR CSR ADDRESS
;   2321  1     !       2ND ARG =       OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;   2322  1     !       3RD ARG =       OFFSET INTO P-TABLE FOR DRIVE NUMBER
;   2323  1     !--
;   2324  1
;   2325  1     ENDPROT;
;   2326  1
;   2327  1     !++
;   2328  1     !       INSERT BYTE OFFSET FOR DATA NOTED IN COMMENTS ABOVE.  (OFFSET
;   2329  1     !       REFERS TO THE NUMBER OF BYTES FROM THE BEGINNING OF A PTABLE
;   2330  1     !       ENTRY TO THE ITEM IN QUESTION.)  IF THE PARTICULAR
;   2331  1     !       ITEM DOES NOT APPLY, LEAVE ENTRY AS -1.  WHEN THE RUNTIME
;   2332  1     !       SERVICES EXECUTES A GPHARD, IT USES THESE OFFSETS (IF NOT
;   2333  1     !       SET TO -1) TO GET THE ITEMS AND COMPARE WITH THOSE SAVED
;   2334  1     !       IN THE XXDP+ MONITOR.  IF THE UNIT BEING REQUESTED MATCHES THE
;   2335  1     !       LOAD DEVICE, THE RUNTIME SERVICES RETURN AN INCOMPLETE FLAG ON
;   2336  1     !       THE GPHARD.
;   2337  1     !--
;   2338  1
;   2339  1
```

```
;    2340  1     %SBTTL 'GLOBAL DATA SECTION'
;    2341  1
;    2342  1     PSECT
;    2343  1         PLIT   = $PLIT$,
;    2344  1         OWN    = $OWN$,
;    2345  1         GLOBAL = $GLOB$;
;    2346  1
;    2347  1     !++
;    2348  1     !       THE GLOBAL DATA DEFINED IN THIS SECTION IS USED BY MORE THAN ONE
;    2349  1     !       TEST.
;    2350  1     !--
;    2351  1
;    2352  1     GLOBAL
;    2353  1
;    2354  1     !++
;    2355  1     !       COMMUNICATION AREA DECLARATIONS
;    2356  1     !--
;    2357  1
;    2358  1         RCV_D_LIST      : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    2359  1         XMIT_D_LIST     : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    2360  1         RCV_BUFFER      : VECTOR [ B_SIZE, BYTE ],
;    2361  1         XMIT_BUFFER     : VECTOR [ B_SIZE, BYTE ],
;    2362  1         PHYS_ADR        : VECTOR [ 22, BYTE ],
;    2363  1         SETUP_BUFFER    : VECTOR [ SETUB_SIZE, WORD ],
;    2364  1         IOP_TABLE       : VECTOR [ 8, WORD ],
;    2365  1         ETH_STATION_ADR : VECTOR [ 6, WORD ],
;    2366  1         STATION_ADR     : VECTOR [ 4, WORD ],
;    2367  1         PTRN_TABLE      : VECTOR [ 8, BYTE ] INITIAL ( BYTE (
;    2368  1
;    2369  1                           %B'00000000', %B'11111111', %B'10101010', %B'01010101',
;    2370  1                           %B'11001100', %B'00110011', %B'11110000', %B'00001111' ),
;    2371  1
;    2372  1         TARGET_ADR      : VECTOR [ T_SIZE, BYTE ] INITIAL ( BYTE (
;    2373  1
;    2374  1                           %X'00', %X'00', %X'00', %X'00', %X'00', %X'00',      !  1
;    2375  1                           %X'AA', %X'AA', %X'AA', %X'AA', %X'AA', %X'AA',      !  2
;    2376  1                           %X'55', %X'55', %X'55', %X'55', %X'55', %X'55',      !  3
;    2377  1                           %X'FF', %X'FF', %X'FF', %X'FF', %X'FF', %X'FF',      !  4
;    2378  1                           %X'AA', %X'00', %X'00', %X'00', %X'00', %X'00',      !  5
;    2379  1                           %X'AA', %X'00', %X'02', %X'AA', %X'AA', %X'AA',      !  6
;    2380  1                           %X'AA', %X'00', %X'05', %X'55', %X'55', %X'55',      !  7
;    2381  1                           %X'AA', %X'00', %X'04', %X'FF', %X'FF', %X'FF',      !  8
;    2382  1                           %X'AA', %X'00', %X'04', %X'00', %X'00', %X'00',      !  9
;    2383  1                           %X'AA', %X'00', %X'04', %X'18', %X'81', %X'18',      ! 10
;    2384  1                           %X'AA', %X'08', %X'0C', %X'00', %X'0E', %X'0F',      ! 11
;    2385  1                           %X'FF', %X'00', %X'01', %X'02', %X'03', %X'04',      ! 12
;    2386  1                           %X'55', %X'05', %X'06', %X'07', %X'08', %X'09',      ! 13
;    2387  1                           %X'00', %X'F4', %X'FA', %X'44', %X'44', %X'55',      ! 14
;    2388  1                           %X'CC', %X'36', %X'26', %X'27', %X'27', %X'49',      ! 15
;    2389  1                           %X'33', %X'A1', %X'67', %X'BB', %X'4C', %X'9F',      ! 16
;    2390  1                           %X'EB', %X'BE', %X'C7', %X'8F', %X'33', %X'FF' ),    ! 17
;    2391  1
;    2392  1         BD_PROM_DESCR   : VECTOR [ BD_D_SIZE, WORD ] INITIAL ( WORD (
```

```
;    2393  1
;    2394  1                          NEWB,               ! BUFFER NOT USED IF 1
;    2395  1                          V,                  ! VALID ADDRESS IF 1
;    2396  1                          RCV_BUFFER,         ! RCV BUFFER ADDRESS
;    2397  1                          BYTE_COUNT,         ! 1/4 THE BYTE COUNT
;    2398  1                          0,                  ! STATUS WORD 1
;    2399  1                          0,                  ! STATUS WORD 2
;    2400  1
;    2401  1                          NEWB,               ! BUFFER NOT USED IF 1
;    2402  1                          V,                  ! VALID ADDRESS IF 1
;    2403  1                          XMIT_BUFFER,        ! XMIT BUFFER ADDRESS
;    2404  1                          BYTE_COUNT,         ! 1/4 THE BYTE COUNT
;    2405  1                          0,                  ! STATUS WORD 1
;    2406  1                          0,                  ! STATUS WORD 2
;    2407  1
;    2408  1                          NEWB,               ! BUFFER NOT USED IF 1
;    2409  1                          E,                  ! VALID ADDRESS IF 1
;    2410  1                          0,                  ! 2 EXTRA WORDS
;    2411  1                          0 )),               !
```

```
;    2412  1      !++
;    2413  1      !       HARDWARE P TABLE STORAGE DECLARATIONS
;    2414  1      ! -
;    2415  1
;    2416  1          HWP_TABLE    : REF BLOCK [ HWP SIZE, WORD ] FIELD ( HWP FIELDS ),
;    2417  1
;    2418  1
;    2419  1      !++
;    2420  1      !       SOFTWARE P TABLE STORAGE DECLARATIONS
;    2421  1      !
;    2422  1
;    2423  1          SWP_TABLE    : REF BLOCK [ SWP SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    2424  1
;    2425  1      !++
;    2426  1      !       SYSTEM CLOCK STORAGE DECLARATIONS
;    2427  1      ! -
;    2428  1
;    2429  1          CLK_ADR              : WORD,                 ! LOC. TO RETURN CLOCK ADDR.
;    2430  1          CLK_TYPE             : WORD,                 ! TYPE OF CLOCK ON SYSTEM
;    2431  1                                                       ! (0=NONE,-1=L-CLOCK,1=P_CLOCK)
;    2432  1          CLK_VEC              : WORD,                 ! CLOCK INTERRUPT VECTOR ADR
;    2433  1          CLK_CSR              : WORD,                 ! STORE CSR ADDR FOR CLOCK HERE
;    2434  1          CLK_START            : WORD,                 ! STORE CLOCK START VALUE
;    2435  1          CLK_HERTZ            : WORD,                 ! TOTAL # OF CLOCK INTERRUPTS
;    2436  1          TICKS                : WORD,                 ! CLOCK RATE
;    2437  1          SECONDS              : WORD INITIAL (0),     ! STORE SECONDS
;    2438  1          MINUTES              : WORD INITIAL (0),     ! STORE MINUTES
;    2439  1          CANCEL_TIMER         : WORD,                 !
;    2440  1
;    2441  1      !++
;    2442  1      !
;    2443  1      !       MISCELLANEOUS DATA DECLARATIONS
;    2444  1      !
;    2445  1      !--
;    2446  1
;    2447  1          XBUF_LENGTH          : WORD,                 ! XMIT BUFFER LENGTH IN WORDS
;    2448  1          RBUF_LENGTH          : WORD,                 ! RCV BUFFER LENGTH IN BYTES
;    2449  1          FREE_MEM_ADR         : WORD,                 ! FREE MEMORY BEGIN ADR
;    2450  1          MEM_SIZE             : WORD,                 ! FREE MEMORY SIZE
;    2451  1          INTERRUPT_FLG        : WORD,                 ! 1 = INTERRUPT OCCURED
;    2452  1          DEQNA_NO             : WORD,                 ! DEQNA UNDER TEST THIS PASS
;    2453  1          COUNTER              : WORD,                 ! INETATION COUNTER, INDEX
;    2454  1          CHECKSUM             : WORD,                 ! EXPECTED PROM CHECKSUM
;    2455  1          BUF_LENGTH           : WORD,                 ! XMIT BUFFER SIZE IN WORDS
;    2456  1          CSR_WORD             : WORD,                 !
;    2457  1
;    2458  1          REG_ADR              : REF REG_STR FIELD ( IOP_FIELDS ),
;    2459  1          IOP_DATA             : REF REG_STR FIELD ( IOP_FIELDS ),
;    2460  1          GET_ADR              : REF ADR_STR FIELD ( IOP_FIELDS ),
;    2461  1
```

```
;    2462  1      !++
;    2463  1      !
;    2464  1      !   TEMPORARY STORAGE DATA DECLARATIONS
;    2465  1      !
;    2466  1      !--
;    2467  1
;    2468  1          TMP_IOP_ADR         : WORD,          ! I/O PAGE REGISTER ADDRESS
;    2469  1          TMP_REG_DATA        : WORD,          ! I/O PAGE REG CONTENTS
;    2470  1          TEMP1               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2471  1          TEMP2               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2472  1          TEMP3               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2473  1          TEMP4               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2474  1          TEMP5               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2475  1          TEMP6               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2476  1          TEMP7               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2477  1          TEMP8               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2478  1          TEMP9               : WORD,          ! TEMPORARY STORAGE LOCATION
;    2479  1          P1                  : WORD,          ! PARAMETER #1
;    2480  1          P2                  : WORD,          ! PARAMETER #2
;    2481  1          P3                  : WORD,          ! PARAMETER #3
;    2482  1          P4                  : WORD,          ! PARAMETER #4
;    2483  1          P5                  : WORD,          ! PARAMETER #5
;    2484  1          TBYTE1              : BYTE,          !
;    2485  1          TBYTE2              : BYTE,          !
;    2486  1          TBYTE3              : BYTE,          !
;    2487  1          TBYTE4              : BYTE,          !
;    2488  1          TADR1               : WORD,          !
;    2489  1          TADR2               : WORD;          !
;    2490  1
;    2491  1
;    2492  1
;    2493  1      !++
;    2494  1      !       THE ERRTBL MACRO IS REQUIRED WHETHER OR NOT YOU REPORT ERRORS USING
;    2495  1      !       THE "ERROR" MACRO.  THE ERRTBL MACRO EXPANDS INTO FOUR WORDS THAT
;    2496  1      !       ARE USED BY THE RUNTIME SERVICES DURING AN ERROR CALL: ERROR TYPE,
;    2497  1      !       ERROR NUMBER, ADDRESS OF ERROR MESSAGE AND ADDRESS OF MESSAGE
;    2498  1      !       BLOCK.  THERE MUST BE ONLY ONE ERRTBL IN ANY PROGRAM.  THIS SECTION
;    2499  1      !       IS NOT OPTIONAL.
;    2500  1      !--
;    2501  1
;    2502  1
```

```
;    2503  1
;    2504  1     %SBTTL 'GLOBAL TEXT SECTION'
;    2505  1
;    2506  1     !++
;    2507  1     !       THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS, MESSAGES,
;    2508  1     !       AND ASCII INFORMATION THAT IS USED IN MORE THAN ONE TEST.
;    2509  1     !--
;    2510  1
;    2511  1     GLOBAL BIND
;    2512  1
;    2513  1         DESCR_LIST    = RCV_D_LIST,
;    2514  1         DATA_BUFFER   = RCV_BUFFER,
;    2515  1
;    2516  1     !++
;    2517  1     !       HARDWARE AND SOFTWARE QUESTIONS
;    2518  1     !--
;    2519  1
;    2520  1         QST01 = UPLIT (%ASCIZ'DEQNA I/O PAGE ADR     '),
;    2521  1         QST02 = UPLIT (%ASCIZ'INTERRUPT VECTOR ADR   '),
;    2522  1         QST03 = UPLIT (%ASCIZ'BR LEVEL               '),
;    2523  1         QST04 = UPLIT (%ASCIZ'DO YOU WANT TO DO MANUAL INTERVENTION ? '),
;    2524  1         QST05 = UPLIT (%ASCIZ'ERROR LIMIT FOR DROPPING DEQNA ? '),
;    2525  1         QST06 = UPLIT (%ASCIZ''),
;    2526  1         QST07 = UPLIT (%ASCIZ''),
;    2527  1         QST08 = UPLIT (%ASCIZ''),
;    2528  1         QST09 = UPLIT (%ASCIZ''),
;    2529  1
;    2530  1     !++
;    2531  1     !       THE FOLLOWING DBM'S ARE DEBUG MESSAGES, AND SHOULD BE REMOVED BEFORE
;    2532  1     !       RELEASING THE PROGRAM.  THEY INCLUDE THE NAMES OF EACH ROUTINE, PLUS
;    2533  1     !       FORMAT STATEMENTS FOR PRINTING OUT OTHER INFORMATION.
;    2534  1     !--
;    2535  1
;    2536  1         DBM01 = UPLIT (%ASCIZ'%N%A INITIALIZE CODE SECTION '),
;    2537  1         DBM02 = UPLIT (%ASCIZ'%N%A START COMMAND '),
;    2538  1         DBM03 = UPLIT (%ASCIZ'%N%A RESTART COMMAND '),
;    2539  1         DBM04 = UPLIT (%ASCIZ'%N%A CONTINUE COMMAND '),
;    2540  1         DBM05 = UPLIT (%ASCIZ'%N%A AUTODROP CODE SECTION '),
;    2541  1         DBM06 = UPLIT (%ASCIZ'%N%A CLEAN-UP CODE SECTION '),
;    2542  1         DBM33 = UPLIT (%ASCIZ'%N%A DROP DEVICE CODE SECTION '),
;    2543  1         DBM34 = UPLIT (%ASCIZ'%N%A ADD DEVICE CODE SECTION '),
;    2544  1         DBM35 = UPLIT (%ASCIZ'%EXTENDED ERROR MESSAGE PRTNT ROUTINE %N'),
;    2545  1         DBM36 = UPLIT (%ASCIZ'%N%A WE HAVE P-CLOCK ON THE SYSTEM '),
;    2546  1         DBM37 = UPLIT (%ASCIZ'%N%A WE HAVE L-CLOCK ON THE SYSTEM '),
;    2547  1         DBM38 = UPLIT (%ASCIZ'%N%A CLOCK VECTOR: %O6%A  CLOCK I/OP ADR: %O6%N%N'),
;    2548  1         DBM39 = UPLIT (%ASCIZ'%N%A DEQNA ADR: %O6%A  VEC: %O6%N%N'),
;    2549  1         DBM40 = UPLIT (%ASCIZ'%N%A CSR ADR: %O6%A   CSR = %O6%N%N'),
;    2550  1         DBM41 = UPLIT (%ASCIZ'%N%A STATION ADR: ACTUAL = %O6%A EXPECTED = %O6%N%N'),
;    2551  1         DBM42 = UPLIT (%ASCIZ'%N%A IOP ADR = %O6%N'),
;    2552  1         DBM43 = UPLIT (%ASCIZ'%N%A EXPECTED DESCR - INDEX=%O6%A EXPECTED = %O6%N'),
;    2553  1         DBM44 = UPLIT (%ASCIZ'%N%A INTR FLG = %O6%A ACTUAL = %O6%A EXPECTED = %O6%N'),
;    2554  1         DBM45 = UPLIT (%ASCIZ'%N%A XMIT = %O6%A RCV = %O6%N'),
;    2555  1         DBM46 = UPLIT (%ASCIZ'%N%A CA BIT IS OK, CSR = %O6%N%N'),
```

```
;    2556  1
;    2557  1      !++
;    2558  1      !       SYSTEM ERROR MESSAGES
;    2559  1      ! -
;    2560  1
;    2561  1         ERR01 = UPLIT (%ASCIZ'%N%ANO CLOCK WAS FOUND ON THE SYSTEM'),
;    2562  1         ERR02 = UPLIT (%ASCIZ'%AI/O PAGE REG NOT PRESENT AT '),
;    2563  1         ERR03 = UPLIT (%ASCIZ'%AERROR - TOO MANY DEVICES '),
;    2564  1         ERR04 = UPLIT (%ASCIZ'%N%A PROM CHECKSUM: ACT = %O6%A EXP = %O6%N'),
;    2565  1         ERR05 = UPLIT (%ASCIZ'%A UNABLE TO RESET DEQNA: ADR: %O6%A  CSR = %O6%N'),
;    2566  1         ERR06 = UPLIT (%ASCIZ'%A'),
;    2567  1         ERR07 = UPLIT (%ASCIZ'%A'),
;    2568  1         ERR08 = UPLIT (%ASCIZ'%A'),
;    2569  1
;    2570  1      !++
;    2571  1      !       FORMATTED ASCIC STRINGS
;    2572  1      !--
;    2573  1         E0001 = UPLIT (%ASCIZ'DEQNA FATAL ERROR DETECTED '),
;    2574  1         E0101 = UPLIT (%ASCIZ'%N%AREGISTER FAILED TO RESPOND AT ADDRESS:  %O6%N'),
;    2575  1         E0201 = UPLIT (%ASCIZ'%ACSR BITS STUCK AT 0 - CSR ADR: %O6%A  CSR = %O6%N'),
;    2576  1         E0202 = UPLIT (%ASCIZ'%ACSR BITS STUCK AT 1 - CSR ADR: %O6%A  CSR = %O6%N'),
;    2577  1         E0301 = UPLIT (%ASCIZ'%N%ABAD ST. ADR CHECKSUM: ACT = %O6%A EXP = %O6%N'),
;    2578  1         E0401 = UPLIT (%ASCIZ'%N%A CSR ADR: %O6%A ACT VEC = %O6%A EXP VEC = %O6%N'),
;    2579  1         E0501 = UPLIT (%ASCIZ'%N%A RCV LIST EXPECTED TO BE VALID, CSR BIT 5 WAS SET%N'),
;    2580  1         E0502 = UPLIT (%ASCIZ'%N%A B/D DESCR INDEX = %O6%A ACT = %O6%A EXP = %O6%N'),
;    2581  1         E0503 = UPLIT (%ASCIZ'%N%A B/D PROM CHECKSUM OFFSET = %O6%A ACT = %O6%A EXP = %O6%N'),
;    2582  1         E0601 = UPLIT (%ASCIZ'%N%A IOP ADR: %O6%A ACT LEV = %O6%A EXP LEV = %O6%N'),
;    2583  1         E0801 = UPLIT (%ASCIZ'%N%A IOP ADR: %O6%A ACT LEV = %O6%A EXP LEV = %O6%N'),
;    2584  1
;    2585  1
;    2586  1      !++
;    2587  1      !       DEVICE ERROR MESSAGES
;    2588  1      !++
;    2589  1
;    2590  1          MSG01  = UPLIT (%ASCIZ'%N%N%A   DEQNA ADDRESS: %O6%A.  STATION ADDRESS: '),
;    2591  1          MSG02  = UPLIT (%ASCIZ'%A       ACTUAL DATA = %O6%A     EXPECTED DATA = %O6%N'),
;    2592  1          MSG03  = UPLIT (%ASCIZ'%A                                XMIT DESCRIPTOR    RCV DESCRIPTOR %N'),
;    2593  1          MSG04  = UPLIT (%ASCIZ'%A       FLAG WORD                    %O6%A              %O6%N'),
;    2594  1          MSG05  = UPLIT (%ASCIZ'%A       HIGH ORDER ADDR BITS         %O6%A              %O6%N'),
;    2595  1          MSG06  = UPLIT (%ASCIZ'%A       LOW  ORDER ADDR BITS         %O6%A              %O6%N'),
;    2596  1          MSG07  = UPLIT (%ASCIZ'%A       PACKET LENGTH ( WD )         %O6%A              %O6%N'),
;    2597  1          MSG08  = UPLIT (%ASCIZ'%A       STATUS WORD 1                %O6%A              %O6%N'),
;    2598  1          MSG09  = UPLIT (%ASCIZ'%A       STATUS WORD 2                %O6%A              %O6%N'),
;    2599  1          MSG10  = UPLIT (%ASCIZ'%A       DEQNA CSR REGISTER           %O6%N'),
;    2600  1          MSG11  = UPLIT (%ASCIZ'%A       DEQNA I/O PAGE ADR           %O6%N%N'),
;    2601  1          MSG12  = UPLIT (%ASCIZ'%A BAD CSR, ACTUAL = %O6%A EXPECTED = %O6%N%N'),
;    2602  1          MSG13  = UPLIT (%ASCIZ'%A BAD TRANSMIT FLAG WORD AT I/O PAGE ADR = %O6%N'),
;    2603  1          MSG14  = UPLIT (%ASCIZ'%A BAD TRANSMIT STATUS WD 1 AT I/O PAGE ADR = %O6%N'),
;    2604  1          MSG15  = UPLIT (%ASCIZ'%A BAD RECEIVE FLAG WORD AT I/O PAGE ADR = %O6%N'),
;    2605  1          MSG16  = UPLIT (%ASCIZ'%A BAD RECEIVE STATUS WD 1 AT I/O PAGE ADR = %O6%N'),
;    2606  1          MSG17  = UPLIT (%ASCIZ'%A BAD RECEIVE BUFFER LENGTH AT I/O PAGE ADR = %O6%N'),
;    2607  1          MSG18  = UPLIT (%ASCIZ'%A BAD CSR = %O6%N'),
;    2608  1          MSG19  = UPLIT (%ASCIZ'%A LOOPBACK PACKET UNABLE TO SET CA BIT, CSR = %O6%N'),
```

```
;    2609  1          MSG20  = UPLIT (%ASCIZ'%A LOOPBACK PACKET UNABLE TO CLEAR CA BIT, CSR = %O6%N'),
;    2610  1          MSG21  = UPLIT (%ASCIZ'%A CA BIT OK, BUT RI BIT IS NOT ON, CSR = %O6%N'),
;    2611  1          MSG22  = UPLIT (%ASCIZ'%A CA BIT IN THE CSR WAS SET TOO EARLY, CSR = %O6%N'),
;    2612  1          MSG23  = UPLIT (%ASCIZ'%A %N'),
;    2613  1
;    2614  1      !++
;    2615  1      !       DIAGNOSTIC ERROR MESSAGES
;    2616  1      ! -
;    2617  1
;    2618  1          MSG_1TDR= UPLIT (%ASCIZ' THERE IS NO TDR VALUE'),
;    2619  1          MSG_2TDR= UPLIT (%ASCIZ'%N%A TDR VALUE > 0, TDR = %O6%N'),
;    2620  1          MSG_3TDR= UPLIT (%ASCIZ'%N%A TDR VALUE = 0, TDR = %O6%N'),
;    2621  1          MSG_4TDR= UPLIT (%ASCIZ'%N%A TDR VALUE < 0, TDR = %O6%N'),
;    2622  1
;    2623  1          MSG_PWR = UPLIT (%ASCIZ'POWER DELAY - WAITING'),
;    2624  1          MSG_CSR = UPLIT (%ASCIZ'CSR REGISTER FAILED TO RESPOND'),
;    2625  1          MSG_BR  = UPLIT (%ASCIZ'AT INCORRECT BR LEVEL'),
;    2626  1          MSG_INI = UPLIT (%ASCIZ'%N%A       INTERRUPT AT VEC= %O3%A BR LEVEL= %O1'),
;    2627  1          MSG_END = UPLIT (%ASCIZ'%A   COMPLETED TEST'),
;    2628  1          MSG_CLR = UPLIT (%ASCIZ'RESET DID NO CLEAR RING BUFFER'),
;    2629  1          MSG_NXM = UPLIT (%ASCIZ'NON-EXISTENT MEMORY TEST FAILURE'),
;    2630  1          MSG_LEN = UPLIT (%ASCIZ'%N%APACKET LENGTH = %O6%N');
;    2631  1
;    2632  1
;    2633  1
;    2634  1      !++
;    2635  1      !  ROUTINE TIME_INIT:(     ) =
;    2636  1      !
;    2637  1      !  THIS ROUTINE DOES A 100 MICROSECOND TIMEOUT ON THE RESPONSE TO A HARD
;    2638  1      !  INIT (DONE BY WRITING IP).  IT IS CALLED FOR ANY INIT IN THE PROGRAM
;    2639  1      !  IF THERE IS A P-CLOCK ON THE SYSTEM TO ALLOW THE TIMEOUT.  IT SETS UP
;    2640  1      !  THE P-CLOCK TO INTERRUPT AFTER 100 MICROSECONDS;  THEN IT WRITES IP, STARTS
;    2641  1      !  THE CLOCK AND READS SA LOOKING FOR THE STEP 1 BIT OR AN ERROR.  IF THERE
;    2642  1      !  IS AN INTERRUPT FROM THE CLOCK, THE INTERRUPT SERVICE ROUTINE WILL SET A
;    2643  1      !  FLAG.  THIS ROUTINE CHECKS THAT FLAG BEFORE RETURNING TO THE TEST WHICH
;    2644  1      !  CALLED IT.  IF THE FLAG IS SET, IT RETURNS A CODE SO THAT THE TEST CAN
;    2645  1      !  REPORT THAT THE TIMEOUT OCCURRED.  OTHERWISE, SUCCESS STATUS IS RETURNED
;    2646  1      !  TO THE CALLING TEST.
;    2647  1      !--
;    2648  1
;    2649  1      END
;    2650  0      ELUDOM


                                                .TITLE  ZQNA1 CZQNAAO 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$REV::
000010     101                          .ASCII  /A/
000011     060                          .ASCII  /0/
000012  000000G                 L$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  000154'                 L$HPTP::.WORD   L$HW
000024  000164'                 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  000142'                 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  000170'                 L$PRT:: .WORD   L$PROT
000114  000000                  L$TEST::.WORD   0
000116  000000                  L$DLY:: .WORD   0
000120  000000                  L$HIME::.WORD   0
000122  000007                  D$PCNT::.WORD   7
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                          ERRTYP::.BLKW   1
000144                          ERRNBR::.BLKW   1
```

```
000146                                  ERRMSG::.BLKW   1
000150                                  ERRBLK::.BLKW   1
000152  000000C                         L$HWLEN::
                                                .WORD   <<L$NDHW-L$HWLEN>/2>
000154  174440                          DFSTBL::.WORD   -3340
000156  000700                                  .WORD   700
000160                                  L$NDHW::.BLKW   1
000162  000000C                         L$SWLEN::
                                                .WORD   <<L$NDSW-L$SWLEN>/2>
000164  000001                          DFHTBL::.WORD   1
000166                                  L$NDSW::.BLKW   1
000170  177777                          L$PROT::.WORD    1
000172  177777                                  .WORD   -1
000174  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      102     122     040         P.AAC:  .ASCII  /BR /
000063      114     105     126                 .ASCII  /LEV/
000066      105     114     040                 .ASCII  /EL /
000071      040     040     040                 .ASCII  /   /
000074      040     040     040                 .ASCII  /   /
000077      040     040     040                 .ASCII  /   /
000102      040     040     040                 .ASCII  /   /
000105      040     000     000                 .ASCII  / /<00><00>
000110      104     117     040         P.AAD:  .ASCII  /DO /
000113      131     117     125                 .ASCII  /YOU/
000116      040     127     101                 .ASCII  / WA/
000121      116     124     040                 .ASCII  /NT /
000124      124     117     040                 .ASCII  /TO /
000127      104     117     040                 .ASCII  /DO /
000132      115     101     116                 .ASCII  /MAN/
000135      125     101     114                 .ASCII  /UAL/
000140      040     111     116                 .ASCII  / IN/
000143      124     105     122                 .ASCII  /TER/
000146      126     105     116                 .ASCII  /VEN/
```

```
000151  124  111  117                .ASCII  /TIO/
000154  116  040  077                .ASCII  /N ?/
000157  040  000  000                .ASCII  / /<00><00>
000162  105  122  122        P.AAE:  .ASCII  /ERR/
000165  117  122  040                .ASCII  /OR /
000170  114  111  115                .ASCII  /LIM/
000173  111  124  040                .ASCII  /IT /
000176  106  117  122                .ASCII  /FOR/
000201  040  104  122                .ASCII  / DR/
000204  117  120  120                .ASCII  /OPP/
000207  111  116  107                .ASCII  /ING/
000212  040  104  105                .ASCII  / DE/
000215  121  116  101                .ASCII  /QNA/
000220  040  077  040                .ASCII  / ? /
000223  000                         .ASCII  <00>
000224  000  000        P.AAF:  .ASCII  <00><00>
000226  000  000        P.AAG:  .ASCII  <00><00>
000230  000  000        P.AAH:  .ASCII  <00><00>
000232  000  000        P.AAI:  .ASCII  <00><00>
000234  045  116  045  P.AAJ:  .ASCII  /%N%/
000237  101  040  111                .ASCII  /A I/
000242  116  111  124                .ASCII  /NIT/
000245  111  101  114                .ASCII  /IAL/
000250  111  132  105                .ASCII  /IZE/
000253  040  103  117                .ASCII  / CO/
000256  104  105  040                .ASCII  /DE /
000261  123  105  103                .ASCII  /SEC/
000264  124  111  117                .ASCII  /TIO/
000267  116  040  000                .ASCII  /N /<00>
000272  045  116  045        P.AAK:  .ASCII  /%N%/
000275  101  040  123                .ASCII  /A S/
000300  124  101  122                .ASCII  /TAR/
000303  124  040  103                .ASCII  /T C/
000306  117  115  115                .ASCII  /OMM/
000311  101  116  104                .ASCII  /AND/
000314  040  000                     .ASCII  / /<00>
000316  045  116  045        P.AAL:  .ASCII  /%N%/
000321  101  040  122                .ASCII  /A R/
000324  105  123  124                .ASCII  /EST/
000327  101  122  124                .ASCII  /ART/
000332  040  103  117                .ASCII  / CO/
000335  115  115  101                .ASCII  /MMA/
000340  116  104  040                .ASCII  /ND /
000343  000                         .ASCII1 <00>
000344  045  116  045        P.AAM:  .ASCII  /%N%/
000347  101  040  103                .ASCII  /A C/
000352  117  116  124                .ASCII  /ONT/
000355  111  116  125                .ASCII  /INU/
000360  105  040  103                .ASCII  /E C/
000363  117  115  115                .ASCII  /OMM/
000366  101  116  104                .ASCII  /AND/
000371  040  000  000                .ASCII  / /<00><00>
000374  045  116  045        P.AAN:  .ASCII  /%N%/
```

```
000377  101  040  101                .ASCII  /A A/
000402  125  124  117                .ASCII  /UTO/
000405  104  122  117                .ASCII  /DRO/
000410  120  040  103                .ASCII  /P C/
000413  117  104  105                .ASCII  /ODE/
000416  040  123  105                .ASCII  / SE/
000421  103  124  111                .ASCII  /CTI/
000424  117  116  040                .ASCII  /ON /
000427  000                         .ASCII  <00>
000430  045  116  045       P.AAO:   .ASCII  /%N%/
000433  101  040  103                .ASCII  /A C/
000436  114  105  101                .ASCII  /LEA/
000441  116  055  125                .ASCII  /N-U/
000444  120  040  103                .ASCII  /P C/
000447  117  104  105                .ASCII  /ODE/
000452  040  123  105                .ASCII  / SE/
000455  103  124  111                .ASCII  /CTI/
000460  117  116  040                .ASCII  /ON /
000463  000                         .ASCII  <00>
000464  045  116  045       P.AAP:   .ASCII  /%N%/
000467  101  040  104                .ASCII  /A D/
000472  122  117  120                .ASCII  /ROP/
000475  040  104  105                .ASCII  / DE/
000500  126  111  103                .ASCII  /VIC/
000503  105  040  103                .ASCII  /E C/
000506  117  104  105                .ASCII  /ODE/
000511  040  123  105                .ASCII  / SE/
000514  103  124  111                .ASCII  /CTI/
000517  117  116  040                .ASCII  /ON /
000522  000  000                    .ASCII  <00><00>
000524  045  116  045       P.AAQ:   .ASCII  /%N%/
000527  101  040  101                .ASCII  /A A/
000532  104  104  040                .ASCII  /DD /
000535  104  105  126                .ASCII  /DEV/
000540  111  103  105                .ASCII  /ICE/
000543  040  103  117                .ASCII  / CO/
000546  104  105  040                .ASCII  /DE /
000551  123  105  103                .ASCII  /SEC/
000554  124  111  117                .ASCII  /TIO/
000557  116  040  000                .ASCII  /N /<00>
000562  045  105  130       P.AAR:   .ASCII  /%EX/
000565  124  105  116                .ASCII  /TEN/
000570  104  105  104                .ASCII  /DED/
000573  040  105  122                .ASCII  / ER/
000576  122  117  122                .ASCII  /ROR/
000601  040  115  105                .ASCII  / ME/
000604  123  123  101                .ASCII  /SSA/
000607  107  105  040                .ASCII  /GE /
000612  120  122  111                .ASCII  /PRI/
000615  116  124  040                .ASCII  /NT /
000620  122  117  125                .ASCII  /ROU/
000623  124  111  116                .ASCII  /TIN/
000626  105  040  045                .ASCII  /E %/
```

```
000631  116  000  000                .ASCII  /N/<00><00>
000634  045  116  045        P.AAS:  .ASCII  /%N%/
000637  101  040  127                .ASCII  /A W/
000642  105  040  110                .ASCII  /E H/
000645  101  126  105                .ASCII  /AVE/
000650  040  120  055                .ASCII  / P-/
000653  103  114  117                .ASCII  /CLO/
000656  103  113  040                .ASCII  /CK /
000661  117  116  040                .ASCII  /ON /
000664  124  110  105                .ASCII  /THE/
000667  040  123  131                .ASCII  / SY/
000672  123  124  105                .ASCII  /STE/
000675  115  040  000                .ASCII  /M /<00>
000700  045  116  045        P.AAT:  .ASCII  /%N%/
000703  101  040  127                .ASCII  /A W/
000706  105  040  110                .ASCII  /E H/
000711  101  126  105                .ASCII  /AVE/
000714  040  114  055                .ASCII  / L-/
000717  103  114  117                .ASCII  /CLO/
000722  103  113  040                .ASCII  /CK /
000725  117  116  040                .ASCII  /ON /
000730  124  110  105                .ASCII  /THE/
000733  040  123  131                .ASCII  / SY/
000736  123  124  105                .ASCII  /STE/
000741  115  040  000                .ASCII  /M /<00>
000744  045  116  045        P.AAU:  .ASCII  /%N%/
000747  101  040  103                .ASCII  /A C/
000752  114  117  103                .ASCII  /LOC/
000755  113  040  126                .ASCII  /K V/
000760  105  103  124                .ASCII  /ECT/
000763  117  122  072                .ASCII  /OR:/
000766  040  045  117                .ASCII  / %O/
000771  066  045  101                .ASCII  /6%A/
000774  040  040  103                .ASCII  /  C/
000777  114  117  103                .ASCII  /LOC/
001002  113  040  111                .ASCII  /K I/
001005  057  117  120                .ASCII  <57>/OP/
001010  040  101  104                .ASCII  / AD/
001013  122  072  040                .ASCII  /R: /
001016  045  117  066                .ASCII  /%O6/
001021  045  116  045                .ASCII  /%N%/
001024  116  000                    .ASCII  /N/<00>
001026  045  116  045        P.AAV:  .ASCII  /%N%/
001031  101  040  104                .ASCII  /A D/
001034  105  121  116                .ASCII  /EQN/
001037  101  040  101                .ASCII  /A A/
001042  104  122  072                .ASCII  /DR:/
001045  040  045  117                .ASCII  / %O/
001050  066  045  101                .ASCII  /6%A/
001053  040  040  126                .ASCII  /  V/
001056  105  103  072                .ASCII  /EC:/
001061  040  045  117                .ASCII  / %O/
001064  066  045  116                .ASCII  /6%N/
```

```
001067  045  116  000                .ASCII  /%N/<00>
001072  045  116  045        P.AAW:  .ASCII  /%N%/
001075  101  040  103                .ASCII  /A C/
001100  123  122  040                .ASCII  /SR /
001103  101  104  122                .ASCII  /ADR/
001106  072  040  045                .ASCII  /: %/
001111  117  066  045                .ASCII  /06%/
001114  101  040  040                .ASCII  /A  /
001117  040  103  123                .ASCII  / CS/
001122  122  040  075                .ASCII  /R =/
001125  040  045  117                .ASCII  / %O/
001130  066  045  116                .ASCII  /6%N/
001133  045  116  000                .ASCII  /%N/<00>
001136  045  116  045        P.AAX:  .ASCII  /%N%/
001141  101  040  123                .ASCII  /A S/
001144  124  101  124                .ASCII  /TAT/
001147  111  117  116                .ASCII  /ION/
001152  040  101  104                .ASCII  / AD/
001155  122  072  040                .ASCII  /R: /
001160  101  103  124                .ASCII  /ACT/
001163  125  101  114                .ASCII  /UAL/
001166  040  075  040                .ASCII  / = /
001171  045  117  066                .ASCII  /%06/
001174  045  101  040                .ASCII  /%A /
001177  105  130  120                .ASCII  /EXP/
001202  105  103  124                .ASCII  /ECT/
001205  105  104  040                .ASCII  /ED /
001210  075  040  045                .ASCII  /= %/
001213  117  066  045                .ASCII  /06%/
001216  116  045  116                .ASCII  /N%N/
001221  000                        .ASCII  <00>
001222  045  116  045        P.AAY:  .ASCII  /%N%/
001225  101  040  111                .ASCII  /A I/
001230  117  120  040                .ASCII  /OP /
001233  101  104  122                .ASCII  /ADR/
001236  040  075  040                .ASCII  / = /
001241  045  117  066                .ASCII  /%06/
001244  045  116  000                .ASCII  /%N/<00>
001247  000                        .ASCII  <00>
001250  045  116  045        P.AAZ:  .ASCII  /%N%/
001253  101  040  105                .ASCII  /A E/
001256  130  120  105                .ASCII  /XPE/
001261  103  124  105                .ASCII  /CTE/
001264  104  040  104                .ASCII  /D D/
001267  105  123  103                .ASCII  /ESC/
001272  122  040  055                .ASCII  /R -/
001275  040  111  116                .ASCII  / IN/
001300  104  105  130                .ASCII  /DEX/
001303  075  045  117                .ASCII  /=%O/
001306  066  045  101                .ASCII  /6%A/
001311  040  105  130                .ASCII  / EX/
001314  120  105  103                .ASCII  /PEC/
001317  124  105  104                .ASCII  /TED/
```

M3


```
001322  040  075  040              .ASCII  / = /
001325  045  117  066              .ASCII  /%06/
001330  045  116  000              .ASCII  /%N/<00>
001333  000                       .ASCII  <00>
001334  045  116  045  P.ABA:     .ASCII  /%N%/
001337  101  040  111              .ASCII  /A I/
001342  116  124  122              .ASCII  /NTR/
001345  040  106  114              .ASCII  / FL/
001350  107  040  075              .ASCII  /G =/
001353  040  045  117              .ASCII  / %O/
001356  066  045  101              .ASCII  /6%A/
001361  040  101  103              .ASCII  / AC/
001364  124  125  101              .ASCII  /TUA/
001367  114  040  075              .ASCII  /L =/
001372  040  045  117              .ASCII  / %O/
001375  066  045  101              .ASCII  /6%A/
001400  040  105  130              .ASCII  / EX/
001403  120  105  103              .ASCII  /PEC/
001406  124  105  104              .ASCII  /TED/
001411  040  075  040              .ASCII  / = /
001414  045  117  066              .ASCII  /%06/
001417  045  116  000              .ASCII  /%N/<00>
001422  045  116  045  P.ABB:     .ASCII  /%N%/
001425  101  040  130              .ASCII  /A X/
001430  115  111  124              .ASCII  /MIT/
001433  040  075  040              .ASCII  / = /
001436  045  117  066              .ASCII  /%06/
001441  045  101  040              .ASCII  /%A /
001444  122  103  126              .ASCII  /RCV/
001447  040  075  040              .ASCII  / = /
001452  045  117  066              .ASCII  /%06/
001455  045  116  000              .ASCII  /%N/<00>
001460  045  116  045  P.ABC:     .ASCII  /%N%/
001463  101  040  103              .ASCII  /A C/
001466  101  040  102              .ASCII  /A B/
001471  111  124  040              .ASCII  /IT /
001474  111  123  040              .ASCII  /IS /
001477  117  113  054              .ASCII  /OK,/
001502  040  103  123              .ASCII  / CS/
001505  122  040  075              .ASCII  /R =/
001510  040  045  117              .ASCII  / %O/
001513  066  045  116              .ASCII  /6%N/
001516  045  116  000              .ASCII  /%N/<00>
001521  000                       .ASCII  <00>
001522  045  116  045  P.ABD:     .ASCII  /%N%/
001525  101  116  117              .ASCII  /ANO/
001530  040  103  114              .ASCII  / CL/
001533  117  103  113              .ASCII  /OCK/
001536  040  127  101              .ASCII  / WA/
001541  123  040  106              .ASCII  /S F/
001544  117  125  116              .ASCII  /OUN/
001547  104  040  117              .ASCII  /D O/
001552  116  040  124              .ASCII  /N T/
```

```
001555  110  105  040              .ASCII  /HE /
001560  123  131  123              .ASCII  /SYS/
001563  124  105  115              .ASCII  /TEM/
001566  000  000                   .ASCII  <00><00>
001570  045  101  111  P.ABE:      .ASCII  /%AI/
001573  057  117  040              .ASCII  <57>/O /
001576  120  101  107              .ASCII  /PAG/
001601  105  040  122              .ASCII  /E R/
001604  105  107  040              .ASCII  /EG /
001607  116  117  124              .ASCII  /NOT/
001612  040  120  122              .ASCII  / PR/
001615  105  123  105              .ASCII  /ESE/
001620  116  124  040              .ASCII  /NT /
001623  101  124  040              .ASCII  /AT /
001626  000  000                   .ASCII  <00><00>
001630  045  101  105  P.ABF:      .ASCII  /%AE/
001633  122  122  117              .ASCII  /RRO/
001636  122  040  055              .ASCII  /R -/
001641  040  124  117              .ASCII  / TO/
001644  117  040  115              .ASCII  /O M/
001647  101  116  131              .ASCII  /ANY/
001652  040  104  105              .ASCII  / DE/
001655  126  111  103              .ASCII  /VIC/
001660  105  123  040              .ASCII  /ES /
001663  000                       .ASCII  <00>
001664  045  116  045  P.ABG:      .ASCII  /%N%/
001667  101  040  120              .ASCII  /A P/
001672  122  117  115              .ASCII  /ROM/
001675  040  103  110              .ASCII  / CH/
001700  105  103  113              .ASCII  /ECK/
001703  123  125  115              .ASCII  /SUM/
001706  072  040  101              .ASCII  /: A/
001711  103  124  040              .ASCII  /CT /
001714  075  040  045              .ASCII  /= %/
001717  117  066  045              .ASCII  /06%/
001722  101  040  105              .ASCII  /A E/
001725  130  120  040              .ASCII  /XP /
001730  075  040  045              .ASCII  /= %/
001733  117  066  045              .ASCII  /06%/
001736  116  000                   .ASCII  /N/<00>
001740  045  101  040  P.ABH:      .ASCII  /%A /
001743  125  116  101              .ASCII  /UNA/
001746  102  114  105              .ASCII  /BLE/
001751  040  124  117              .ASCII  / TO/
001754  040  122  105              .ASCII  / RE/
001757  123  105  124              .ASCII  /SET/
001762  040  104  105              .ASCII  / DE/
001765  121  116  101              .ASCII  /QNA/
001770  072  040  101              .ASCII  /: A/
001773  104  122  072              .ASCII  /DR:/
001776  040  045  117              .ASCII  / %O/
002001  066  045  101              .ASCII  /6%A/
002004  040  040  103              .ASCII  /  C/
```

```
002007  123  122  040                .ASCII  /SR /
002012  075  040  045                .ASCII  /= %/
002015  117  066  045                .ASCII  /06%/
002020  116  000                     .ASCII  /N/<00>
002022  045  101  000       P.ABI:   .ASCII  /%A/<00>
002025  000                          .ASCII  <00>
002026  045  101  000       P.ABJ:   .ASCII  /%A/<00>
002031  000                          .ASCII  <00>
002032  045  101  000       P.ABK:   .ASCII  /%A/<00>
002035  000                          .ASCII  <00>
002036  104  105  121       P.ABL:   .ASCII  /DEQ/
002041  116  101  040                .ASCII  /NA /
002044  106  101  124                .ASCII  /FAT/
002047  101  114  040                .ASCII  /AL /
002052  105  122  122                .ASCII  /ERR/
002055  117  122  040                .ASCII  /OR /
002060  104  105  124                .ASCII  /DET/
002063  105  103  124                .ASCII  /ECT/
002066  105  104  040                .ASCII  /ED /
002071  000                          .ASCII  <00>
002072  045  116  045       P.ABM:   .ASCII  /%N%/
002075  101  122  105                .ASCII  /ARE/
002100  107  111  123                .ASCII  /GIS/
002103  124  105  122                .ASCII  /TER/
002106  040  106  101                .ASCII  / FA/
002111  111  114  105                .ASCII  /ILE/
002114  104  040  124                .ASCII  /D T/
002117  117  040  122                .ASCII  /O R/
002122  105  123  120                .ASCII  /ESP/
002125  117  116  104                .ASCII  /OND/
002130  040  101  124                .ASCII  / AT/
002133  040  101  104                .ASCII  / AD/
002136  104  122  105                .ASCII  /DRE/
002141  123  123  072                .ASCII  /SS:/
002144  040  040  045                .ASCII  /  %/
002147  117  066  045                .ASCII  /06%/
002152  116  000                     .ASCII  /N/<00>
002154  045  101  103       P.ABN:   .ASCII  /%AC/
002157  123  122  040                .ASCII  /SR /
002162  102  111  124                .ASCII  /BIT/
002165  123  040  123                .ASCII  /S S/
002170  124  125  103                .ASCII  /TUC/
002173  113  040  101                .ASCII  /K A/
002176  124  040  060                .ASCII  /T 0/
002201  040  055  040                .ASCII  / - /
002204  103  123  122                .ASCII  /CSR/
002207  040  101  104                .ASCII  / AD/
002212  122  072  040                .ASCII  /R: /
002215  045  117  066                .ASCII  /%06/
002220  045  101  040                .ASCII  /%A /
002223  040  103  123                .ASCII  / CS/
002226  122  040  075                .ASCII  /R =/
002231  040  045  117                .ASCII  / %O/
```

```
002234  066  045  116                    .ASCII  /6≠N/
002237  000                              .ASCII  <00>
002240  045  101  103          P.ABO:    .ASCII  /≠AC/
002243  123  122  040                    .ASCII  /SR /
002246  102  111  124                    .ASCII  /BIT/
002251  123  040  123                    .ASCII  /S S/
002254  124  125  103                    .ASCII  /TUC/
002257  113  040  101                    .ASCII  /K A/
002262  124  040  061                    .ASCII  /T 1/
002265  040  055  040                    .ASCII  / - /
002270  103  123  122                    .ASCII  /CSR/
002273  040  101  104                    .ASCII  / AD/
002276  122  072  040                    .ASCII  /R: /
002301  045  117  066                    .ASCII  /≠O6/
002304  045  101  040                    .ASCII  /≠A /
002307  040  103  123                    .ASCII  / CS/
002312  122  040  075                    .ASCII  /R =/
002315  040  045  117                    .ASCII  / ≠O/
002320  066  045  116                    .ASCII  /6≠N/
002323  000                              .ASCII  <00>
002324  045  116  045          P.ABP:    .ASCII  /≠N≠/
002327  101  102  101                    .ASCII  /ABA/
002332  104  040  123                    .ASCII  /D S/
002335  124  056  040                    .ASCII  /T. /
002340  101  104  122                    .ASCII  /ADR/
002343  040  103  110                    .ASCII  / CH/
002346  105  103  113                    .ASCII  /ECK/
002351  123  125  115                    .ASCII  /SUM/
002354  072  040  101                    .ASCII  /: A/
002357  103  124  040                    .ASCII  /CT /
002362  075  040  045                    .ASCII  /= ≠/
002365  117  066  045                    .ASCII  /06≠/
002370  101  040  105                    .ASCII  /A E/
002373  130  120  040                    .ASCII  /XP /
002376  075  040  045                    .ASCII  /= ≠/
002401  117  066  045                    .ASCII  /06≠/
002404  116  000                        .ASCII  /N/<00>
002406  045  116  045          P.ABQ:    .ASCII  /≠N≠/
002411  101  040  103                    .ASCII  /A C/
002414  123  122  040                    .ASCII  /SR /
002417  101  104  122                    .ASCII  /ADR/
002422  072  040  045                    .ASCII  /: ≠/
002425  117  066  045                    .ASCII  /06≠/
002430  101  040  101                    .ASCII  /A A/
002433  103  124  040                    .ASCII  /CT /
002436  126  105  103                    .ASCII  /VEC/
002441  040  075  040                    .ASCII  / = /
002444  045  117  066                    .ASCII  /≠06/
002447  045  101  040                    .ASCII  /≠A /
002452  105  130  120                    .ASCII  /EXP/
002455  040  126  105                    .ASCII  / VE/
002460  103  040  075                    .ASCII  /C =/
002463  040  045  117                    .ASCII  / ≠O/
```

```
002466  066  045  116                .ASCII  /6%N/
002471  000                         .ASCII  <00>
002472  045  116  045       P.ABR:   .ASCII  /%N%/
002475  101  040  122                .ASCII  /A R/
002500  103  126  040                .ASCII  /CV /
002503  114  111  123                .ASCII  /LIS/
002506  124  040  105                .ASCII  /T E/
002511  130  120  105                .ASCII  /XPE/
002514  103  124  105                .ASCII  /CTE/
002517  104  040  124                .ASCII  /D T/
002522  117  040  102                .ASCII  /O B/
002525  105  040  126                .ASCII  /E V/
002530  101  114  111                .ASCII  /ALI/
002533  104  054  040                .ASCII  /D, /
002536  103  123  122                .ASCII  /CSR/
002541  040  102  111                .ASCII  / BI/
002544  124  040  065                .ASCII  /T 5/
002547  040  127  101                .ASCII  / WA/
002552  123  040  123                .ASCII  /S S/
002555  105  124  045                .ASCII  /ET%/
002560  116  000                    .ASCII  /N/<00>
002562  045  116  045       P.ABS:   .ASCII  /%N%/
002565  101  040  102                .ASCII  /A B/
002570  057  104  040                .ASCII  <57>/D /
002573  104  105  123                .ASCII  /DES/
002576  103  122  040                .ASCII  /CR /
002601  111  116  104                .ASCII  /IND/
002604  105  130  040                .ASCII  /EX /
002607  075  040  045                .ASCII  /= %/
002612  117  066  045                .ASCII  /06%/
002615  101  040  101                .ASCII  /A A/
002620  103  124  040                .ASCII  /CT /
002623  075  040  045                .ASCII  /= %/
002626  117  066  045                .ASCII  /06%/
002631  101  040  105                .ASCII  /A E/
002634  130  120  040                .ASCII  /XP /
002637  075  040  045                .ASCII  /= %/
002642  117  066  045                .ASCII  /06%/
002645  116  000  000                .ASCII  /N/<00><00>
002650  045  116  045       P.ABT:   .ASCII  /%N%/
002653  101  040  102                .ASCII  /A B/
002656  057  104  040                .ASCII  <57>/D /
002661  120  122  117                .ASCII  /PRO/
002664  115  040  103                .ASCII  /M C/
002667  110  105  103                .ASCII  /HEC/
002672  113  123  125                .ASCII  /KSU/
002675  115  040  117                .ASCII  /M O/
002700  106  106  123                .ASCII  /FFS/
002703  105  124  040                .ASCII  /ET /
002706  075  040  045                .ASCII  /= %/
002711  117  066  045                .ASCII  /06%/
002714  101  040  101                .ASCII  /A A/
002717  103  124  040                .ASCII  /CT /
```

```
002722  075  040  045                .ASCII  /= %/
002725  117  066  045                .ASCII  /06%/
002730  101  040  105                .ASCII  /A E/
002733  130  120  040                .ASCII  /XP /
002736  075  040  045                .ASCII  /= %/
002741  117  066  045                .ASCII  /06%/
002744  116  000                     .ASCII  /N/<00>
002746  045  116  045       P.ABU:   .ASCII  /%N%/
002751  101  040  111                .ASCII  /A I/
002754  117  120  040                .ASCII  /OP /
002757  101  104  122                .ASCII  /ADR/
002762  072  040  045                .ASCII  /: %/
002765  117  066  045                .ASCII  /06%/
002770  101  040  101                .ASCII  /A A/
002773  103  124  040                .ASCII  /CT /
002776  114  105  126                .ASCII  /LEV/
003001  040  075  040                .ASCII  / = /
003004  045  117  066                .ASCII  /%06/
003007  045  101  040                .ASCII  /%A /
003012  105  130  120                .ASCII  /EXP/
003015  040  114  105                .ASCII  / LE/
003020  126  040  075                .ASCII  /V =/
003023  040  045  117                .ASCII  / %O/
003026  066  045  116                .ASCII  /6%N/
003031  000                          .ASCII  <00>
003032  045  116  045       P.ABV:   .ASCII  /%N%/
003035  101  040  111                .ASCII  /A I/
003040  117  120  040                .ASCII  /OP /
003043  101  104  122                .ASCII  /ADR/
003046  072  040  045                .ASCII  /: %/
003051  117  066  045                .ASCII  /06%/
003054  101  040  101                .ASCII  /A A/
003057  103  124  040                .ASCII  /CT /
003062  114  105  126                .ASCII  /LEV/
003065  040  075  040                .ASCII  / = /
003070  045  117  066                .ASCII  /%06/
003073  045  101  040                .ASCII  /%A /
003076  105  130  120                .ASCII  /EXP/
003101  040  114  105                .ASCII  / LE/
003104  126  040  075                .ASCII  /V =/
003107  040  045  117                .ASCII  / %O/
003112  066  045  116                .ASCII  /6%N/
003115  000                          .ASCII  <00>
003116  045  116  045       P.ABW:   .ASCII  /%N%/
003121  116  045  101                .ASCII  /N%A/
003124  040  040  040                .ASCII  /   /
003127  104  105  121                .ASCII  /DEQ/
003132  116  101  040                .ASCII  /NA /
003135  101  104  104                .ASCII  /ADD/
003140  122  105  123                .ASCII  /RES/
003143  123  072  040                .ASCII  /S: /
003146  045  117  066                .ASCII  /%06/
003151  045  101  054                .ASCII  /%A,/
```

```
003154  040  040  123                .ASCII  /  S/
003157  124  101  124                .ASCII  /TAT/
003162  111  117  116                .ASCII  /ION/
003165  040  101  104                .ASCII  / AD/
003170  104  122  105                .ASCII  /DRE/
003173  123  123  072                .ASCII  /SS:/
003176  040  000                     .ASCII  / /<00>
003200  045  101  040       P.ABX:   .ASCII  /%A /
003203  040  040  040                .ASCII  /   /
003206  040  040  101                .ASCII  /  A/
003211  103  124  125                .ASCII  /CTU/
003214  101  114  040                .ASCII  /AL /
003217  104  101  124                .ASCII  /DAT/
003222  101  040  075                .ASCII  /A =/
003225  040  045  117                .ASCII  / %O/
003230  066  045  101                .ASCII  /6%A/
003233  040  040  040                .ASCII  /   /
003236  040  040  105                .ASCII  /  E/
003241  130  120  105                .ASCII  /XPE/
003244  103  124  105                .ASCII  /CTE/
003247  104  040  104                .ASCII  /D D/
003252  101  124  101                .ASCII  /ATA/
003255  040  075  040                .ASCII  / = /
003260  045  117  066                .ASCII  /%O6/
003263  045  116  000                .ASCII  /%N/<00>
003266  045  101  040       P.ABY:   .ASCII  /%A /
003271  040  040  040                .ASCII  /   /
003274  040  040  040                .ASCII  /   /
003277  040  040  040                .ASCII  /   /
003302  040  040  040                .ASCII  /   /
003305  040  040  040                .ASCII  /   /
003310  040  040  040                .ASCII  /   /
003313  040  040  040                .ASCII  /   /
003316  040  040  040                .ASCII  /   /
003321  040  040  040                .ASCII  /   /
003324  130  115  111                .ASCII  /XMI/
003327  124  040  104                .ASCII  /T D/
003332  105  123  103                .ASCII  /ESC/
003335  122  111  120                .ASCII  /RIP/
003340  124  117  122                .ASCII  /TOR/
003343  040  040  040                .ASCII  /   /
003346  040  122  103                .ASCII  / RC/
003351  126  040  104                .ASCII  /V D/
003354  105  123  103                .ASCII  /ESC/
003357  122  111  120                .ASCII  /RIP/
003362  124  117  122                .ASCII  /TOR/
003365  040  045  116                .ASCII  / %N/
003370  000  000                     .ASCII  <00><00>
003372  045  101  040       P.ABZ:   .ASCII  /%A /
003375  040  040  040                .ASCII  /   /
003400  040  040  106                .ASCII  /  F/
003403  114  101  107                .ASCII  /LAG/
003406  040  127  117                .ASCII  / WO/
```

```
003411  122  104  040              .ASCII  /RD /
003414  040  040  040              .ASCII  /   /
003417  040  040  040              .ASCII  /   /
003422  040  040  040              .ASCII  /   /
003425  040  040  040              .ASCII  /   /
003430  040  040  040              .ASCII  /   /
003433  040  040  045              .ASCII  /  %/
003436  117  066  045              .ASCII  /06%/
003441  101  040  040              .ASCII  /A  /
003444  040  040  040              .ASCII  /   /
003447  040  040  040              .ASCII  /   /
003452  040  040  040              .ASCII  /   /
003455  040  045  117              .ASCII  / %O/
003460  066  045  116              .ASCII  /6%N/
003463  000                       .ASCII  <00>
003464  045  101  040  P.ACA:      .ASCII  /%A /
003467  040  040  040              .ASCII  /   /
003472  040  040  110              .ASCII  /  H/
003475  111  107  110              .ASCII  /IGH/
003500  040  117  122              .ASCII  / OR/
003503  104  105  122              .ASCII  /DER/
003506  040  101  104              .ASCII  / AD/
003511  104  122  040              .ASCII  /DR /
003514  102  111  124              .ASCII  /BIT/
003517  123  040  040              .ASCII  /S  /
003522  040  040  040              .ASCII  /   /
003525  040  040  045              .ASCII  /  %/
003530  117  066  045              .ASCII  /06%/
003533  101  040  040              .ASCII  /A  /
003536  040  040  040              .ASCII  /   /
003541  040  040  040              .ASCII  /   /
003544  040  040  040              .ASCII  /   /
003547  040  045  117              .ASCII  / %O/
003552  066  045  116              .ASCII  /6%N/
003555  000                       .ASCII  <00>
003556  045  101  040  P.ACB:      .ASCII  /%A /
003561  040  040  040              .ASCII  /   /
003564  040  040  114              .ASCII  /  L/
003567  117  127  040              .ASCII  /OW /
003572  040  117  122              .ASCII  / OR/
003575  104  105  122              .ASCII  /DER/
003600  040  101  104              .ASCII  / AD/
003603  104  122  040              .ASCII  /DR /
003606  102  111  124              .ASCII  /BIT/
003611  123  040  040              .ASCII  /S  /
003614  040  040  040              .ASCII  /   /
003617  040  040  045              .ASCII  /  %/
003622  117  066  045              .ASCII  /06%/
003625  101  040  040              .ASCII  /A  /
003630  040  040  040              .ASCII  /   /
003633  040  040  040              .ASCII  /   /
003636  040  040  040              .ASCII  /   /
003641  040  045  117              .ASCII  / %O/
```

```
003644  066  045  116                  .ASCII  /6%N/
003647  000                            .ASCII  <00>
003650  045  101  040        P.ACC:    .ASCII  /%A /
003653  040  040  040                  .ASCII  /  /
003656  040  040  120                  .ASCII  /  P/
003661  101  103  113                  .ASCII  /ACK/
003664  105  124  040                  .ASCII  /ET /
003667  114  105  116                  .ASCII  /LEN/
003672  107  124  110                  .ASCII  /GTH/
003675  040  050  040                  .ASCII  / ( /
003700  127  104  040                  .ASCII  /WD /
003703  051  040  040                  .ASCII  /) /
003706  040  040  040                  .ASCII  /  /
003711  040  040  045                  .ASCII  /  %/
003714  117  066  045                  .ASCII  /06%/
003717  101  040  040                  .ASCII  /A  /
003722  040  040  040                  .ASCII  /  /
003725  040  040  040                  .ASCII  /  /
003730  040  040  040                  .ASCII  /  /
003733  040  045  117                  .ASCII  / %O/
003736  066  045  116                  .ASCII  /6%N/
003741  000                            .ASCII  <00>
003742  045  101  040        P.ACD:    .ASCII  /%A /
003745  040  040  040                  .ASCII  /  /
003750  040  040  123                  .ASCII  /  S/
003753  124  101  124                  .ASCII  /TAT/
003756  125  123  040                  .ASCII  /US /
003761  127  117  122                  .ASCII  /WOR/
003764  104  040  061                  .ASCII  /D 1/
003767  040  040  040                  .ASCII  /  /
003772  040  040  040                  .ASCII  /  /
003775  040  040  040                  .ASCII  /  /
004000  040  040  040                  .ASCII  /  /
004003  040  040  045                  .ASCII  /  %/
004006  117  066  045                  .ASCII  /06%/
004011  101  040  040                  .ASCII  /A  /
004014  040  040  040                  .ASCII  /  /
004017  040  040  040                  .ASCII  /  /
004022  040  040  040                  .ASCII  /  /
004025  040  045  117                  .ASCII  / %O/
004030  066  045  116                  .ASCII  /6%N/
004033  000                            .ASCII  <00>
004034  045  101  040        P.ACE:    .ASCII  /%A /
004037  040  040  040                  .ASCII  /  /
004042  040  040  123                  .ASCII  /  S/
004045  124  101  124                  .ASCII  /TAT/
004050  125  123  040                  .ASCII  /US /
004053  127  117  122                  .ASCII  /WOR/
004056  104  040  062                  .ASCII  /D 2/
004061  040  040  040                  .ASCII  /  /
004064  040  040  040                  .ASCII  /  /
004067  040  040  040                  .ASCII  /  /
004072  040  040  040                  .ASCII  /  /
```

```
004075  040  040  045              .ASCII  /  %/
004100  117  066  045              .ASCII  /06%/
004103  101  040  040              .ASCII  /A  /
004106  040  040  040              .ASCII  /   /
004111  040  040  040              .ASCII  /   /
004114  040  040  040              .ASCII  /   /
004117  040  045  117              .ASCII  / %O/
004122  066  045  116              .ASCII  /6%N/
004125  000                       .ASCII  <00>
004126  045  101  040  P.ACF:     .ASCII  /%A /
004131  040  040  040              .ASCII  /   /
004134  040  040  104              .ASCII  /  D/
004137  105  121  116              .ASCII  /EQN/
004142  101  040  103              .ASCII  /A C/
004145  123  122  040              .ASCII  /SR /
004150  122  105  107              .ASCII  /REG/
004153  111  123  124              .ASCII  /IST/
004156  105  122  040              .ASCII  /ER /
004161  040  040  040              .ASCII  /   /
004164  040  040  040              .ASCII  /   /
004167  040  040  045              .ASCII  /  %/
004172  117  066  045              .ASCII  /06%/
004175  116  000  000              .ASCII  /N/<00><00>
004200  045  101  040  P.ACG:     .ASCII  /%A /
004203  040  040  040              .ASCII  /   /
004206  040  040  104              .ASCII  /  D/
004211  105  121  116              .ASCII  /EQN/
004214  101  040  111              .ASCII  /A I/
004217  057  117  040              .ASCII  <57>/O /
004222  120  101  107              .ASCII  /PAG/
004225  105  040  101              .ASCII  /E A/
004230  104  122  040              .ASCII  /DR /
004233  040  040  040              .ASCII  /   /
004236  040  040  040              .ASCII  /   /
004241  040  040  045              .ASCII  /  %/
004244  117  066  045              .ASCII  /06%/
004247  116  045  116              .ASCII  /N%N/
004252  000  000                   .ASCII  <00><00>
004254  045  101  040  P.ACH:     .ASCII  /%A /
004257  102  101  104              .ASCII  /BAD/
004262  040  103  123              .ASCII  / CS/
004265  122  054  040              .ASCII  /R, /
004270  101  103  124              .ASCII  /ACT/
004273  125  101  114              .ASCII  /UAL/
004276  040  075  040              .ASCII  / = /
004301  045  117  066              .ASCII  /%06/
004304  045  101  040              .ASCII  /%A /
004307  105  130  120              .ASCII  /EXP/
004312  105  103  124              .ASCII  /ECT/
004315  105  104  040              .ASCII  /ED /
004320  075  040  045              .ASCII  /= %/
004323  117  066  045              .ASCII  /06%/
004326  116  045  116              .ASCII  /N%N/
```

```
004331  000                             .ASCII  <00>
004332  045  101  040          P.ACI:  .ASCII  /%A /
004335  102  101  104                  .ASCII  /BAD/
004340  040  124  122                  .ASCII  / TR/
004343  101  116  123                  .ASCII  /ANS/
004346  115  111  124                  .ASCII  /MIT/
004351  040  106  114                  .ASCII  / FL/
004354  101  107  040                  .ASCII  /AG /
004357  127  117  122                  .ASCII  /WOR/
004362  104  040  101                  .ASCII  /D A/
004365  124  040  111                  .ASCII  /T I/
004370  057  117  040                  .ASCII  <57>/O /
004373  120  101  107                  .ASCII  /PAG/
004376  105  040  101                  .ASCII  /E A/
004401  104  122  040                  .ASCII  /DR /
004404  075  040  045                  .ASCII  /= %/
004407  117  066  045                  .ASCII  /06%/
004412  116  000                       .ASCII  /N/<00>
004414  045  101  040          P.ACJ:  .ASCII  /%A /
004417  102  101  104                  .ASCII  /BAD/
004422  040  124  122                  .ASCII  / TR/
004425  101  116  123                  .ASCII  /ANS/
004430  115  111  124                  .ASCII  /MIT/
004433  040  123  124                  .ASCII  / ST/
004436  101  124  125                  .ASCII  /ATU/
004441  123  040  127                  .ASCII  /S W/
004444  104  040  061                  .ASCII  /D 1/
004447  040  101  124                  .ASCII  / AT/
004452  040  111  057                  .ASCII  / I/<57>
004455  117  040  120                  .ASCII  /O P/
004460  101  107  105                  .ASCII  /AGE/
004463  040  101  104                  .ASCII  / AD/
004466  122  040  075                  .ASCII  /R =/
004471  040  045  117                  .ASCII  / %O/
004474  066  045  116                  .ASCII  /6%N/
004477  000                             .ASCII  <00>
004500  045  101  040          P.ACK:  .ASCII  /%A /
004503  102  101  104                  .ASCII  /BAD/
004506  040  122  105                  .ASCII  / RE/
004511  103  105  111                  .ASCII  /CEI/
004514  126  105  040                  .ASCII  /VE /
004517  106  114  101                  .ASCII  /FLA/
004522  107  040  127                  .ASCII  /G W/
004525  117  122  104                  .ASCII  /ORD/
004530  040  101  124                  .ASCII  / AT/
004533  040  111  057                  .ASCII  / I/<57>
004536  117  040  120                  .ASCII  /O P/
004541  101  107  105                  .ASCII  /AGE/
004544  040  101  104                  .ASCII  / AD/
004547  122  040  075                  .ASCII  /R =/
004552  040  045  117                  .ASCII  / %O/
004555  066  045  116                  .ASCII  /6%N/
004560  000  000                       .ASCII  <00><00>
```

```
004562  045  101  040        P.ACL:  .ASCII  /%A /
004565  102  101  104                .ASCII  /BAD/
004570  040  122  105                .ASCII  / RE/
004573  103  105  111                .ASCII  /CEI/
004576  126  105  040                .ASCII  /VE /
004601  123  124  101                .ASCII  /STA/
004604  124  125  123                .ASCII  /TUS/
004607  040  127  104                .ASCII  / WD/
004612  040  061  040                .ASCII  / 1 /
004615  101  124  040                .ASCII  /AT /
004620  111  057  117                .ASCII  /I/<57>/O/
004623  040  120  101                .ASCII  / PA/
004626  107  105  040                .ASCII  /GE /
004631  101  104  122                .ASCII  /ADR/
004634  040  075  040                .ASCII  / = /
004637  045  117  066                .ASCII  /%O6/
004642  045  116  000                .ASCII  /%N/<00>
004645  000                        .ASCII  <00>
004646  045  101  040        P.ACM:  .ASCII  /%A /
004651  102  101  104                .ASCII  /BAD/
004654  040  122  105                .ASCII  / RE/
004657  103  105  111                .ASCII  /CEI/
004662  126  105  040                .ASCII  /VE /
004665  102  125  106                .ASCII  /BUF/
004670  106  105  122                .ASCII  /FER/
004673  040  114  105                .ASCII  / LE/
004676  116  107  124                .ASCII  /NGT/
004701  110  040  101                .ASCII  /H A/
004704  124  040  111                .ASCII  /T I/
004707  057  117  040                .ASCII  <57>/O /
004712  120  101  107                .ASCII  /PAG/
004715  105  040  101                .ASCII  /E A/
004720  104  122  040                .ASCII  /DR /
004723  075  040  045                .ASCII  /= %/
004726  117  066  045                .ASCII  /O6%/
004731  116  000  000                .ASCII  /N/<00><00>
004734  045  101  040        P.ACN:  .ASCII  /%A /
004737  102  101  104                .ASCII  /BAD/
004742  040  103  123                .ASCII  / CS/
004745  122  040  075                .ASCII  /R =/
004750  040  045  117                .ASCII  / %O/
004753  066  045  116                .ASCII  /6%N/
004756  000  000                    .ASCII  <00><00>
004760  045  101  040        P.ACO:  .ASCII  /%A /
004763  114  117  117                .ASCII  /LOO/
004766  120  102  101                .ASCII  /PBA/
004771  103  113  040                .ASCII  /CK /
004774  120  101  103                .ASCII  /PAC/
004777  113  105  124                .ASCII  /KET/
005002  040  125  116                .ASCII  / UN/
005005  101  102  114                .ASCII  /ABL/
005010  105  040  124                .ASCII  /E T/
005013  117  040  123                .ASCII  /O S/
```

```
005016  105  124  040                .ASCII  /ET /
005021  103  101  040                .ASCII  /CA /
005024  102  111  124                .ASCII  /BIT/
005027  054  040  103                .ASCII  /, C/
005032  123  122  040                .ASCII  /SR /
005035  075  040  045                .ASCII  /= %/
005040  117  066  045                .ASCII  /O6%/
005043  116  000  000                .ASCII  /N/<00><00>
005046  045  101  040  P.ACP:        .ASCII  /%A /
005051  114  117  117                .ASCII  /LOO/
005054  120  102  101                .ASCII  /PBA/
005057  103  113  040                .ASCII  /CK /
005062  120  101  103                .ASCII  /PAC/
005065  113  105  124                .ASCII  /KET/
005070  040  125  116                .ASCII  / UN/
005073  101  102  114                .ASCII  /ABL/
005076  105  040  124                .ASCII  /E T/
005101  117  040  103                .ASCII  /O C/
005104  114  105  101                .ASCII  /LEA/
005107  122  040  103                .ASCII  /R C/
005112  101  040  102                .ASCII  /A B/
005115  111  124  054                .ASCII  /IT,/
005120  040  103  123                .ASCII  / CS/
005123  122  040  075                .ASCII  /R =/
005126  040  045  117                .ASCII  / %O/
005131  066  045  116                .ASCII  /6%N/
005134  000  000                     .ASCII  <00><00>
005136  045  101  040  P.ACQ:        .ASCII  /%A /
005141  103  101  040                .ASCII  /CA /
005144  102  111  124                .ASCII  /BIT/
005147  040  117  113                .ASCII  / OK/
005152  054  040  102                .ASCII  /, B/
005155  125  124  040                .ASCII  /UT /
005160  122  111  040                .ASCII  /RI /
005163  102  111  124                .ASCII  /BIT/
005166  040  111  123                .ASCII  / IS/
005171  040  116  117                .ASCII  / NO/
005174  124  040  117                .ASCII  /T O/
005177  116  054  040                .ASCII  /N, /
005202  103  123  122                .ASCII  /CSR/
005205  040  075  040                .ASCII  / = /
005210  045  117  066                .ASCII  /%O6/
005213  045  116  000                .ASCII  /%N/<00>
005216  045  101  040  P.ACR:        .ASCII  /%A /
005221  103  101  040                .ASCII  /CA /
005224  102  111  124                .ASCII  /BIT/
005227  040  111  116                .ASCII  / IN/
005232  040  124  110                .ASCII  / TH/
005235  105  040  103                .ASCII  /E C/
005240  123  122  040                .ASCII  /SR /
005243  127  101  123                .ASCII  /WAS/
005246  040  123  105                .ASCII  / SE/
005251  124  040  124                .ASCII  /T T/
```

```
005254  117  117  040                .ASCII  /OO /
005257  105  101  122                .ASCII  /EAR/
005262  114  131  054                .ASCII  /LY,/
005265  040  103  123                .ASCII  / CS/
005270  122  040  075                .ASCII  /R =/
005273  040  045  117                .ASCII  / %O/
005276  066  045  116                .ASCII  /6%N/
005301  000                        .ASCII  <00>
005302  045  101  040        P.ACS:  .ASCII  /%A /
005305  045  116  000                .ASCII  /%N/<00>
005310  040  124  110        P.ACT:  .ASCII  / TH/
005313  105  122  105                .ASCII  /ERE/
005316  040  111  123                .ASCII  / IS/
005321  040  116  117                .ASCII  / NO/
005324  040  124  104                .ASCII  / TD/
005327  122  040  126                .ASCII  /R V/
005332  101  114  125                .ASCII  /ALU/
005335  105  000  000                .ASCII  /E/<00><00>
005340  045  116  045        P.ACU:  .ASCII  /%N%/
005343  101  040  124                .ASCII  /A T/
005346  104  122  040                .ASCII  /DR /
005351  126  101  114                .ASCII  /VAL/
005354  125  105  040                .ASCII  /UE /
005357  076  040  060                .ASCII  /> 0/
005362  054  040  124                .ASCII  /, T/
005365  104  122  040                .ASCII  /DR /
005370  075  040  045                .ASCII  /= %/
005373  117  066  045                .ASCII  /06%/
005376  116  000                    .ASCII  /N/<00>
005400  045  116  045        P.ACV:  .ASCII  /%N%/
005403  101  040  124                .ASCII  /A T/
005406  104  122  040                .ASCII  /DR /
005411  126  101  114                .ASCII  /VAL/
005414  125  105  040                .ASCII  /UE /
005417  075  040  060                .ASCII  /= 0/
005422  054  040  124                .ASCII  /, T/
005425  104  122  040                .ASCII  /DR /
005430  075  040  045                .ASCII  /= %/
005433  117  066  045                .ASCII  /06%/
005436  116  000                    .ASCII  /N/<00>
005440  045  116  045        P.ACW:  .ASCII  /%N%/
005443  101  040  124                .ASCII  /A T/
005446  104  122  040                .ASCII  /DR /
005451  126  101  114                .ASCII  /VAL/
005454  125  105  040                .ASCII  /UE /
005457  074  040  060                .ASCII  /< 0/
005462  054  040  124                .ASCII  /, T/
005465  104  122  040                .ASCII  /DR /
005470  075  040  045                .ASCII  /= %/
005473  117  066  045                .ASCII  /06%/
005476  116  000                    .ASCII  /N/<00>
005500  120  117  127        P.ACX:  .ASCII  /POW/
005503  105  122  040                .ASCII  /ER /
```

```
005506  104  105  114                .ASCII  /DEL/
005511  101  131  040                .ASCII  /AY /
005514  055  040  127                .ASCII  /- W/
005517  101  111  124                .ASCII  /AIT/
005522  111  116  107                .ASCII  /ING/
005525  000                        .ASCII  <00>
005526  103  123  122        P.ACY:  .ASCII  /CSR/
005531  040  122  105                .ASCII  / RE/
005534  107  111  123                .ASCII  /GIS/
005537  124  105  122                .ASCII  /TER/
005542  040  106  101                .ASCII  / FA/
005545  111  114  105                .ASCII  /ILE/
005550  104  040  124                .ASCII  /D T/
005553  117  040  122                .ASCII  /O R/
005556  105  123  120                .ASCII  /ESP/
005561  117  116  104                .ASCII  /OND/
005564  000  000                    .ASCII  <00><00>
005566  101  124  040        P.ACZ:  .ASCII  /AT /
005571  111  116  103                .ASCII  /INC/
005574  117  122  122                .ASCII  /ORR/
005577  105  103  124                .ASCII  /ECT/
005602  040  102  122                .ASCII  / BR/
005605  040  114  105                .ASCII  / LE/
005610  126  105  114                .ASCII  /VEL/
005613  000                        .ASCII  <00>
005614  045  116  045        P.ADA:  .ASCII  /%N%/
005617  101  040  040                .ASCII  /A  /
005622  040  040  040                .ASCII  /   /
005625  040  040  111                .ASCII  /  I/
005630  116  124  105                .ASCII  /NTE/
005633  122  122  125                .ASCII  /RRU/
005636  120  124  040                .ASCII  /PT /
005641  101  124  040                .ASCII  /AT /
005644  126  105  103                .ASCII  /VEC/
005647  075  040  045                .ASCII  /= %/
005652  117  063  045                .ASCII  /03%/
005655  101  040  102                .ASCII  /A B/
005660  122  040  114                .ASCII  /R L/
005663  105  126  105                .ASCII  /EVE/
005666  114  075  040                .ASCII  /L= /
005671  045  117  061                .ASCII  /%01/
005674  000  000                    .ASCII  <00><00>
005676  045  101  040        P.ADB:  .ASCII  /%A /
005701  040  040  103                .ASCII  /  C/
005704  117  115  120                .ASCII  /OMP/
005707  114  105  124                .ASCII  /LET/
005712  105  104  040                .ASCII  /ED /
005715  124  105  123                .ASCII  /TES/
005720  124  000                    .ASCII  /T/<00>
005722  122  105  123        P.ADC:  .ASCII  /RES/
005725  105  124  040                .ASCII  /ET /
005730  104  111  104                .ASCII  /DID/
005733  040  116  117                .ASCII  / NO/
```

```
005736  040  103  114                    .ASCII  / CL/
005741  105  101  122                    .ASCII  /EAR/
005744  040  122  111                    .ASCII  / RI/
005747  116  107  040                    .ASCII  /NG /
005752  102  125  106                    .ASCII  /BUF/
005755  106  105  122                    .ASCII  /FER/
005760  000  000                         .ASCII  <00><00>
005762  116  117  116          P.ADD:    .ASCII  /NON/
005765  055  105  130                    .ASCII  /-EX/
005770  111  123  124                    .ASCII  /IST/
005773  105  116  124                    .ASCII  /ENT/
005776  040  115  105                    .ASCII  / ME/
006001  115  117  122                    .ASCII  /MOR/
006004  131  040  124                    .ASCII  /Y T/
006007  105  123  124                    .ASCII  /EST/
006012  040  106  101                    .ASCII  / FA/
006015  111  114  125                    .ASCII  /ILU/
006020  122  105  000                    .ASCII  /RE/<00>
006023  000                             .ASCII  <00>
006024  045  116  045          P.ADE:    .ASCII  /%N%/
006027  101  120  101                    .ASCII  /APA/
006032  103  113  105                    .ASCII  /CKE/
006035  124  040  114                    .ASCII  /T L/
006040  105  116  107                    .ASCII  /ENG/
006043  124  110  040                    .ASCII  /TH /
006046  075  040  045                    .ASCII  /= %/
006051  117  066  045                    .ASCII  /06%/
006054  116  000                         .ASCII  /N/<00>

000000                                  .PSECT  $GLOB$,  D
000000                          RCV.D.LIST::
                                        .BLKW   40
000100                          XMIT.D.LIST::
                                        .BLKW   40
000200                          RCV.BUFFER::
                                        .BLKW   4000
010200                          XMIT.BUFFER::
                                        .BLKW   4000
020200                          PHYS.ADR::
                                        .BLKW   13
020226                          SETUP.BUFFER::
                                        .BLKW   400
021226                          IOP.TABLE::
                                        .BLKW   10
021246                          ETH.STATION.ADR::
                                        .BLKW   6
021262                          STATION.ADR::
                                        .BLKW   4
021272                          PTRN.TABLE::
021272  000                             .BYTE   0
021273  377                             .BYTE   377
```

```
021274      252                                 .BYTE   252
021275      125                                 .BYTE   125
021276      314                                 .BYTE   314
021277      063                                 .BYTE   63
021300      360                                 .BYTE   360
021301      017                                 .BYTE   17
021302                                  TARGET.ADR::
021302      000                                 .BYTE   0
021303      000                                 .BYTE   0
021304      000                                 .BYTE   0
021305      000                                 .BYTE   0
021306      000                                 .BYTE   0
021307      000                                 .BYTE   0
021310      252                                 .BYTE   252
021311      252                                 .BYTE   252
021312      252                                 .BYTE   252
021313      252                                 .BYTE   252
021314      252                                 .BYTE   252
021315      252                                 .BYTE   252
021316      125                                 .BYTE   125
021317      125                                 .BYTE   125
021320      125                                 .BYTE   125
021321      125                                 .BYTE   125
021322      125                                 .BYTE   125
021323      125                                 .BYTE   125
021324      377                                 .BYTE   377
021325      377                                 .BYTE   377
021326      377                                 .BYTE   377
021327      377                                 .BYTE   377
021330      377                                 .BYTE   377
021331      377                                 .BYTE   377
021332      252                                 .BYTE   252
021333      000                                 .BYTE   0
021334      000                                 .BYTE   0
021335      000                                 .BYTE   0
021336      000                                 .BYTE   0
021337      000                                 .BYTE   0
021340      252                                 .BYTE   252
021341      000                                 .BYTE   0
021342      002                                 .BYTE   2
021343      252                                 .BYTE   252
021344      252                                 .BYTE   252
021345      252                                 .BYTE   252
021346      252                                 .BYTE   252
021347      000                                 .BYTE   0
021350      005                                 .BYTE   5
021351      125                                 .BYTE   125
021352      125                                 .BYTE   125
021353      125                                 .BYTE   125
021354      252                                 .BYTE   252
021355      000                                 .BYTE   0
021356      004                                 .BYTE   4
021357      377                                 .BYTE   377
```

```
021360    377                         .BYTE   377
021361    377                         .BYTE   377
021362    252                         .BYTE   252
021363    000                         .BYTE   0
021364    004                         .BYTE   4
021365    000                         .BYTE   0
021366    000                         .BYTE   0
021367    000                         .BYTE   0
021370    252                         .BYTE   252
021371    000                         .BYTE   0
021372    004                         .BYTE   4
021373    030                         .BYTE   30
021374    201                         .BYTE   201
021375    030                         .BYTE   30
021376    252                         .BYTE   252
021377    013                         .BYTE   13
021400    014                         .BYTE   14
021401    015                         .BYTE   15
021402    016                         .BYTE   16
021403    017                         .BYTE   17
021404    377                         .BYTE   377
021405    000                         .BYTE   0
021406    001                         .BYTE   1
021407    002                         .BYTE   2
021410    003                         .BYTE   3
021411    004                         .BYTE   4
021412    125                         .BYTE   125
021413    005                         .BYTE   5
021414    006                         .BYTE   6
021415    007                         .BYTE   7
021416    010                         .BYTE   10
021417    011                         .BYTE   11
021420    000                         .BYTE   0
021421    364                         .BYTE   364
021422    372                         .BYTE   372
021423    104                         .BYTE   104
021424    104                         .BYTE   104
021425    125                         .BYTE   125
021426    314                         .BYTE   314
021427    066                         .BYTE   66
021430    046                         .BYTE   46
021431    047                         .BYTE   47
021432    047                         .BYTE   47
021433    111                         .BYTE   111
021434    063                         .BYTE   63
021435    241                         .BYTE   241
021436    147                         .BYTE   147
021437    273                         .BYTE   273
021440    114                         .BYTE   114
021441    237                         .BYTE   237
021442    353                         .BYTE   353
021443    276                         .BYTE   276
021444    307                         .BYTE   307
```

```
021445     217                          .BYTE   217
021446     063                          .BYTE   63
021447     377                          .BYTE   377
021450                                  .BLKB   22
021472                          BD.PROM.DESCR::
021472  100000                          .WORD   -100000
021474  100000                          .WORD   -100000
021476  000200'                         .WORD   RCV.BUFFER
021500  174000                          .WORD   -4000
021502  000000                          .WORD   0
021504  000000                          .WORD   0
021506  100000                          .WORD   -100000
021510  100000                          .WORD   -100000
021512  010200'                         .WORD   XMIT.BUFFER
021514  174000                          .WORD   -4000
021516  000000                          .WORD   0
021520  000000                          .WORD   0
021522  100000                          .WORD   -100000
021524  020000                          .WORD   20000
021526  000000                          .WORD   0
021530  000000                          .WORD   0
021532                          HWP.TABLE::
                                        .BLKW   1
021534                          SWP.TABLE::
                                        .BLKW   1
021536                          CLK.ADR::
                                        .BLKW   1
021540                          CLK.TYPE::
                                        .BLKW   1
021542                          CLK.VEC::
                                        .BLKW   1
021544                          CLK.CSR::
                                        .BLKW   1
021546                          CLK.START::
                                        .BLKW   1
021550                          CLK.HERTZ::
                                        .BLKW   1
021552                          TICKS:: .BLKW   1
021554  000000                  SECONDS::
                                        .WORD   0
021556  000000                  MINUTES::
                                        .WORD   0
021560                          CANCEL.TIMER::
                                        .BLKW   1
021562                          XBUF.LENGTH::
                                        .BLKW   1
021564                          RBUF.LENGTH::
                                        .BLKW   1
021566                          FREE.MEM.ADR::
                                        .BLKW   1
021570                          MEM.SIZE::
                                        .BLKW   1
021572                          INTERRUPT.FLG::
```

```
                                .BLKW   1
021574                  DEQNA.NO::
                                .BLKW   1
021576                  COUNTER::
                                .BLKW   1
021600                  CHECKSUM::
                                .BLKW   1
021602                  BUF.LENGTH::
                                .BLKW   1
021604                  CSR.WORD::
                                .BLKW   1
021606                  REG.ADR::
                                .BLKW   1
021610                  IOP.DATA::
                                .BLKW   1
021612                  GET.ADR::
                                .BLKW   1
021614                  TMP.IOP.ADR::
                                .BLKW   1
021616                  TMP.REG.DATA::
                                .BLKW   1
021620                  TEMP1::  .BLKW   1
021622                  TEMP2::  .BLKW   1
021624                  TEMP3::  .BLKW   1
021626                  TEMP4::  .BLKW   1
021630                  TEMP5::  .BLKW   1
021632                  TEMP6::  .BLKW   1
021634                  TEMP7::  .BLKW   1
021636                  TEMP8::  .BLKW   1
021640                  TEMP9::  .BLKW   1
021642                  P1::     .BLKW   1
021644                  P2::     .BLKW   1
021646                  P3::     .BLKW   1
021650                  P4::     .BLKW   1
021652                  P5::     .BLKW   1
021654                  TBYTE1::.BLKB   1
021655                  TBYTE2::.BLKB   1
021656                  TBYTE3::.BLKB   1
021657                  TBYTE4::.BLKB   1
021660                  TADR1::  .BLKW   1
021662                  TADR2::  .BLKW   1

                        .GLOBL  L$SOFT, T$PTHV, L$RPT, L$INIT
                        .GLOBL  L$CLEAN, L$LAST, L$HARD, L$DVTYP
                        .GLOBL  L$DESC, L$DU, L$AU, L$AUTO, T1
                        .GLOBL  T2, T3, T4, T5, T6, T7

100000                  BIT15=:         -100000
040000                  BIT14=:         40000
020000                  BIT13=:         20000
010000                  BIT12=:         10000
```

```
004000                  BIT11==         4000
002000                  BIT10==         2000
001000                  BIT09==         1000
000400                  BIT08==         400
000200                  BIT07==         200
000100                  BIT06==         100
000040                  BIT05==         40
000020                  BIT04==         20
000010                  BIT03==         10
000004                  BIT02==         4
000002                  BIT01==         2
000001                  BIT00==         1
001000                  BIT9==          1000
000400                  BIT8==          400
000200                  BIT7==          200
000100                  BIT6==          100
000040                  BIT5==          40
000020                  BIT4==          20
000010                  BIT3==          10
000004                  BIT2==          4
000002                  BIT1==          2
000001                  BIT0==          1
000040                  EF.START==      40
000037                  EF.RESTART==    37
000036                  EF.CONTINUE==   36
000035                  EF.NEW==        35
000034                  EF.PWR==        34
000340                  PRI07==         340
000300                  PRI06==         300
000240                  PRI05==         240
000200                  PRI04==         200
000140                  PRI03==         140
000100                  PRI02==         100
000040                  PRI01==         40
000000                  PRI00==         0
000004                  EVL==           4
000010                  LOT==           10
000020                  ADR==           20
000040                  IDU==           40
000100                  ISR==           100
000200                  UAM==           200
000400                  BOE==           400
001000                  PNT==           1000
002000                  PRI==           2000
004000                  IXE==           4000
010000                  IBE==           10000
020000                  IER==           20000
040000                  LOE==           40000
100000                  HOE==           -100000
000142'                 L$ERRTBL==      ERRTYP
000164'                 L$SW==          L$SWLEN*2
000154'                 L$HW==          L$HWLEN*2
000011'                 L$DEPO==        L$REV*1
```

H5


```
000154'                 HWPTBL--        L$HWLEN*2
000164'                 SWPTBL--        L$SWLEN*2
000000'                 DESCR.LIST--    RCV.D.LIST
000200'                 DATA.BUFFER--   RCV.BUFFER
000000'                 QST01--         P.AAA
000030'                 QST02--         P.AAB
000060'                 QST03--         P.AAC
000110'                 QST04--         P.AAD
000162'                 QST05--         P.AAE
000224'                 QST06--         P.AAF
000226'                 QST07--         P.AAG
000230'                 QST08--         P.AAH
000232'                 QST09--         P.AAI
000234'                 DBM01--         P.AAJ
000272'                 DBM02--         P.AAK
000316'                 DBM03--         P.AAL
000344'                 DBM04--         P.AAM
000374'                 DBM05--         P.AAN
000430'                 DBM06--         P.AAO
000464'                 DBM33--         P.AAP
000524'                 DBM34--         P.AAQ
000562'                 DBM35--         P.AAR
000634'                 DBM36--         P.AAS
000700'                 DBM37--         P.AAT
000744'                 DBM38--         P.AAU
001026'                 DBM39--         P.AAV
001072'                 DBM40--         P.AAW
001136'                 DBM41--         P.AAX
001222'                 DBM42--         P.AAY
001250'                 DBM43--         P.AAZ
001334'                 DBM44--         P.ABA
001422'                 DBM45--         P.ABB
001460'                 DBM46--         P.ABC
001522'                 ERR01--         P.ABD
001570'                 ERR02--         P.ABE
001630'                 ERR03--         P.ABF
001664'                 ERR04--         P.ABG
001740'                 ERR05--         P.ABH
002022'                 ERR06--         P.ABI
002026'                 ERR07--         P.ABJ
002032'                 ERR08--         P.ABK
002036'                 E0001--         P.ABL
002072'                 E0101--         P.ABM
002154'                 E0201--         P.ABN
002240'                 E0202--         P.ABO
002324'                 E0301--         P.ABP
002406'                 E0401--         P.ABQ
002472'                 E0501--         P.ABR
002562'                 E0502--         P.ABS
002650'                 E0503--         P.ABT
002746'                 E0601--         P.ABU
003032'                 E0801--         P.ABV
003116'                 MSG01--         P.ABW
```

```
003200'                         MSG02**         P.ABX
003266'                         MSG03**         P.ABY
003372'                         MSG04**         P.ABZ
003464'                         MSG05**         P.ACA
003556'                         MSG06**         P.ACB
003650'                         MSG07**         P.ACC
003742'                         MSG08**         P.ACD
004034'                         MSG09**         P.ACE
004126'                         MSG10**         P.ACF
004200'                         MSG11**         P.ACG
004254'                         MSG12**         P.ACH
004332'                         MSG13**         P.ACI
004414'                         MSG14**         P.ACJ
004500'                         MSG15**         P.ACK
004562'                         MSG16**         P.ACL
004646'                         MSG17**         P.ACM
004734'                         MSG18**         P.ACN
004760'                         MSG19**         P.ACO
005046'                         MSG20**         P.ACP
005136'                         MSG21**         P.ACQ
005216'                         MSG22**         P.ACR
005302'                         MSG23**         P.ACS
005310'                         MSG.1TDR**      P.ACT
005340'                         MSG.2TDR**      P.ACU
005400'                         MSG.3TDR**      P.ACV
005440'                         MSG.4TDR**      P.ACW
005500'                         MSG.PWR**       P.ACX
005526'                         MSG.CSR**       P.ACY
005566'                         MSG.BR**        P.ACZ
005614'                         MSG.INI**       P.ADA
005676'                         MSG.END**       P.ADB
005722'                         MSG.CLR**       P.ADC
005762'                         MSG.NXM**       P.ADD
006024'                         MSG.LEN**       P.ADE


;                               PSECT SUMMARY
;
;       Psect Name              Words     Attributes
;         $CODE$                   63       RO  .  I  .  LCL,  REL,  CON
;         $GLOB$                 4570       RW  .  D  .  LCL,  REL,  CON
;         $PLIT$                 1559       RO  .  D  .  LCL,  REL,  CON


;                           Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[MAZURCZYK.SDC]QNALIB.L16;2
```

```
;                                       157       65       41       12        00:00.1



;                                 COMMAND QUALIFIERS

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

; Size:          0 code + 6192 data words
; Run Time:         00:22.7
; Elapsed Time:     01:06.8
; Lines/CPU Min:     7019
; Lexemes/CPU-Min: 35327
; Memory Used:  223 pages
; Compilation Complete
```

```
;    0001  0     MODULE ZQNA2 (%TITLE 'CZQNAA0 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
```

L5


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

```
;    1515  1      !++
;    1516  1      !       EXTERNAL DATA USED BY THIS MODULE
;    1517  1      ! -
;    1518  1
;    1519  1      EXTERNAL
;    1520  1
;    1521  1      !++
;    1522  1      !       COMMUNICATION AREA DECLARATIONS
;    1523  1      !--
;    1524  1
;    1525  1
;    1526  1      !++
;    1527  1      !       COMMUNICATION AREA DECLARATIONS
;    1528  1      !--
;    1529  1
;    1530  1          IOP_TABLE        : VECTOR [ 8, WORD ],
;    1531  1
;    1532  1
;    1533  1      !++
;    1534  1      !       HARDWARE P-TABLE STORAGE DECLARATIONS
;    1535  1      !--
;    1536  1
;    1537  1          HWP_TABLE    : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;    1538  1
;    1539  1
;    1540  1      !++
;    1541  1      !       SOFTWARE P-TABLE STORAGE DECLARATIONS
;    1542  1      !--
;    1543  1
;    1544  1          SWP_TABLE    : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    1545  1
;    1546  1
;    1547  1      !++
;    1548  1      !       SYSTEM CLOCK STORAGE DECLARATIONS
;    1549  1      !--
;    1550  1
;    1551  1          TICKS                : WORD,                 ! CLOCK RATE
;    1552  1          SECONDS              : WORD,                 ! STORE SECONDS
;    1553  1          MINUTES              : WORD,                 ! STORE MINUTES
;    1554  1          CANCEL_TIMER         : WORD,                 !
;    1555  1          CLK_VEC              : WORD,                 ! CLOCK INTERRUPT VECTOR ADR
;    1556  1          CLK_CSR              : WORD,                 ! STORE CSR ADDR FOR CLOCK HERE
;    1557  1          CLK_START            : WORD,                 ! STORE CLOCK START VALUE
;    1558  1          CLK_HERTZ            : WORD,                 ! TOTAL # OF CLOCK INTERRUPTS
;    1559  1          CLK_ADR              : WORD,                 ! LOC. TO RETURN CLOCK ADDR.
;    1560  1          CLK_TYPE             : WORD,                 ! TYPE OF CLOCK ON SYSTEM
;    1561  1                                                       ! (0=NONE,-1=L-CLOCK,1=P_CLOCK)
;    1562  1      !++
;    1563  1      !       MISCELLANEOUS DATA DECLARATIONS
;    1564  1      !--
;    1565  1
;    1566  1          FREE_MEM_ADR         : WORD,                 ! FREE MEMORY BEGIN ADR
;    1567  1          MEM_SIZE             : WORD,                 ! FREE MEMORY SIZE
```

```
;    1568  1          INTERRUPT_FLG       : WORD,                 ! 1 = INTERRUPT OCCURED
;    1569  1
;    1570  1          REG_ADR             : REF REG_STR FIELD ( IOP_FIELDS ),
;    1571  1          IOP_DATA            : REF REG_STR FIELD ( IOP_FIELDS ),
;    1572  1          GET_ADR             : REF ADR_STR FIELD ( IOP_FIELDS ),
;    1573  1
;    1574  1      !++
;    1575  1      !   TEMPORARY STORAGE DATA DECLARATIONS
;    1576  1      !--
;    1577  1
;    1578  1          TMP_IOP_ADR         : WORD,                 ! I/O PAGE REGISTER ADDRESS
;    1579  1          TMP_REG_DATA        : WORD,                 ! I/O PAGE REG CONTENTS
;    1580  1          TEMP1               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1581  1          TEMP2               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1582  1          TEMP3               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1583  1          TEMP4               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1584  1          TEMP5               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1585  1          TEMP6               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1586  1          TEMP7               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1587  1          TEMP8               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1588  1          TEMP9               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1589  1
```

```
;    1590  1
;    1591  1      !++
;    1592  1      !       DEBUG MESSAGES DECLARED EXTERMALLY
;    1593  1      !--
;    1594  1
;    1595  1          DBM01,DBM02,DBM03,DBM04,DBM05,DBM06,DBM33,DBM34,DBM35,DBM36,DBM37,
;    1596  1          DBM38,DBM39,DBM42,
;    1597  1
;    1598  1          ERR01,ERR02,ERR03,
;    1599  1
;    1600  1      !++
;    1601  1      !       DIAGNOSTIC ERROR MESSAGES DECLARED EXTERNALLY
;    1602  1      !--
;    1603  1
;    1604  1          MSG_PWR,
;    1605  1
;    1606  1      !++
;    1607  1      !       QUESTIONS DECLARED EXTERNALLY
;    1608  1      !-
;    1609  1
;    1610  1          QST01, QST02, QST03, QST04, QST05, QST07, QST08, QST09;
;    1611  1
;    1612  1
```

```
;    1613  1     %SBTTL 'TYPE AND DESCRIPTION'
;    1614  1
;    1615  1     !++
;    1616  1     !       NAMES OF DEVICES SUPPORTED BY PROGRAM
;    1617  1     !--
;    1618  1
;    1619  1     EQUALS;
;    1620  1     DEVTYP (%ASCIZ'DEQNA/M7504');
;    1621  1
;    1622  1     !++
;    1623  1     !       TEST DESCRIPTION
;    1624  1     !--
;    1625  1
;    1626  1     DESCRIPT (%ASCIZ'DEQNA FUNCTIONAL TEST');
;    1627  1
```

```
;    1628  1     %SBTTL 'HARDWARE PARAMETER CODING SECTION'
;    1629  1
;    1630  1     !++
;    1631  1     !       THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;    1632  1     !       THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    1633  1     !       MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    1634  1     !       INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    1635  1     !       MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    1636  1     !       WITH THE OPERATOR.
;    1637  1     !--
;    1638  1
;    1639  1     !++
;    1640  1     !       THIS CODE IS USED BY THE SUPERVISOR TO INTERROGATE THE OPERATOR
;    1641  1     !       FOR DEVICE INFORMATION TO PUT IN THE P-TABLE.  THIS CODE IS USED
;    1642  1     !       IN CONJUNCTION WITH THE DEFAULT P-TABLE TEMPLATE.  THE MACROS
;    1643  1     !       USED IN THIS SECTION ARE "GPRMD", "GPRMA".
;    1644  1     !++
;    1645  1
;    1646  1     BGNHRD;
;    1647  1     GPRMA (QST01, %O'0', 0, %O'174440', %O'174460', YES, 1);  ! I/O PAGE  ADDRESS ?
;    1648  1     GPRMA (QST02, %O'2', 0, %O'700',    %O'704',    YES, 1);  ! INTERRUPT VECTOR ADDR ?
;    1649  1     ENDHRD;
;    1650  1
;    1651  1
```

```
;    1652  1      %SBTTL 'SOFTWARE PARAMETER CODING SECTION'
;    1653  1
;    1654  1      !++
;    1655  1      !       THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;    1656  1      !       THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    1657  1      !       MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    1658  1      !       INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    1659  1      !       MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    1660  1      !       WITH THE OPERATOR.
;    1661  1      !--
;    1662  1      !
;    1663  1      BGNSFT;
;    1664  1      !GPRML (QST05, %O'0', %O'1', YES, 1);     ! DO YOU WANT TO DO MANUAL INTERVENTION
;    1665  1      ENDSFT;
;    1666  1
;    1667  1
```

```
;    1668  1     %SBTTL 'REPORT CODING SECTION'
;    1669  1
;    1670  1     !++
;    1671  1     !
;    1672  1     !       THE REPORT CODING SECTION CONTAINS THE
;    1673  1     !       "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;    1674  1     !--
;    1675  1
;    1676  2     BGNRPT;
;    1677  2     TEMP1 = 1;
;    1678  2     !PRINTF (DBM02);                                    !'REPORT CODE SECTION'
;    1679  2
;    1680  2     !++
;    1681  2     !       THIS SECTION CONTAINS THE CODE FOR PRINTING
;    1682  2     !       STATISTICAL INFORMATION GATHERED BY THE DIAGNOSTIC.  IT IS
;    1683  2     !       EXECUTED BY THE OPERATOR COMMAND "PRINT" OR BY THE MACRO CALL
;    1684  2     !       "DORPT".  USE THE PRINTS MACRO TO PRINT THE INFORMATION.
;    1685  2     !       USE FORMAT STATEMENTS AS IN THE PRINTB/PRINTX MACROS.  IT IS
;    1686  2     !       THE PROGRAMMER'S RESPONSIBILTY TO DEVISE AND IMPLEMENT THE
;    1687  2     !       FORM AND CONTENT OF THE STATISTICS.
;    1688  2     !--
;    1689  2
;    169C  1     ENDRPT;
```

```
                                        .TITLE  ZQNA2 CZQNAAO 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                              L$NDSFT::
                                            .BLKW   1


                                            .GLOBL  RESET.DEQNA, IOP.TABLE, HWP.TABLE
                                            .GLOBL  SWP.TABLE, TICKS, SECONDS, MINUTES
                                            .GLOBL  CANCEL.TIMER, CLK.VEC, CLK.CSR
                                            .GLOBL  CLK.START, CLK.HERTZ, CLK.ADR
                                            .GLOBL  CLK.TYPE, FREE.MEM.ADR, MEM.SIZE
                                            .GLOBL  INTERRUPT.FLG, REG.ADR, IOP.DATA
                                            .GLOBL  GET.ADR, TMP.IOP.ADR, TMP.REG.DATA
                                            .GLOBL  TEMP1, TEMP2, TEMP3, TEMP4, TEMP5
                                            .GLOBL  TEMP6, TEMP7, TEMP8, TEMP9, DBM01
                                            .GLOBL  DBM02, DBM03, DBM04, DBM05, DBM06
                                            .GLOBL  DBM33, DBM34, DBM35, DBM36, DBM37
                                            .GLOBL  DBM38, DBM39, DBM42, ERR01, ERR02
                                            .GLOBL  ERR03, MSG.PWR, QST01, QST02, QST03
                                            .GLOBL  QST04, QST05, QST07, QST08, QST09

100000                              BIT15== -100000
040000                              BIT14== 40000
020000                              BIT13== 20000
010000                              BIT12== 10000
004000                              BIT11== 4000
002000                              BIT10== 2000
001000                              BIT09== 1000
000400                              BIT08== 400
000200                              BIT07== 200
000100                              BIT06== 100
000040                              BIT05== 40
000020                              BIT04== 20
000010                              BIT03== 10
000004                              BIT02== 4
000002                              BIT01== 2
000001                              BIT00== 1
001000                              BIT9==  1000
000400                              BIT8==  400
000200                              BIT7==  200
000100                              BIT6==  100
000040                              BIT5==  40
000020                              BIT4==  20
000010                              BIT3==  10
000004                              BIT2==  4
000002                              BIT1==  2
000001                              BIT0==  1
```

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


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

000000  012737  000001  000000G  LRPT:  MOV     #1,TEMP1                        ;                       1677
000006  000207                         RTS     PC                              ;                       1665

; 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                         ;                       1677
000004  104425                         TRAP    25
000006  000207                         RTS     PC

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

```
;    1695  1     %SBTTL 'INITIALIZE SECTION'
;    1696  1
;    1697  1     !++
;    1698  1     !       THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
;    1699  1     !       AT THE BEGINNING OF EACH PASS.
;    1700  1     ! -
;    1701  1
;    1702  2     BGNINIT;
;    1703  2
;    1704  2     !++
;    1705  2     !       THE INITIALIZE CODE IS EXECUTED UNDER FIVE CONDITIONS.  THERE
;    1706  2     !       ARE SUPERVISOR EVENT FLAGS THAT ARE USED TO LET THE
;    1707  2     !       DIAGNOSTIC KNOW UNDER WHICH CONDITION THE EXECUTION IS TAKING
;    1708  2     !       PLACE.  THE EVENT FLAGS ARE READ USING THE "READEF" MACRO.
;    1709  2     !       THE CONDITIONS UNDER WHICH THE INIT CODE IS EXECUTED AND THE
;    1710  2     !       CORRRESPONDING EVENT FLAGS ARE:
;    1711  2     !               START COMMAND           EF.START
;    1712  2     !               RESTART COMMAND         EF.RESTART
;    1713  2     !               CONTINUE COMMAND        EF.CONTINUE
;    1714  2     !               POWERDOWN/POWERUP       EF.PWR
;    1715  2     !               NEW PASS                EF.NEW
;    1716  2     !       EXAMPLE OF EVENT FLAG USE:
;    1717  2     !               IF READEF(EF.START) THEN
;    1718  2     !                  START_FLAG = 1;
;    1719  2     !       DURING THE INIT CODE, USE THE "GPHARD" MACRO TO OBTAIN P-TABLE
;    1720  2     !       INFORMATION FOR DEVICE TESTING.  GET ONE UNIT'S INFORMATION IF
;    1721  2     !       THIS IS A SEQUENTIAL DIAGNOSTIC. NUMBER OF UNITS AVAILABLE IS IN
;    1722  2     !       A HEADER LOCATION: "L$UNIT".
;    1723  2     !--
;    1724  2
;    1725  2     LOCAL
;    1726  2       START_FLAG,                           ! SET IF THIS PASS IS A START
;    1727  2       DELAY_MULT;                           ! CONTAINS DELAY FACTOR
;    1728  2
;    1729  2     ! X_IDT();
;    1730  2
;    1731  2     !PRINTF (DBM01);
;    1732  2     SETPRI (PRI07);                         ! PRIORITY 7 - NO INTERRUPTS ALLOWED
;    1733  2     START_FLAG = CLEAR_FLG;                 ! CLEAR FLAG BEFORE TESTING IT
;    1734  2
;    1735  2     IF READEF (EF_PWR)                      ! ARE WE HERE BECAUSE OF POWER FAIL?
;    1736  2       THEN
;    1737  3         BEGIN
;    1738  3           PRINTF (MSG_PWR);                 ! "POWER DELAY   WAITING"
;    1739  3
;    1740  3           INCR COUNT FROM 0 TO 60 DO        ! WAIT APPROX. 60 SECONDS
;    1741  4             BEGIN
;    1742  4               DELAY_MULT = 10000;
;    1743  4               DELAY (,DELAY_MULT);
;    1744  4               BREAK;                        ! BREAK FOR APT
;    1745  3             END;
;    1746  2         END;
;    1747  2
```

```
;    1748  2      IF READEF (EF_START)                    ! IS THIS A START ?
;    1749  2        THEN
;    1750  3          BEGIN
;    1751  3      !       PRINTF (DBM02);                 ! 'START COMMAND'
;    1752  3           START_FLAG = TRUE;
;    1753  2          END;
;    1754  2
;    1755  2      IF READEF (EF_RESTART)                  ! IS THIS A RESTART ?
;    1756  2        THEN
;    1757  2      !     PRINTF (DBM03);                   ! 'RESTART COMMAND'
;    1758  2
;    1759  2      IF READEF (EF_CONTINUE)                 ! IS THIS A CONTINUE ?
;    1760  2        THEN
;    1761  2      !     PRINTF (DBM04);                   !  CONTINUE COMMAND'
;    1762  2
;    1763  2      !++
;    1764  2      !
;    1765  2      !       MAKE SURE THAT ONLY ONE (1) DEQNA HAS BEEN SPECIFIED.
;    1766  2      !
;    1767  2      !--
;    1768  2
;    1769  2      !IF .L$UNIT GTRU 1
;    1770  2      !THEN                                   ! MORE THAN 1 UNIT ?
;    1771  2      !    BEGIN
;    1772  2      !    PRINTF (ERR03);                   ! 'ERROR - TOO MANY UNITS'
;    1773  2      !    DOCLN;                            ! RETURN TO SUPERVISOR AND CLEAN UP
;    1774  2      !    END;
;    1775  2
;    1776  2      !++
;    1777  2      !
;    1778  2      !       CHECK TO MAKE SURE THERE IS A CLOCK ON THE SYSTEM.  IF NO_CLOCK,
;    1779  2      !       ABORT TO SUPERVISOR. OTHERWISE, DETERMINE WHETHER CLOCK IS AN L OR
;    1780  2      !       P CLOCK, GET IT'S PARAMETERS.
;    1781  2      !
;    1782  2      !--
;    1783  2
;    1784  2      CLK_TYPE = NO_CLOCK;                    ! SEt FLAG FOR NO CLOCK
;    1785  2
;    1786  3      IF CLOCK (P, CLK_ADR)                   ! IS THERE A P_CLOCK ?
;    1787  2        THEN
;    1788  3          BEGIN
;    1789  3            CLK_TYPE = P_CLOCK;               ! SET THE FLAG FOR P_CLOCK
;    1790  3            CLK_CSR = ..CLK_ADR;              ! SAVE THE CSR ADDRESS
;    1791  3            CLK_HERTZ = .(.CLK_ADR + 6);      ! GET THE CLOCK RATE
;    1792  3            CLK_START = %O'105';              ! SAVE THE STARTTING CLOCK VALUE
;    1793  3      !       PRINTF (DBM36);                 ! 'P-CLOCK ON THE SYSTEM'
;    1794  3          END
;    1795  2        ELSE
;    1796  3          BEGIN
;    1797  3
;    1798  4            IF CLOCK (L, CLK_ADR)           ! IS THERE AN L_CLOCK ?
;    1799  3              THEN
;    1800  4                BEGIN
```

```
;    1801  4                CLK_TYPE = L_CLOCK;          ! SET THE FLAG FOR L_CLOCK
;    1802  4                CLK_CSR = ..CLK_ADR;         ! SAVE THE CSR ADDRESS
;    1803  4                CLK_HERTZ = .(.CLK_ADR + 6);! GET THE CLOCK RATE
;    1804  4                CLK_START = %O'100';         ! SAVE THE STARTING CLOCK VALUE
;    1805  4      !          PRINTF (DBM37);              ! 'L-CLOCK ON THE SYSTEM'
;    1806  3              END;
;    1807  2          END;
;    1808  2
;    1809  2      IF .CLK_TYPE EQLU NO_CLOCK               ! IF NO CLOCK WAS FOUND
;    1810  2        THEN
;    1811  3          BEGIN
;    1812  3            PRINTF (ERR01);                    ! 'NO CLOCK WAS FOUND ON THE SYSTEM'
;    1813  3            DOCLN;                             ! ABORT THE PROGRAM
;    1814  3          END
;    1815  2        ELSE
;    1816  3          BEGIN
;    1817  3            CLK_VEC = .(.CLK_ADR + 4);       ! GET CLOCK VECTOR ADDRESS
;    1818  3            SETVEC (.CLK_VEC, CLK_INT, PRIO5);
;    1819  3                                             ! SET VECTOR & INTERRUPT SERVICE ADDR.
;    1820  3      !       PRINTF (DBM38,..CLK_VEC,..CLK_ADR);       ! 'CLOCK VECTOR AND CLOCK ADR'
;    1821  2          END;
;    1822  2
;    1823  2      !++
;    1824  2      !
;    1825  2      !       DETERMINE THE FREE MEMORY STARTING ADDRESS AND SIZE
;    1826  2      !
;    1827  2      !--
;    1828  2
;    1829  2      MEMORY (FREE_MEM_ADR);                   ! FIND FREE MEMORY STARTING ADR
;    1830  2      MEM_SIZE = .FREE_MEM_ADR;                ! DETERMINE FREE MEMORY SIZE
;    1831  2
;    1832  2      !++
;    1833  2      !
;    1834  2      !       CLEAR HARDWARE P-TABLE ON A START BEFORE DOING THE GPHARDS
;    1835  2      !
;    1836  2      !--
;    1837  2
;    1838  2      IF .START_FLAG OR READEF (EF_NEW) OR READEF (EF_CONTINUE)
;    1839  2        THEN                                   ! IF THIS IS A START
;    1840  3          BEGIN
;    1841  3            LOCAL TABLE_POINTER;
;    1842  3
;    1843  3            INCR INDEX FROM 0 TO HWP_SIZE BY 2 DO     ! ZERO OUT THE TABLES
;    1844  3              (HWP_TABLE + .INDEX) = 0;
;    1845  3
;    1846  3           !++
;    1847  3           !
;    1848  3           !   GET BASE ADDRESS OF HARDWARE P-TABLE AND DEQNA I/O PAGE
;    1849  3           !
;    1850  3           !--
;    1851  3
;    1852  3            IF GPHARD ( 0, TABLE_POINTER ) NEQU 0             ! GET P-TABLE ADDRESS
;    1853  3              THEN
```

```
;     1854  4              BEGIN
;     1855  4                IOP_DATA  = .HWP_TABLE [ ADDR ];
;     1856  4                HWP_TABLE = .TABLE_POINTER;                 ! SAVE HW P-TABLE ADDRESS
;     1857  4                REG_ADR   = .HWP_TABLE [ ADDR ];            ! SAVE I/O PAGE BASE ADDRESS
;     1858  4                GET_ADR   = .HWP_TABLE [ ADDR ];            ! SAVE I/O PAGE BASE ADDRESS
;     1859  4      !         PRINTF (DBM39, .HWP_TABLE[ADDR], .HWP_TABLE[VEC], .HWP_TABLE[BRL]);
;     1860  4                TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;     1861  4                INCR INDEX FROM 0 TO 7 DO
;     1862  5                  BEGIN
;     1863  5                    IOP_TABLE [ .INDEX ] = .TMP_IOP_ADR;
;     1864  5                    TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;     1865  5      !             PRINTF ( DBM42, .IOP_TABLE [ .INDEX ] );
;     1866  4                  END;
;     1867  3              END;
;     1868  2          END;
;     1869  2      RETURN;
;     1870  1      ENDINIT;


                                        .GLOBL  L$DLY


                                        .SBTTL  LINIT INITIALIZE SECTION
000000  004137  000000G         LINIT:  JSR     R1,$SAVE4                       ;                       1690
000004  005746                          TST     -(SP)
000006  012700  000340                  MOV     #340,R0                         ;                       1732
000012  104441                          TRAP    41
000014  005004                          CLR     R4                              ; START.FLAG            1733
000016  012700  000034                  MOV     #34,R0                          ;                       1735
000022  104447                          TRAP    47
000024  103027                          BHIS    6$
000026  012746  000000G                 MOV     #MSG.PWR,-(SP)                  ;                       1738
000032  012746  000001                  MOV     #1,-(SP)
000036  010600                          MOV     SP,R0                           ; SP,*
000040  104417                          TRAP    17
000042  012702  000075                  MOV     #75,R2                          ; *,COUNT               1740
000046  012703  023420          1$:     MOV     #23420,R3                       ; *,DELAY.MULT          1742
000052  010301                          MOV     R3,R1                           ; DELAY.MULT,$$TMP2     1743
000054  001410                  2$:     BEQ     5$
000056  013700  000000G                 MOV     L$DLY,R0                        ; *,$$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,*               1740
000102  022626                          CMP     (SP)+,(SP)+                     ;                       1737
000104  012700  000040          6$:     MOV     #40,R0                          ;                       1748
000110  104447                          TRAP    47
000112  103002                          BHIS    7$
000114  012704  000001                  MOV     #1,R4                           ; *,START.FLAG          1752
000120  012700  000037          7$:     MOV     #37,R0                          ;                       1755
```

M6


```
000124  104447                          TRAP    47
000126  103006                          BHIS    8$
000130  012700  000036                  MOV     #36,R0                  ;                               1759
000134  104447                          TRAP    47
000136  103002                          BHIS    8$
000140  005037  000000G                 CLR     CLK.TYPE                ;                               1784
000144  012700  000120          8$:     MOV     #120,R0                 ;                               1786
000150  104462                          TRAP    62
000152  103016                          BHIS    9$
000154  010037  000000G                 MOV     R0,CLK.ADR              ; R0,*
000160  012737  000001  000000G         MOV     #1,CLK.TYPE             ;                               1789
000166  011037  000000G                 MOV     (R0),CLK.CSR            ; CLK.ADR,*                     1790
000172  016037  000006  000000G         MOV     6(R0),CLK.HERTZ         ;                               1791
000200  012737  000105  000000G         MOV     #105,CLK.START          ;                               1792
000206  000421                          BR      10$                     ;                               1786
000210  012700  000114          9$:     MOV     #114,R0                 ;                               1798
000214  104462                          TRAP    62
000216  103015                          BHIS    10$
000220  010037  000000G                 MOV     R0,CLK.ADR              ; R0,*
000224  012737  000001  000000G         MOV     #1,CLK.TYPE             ;                               1801
000232  011037  000000G                 MOV     (R0),CLK.CSR            ; CLK.ADR,*                     1802
000236  016037  000006  000000G         MOV     6(R0),CLK.HERTZ         ;                               1803
000244  012737  000100  000000G         MOV     #100,CLK.START          ;                               1804
000252  005737  000000G         10$:    TST     CLK.TYPE                ;                               1809
000256  001010                          BNE     11$
000260  012746  000000G                 MOV     #ERR01,-(SP)            ;                               1812
000264  012746  000001                  MOV     #1,-(SP)
000270  010600                          MOV     SP,R0                   ; SP,*
000272  104417                          TRAP    17
000274  104444                          TRAP    44
000276  000417                          BR      12$                     ;                               1809
000300  013700  000000G         11$:    MOV     CLK.ADR,R0              ;                               1817
000304  016037  000004  000000G         MOV     4(R0),CLK.VEC
000312  012746  000240                  MOV     #240,-(SP)              ;                               1818
000316  012746  000000V                 MOV     #CLK.INT,-(SP)
000322  013746  000000G                 MOV     CLK.VEC,-(SP)
000326  012746  000003                  MOV     #3,-(SP)
000332  104437                          TRAP    37
000334  022626                          CMP     (SP)+,(SP)+             ;                               1816
000336  104431                          TRAP    31                      ;                               1829
000340  010037  000000G         12$:    MOV     R0,FREE.MEM.ADR
000344  010037  000000G                 MOV     R0,MEM.SIZE             ; FREE.MEM.ADR,*                1830
000350  006004                          ROR     R4                      ; START.FLAG                    1838
000352  103410                          BLO     13$
000354  012700  000035                  MOV     #35,R0
000360  104447                          TRAP    47
000362  103404                          BCS     13$
000364  012700  000036                  MOV     #36,R0
000370  104447                          TRAP    47
000372  103044                          BHIS    16$
000374  005000                  13$:    CLR     R0                      ; INDEX                         1843
000376  005060  000000G         14$:    CLR     HWP.TABLE(R0)           ; *(INDEX)                      1844
000402  062700  000002                  ADD     #2,R0                   ; *,INDEX                       1843
```

```
000406  020027  000002                          CMP     R0,#2                           ; INDEX,*
000412  003771                                  BLE     14$
000414  005000                                  CLR     R0                              ;                                       1852
000416  104442                                  TRAP    42
000420  005700                                  TST     R0                              ; TABLE.POINTER
000422  001430                                  BEQ     16$
000424  017737  000000G 000000G                 MOV     @HWP.TABLE,IOP.DATA             ;                                       1855
000432  010037  000000G                         MOV     R0,HWP.TABLE                    ; TABLE.POINTER,*                       1856
000436  011000                                  MOV     (R0),R0                         ; HWP.TABLE,*                           1857
000440  010037  000000G                         MOV     R0,REG.ADR
000444  010037  000000G                         MOV     R0,GET.ADR                      ;                                       1858
000450  010037  000000G                         MOV     R0,TMP.IOP.ADR                  ;                                       1860
000454  005000                                  CLR     R0                              ; INDEX                                 1861
000456  013760  000000G 000000G         15$:    MOV     TMP.IOP.ADR,IOP.TABLE(R0)       ; *,*(INDEX)                            1863
000464  062737  000002  000000G                 ADD     #2,TMP.IOP.ADR                  ;                                       1864
000472  062700  000002                          ADD     #2,R0                           ; *,INDEX                               1861
000476  020027  000016                          CMP     R0,#16                          ; INDEX,*
000502  003765                                  BLE     15$
000504  062706  000006                 16$:    ADD     #6,SP                           ;                                       1690
000510  000207                                  RTS     PC

; Routine Size:  165 words,    Routine Base:  AA$CODE$ + 0020
; Maximum stack depth per invocation:  12 words


                                        .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  000020'         L$INIT::JSR     PC,LINIT                        ;                                       1869
000004  104411                          TRAP    11
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  AA$CODE$ + 0532
; Maximum stack depth per invocation:  2 words


;    1871  1
;    1872  1
;    1873  1
```

```
;    1874  1     %SBTTL 'AUTODROP SECTION'
;    1875  1
;    1876  1     !++
;    1877  1     !
;    1878  1     !       THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;    1879  1     !       THE "ADR" FLAG WAS SET.  THE UNIT UNDER TEST IS CHECKED TO
;    1880  1     !       SEE IF IT WILL RESPOND.  IF IT DOESN'T IT IS IMMEDIATELY
;    1881  1     !       DROPPED FROM TESTING.
;    1882  1     !
;    1883  1     !--
;    1884  1
;    1885  2     BGNAUTO;
;    1886  2     !PRINTF (DBM05);
;    1887  2     RETURN;
;    1888  1     ENDAUTO;


                                        .SBTTL  LAUTO AUTODROP SECTION
000000  000207                  LAUTO:  RTS     PC                      ;                                       1870

; Routine Size:  1 words,       Routine Base:  AA$CODE$ + 0542
; Maximum stack depth per invocation:  0 words


                                        .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  000542'         L$AUTO::JSR     PC,LAUTO                ;                                       1887
000004  104461                          TRAP    61
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0544
; Maximum stack depth per invocation:  2 words


;    1889  1
;    1890  1
```

```
;    1891  1      %SBTTL 'CLEANUP CODING SECTION
;    1892  1
;    1893  1      !++
;    1894  1      !
;    1895  1      !       THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
;    1896  1      !       AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
;    1897  1      !
;    1898  1      !--
;    1899  1
;    1900  2      BGNCLN;
;    1901  2      !PRINTF (DBM06);                                ! 'CLEAN-UP CODE SECTION'
;    1902  2      RESET_DEQNA ( );
;    1903  2      .CLK_CSR = ZERO;                          ! TURN OFF THE CLOCK
;    1904  2      RETURN;
;    1905  2
;    1906  2      !++
;    1907  2      !
;    1908  2      !       INSERT YOUR CLEANUP CODING.  THIS CODING SHOULD
;    1909  2      !       RESTORE YOUR TEST-DEVICE TO A NEUTRAL STATE.
;    1910  2      !       THIS CODE WILL BE EXECUTED AFTER EACH PASS AND AFTER THE
;    1911  2      !       PROGRAM IS INTERRUPTED BY "^C".
;    1912  2      !
;    1913  2      !--
;    1914  2
;    1915  1      ENDCLN;


                             .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  004737  000000G  LCLEAN: JSR     PC,RESET.DEQNA                  ;  1902
000004  005077  000000G          CLR     @CLK.CSR                        ;  1903
000010  000207                   RTS     PC                              ;  1888

; Routine Size:  5 words,      Routine Base:  AA$CODE$ + 0554
; Maximum stack depth per invocation:  1 word


                             .SBTTL  L$CLEAN CLEANUP CODING SECTION
000000  004737  000554'  L$CLEAN::
                         JSR     PC,LCLEAN                       ;  1904
000004  104412           TRAP    12
000006  000207           RTS     PC

; Routine Size:  4 words,      Routine Base:  AA$CODE$ + 0566
; Maximum stack depth per invocation:  2 words

;    1916  1
;    1917  1
```

```
;    1918  1     %SBTTL 'DROP UNIT SECTION'
;    1919  1
;    1920  1     !++
;    1921  1     !
;    1922  1     !       THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;    1923  1     !       TO NO LONGER BE TESTED.
;    1924  1     !
;    1925  1     !--
;    1926  1
;    1927  2     BGNDU;
;    1928  2     !PRINTF (DBM33);
;    1929  2     RETURN;
;    1930  2
;    1931  2     !++
;    1932  2     !
;    1933  2     !       INSERT DROP CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
;    1934  2     !       A "DROP" COMMAND OR A "DODU" MACRO EXECUTION.  THE PURPOSE
;    1935  2     !       OR THIS CODE IS TO DO ANY NECESSARY HOUSEKEEPING AFTER A
;    1936  2     !       UNIT HAS BEEN DROPPED.
;    1937  2     !
;    1938  2     !--
;    1939  2
;    1940  1     ENDDU;


                                        .SBTTL  LDU DROP UNIT SECTION
000000  000207                  LDU:    RTS     PC                              ;                       1915

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0576
; Maximum stack depth per invocation:  0 words



                                        .SBTTL  L$DU DROP UNIT SECTION
000000  004737  000576'          L$DU:: JSR     PC,LDU                          ;                       1929
000004  104453                          TRAP    53
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0600
; Maximum stack depth per invocation:  2 words


;    1941  1
;    1942  1
```

```
;    1943  1     %SBTTL 'ADD UNIT SECTION'
;    1944  1
;    1945  1     !++
;    1946  1     !
;    1947  1     !       THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;    1948  1     !       TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;    1949  1     !       TO THE TEST CYCLE.
;    1950  1     !
;    1951  1     !--
;    1952  1
;    1953  2     BGNAU;
;    1954  2     !PRINTF (D8M34);
;    1955  2     RETURN;
;    1956  2
;    1957  2     !++
;    1958  2     !
;    1959  2     !       INSERT ADD CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
;    1960  2     !       AN "ADD" COMMAND.  THE PURPOSE OF THIS CODE IS TO DO ANY
;    1961  2     !       HOUSEKEEPING THAT MAY BE NECESSARY AFTER A UNIT HAS BEEN ADDED.
;    1962  2     !
;    1963  2     !--
;    1964  2
;    1965  1     ENDAU;


                                        .SBTTL  LAU ADD UNIT SECTION
000000  000207                  LAU:    RTS     PC                      ;  1940

; Routine Size:  1 word,        Routine Base:  AA$CODE$ + 0610
; Maximum stack depth per invocation:  0 words

                                        .SBTTL  L$AU ADD UNIT SECTION
000000  004737  000610'          L$AU:: JSR     PC,LAU                  ;  1955
000004  104452                         TRAP    52
000006  000207                         RTS     PC

; Routine Size:  4 words,       Routine Base:  AA$CODE$ + 0612
; Maximum stack depth per invocation:  2 words


;    1966  1
;    1967  1
```

F 1


```
;    1968  1
;    1969  2     BGNSRV (NXM_INT);
;    1970  2
;    1971  2     !++
;    1972  2     !
;    1973  2     !       GLOBAL LOCATION "INTERRUPT_FLG" IS SET TO TRUE WHICH INDICATES
;    1974  2     !       THE INITIALIZATION SEQUENCE INTERRUPT OCCURED.
;    1975  2     !
;    1976  2     !--
;    1977  2
;    1978  2        INTERRUPT_FLG     = %O'177777';
;    1979  2        CANCEL_TIMER      = %O'177777';
;    1980  2
;    1981  1     ENDSRV;


                                        .SBTTL  NXM.INT ADD UNIT SECTION
000000  012737  177777  000000G     NXM.INT::
                                        MOV     #-1,INTERRUPT.FLG               ;                1978
000006  012737  177777  000000G             MOV     #-1,CANCEL.TIMER            ;                1979
000014  000002                              RTI                                 ;                1969

; Routine Size:  7 words,       Routine Base:  AA$CODE$ + 0622
; Maximum stack depth per invocation:  0 words
```

```
;    1982  1
;    1983  1
;    1984  2     BGNSRV (CLK_INT);
;    1985  2
;    1986  2     !++
;    1987  2     !
;    1988  2     !       THE CLOCK INTERRUPT SERVICE ROUTINE IS ENTERED AT THE CLOCK RATE
;    1989  2     !
;    1990  2     !--
;    1991  2
;    1992  2        TICKS = .TICKS + 1;                  ! INCREMENT THE NUMBER OF TICK
;    1993  2        IF .TICKS EQLU .CLK_HERTZ            ! IF TOTAL NUMBER OF TICK = 60
;    1994  2        THEN                                 ! THEN
;    1995  3           BEGIN
;    1996  3              TICKS = 0;                     ! RESET TICK TO ZERO
;    1997  3              SECONDS = .SECONDS + 1;        ! INCREMENT THEN SECOND
;    1998  3              IF .SECONDS EQLU 60            ! IF SECOND = 60
;    1999  3              THEN                           ! THEN
;    2000  4                 BEGIN
;    2001  4                    SECONDS = 0;             ! RESET SECOND TO ZERO
;    2002  4                    MINUTES = .MINUTES + 1;  ! INCREMENT THE MINUTES
;    2003  3                 END;
;    2004  2           END;
;    2005  1     ENDSRV;


                                                .SBTTL  CLK.INT ADD UNIT SECTION
000000  005237  000000G                  CLK.INT::
000004  023737  000000G 000000G                 INC     TICKS                   ;    1992
000012  001014                                  CMP     TICKS,CLK.HERTZ         ;    1993
000014  005037  000000G                         BNE     1$
000020  005237  000000G                         CLR     TICKS                   ;    1996
000024  023727  000000G 000074                 INC     SECONDS                 ;    1997
000032  001004                                  CMP     SECONDS,#74             ;    1998
000034  005037  000000G                         BNE     1$
000040  005237  000000G                         CLR     SECONDS                 ;    2001
000044  000002                                  INC     MINUTES                 ;    2002
                                        1$:     RTI                             ;    1984

; Routine Size:  19 words,      Routine Base:  AA$CODE$ + 0640
; Maximum stack depth per invocation:  0 words


;    2006  1
;    2007  1     END
;    2008  0     ELUDOM


;                                       OTS external references
                                                .GLOBL  $SAVE4
```

```
;                                PSECT SUMMARY
;
;       Psect Name                  Words    Attributes
;        $CODE$                       31       RO ,  I  ,  LCL,  REL,  CON
;        AA$CODE$                    227       RO ,  I  ,  LCL,  REL,  CON




;                         Library Statistics
;
;                                  - -   - Symbols    ---        Pages      Processing
;       File                      Total    Loaded   Percent      Mapped     Time
;
;    DISK$USER2:[MAZURCZYK.SDC]QNALIB.L16;2
;                                  157       50        31          12        00:00.1




;                                COMMAND QUALIFIERS

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

; Size:          227 code + 31 data words
; Run Time:         00:16.1
; Elapsed Time:     00:40.8
; Lines/CPU Min:     7501
; Lexemes/CPU-Min: 41278
; Memory Used:  215 pages
; Compilation Complete
```

```
;    0001  0     MODULE ZQNA3 (%TITLE 'CZQNAAO 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          ERROR$REPORT            : NOVALUE,          ! PRINT EXTENDED ERROR MESSAGE
;    1511  1          E1$REPORT               : NOVALUE,          ! PRINT EXTENDED ERROR MESSAGE
;    1512  1          CLK_INT                 : L$ISR NOVALUE,    ! CLK INTERRUPT SERVICE ROUTINE
;    1513  1          NXM_INT                 : L$ISR NOVALUE,    ! NXM INTERRUPT SERVICE ROUTINE
;    1514  1          VER_DESCR_STATUS        : NOVALUE,
;    1515  1          CLR_BUFFERS             : NOVALUE,
;    1516  1          CLR_DESCRIPTORS         : NOVALUE,
;    1517  1          CLR_RBUFFER             : NOVALUE,
;    1518  1          CLR_XBUFFER             : NOVALUE,
;    1519  1          SET_RDESCR_LIST         : NOVALUE,
;    1520  1          SET_XDESCR_LIST         : NOVALUE,
;    1521  1          CHK_CSR_STATUS          : NOVALUE,
;    1522  1          CHK_RCV_STATUS          : NOVALUE,
;    1523  1          CHK_XMIT_STATUS         : NOVALUE,
;    1524  1          COMPARE_PACKETS         : NOVALUE,
;    1525  1          XMIT_AND_RCV_PACKET     : NOVALUE,
;    1526  1          WRT_STATION_ADR         : NOVALUE,
;    1527  1          WALKING_BIT             : NOVALUE,
;    1528  1          RESET_DEQNA             : NOVALUE;
;    1529  1
;    1530  1
```

```
;    1531  1
;    1532  1      EXTERNAL
;    1533  1
;    1534  1      !++
;    1535  1      !       COMMUNICATION AREA DECLARATIONS
;    1536  1      !--
;    1537  1
;    1538  1          RCV_D_LIST      : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1539  1          XMIT_D_LIST     : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1540  1          DESCR_LIST      : BLOCK  [ DESCR_SIZE, WORD] FIELD ( DL_FIELDS ),
;    1541  1          RCV_BUFFER      : VECTOR [ B_SIZE, BYTE ],
;    1542  1          XMIT_BUFFER     : VECTOR [ B_SIZE, BYTE ],
;    1543  1          DATA_BUFFER     : VECTOR [ BUF_SIZE, BYTE ],
;    1544  1          PHYS_ADR        : VECTOR [ 22, BYTE ],
;    1545  1          ETH_STATION_ADR : VECTOR [ 6, WORD ],
;    1546  1          SETUP_BUFFER    : VECTOR [ SETUB_SIZE, WORD ],
;    1547  1          IOP_TABLE       : VECTOR [ 8, WORD ],
;    1548  1          BD_PROM_DESCR   : VECTOR [ BD_D_SIZE, WORD ],
;    1549  1
;    1550  1      !++
;    1551  1      !       HARDWARE P-TABLE STORAGE DECLARATIONS
;    1552  1      !--
;    1553  1
;    1554  1          HWP_TABLE    : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;    1555  1
;    1556  1
;    1557  1      !++
;    1558  1      !       SOFTWARE P-TABLE STORAGE DECLARATIONS
;    1559  1      !--
;    1560  1
;    1561  1          SWP_TABLE    : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    1562  1
;    1563  1
;    1564  1      !++
;    1565  1      !       SYSTEM CLOCK STORAGE DECLARATIONS
;    1566  1      !--
;    1567  1
;    1568  1          XBUF_LENGTH          : WORD,                 ! XMIT BUFFER LENGTH IN WORDS
;    1569  1          RBUF_LENGTH          : WORD,                 ! RCV BUFFER LENGTH IN BYTES
;    1570  1          TICKS                : WORD,                 ! CLOCK RATE
;    1571  1          SECONDS              : WORD,                 ! STORE SECONDS
;    1572  1          MINUTES              : WORD,                 ! STORE MINUTES
;    1573  1          CANCEL_TIMER         : WORD,                 !
;    1574  1          CLK_VEC              : WORD,                 ! CLOCK INTERRUPT VECTOR ADR
;    1575  1          CLK_CSR              : WORD,                 ! STORE CSR ADDR FOR CLOCK HERE
;    1576  1          CLK_START            : WORD,                 ! STORE CLOCK START VALUE
;    1577  1          CLK_HERTZ            : WORD,                 ! TOTAL # OF CLOCK INTERRUPTS
;    1578  1          CLK_ADR              : WORD,                 ! LOC. TO RETURN CLOCK ADDR.
;    1579  1          CLK_TYPE             : WORD,                 ! TYPE OF CLOCK ON SYSTEM
;    1580  1                                                       ! (0=NONE,-1=L-CLOCK,1=P_CLOCK)
;    1581  1      !++
;    1582  1      !       MISCELLANEOUS DATA DECLARATIONS
;    1583  1      !--
```

```
;    1584  1
;    1585  1          FREE_MEM_ADR        : WORD,                 ! FREE MEMORY BEGIN ADR
;    1586  1          MEM_SIZE            : WORD,                 ! FREE MEMORY SIZE
;    1587  1          INTERRUPT_FLG       : WORD,                 ! 1 = INTERRUPT OCCURED
;    1588  1          COUNTER             : WORD,                 !
;    1589  1          CHECKSUM            : WORD,                 !
;    1590  1          PRI00, PRI01, PRI02, PRI03, PRI04, PRI05, PRI06, PRI07,
;    1591  1          DEQNA_NO,
;    1592  1
;    1593  1          REG_ADR             : REF REG_STR FIELD ( IOP_FIELDS ),
;    1594  1          GET_ADR             : REF ADR_STR FIELD ( IOP_FIELDS ),
;    1595  1          IOP_DATA            : REF REG_STR FIELD ( IOP_FIELDS ),
;    1596  1          STATION_ADR         : VECTOR [ 4, WORD ],
;    1597  1          PTRN_TABLE          : VECTOR [ 8, BYTE ],
;    1598  1
;    1599  1      !++
;    1600  1      !   TEMPORARY STORAGE DATA DECLARATIONS
;    1601  1      !-
;    1602  1
;    1603  1          TMP_IOP_ADR         : WORD,                 ! I/O PAGE REGISTER ADDRESS
;    1604  1          TMP_REG_DATA        : WORD,                 ! I/O PAGE REG CONTENTS
;    1605  1          TEMP1               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1606  1          TEMP2               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1607  1          TEMP3               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1608  1          TEMP4               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1609  1          TEMP5               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1610  1          TEMP6               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1611  1          TEMP7               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1612  1          TEMP8               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1613  1          TEMP9               : WORD,                 ! TEMPORARY STORAGE LOCATION
;    1614  1          TBYTE1              : BYTE,                 !
;    1615  1          TBYTE2              : BYTE,                 !
;    1616  1          TBYTE3              : BYTE,                 !
;    1617  1          TBYTE4              : BYTE,                 !
;    1618  1          TADR1               : WORD,                 !
;    1619  1          TADR2               : WORD,                 !
;    1620  1
;    1621  1
```

```
;    1622  1      !++
;    1623  1      !       DEBUG MESSAGES DECLARED EXTERMALLY
;    1624  1      ! -
;    1625  1
;    1626  1          DBM33,DBM34,DBM35,DBM36,DBM37,DBM38,DBM39,DBM40,DBM41,DBM42,
;    1627  1          DBM43, DBM44, DBM45, DBM46,
;    1628  1
;    1629  1      !++
;    1630  1      !       DIAGNOSTIC ERROR MESSAGES DECLARED EXTERNALLY
;    1631  1      ! -
;    1632  1
;    1633  1          E0001,
;    1634  1          ERR05, ERR06, ERR07, ERR08,
;    1635  1          E0101,
;    1636  1          E0201, E0202,
;    1637  1          E0301,
;    1638  1          E0401,
;    1639  1          E0501, E0502, E0503,
;    1640  1          E0601,
;    1641  1
;    1642  1          MSG01, MSG18, MSG19, MSG20, MSG21, MSG22, MSG23;
;    1643  1
;    1644  1
```

```
;    1645  1      %SBTTL 'TEST 1 - NON-EXISTANT I/O PAGE REGISTER TEST'
;    1646  1      !++
;    1647  1      !
;    1648  1      !  TEST 1:      NON-EXISTANT I/O PAGE REGISTER TEST
;    1649  1      !
;    1650  1      !  DESCRIPTION:
;    1651  1      !
;    1652  1      !       This test verifies that all the device registers residing in the
;    1653  1      !       I/O Page can be accessed without forcing a non-existant memory (NXM)
;    1654  1      !       interrupt. If the operator specifies loop on error, the program
;    1655  1      !       re-executes the code that detected the error until ^C is entered.
;    1656  1      !
;    1657  1      !       Hardware tested:         Q-Bus to DEQNA Slave Registers Interface
;    1658  1      !
;    1659  1      !
;    1660  1      !       Processing:
;    1661  1      !
;    1662  1      !            BEGIN
;    1663  1      !               get ready for NXM interrupt
;    1664  1      !               REPEAT for every I/O page register
;    1665  1      !
;    1666  1      !                  read I/O page register
;    1667  1      !                  IF NXM occured
;    1668  1      !                  THEN
;    1669  1      !                     print error message if not inhibited
;    1670  1      !                  ENDIF
;    1671  1      !
;    1672  1      !               ENDREPEAT
;    1673  1      !            END
;    1674  1      !               write any data pattern into the first 2 I/O page regs.
;    1675  1      !               IF NXM occured
;    1676  1      !               THEN
;    1677  1      !                  print error message if not inhibited
;    1678  1      !               ENDIF
;    1679  1      !
;    1680  1      !
;    1681  1      !--
```

```
;    1682  3     BGNTST;
;    1683  3
;    1684  3     SETVEC (4, NXM_INT, PRI07);                  ! SET UP FOR AN NXM INTERRUPT
;    1685  3     DELAY (MS_DELAY);                            ! DELAY 50 * 100 us = 5 ms
;    1686  3     INTERRUPT_FLG = CLEAR_FLG;                   ! CLEAR OUT NEX FLAG
;    1687  3     TMP_IOP_ADR = .MMP_TABLE [ ADDR ];
;    1688  3     INCR INDEX FROM 0 TO 7 DO
;    1689  4       BEGIN
;    1690  6         BGNSUB;
;    1691  6         TEMP1 = ..TMP_IOP_ADR;
;    1692  6         DELAY(7);
;    1693  6
;    1694  6         IF .INTERRUPT_FLG EQLU WORD_LIMIT        ! SEE IF WE GOT A NXM INTRT
;    1695  6           THEN
;    1696  7             BEGIN                                ! ADDRESS NOT THERE
;    1697  7               INTERRUPT_FLG = CLEAR_FLG;         ! CLEAR TRAP FLAG
;    1698  7               PRINTB ( E0101, .TMP_IOP_ADR );
;    1699  7               ERROF (0101, E0001, E1$REPORT);    ! 'I/O PAGE REG. NOT PRESENT
;    1700  7               DODU (DEQNA_NO);                   ! AND DROP THE UNIT
;    1701  7               DOCLN;
;    1702  6             END;
;    1703  4         ENDSUB;
;    1704  4         TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;    1705  3       END;
;    1706  3
;    1707  3     TMP_IOP_ADR = .MMP_TABLE [ ADDR ];
;    1708  3
;    1709  3     INCR INDEX FROM 0 TO 1 DO
;    1710  4       BEGIN
;    1711  6         BGNSUB;
;    1712  6         TEMP1 = ..TMP_IOP_ADR;
;    1713  6         DELAY(7);
;    1714  6         IF .INTERRUPT_FLG EQLU WORD_LIMIT        ! SEE IF WE GOT A NXM INTRT
;    1715  6           THEN
;    1716  7             BEGIN                                ! ADDRESS NOT THERE
;    1717  7               INTERRUPT_FLG = CLEAR_FLG;         ! CLEAR TRAP FLAG
;    1718  7               PRINTB ( E0101, .TMP_IOP_ADR );
;    1719  7               ERROF (0101, E0001, E1$REPORT);    ! 'I/O PAGE REG. NOT PRESENT'
;    1720  7               DODU (DEQNA_NO);                   ! AND DROP THE UNIT
;    1721  7               DOCLN;
;    1722  6             END;
;    1723  4         ENDSUB;
;    1724  4         TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;    1725  3       END;
;    1726  3
;    1727  3     CLRVEC (4);                                  ! CLEAR INTERRUPT VECTOR
;    1728  3
;    1729  1     ENDTST;


                                        .TITLE  ZQNA3 CZQNAA0 DEQNA FUNCTIONAL TEST
                                        .IDENT  /V01.0/
                                        .ENABL  AMA
```

```
                .GLOBL  ERROR$REPORT, E1$REPORT, CLK.INT
                .GLOBL  NXM.INT, VER.DESCR.STATUS, CLR.BUFFERS
                .GLOBL  CLR.DESCRIPTORS, CLR.RBUFFER, CLR.XBUFFER
                .GLOBL  SET.RDESCR.LIST, SET.XDESCR.LIST
                .GLOBL  CHK.CSR.STATUS, CHK.RCV.STATUS
                .GLOBL  CHK.XMIT.STATUS, COMPARE.PACKETS
                .GLOBL  XMIT.AND.RCV.PACKET, WRT.STATION.ADR
                .GLOBL  WALKING.BIT, RESET.DEQNA, RCV.D.LIST
                .GLOBL  XMIT.D.LIST, DESCR.LIST, RCV.BUFFER
                .GLOBL  XMIT.BUFFER, DATA.BUFFER, PHYS.ADR
                .GLOBL  ETH.STATION.ADR, SETUP.BUFFER
                .GLOBL  IOP.TABLE, BD.PROM.DESCR, MMP.TABLE
                .GLOBL  SMP.TABLE, XBUF.LENGTH, RBUF.LENGTH
                .GLOBL  TICKS, SECONDS, MINUTES, CANCEL.TIMER
                .GLOBL  CLK.VEC, CLK.CSR, CLK.START, CLK.HERTZ
                .GLOBL  CLK.ADR, CLK.TYPE, FREE.MEM.ADR
                .GLOBL  MEM.SIZE, INTERRUPT.FLG, COUNTER
                .GLOBL  CHECKSUM, PRI00, PRI01, PRI02
                .GLOBL  PRI03, PRI04, PRI05, PRI06, PRI07
                .GLOBL  DEQNA.NO, REG.ADR, GET.ADR, IOP.DATA
                .GLOBL  STATION.ADR, PTRN.TABLE, TMP.IOP.ADR
                .GLOBL  TMP.REG.DATA, TEMP1, TEMP2, TEMP3
                .GLOBL  TEMP4, TEMP5, TEMP6, TEMP7, TEMP8
                .GLOBL  TEMP9, TBYTE1, TBYTE2, TBYTE3
                .GLOBL  TBYTE4, TADR1, TADR2, DBM33, DBM34
                .GLOBL  DBM35, DBM36, DBM37, DBM38, DBM39
                .GLOBL  DBM40, DBM41, DBM42, DBM43, DBM44
                .GLOBL  DBM45, DBM46, E0001, ERR05, ERR06
                .GLOBL  ERR07, ERR08, E0101, E0201, E0202
                .GLOBL  E0301, E0401, E0501, E0502, E0503
                .GLOBL  E0601, MSG01, MSG18, MSG19, MSG20
                .GLOBL  MSG21, MSG22, MSG23, L$DLY

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

000000

000000  004137  000000G       $T1:    JSR     R1,$SAVE2                       ;                       1642
000004  005746                        TST     -(SP)
000006  012746  000000G               MOV     #PRI07,-(SP)                    ;                       1684
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                          ; .,$$TMP2              1685
000034  001410              1$:       BEQ     4$
000036  013700  000000G               MOV     L$DLY,R0                        ; .,$$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$:     CLR     INTERRUPT.FLG                   ;                               1686
000062  017737  000000G 000000G                 MOV     @HWP.TABLE,TMP.IOP.ADR          ;                               1687
000070  012702  000010                         MOV     #10,R2                          ; *,INDEX                       1688
000074  104402                          5$:     TRAP    2                               ;                               1689
000076  017737  000000G 000000G                 MOV     @TMP.IOP.ADR,TEMP1              ;                               1691
000104  012701  000007                         MOV     #7,R1                           ; *,$$TMP2                      1692
000110  001410                          6$:     BEQ     9$
000112  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000116  001403                                 BEQ     8$
000120  005066  000010                  7$:     CLR     10(SP)                          ; $$TMP
000124  077003                                 SOB     R0,7$                           ; $$TMP1,*
000126  005301                          8$:     DEC     R1                              ; $$TMP2
000130  000767                                 BR      6$
000132  023727  000000G 177777          9$:     CMP     INTERRUPT.FLG,#-1               ;                               1694
000140  001023                                 BNE     10$
000142  005037  000000G                         CLR     INTERRUPT.FLG                   ;                               1697
000146  013716  000000G                         MOV     TMP.IOP.ADR,(SP)                ;                               1698
000152  012746  000000G                         MOV     #E0101,-(SP)
000156  012746  000002                         MOV     #2,-(SP)
000162  010600                                 MOV     SP,R0                           ; SP,*
000164  104414                                 TRAP    14
000166  104455                                 TRAP    55                              ;                               1699
000170  000145                                 .WORD   145
000172  000000G                                .WORD   E0001
000174  000000G                                .WORD   E1$REPORT
000176  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                               1700
000202  104451                                 TRAP    51
000204  104444                                 TRAP    44
000206  022626                                 CMP     (SP)+,(SP)+                     ;                               1696
000210  104467                          10$:    TRAP    67                              ;                               1702
000212  006000                                 ROR     R0
000214  103727                                 BLO     5$
000216  062737  000002  000000G                ADD     #2,TMP.IOP.ADR                  ;                               1704
000224  077255                                 SOB     R2,5$                           ; INDEX,*                       1688
000226  017737  000000G 000000G                 MOV     @HWP.TABLE,TMP.IOP.ADR          ;                               1707
000234  012702  000002                         MOV     #2,R2                           ; *,INDEX                       1709
000240  104402                          11$:    TRAP    2                               ;                               1710
000242  017737  000000G 000000G                 MOV     @TMP.IOP.ADR,TEMP1              ;                               1712
000250  012701  000007                         MOV     #7,R1                           ; *,$$TMP2                      1713
000254  001410                          12$:    BEQ     15$
000256  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000262  001403                                 BEQ     14$
000264  005066  000010                  13$:    CLR     10(SP)                          ; $$TMP
000270  077003                                 SOB     R0,13$                          ; $$TMP1,*
000272  005301                          14$:    DEC     R1                              ; $$TMP2
000274  000767                                 BR      12$
000276  023727  000000G 177777          15$:    CMP     INTERRUPT.FLG,#-1               ;                               1714
000304  001023                                 BNE     16$
000306  005037  000000G                         CLR     INTERRUPT.FLG                   ;                               1717
000312  013716  000000G                         MOV     TMP.IOP.ADR,(SP)                ;                               1718
000316  012746  000000G                         MOV     #E0101,-(SP)
000322  012746  000002                         MOV     #2,-(SP)
000326  010600                                 MOV     SP,R0                           ; SP,*
```

```
000330  104414                          TRAP    14
000332  104455                          TRAP    55                              ;                               1719
000334  000145                          .WORD   145
000336  000000G                         .WORD   E0001
000340  000000G                         .WORD   E1$REPORT
000342  012700  000000G                 MOV     @DEQNA.NO,R0                    ;                               1720
000346  104451                          TRAP    51
000350  104444                          TRAP    44
000352  022626                          CMP     (SP)+,(SP)+                     ;                               1716
000354  104467                  16$:    TRAP    67                              ;                               1722
000356  006000                          ROR     R0
000360  103727                          BLO     11$
000362  062737  000002  000000G         ADD     @2,TMP.IOP.ADR                  ;                               1724
000370  077255                          SOB     R2,11$                          ; INDEX,*                       1709
000372  012700  000004                  MOV     @4,R0                           ;                               1727
000376  104436                          TRAP    36
000400  062706  000012                  ADD     @12,SP                          ;                               1642
000404  000207                          RTS     PC

; Routine Size:  131 words,     Routine Base:  AB$CODE$ + 0000
; Maximum stack depth per invocation:  12 words



                                        .SBTTL  T1 TEST 1 - NON-EXISTANT I/O PAGE REGISTER TEST
000000  004737  000000'          T1::
000000                           1$:    JSR     PC,$T1                          ;                               1727
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 0406
; Maximum stack depth per invocation:  2 words


;    1730  1
;    1731  1
```

```
;    1732  1     %SBTTL 'TEST 2 - CSR STATIC BIT TEST'
;    1733  1     !++
;    1734  1     !
;    1735  1     !  TEST 2:       CSR STATIC BIT TEST
;    1736  1     !
;    1737  1     !  DESCRIPTION:
;    1738  1     !
;    1739  1     !        This test verifies that the CSR register static bits can be set
;    1740  1     !       and cleared as specified.  The host writes data patterns to this
;    1741  1     !       register and reads them back verifying  no static
;    1742  1     !       (stuck at 1 / stuck at 0) faults occur. If the operator specifies
;    1743  1     !       loop on error, the program re-executes the code that detected the
;    1744  1     !       error until ^C is entered.
;    1745  1     !
;    1746  1     !       Hardware tested:                Q-Bus to DMA Interface
;    1747  1     !
;    1748  1     !       Processing:
;    1749  1     !
;    1750  1     !            BEGIN
;    1751  1     !
;    1752  1     !               reset device and check for expected CSR status
;    1753  1     !               IF error
;    1754  1     !               THEN
;    1755  1     !                  print error message if not inhibited
;    1756  1     !               ENDIF
;    1757  1     !
;    1758  1     !               set static bits and check for expected CSR status
;    1759  1     !               IF error
;    1760  1     !               THEN
;    1761  1     !                  print error message if not inhibited
;    1762  1     !               ENDIF
;    1763  1     !
;    1764  1     !               clear static bits and check for expected CSR status
;    1765  1     !               IF error
;    1766  1     !               THEN
;    1767  1     !                  print error message if not inhibited
;    1768  1     !               ENDIF
;    1769  1     !
;    1770  1     !            END
;    1771  1     !--
```

```
;    1772  3     BGNTST;
;    1773  3
;    1774  3     !++
;    1775  3     !
;    1776  3     !  RESET THE DEVICE AND CHECK FOR NOMINAL CSR STATUS
;    1777  3     !
;    1778  3     !--
;    1779  3
;    1780  3     RESET_DEQNA ( );
;    1781  3
;    1782  3     !++
;    1783  3     !
;    1784  3     !  CHECK IF CSR STATIC BITS (BIT 0,3,8 AND 9) ARE NOT STUCK AT 0
;    1785  3     !
;    1786  3     !--
;    1787  3
;    1788  3     PUT_BIT ( CSR, ALL_BITS, PATRN1 );
;    1789  3     DELAY ( 50 );
;    1790  3     IF GET_BIT [ CSR, ALL_BITS ] NEQU PATRN2
;    1791  3         THEN
;    1792  4             BEGIN
;    1793  4             PRINTB ( E0201, .GET_ADR [CSR_ALL], GET_BIT [ CSR, ALL_BITS ]);
;    1794  4             ERRDF ( 0201, E0001, E1$REPORT );
;    1795  4             DODU (DEQNA_NO);
;    1796  4             DOCLN;
;    1797  3             END;
;    1798  3
;    1799  3     !++
;    1800  3     !
;    1801  3     !  CHECK IF CSR STATIC BITS (BIT 0,3,8 AND 9) ARE NOT STUCK AT 1
;    1802  3     !
;    1803  3     !--
;    1804  3
;    1805  3     PUT_BIT ( CSR, ALL_BITS, ZERO );
;    1806  3     DELAY ( 100 );
;    1807  3     IF GET_BIT [ CSR, ALL_BITS ] NEQU CSR_2_STATUS
;    1808  3         THEN
;    1809  4             BEGIN
;    1810  4             PRINTB ( E0202, .GET_ADR [CSR_ALL], GET_BIT [ CSR, ALL_BITS ]);
;    1811  4             ERRDF ( 0202, E0001, E1$REPORT );
;    1812  4             DODU (DEQNA_NO);
;    1813  4             DOCLN;
;    1814  3             END;
;    1815  3
;    1816  3     RESET_DEQNA ( );
;    1817  3
;    1818  1     ENDTST;


                                        .SBTTL  $T2 TEST 2 - CSR STATIC BIT TEST
000000  004137  000000G         $T2:    JSR     R1,$SAVE2                       ;                       1729
000004  162706  000016                  SUB     #16,SP
000010  004737  000000G                 JSR     PC,RESET.DEQNA                  ;                       1780
```

```
000014  013700  000000G                        MOV     REG.ADR,R0                      ;                                       1788
000020  012760  001411  000016                 MOV     #1411,16(R0)
000026  012702  000062                         MOV     #62,R2                          ; *,$$TMP2                              1789
000032  001410                          1$:    BEQ     4$
000034  013701  000000G                        MOV     L$DLY,R1                        ; *,$$TMP1
000040  001403                                 BEQ     3$
000042  005066  000014                  2$:    CLR     14(SP)                          ; $$TMP
000046  077103                                 SOB     R1,2$                           ; $$TMP1,*
000050  005302                          3$:    DEC     R2                              ; $$TMP2
000052  000767                                 BR      1$
000054  016016  000016                  4$:    MOV     16(R0),(SP)                     ; *,TMP.LOCATION                        1790
000060  021627  011471                         CMP     (SP),#11471                     ; TMP.LOCATION,*
000064  001433                                 BEQ     5$
000066  011666  000002                         MOV     (SP),2(SP)                      ; *,TMP.LOCATION                        1793
000072  011646                                 MOV     (SP),-(SP)                      ; TMP.LOCATION,*
000074  013766  000000G 000006                 MOV     GET.ADR,6(SP)                   ; *,TMP.LOCATION
000102  062766  000016  000006                 ADD     #16,6(SP)                       ; *,TMP.LOCATION
000110  016646  000006                         MOV     6(SP),-(SP)                     ; TMP.LOCATION,*
000114  012746  000000G                        MOV     #E0201,-(SP)
000120  012746  000003                         MOV     #3,-(SP)
000124  010600                                 MOV     SP,R0                           ; SP,*
000126  104414                                 TRAP    14
000130  104455                                 TRAP    55                              ;                                       1794
000132  000311                                 .WORD   311
000134  000000G                                .WORD   E0001
000136  000000G                                .WORD   E1$REPORT
000140  012700  000000G                        MOV     #DEQNA.NO,R0                    ;                                       1795
000144  104451                                 TRAP    51
000146  104444                                 TRAP    44
000150  062706  000010                         ADD     #10,SP                          ;                                       1792
000154  013700  000000G                 5$:    MOV     REG.ADR,R0                      ;                                       1805
000160  005060  000016                         CLR     16(R0)
000164  012702  000144                         MOV     #144,R2                         ; *,$$TMP2                              1806
000170  001410                          6$:    BEQ     9$
000172  013701  000000G                        MOV     L$DLY,R1                        ; *,$$TMP1
000176  001403                                 BEQ     8$
000200  005066  000014                  7$:    CLR     14(SP)                          ; $$TMP
000204  077103                                 SOB     R1,7$                           ; $$TMP1,*
000206  005302                          8$:    DEC     R2                              ; $$TMP2
000210  000767                                 BR      6$
000212  016066  000016  000006          9$:    MOV     16(R0),6(SP)                    ; *,TMP.LOCATION                        1807
000220  026627  000006  010060                 CMP     6(SP),#10060                    ; TMP.LOCATION,*
000226  001435                                 BEQ     10$
000230  016666  000006  000010                 MOV     6(SP),10(SP)                    ; *,TMP.LOCATION                        1810
000236  016646  000010                         MOV     10(SP),-(SP)                    ; TMP.LOCATION,*
000242  013766  000000G 000014                 MOV     GET.ADR,14(SP)                  ; *,TMP.LOCATION
000250  062766  000016  000014                 ADD     #16,14(SP)                      ; *,TMP.LOCATION
000256  016646  000014                         MOV     14(SP),-(SP)                    ; TMP.LOCATION,*
000262  012746  000000G                        MOV     #E0202,-(SP)
000266  012746  000003                         MOV     #3,-(SP)
000272  010600                                 MOV     SP,R0                           ; SP,*
000274  104414                                 TRAP    14
000276  104455                                 TRAP    55                              ;                                       1811
```

```
000300  000312                                  .WORD   312
000302  000000G                                 .WORD   E0001
000304  000000G                                 .WORD   E1$REPORT
000306  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                       1812
000312  104451                                  TRAP    51
000314  104444                                  TRAP    44
000316  062706  000010                          ADD     #10,SP                          ;                       1809
000322  004737  000000G             10$:        JSR     PC,RESET.DEQNA                  ;                       1816
000326  062706  000016                          ADD     #16,SP                          ;                       1729
000332  000207                                  RTS     PC

; Routine Size:  110 words,     Routine Base:  AB$CODE$ + 0422
; Maximum stack depth per invocation:  16 words



                                                .SBTTL  T2 TEST 2 - CSR STATIC BIT TEST
000000  004737  000422'            T2::
000000                              1$:         JSR     PC,$T2                          ;                       1816
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 0756
; Maximum stack depth per invocation:  2 words
;    1819  1
;    1820  1
```

```
;    1821  1     %SBTTL 'TEST 3 - ETHERNET STATION ADDRESS VERIFY TEST'
;    1822  1     !++
;    1823  1     !
;    1824  1     !  TEST 3:      ETHERNET STATION ADDRESS VERIFY TEST
;    1825  1     !
;    1826  1     !  DESCRIPTION:
;    1827  1     !
;    1828  1     !       This test verifies that  the Ethernet Station Address PROM can be
;    1829  1     !       read and loaded to host memory correctly. Ethernet Station Address is
;    1830  1     !       verified and checksum is computed from PROM data read and this checksum
;    1831  1     !       is compared to the checksum stored in the Ethernet Station Address
;    1832  1     !       PROM. Ethernet Station Address is always printed out on the console in
;    1833  1     !       the Ethernet standard format. If the address is not proper, the error
;    1834  1     !       is recorded and an appropriate error message is printed out on the
;    1835  1     !       console. If the operator specifies loop on error, the program
;    1836  1     !       re-executes the code that detected the error until ^C is entered.
;    1837  1     !
;    1838  1     !       Hardware tested:        Station Address PROM
;    1839  1     !                               Q-Bus DMA Interface
;    1840  1     !       Processing:
;    1841  1     !
;    1842  1     !            BEGIN
;    1843  1     !
;    1844  1     !               read DEQNA Station Address PROM and checksum
;    1845  1     !               save copy of Station Address PROM in host memory
;    1846  1     !               print Station Address on the console in standard format
;    1847  1     !               compute Station Address ROM checksum
;    1848  1     !               IF checksum read not equal checksum computed
;    1849  1     !               THEN
;    1850  1     !                  print error message if not inhibited
;    1851  1     !               ENDIF
;    1852  1     !               IF Station Address
;    1853  1     !                     [all 0's]
;    1854  1     !                  OR [all 1's]:
;    1855  1     !                  OR [not assigned to DEQNA space]:
;    1856  1     !                  OR [multicast bit set]:
;    1857  1     !               THEN
;    1858  1     !                  print error message if not inhibited
;    1859  1     !               ENDIF
;    1860  1     !
;    1861  1     !            END
;    1862  1     !--
```

```
;    1863  3      BGNTST;
;    1864  3
;    1865  3      RESET_DEQNA ( );
;    1866  3
;    1867  3      !++
;    1868  3      !  READ STATION ADDRESS FROM DEQNA PROM, SAVE IT, AND COMPUTE EXPECTED
;    1869  3      !  CHECKSUM.
;    1870  3      !--
;    1871  3
;    1872  3      CHECKSUM = 0;
;    1873  3
;    1874  3      INCR INDEX FROM 0 TO 5 BY 2 DO
;    1875  4        BEGIN
;    1876  4          TEMP1  = .REG_ADR [ .INDEX, ALL_BITS ];
;    1877  4          TEMP1  = .TEMP1 ^ 8;
;    1878  4          TEMP2 =  .REG_ADR [ .INDEX + 1, ALL_BITS ];
;    1879  4          STATION_ADR [ .COUNTER ] = .TEMP1 OR ( .TEMP2 AND %O'000377' );
;    1880  4
;    1881  4          IF ( .CHECKSUM AND %O'100000' ) NEQU ZERO
;    1882  4            THEN
;    1883  5              BEGIN
;    1884  5                CHECKSUM = .CHECKSUM ^ 1;
;    1885  5                CHECKSUM = .CHECKSUM + 1;
;    1886  5              END
;    1887  4            ELSE
;    1888  4              CHECKSUM = .CHECKSUM ^ 1;
;    1889  4
;    1890  4          CHECKSUM = .CHECKSUM + .STATION_ADR [ .COUNTER ];
;    1891  4
;    1892  4          IF .CHECKSUM GTRU WORD_LIMIT
;    1893  4            THEN
;    1894  4              CHECKSUM = .CHECKSUM + 1;
;    1895  4
;    1896  4          COUNTER = .COUNTER + 1;
;    1897  3        END;
;    1898  3
;    1899  3      !++
;    1900  3      !  PRINT ETHERNET STATION ADDRESS ON THE CONSOLE
;    1901  3      !--
;    1902  3
;    1903  3      COUNTER  = 18;
;    1904  3      PHYS_ADR [ 0  ] = %C'%';
;    1905  3      PHYS_ADR [ 1  ] = %C'A';
;    1906  3      PHYS_ADR [ 19 ] = %C' ';
;    1907  3      PHYS_ADR [ 20 ] = %C'%';
;    1908  3      PHYS_ADR [ 21 ] = %C'N';
;    1909  3
;    1910  3      DECR INDEX1 FROM 2 TO 0 DO
;    1911  4        BEGIN
;    1912  4          TEMP3 = .STATION_ADR [ .INDEX1 ];
;    1913  4          INCR INDEX2 FROM 0 TO 1 DO
;    1914  5            BEGIN
;    1915  5              INCR INDEX3 FROM 0 TO 1 DO
```

```
;    1916  6                BEGIN
;    1917  6                  TEMP1 = .TEMP3 AND %X'F';
;    1918  6                  IF .TEMP1 LEQU %DECIMAL'9'
;    1919  6                    THEN
;    1920  6                      TBYTE1 = %C'0' + .TEMP1
;    1921  6                    ELSE
;    1922  6                      TBYTE1 = %C'A' + ( .TEMP1 - %DECIMAL'10' );
;    1923  6                  PHYS_ADR [ .COUNTER ] = .TBYTE1;
;    1924  6                  COUNTER = .COUNTER - 1;
;    1925  6                  TEMP3 = .TEMP3 ^ ( -4 );
;    1926  5                END;
;    1927  5
;    1928  5              IF .COUNTER GTRU 2
;    1929  5                THEN
;    1930  5                  PHYS_ADR [ .COUNTER ] = %C'-';
;    1931  5
;    1932  5              COUNTER = .COUNTER - 1;
;    1933  4            END;
;    1934  3        END;
;    1935  3
;    1936  3      PRINTB ( MSG01, .HWP_TABLE [ ADDR ] );
;    1937  3      PRINTB ( PHYS_ADR );
;    1938  3
;    1939  3      !++
;    1940  3      !  READ ACTUAL CHECKSUM FROM DEQNA STATION ADDRESS PROM AND COMPARE IT TO
;    1941  3      !  THE EXPECTED CHECKSUM COMPUTED ABOVE.
;    1942  3      !--
;    1943  3
;    1944  3      PUT_BIT ( CSR, LB, EXT_LOOPBACK );
;    1945  3      DELAY ( 50 );
;    1946  3      TEMP1  = .REG_ADR [ 1, ALL_BITS ];
;    1947  3      TEMP1  = .TEMP1 ^ 8;
;    1948  3      TEMP2 =  .REG_ADR [ 0, ALL_BITS ];
;    1949  3      STATION_ADR [ CHSUM ] = .TEMP1 OR ( .TEMP2 AND %O'000377' );
;    1950  3      PUT_BIT ( CSR, LB, ZERO );
;    1951  3      IF .CHECKSUM NEQU .STATION_ADR [ CHSUM ]
;    1952  3          THEN
;    1953  4              BEGIN
;    1954  4              PRINTB ( E0301, .STATION_ADR [ CHSUM ], .CHECKSUM );
;    1955  4              ERRDF (0301, E0001, E1$REPORT);      !
;    1956  4              DODU (DEQNA_NO);                        ! AND DROP THE UNIT
;    1957  4              DOCLN;
;    1958  3              END;
;    1959  1      ENDTST;


                                        .SBTTL  $T3 TEST 3 - ETHERNET STATION ADDRESS VERIFY TEST
000000  004137  000000G          $T3:   JSR     R1,$SAVE4                       ;                    1818
000004  162706  000012                  SUB     #12,SP
000010  004737  000000G                 JSR     PC,RESET.DEQNA                  ;                    1865
000014  005037  000000G                 CLR     CHECKSUM                        ;                    1872
000020  005002                          CLR     R2                              ; INDEX              1874
000022  010200                  1$:     MOV     R2,R0                           ; INDEX,*            1876
```

```
000024  006300                                  ASL     R0
000026  063700  000000G                         ADD     REG.ADR,R0
000032  011016                                  MOV     (R0),(SP)       ; *,TMP.LOCATION
000034  011600                                  MOV     (SP),R0         ;                               1877
000036  072027  000010                          ASH     #10,R0
000042  010037  000000G                         MOV     R0,TEMP1
000046  010200                                  MOV     R2,R0           ; INDEX,*                       1878
000050  006300                                  ASL     R0
000052  063700  000000G                         ADD     REG.ADR,R0
000056  016066  000002  000002                  MOV     2(R0),2(SP)     ; *,TMP.LOCATION
000064  016637  000002  000000G                 MOV     2(SP),TEMP2     ; TMP.LOCATION,*
000072  013700  000000G                         MOV     COUNTER,R0      ;                               1879
000076  006300                                  ASL     R0
000100  005001                                  CLR     R1
000102  156601  000002                          BISB    2(SP),R1        ; TEMP2,*
000106  053701  000000G                         BIS     TEMP1,R1
000112  010160  000000G                         MOV     R1,STATION.ADR(R0)
000116  013701  000000G                         MOV     CHECKSUM,R1     ;                               1884
000122  006301                                  ASL     R1
000124  032737  100000  000000G                 BIT     #-100000,CHECKSUM ;                             1881
000132  001405                                  BEQ     2$
000134  010137  000000G                         MOV     R1,CHECKSUM     ;                               1884
000140  005237  000000G                         INC     CHECKSUM        ;                               1885
000144  000402                                  BR      3$              ;                               1881
000146  010137  000000G          2$:            MOV     R1,CHECKSUM     ;                               1888
000152  066037  000000G 000000G    3$:           ADD     STATION.ADR(R0),CHECKSUM ;                      1890
000160  005237  000000G                         INC     COUNTER         ;                               1896
000164  062702  000002                          ADD     #2,R2           ; *,INDEX                       1874
000170  020227  000005                          CMP     R2,#5           ; INDEX,*
000174  003712                                  BLE     1$
000176  012737  000022  000000G                 MOV     #22,COUNTER     ;                               1903
000204  112737  000045  000000G                 MOVB    #45,PHYS.ADR    ;                               1904
000212  112737  000101  000001G                 MOVB    #101,PHYS.ADR+1 ;                               1905
000220  112737  000040  000023G                 MOVB    #40,PHYS.ADR+23 ;                               1906
000226  112737  000045  000024G                 MOVB    #45,PHYS.ADR+24 ;                               1907
000234  112737  000116  000025G                 MOVB    #116,PHYS.ADR+25 ;                              1908
000242  012701  000004                          MOV     #4,R1           ; *,INDEX1                      1910
000246  016137  000000G 000000G    4$:           MOV     STATION.ADR(R1),TEMP3 ; *(INDEX1),*             1912
000254  012703  000002                          MOV     #2,R3           ; *,INDEX2                      1913
000260  012702  000002              5$:           MOV     #2,R2           ; *,INDEX3                      1915
000264  013737  000000G 000000G    6$:           MOV     TEMP3,TEMP1     ;                               1917
000272  042737  177760  000000G                 BIC     #177760,TEMP1
000300  013700  000000G                         MOV     TEMP1,R0        ;                               1918
000304  020027  000011                          CMP     R0,#11
000310  101004                                  BHI     7$
000312  010004                                  MOV     R0,R4           ;                               1920
000314  062704  000060                          ADD     #60,R4
000320  000403                                  BR      8$
000322  010004              7$:                 MOV     R0,R4           ;                               1922
000324  062704  000067                          ADD     #67,R4
000330  110437  000000G          8$:            MOVB    R4,TBYTE1
000334  013700  000000G                         MOV     COUNTER,R0      ;                               1923
000340  110460  000000G                         MOVB    R4,PHYS.ADR(R0) ; TBYTE1,*
```

```
000344  005337  000000G                        DEC     COUNTER                         ;                                       1924
000350  013700  000000G                        MOV     TEMP3,R0                        ;                                       1925
000354  072027  177774                         ASH     #-4,R0
000360  010037  000000G                        MOV     R0,TEMP3
000364  077241                                 SOB     R2,6$                           ; INDEX3,*                              1915
000366  013702  000000G                        MOV     COUNTER,R2                      ;                                       1928
000372  020227  000002                         CMP     R2,#2
000376  101403                                 BLOS    9$
000400  112762  000055  000000G                MOVB    #55,PHYS.ADR(R2)                ;                                       1930
000406  005337  000000G              9$:       DEC     COUNTER                         ;                                       1932
000412  077356                                 SOB     R3,5$                           ; INDEX2,*                              1913
000414  162701  000002                         SUB     #2,R1                           ; *,INDEX1                              1910
000420  100312                                 BPL     4$
000422  017746  000000G                        MOV     @HWP.TABLE,-(SP)                ;                                       1936
000426  012746  000000G                        MOV     #MSG01,-(SP)
000432  012746  000002                         MOV     #2,-(SP)
000436  010600                                 MOV     SP,R0                           ; SP,*
000440  104414                                 TRAP    14
000442  012716  000000G                        MOV     #PHYS.ADR,(SP)                  ;                                       1937
000446  012746  000001                         MOV     #1,-(SP)
000452  010600                                 MOV     SP,R0                           ; SP,*
000454  104414                                 TRAP    14
000456  013701  000000G                        MOV     REG.ADR,R1                      ;                                       1944
000462  052761  001400  000016                 BIS     #1400,16(R1)
000470  012702  000062                         MOV     #62,R2                          ; *,$$TMP2                              1945
000474  001410                       10$:      BEQ     13$
000476  013700  000000G                        MOV     L$DLY,R0                        ; *,$$TMP1
000502  001403                                 BEQ     12$
000504  005066  000020                 11$:    CLR     20(SP)                          ; $$TMP
000510  077003                                 SOB     R0,11$                          ; $$TMP1,*
000512  005302                       12$:      DEC     R2                              ; $$TMP2
000514  000767                                 BR      10$
000516  016166  000002  000014       13$:      MOV     2(R1),14(SP)                    ; *,TMP.LOCATION                        1946
000524  016600  000014                         MOV     14(SP),R0                       ; TEMP1,*                               1947
000530  072027  000010                         ASH     #10,R0
000534  010037  000000G                        MOV     R0,TEMP1
000540  011166  000016                         MOV     (R1),16(SP)                     ; *,TMP.LOCATION                        1948
000544  011137  000000G                        MOV     (R1),TEMP2                      ; TMP.LOCATION,*
000550  005037  000006G                        CLR     STATION.ADR+6                   ;                                       1949
000554  111137  000006G                        MOVB    (R1),STATION.ADR+6              ; TEMP2,*
000560  050037  000006G                        BIS     R0,STATION.ADR+6                ; TEMP1,*
000564  042761  001400  000016                 BIC     #1400,16(R1)                    ;                                       1950
000572  023737  000000G 000006G               CMP     CHECKSUM,STATION.ADR+6          ;                                       1951
000600  001424                                 BEQ     14$
000602  013716  000000G                        MOV     CHECKSUM,(SP)                   ;                                       1954
000606  013746  000006G                        MOV     STATION.ADR+6,-(SP)
000612  012746  000000G                        MOV     #E0301,-(SP)
000616  012746  000003                         MOV     #3,-(SP)
000622  010600                                 MOV     SP,R0                           ; SP,*
000624  104414                                 TRAP    14
000626  104455                                 TRAP    55                              ;                                       1955
000630  000455                                 .WORD   455
000632  000000G                                .WORD   E0001
```

```
000634  000000G                                 .WORD   E1$REPORT
000636  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                       1956
000642  104451                                  TRAP    51
000644  104444                                  TRAP    44
000646  062706  000006                          ADD     #6,SP                           ;
000652  062706  000022                  14$:    ADD     #22,SP                          ;                       1953
000656  000207                                  RTS     PC                              ;                       1818

; Routine Size:  216 words,     Routine Base:  AB$CODE$ + 0772
; Maximum stack depth per invocation:  19 words



                                                .SBTTL  T3 TEST 3  ETHERNET STATION ADDRESS VERIFY TEST
000000  004737  000772'                  T3::
000000                                  1$:     JSR     PC,$T3                          ;                       1958
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 1652
; Maximum stack depth per invocation:  2 words


;   1960  1
;   1961  1
```

```
;    1962  1      %SBTTL 'TEST 4   INTERRUPT VECTOR ADDRESS TEST'
;    1963  1      !++
;    1964  1      !
;    1965  1      !  TEST 4:       INTERRUPT VECTOR ADDRESS TEST
;    1966  1      !
;    1967  1      !  DESCRIPTION:
;    1968  1      !
;    1969  1      !       This test verifies that all bits of the vector address register
;    1970  1      !       can be set and cleared as specified.  The host writes data patterns
;    1971  1      !       to this register and reads them back verifying no static
;    1972  1      !       (stuck at 1 / stuck at 0) faults occur. If the operator specifies
;    1973  1      !       loop on error, the program re executes the code that detected the
;    1974  1      !       error until ^C is entered.
;    1975  1      !
;    1976  1      !       NOTE:  Only bits 9:2 of the Interrupt Vector Address Register are
;    1977  1      !              valid, rest read as 0.
;    1978  1      !
;    1979  1      !       The following BINARY data patterns are used:
;    1980  1      !
;    1981  1      !                       00000000        11111111
;    1982  1      !                       10101010        01010101
;    1983  1      !                       11001100        00110011
;    1984  1      !                       11110000        00001111
;    1985  1      !                       walking 1's, 1 propagating thru Vector Address Reg.
;    1986  1      !                       walking 0's, 0 propagating thru Vector Address Reg.
;    1987  1      !
;    1988  1      !       Hardware tested:        Device Vector Address Register
;    1989  1      !                               Slave Interface Registers
;    1990  1      !
;    1991  1      !       Processing:
;    1992  1      !
;    1993  1      !               BEGIN
;    1994  1      !
;    1995  1      !                  reset device
;    1996  1      !                  REPEAT for each pattern
;    1997  1      !                     write pattern to Vector Address Register ( bits 9:2 )
;    1998  1      !                     read pattern from Vector Address Register ( bits 9:2 )
;    1999  1      !                     compare write pattern to read pattern (less noise bits)
;    2000  1      !                     IF not equal
;    2001  1      !                     THEN
;    2002  1      !                        print error message if not inhibited
;    2003  1      !                     ENDIF
;    2004  1      !
;    2005  1      !                  ENDREPEAT
;    2006  1      !               END
;    2007  1      !--
```

```
;    2008  3      BGNTST;
;    2009  3
;    2010  3      RESET_DEQNA ( );
;    2011  3
;    2012  3      !++
;    2013  3      !  WRITE ALTERNATING 0'S AND 1'S TO INTERRUPT VECTOR ADDRESS REGISTER
;    2014  3      !  IN THE I/O PAGE, THEN READ AND COMPARE TO THE WRITE PATTERN
;    2015  3      !--
;    2016  3
;    2017  3      INCR INDEX FROM 0 TO 7 DO
;    2018  4        BEGIN
;    2019  4          TEMP1 = .PTRN_TABLE [ .INDEX ];
;    2020  4          PUT_BIT [ INT_VEC, VEC_ADR, .TEMP1 ];
;    2021  4
;    2022  4          IF GET_BIT [ INT_VEC, VEC_ADR ] NEQU .TEMP1
;    2023  4            THEN
;    2024  5              BEGIN
;    2025  5                PRINTB (E0401, .GET_ADR [ VEC_ALL ], .TEMP1, GET_BIT [ INT_VEC, VEC_ADR ] );
;    2026  5                ERRDF ( 0401, E0001, E1$REPORT );
;    2027  5                DODU ( DEQNA_NO );
;    2028  5                DOCLN;
;    2029  4              END;
;    2030  3        END;
;    2031  3      !++
;    2032  3      !  WRITE WALKING 1 PATTERN INTO THE INTERRUPT VECTOR ADDRESS IN THE I/O PAGE
;    2033  3      !  REGISTER THEN READ AND COMPARE TO THE WRITE PATTERN
;    2034  3      !--
;    2035  3
;    2036  3      TEMP1 = %B'00000001';
;    2037  3
;    2038  3      INCR INDEX FROM 0 TO 7 DO
;    2039  4        BEGIN
;    2040  4          PUT_BIT [ INT_VEC, VEC_ADR, .TEMP1 ];
;    2041  4          IF GET_BIT [ INT_VEC, VEC_ADR ] NEQU .TEMP1
;    2042  4            THEN
;    2043  5              BEGIN
;    2044  5                PRINTB ( E0401, .GET_ADR [ VEC_ALL ], .TEMP1, GET_BIT [ INT_VEC, VEC_ADR ] );
;    2045  5                ERRDF ( 0401, E0001, E1$REPORT );
;    2046  5                DODU ( DEQNA_NO );
;    2047  5                DOCLN;
;    2048  4              END;
;    2049  4          TEMP1 = .TEMP1 ^ 1;
;    2050  3        END;
;    2051  3
;    2052  3      !++
;    2053  3      !  WRITE WALKING 0 PATTERN INTO THE INTERRUPT VECTOR ADDRESS IN THE I/O PAGE
;    2054  3      !  REGISTER THEN READ AND COMPARE TO THE WRITE PATTERN
;    2055  3      !--
;    2056  3
;    2057  3      TEMP1 = %B'11111110';
;    2058  3
;    2059  3      INCR INDEX FROM 0 TO 7 DO
;    2060  4        BEGIN
```

```
;  2061  4          PUT_BIT [ INT_VEC, VEC_ADR, .TEMP1 ];
;  2062  4
;  2063  4          IF GET_BIT [ INT_VEC, VEC_ADR ] NEQU .TEMP1
;  2064  4            THEN
;  2065  5              BEGIN
;  2066  5                PRINTB ( E0401, .GET_ADR [ VEC_ALL ], .TEMP1, GET_BIT [ INT_VEC, VEC ADR ] );
;  2067  5                ERRDF ( 0401, E0001, E1$REPORT );
;  2068  5                DODU ( DEQNA_NO );
;  2069  5                DOCLN;
;  2070  4              END;
;  2071  4
;  2072  4          TEMP1 = (( .TEMP1 + 1 ) + 1 ) AND %O'000377' ;
;  2073  3        END;
;  2074  3
;  2075  1      ENDTST;
```

```
                                        .SBTTL  $T4 TEST 4 - INTERRUPT VECTOR ADDRESS TEST
000000  004137  000000G          $T4:  JSR     R1,$SAVE4                       ;                       1959
000004  162706  000022                 SUB     #22,SP
000010  004737  000000G                JSR     PC,RESET.DEQNA                  ;                       2010
000014  005002                         CLR     R2                              ; INDEX                 2017
000016  116237  000000G 000000G   1$:  MOVB    PTRN.TABLE(R2),TEMP1            ; *(INDEX),*            2019
000024  105037  000001G                CLRB    TEMP1+1
000030  013700  000000G                MOV     REG.ADR,R0                      ;                       2020
000034  013701  000000G                MOV     TEMP1,R1
000040  010103                         MOV     R1,R3
000042  006303                         ASL     R3
000044  006303                         ASL     R3
000046  042703  176003                BIC     #176003,R3
000052  042760  001774  000014       BIC     #1774,14(R0)
000060  050360  000014                BIS     R3,14(R0)
000064  016016  000014                MOV     14(R0),(SP)                     ; *,TMP.LOCATION        2022
000070  010104                         MOV     R1,R4
000072  011603                         MOV     (SP),R3                         ; TMP.LOCATION,*
000074  006203                         ASR     R3
000076  006203                         ASR     R3
000100  042703  177400                BIC     #177400,R3
000104  020304                         CMP     R3,R4
000106  001441                         BEQ     2$
000110  011666  000002                MOV     (SP),2(SP)                      ; *,TMP.LOCATION        2025
000114  011600                         MOV     (SP),R0                         ; TMP.LOCATION,*
000116  006200                         ASR     R0
000120  006200                         ASR     R0
000122  042700  177400                BIC     #177400,R0
000126  010046                         MOV     R0,-(SP)
000130  010146                         MOV     R1,-(SP)
000132  013766  000000G 000010        MOV     GET.ADR,10(SP)                  ; *,TMP.LOCATION
000140  062766  000014  000010        ADD     #14,10(SP)                      ; *,TMP.LOCATION
000146  016646  000010                MOV     10(SP),-(SP)                    ; TMP.LOCATION,*
000152  012746  000000G               MOV     #E0401,-(SP)
000156  012746  000004                MOV     #4,-(SP)
000162  010600                         MOV     SP,R0                           ; SP,*
```

```
000164  104414                          TRAP    14
000166  104455                          TRAP    55                      ;                               2026
000170  000621                          .WORD   621
000172  000000G                         .WORD   E0001
000174  000000G                         .WORD   E1$REPORT
000176  012700  000000G                 MOV     #DEQNA.NO,R0            ;                               2027
000202  104451                          TRAP    51
000204  104444                          TRAP    44
000206  062706  000012                  ADD     #12,SP                  ;                               2024
000212  005202                  2$:     INC     R2                      ; INDEX                         2017
000214  020227  000007                  CMP     R2,#7                   ; INDEX,*
000220  003676                          BLE     1$
000222  012737  000001  000000G         MOV     #1,TEMP1                ;                               2036
000230  012702  000010                  MOV     #10,R2                  ; *,INDEX                       2038
000234  013700  000000G         3$:     MOV     REG.ADR,R0              ;                               2040
000240  013701  000000G                 MOV     TEMP1,R1
000244  010103                          MOV     R1,R3
000246  006303                          ASL     R3
000250  006303                          ASL     R3
000252  042703  176003                  BIC     #176003,R3
000256  042760  001774  000014         BIC     #1774,14(R0)
000264  050360  000014                  BIS     R3,14(R0)
000270  016066  000014  000006         MOV     14(R0),6(SP)            ; *,TMP.LOCATION                2041
000276  010104                          MOV     R1,R4
000300  016603  000006                  MOV     6(SP),R3                ; TMP.LOCATION,*
000304  006203                          ASR     R3
000306  006203                          ASR     R3
000310  042703  177400                  BIC     #177400,R3
000314  020304                          CMP     R3,R4
000316  001443                          BEQ     4$
000320  016666  000006  000010         MOV     6(SP),10(SP)            ; *,TMP.LOCATION                2044
000326  016600  000010                  MOV     10(SP),R0               ; TMP.LOCATION,*
000332  006200                          ASR     R0
000334  006200                          ASR     R0
000336  042700  177400                  BIC     #177400,R0
000342  010046                          MOV     R0,-(SP)
000344  010146                          MOV     R1,-(SP)
000346  013766  000000G 000016         MOV     GET.ADR,16(SP)          ; *,TMP.LOCATION
000354  062766  000014  000016         ADD     #14,16(SP)              ; *,TMP.LOCATION
000362  016646  000016                  MOV     16(SP),-(SP)            ; TMP.LOCATION,*
000366  012746  000000G                 MOV     #E0401,-(SP)
000372  012746  000004                  MOV     #4,-(SP)
000376  010600                          MOV     SP,R0                   ; SP,*
000400  104414                          TRAP    14
000402  104455                          TRAP    55                      ;                               2045
000404  000621                          .WORD   621
000406  000000G                         .WORD   E0001
000410  000000G                         .WORD   E1$REPORT
000412  012700  000000G                 MOV     #DEQNA.NO,R0            ;                               2046
000416  104451                          TRAP    51
000420  104444                          TRAP    44
000422  062706  000012                  ADD     #12,SP                  ;                               2043
000426  006337  000000G         4$:     ASL     TEMP1                   ;                               2049
```

```
000432  005302                                  DEC     R2                              ; INDEX                                 2038
000434  001277                                  BNE     3$
000436  012737  000376  000000G                 MOV     #376,TEMP1                      ;                                       2057
000444  012702  000010                          MOV     #10,R2                          ; *,INDEX                               2059
000450  013700  000000G               5$:      MOV     REG.ADR,R0                      ;                                       2061
000454  013701  000000G                         MOV     TEMP1,R1
000460  010103                                  MOV     R1,R3
000462  006303                                  ASL     R3
000464  006303                                  ASL     R3
000466  042703  176003                          BIC     #176003,R3
000472  042760  001774  000014                 BIC     #1774,14(R0)
000500  050360  000014                          BIS     R3,14(R0)
000504  016066  000014  000014                 MOV     14(R0),14(SP)                   ; *,TMP.LOCATION                        2063
000512  010104                                  MOV     R1,R4
000514  016603  000014                          MOV     14(SP),R3                       ; TMP.LOCATION,*
000520  006203                                  ASR     R3
000522  006203                                  ASR     R3
000524  042703  177400                          BIC     #177400,R3
000530  020304                                  CMP     R3,R4
000532  001443                                  BEQ     6$
000534  016666  000014  000016                 MOV     14(SP),16(SP)                   ; *,TMP.LOCATION                        2066
000542  016600  000016                          MOV     16(SP),R0                       ; TMP.LOCATION,*
000546  006200                                  ASR     R0
000550  006200                                  ASR     R0
000552  042700  177400                          BIC     #177400,R0
000556  010046                                  MOV     R0,-(SP)
000560  010146                                  MOV     R1,-(SP)
000562  013766  000000G 000024                 MOV     GET.ADR,24(SP)                  ; *,TMP.LOCATION
000570  062766  000014  000024                 ADD     #14,24(SP)                      ; *,TMP.LOCATION
000576  016646  000024                          MOV     24(SP),-(SP)                    ; TMP.LOCATION,*
000602  012746  000000G                         MOV     #E0401,-(SP)
000606  012746  000004                          MOV     #4,-(SP)
000612  010600                                  MOV     SP,R0                           ; SP,*
000614  104414                                  TRAP    14
000616  104455                                  TRAP    55                              ;                                       2067
000620  000621                                  .WORD   621
000622  000000G                                 .WORD   E0001
000624  000000G                                 .WORD   E1$REPORT
000626  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                                       2068
000632  104451                                  TRAP    51
000634  104444                                  TRAP    44
000636  062706  000012                          ADD     #12,SP                          ;                                       2065
000642  013700  000000G               6$:      MOV     TEMP1,R0                        ;                                       2072
000646  006300                                  ASL     R0
000650  005200                                  INC     R0
000652  005037  000000G                         CLR     TEMP1
000656  110037  000000G                         MOVB    R0,TEMP1
000662  005302                                  DEC     R2                              ; INDEX                                 2059
000664  001271                                  BNE     5$
000666  062706  000022                          ADD     #22,SP                          ;                                       1959
000672  000207                                  RTS     PC

; Routine Size:  222 words,    Routine Base:  AB$CODE$ + 1666
```

```
; Maximum stack depth per invocation:  21 words



                                                   .SBTTL  T4 TEST 4 - INTERRUPT VECTOR ADDRESS TEST
000000  004737  001666'              T4::
000000                               1$:           JSR     PC,$T4                              ;                                   2073
000004  104466                                     TRAP    66
000006  006000                                     ROR     R0
000010  103773                                     BLO     1$
000012  000207                                     RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 2562
; Maximum stack depth per invocation:  2 words


;   2076  1
;   2077  1
```

```
;    2078  1     %SBTTL 'TEST 5 - INTERRUPT SANITY TEST'
;    2079  1     !++
;    2080  1     !
;    2081  1     !  TEST 5:      INTERRUPT SANITY TEST
;    2082  1     !
;    2083  1     !  DESCRIPTION:
;    2084  1     !
```

```
;  1
;  3   BGNTST;
;  3
;  3   RESET_DEQNA ( );
;  3   SETVEC ( .HWP_TABLE [ VEC ], NXM_INT, PRIO7 );  ! SET UP FOR AN NXM INTERRUPT
;  3   .IOP_TABLE [ INT_VEC ] = .HWP_TABLE [ VEC ];
;  3   TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;  3   COUNTER = 0;
;  3   INTERRUPT_FLG = CLEAR_FLG;                      ! CLEAR OUT NXM FLAG
;  3
;  3   INCR PRIORITY FROM PRIO0 TO PRIO7 BY %O'40' DO
;  4     BEGIN
;  4       INTERRUPT_FLG = CLEAR_FLG;
;  4       SETPRI ( .PRIORITY );                       ! SET PROCESSOR PRI LEVEL
;  4       PUT_BIT ( CSR, IE, SET_IT );                ! ENABLE INTERRUPTS
;  4       DELAY (MS_DELAY);                           ! DELAY 50 x 100 us = 5 ms
;  4
;  4       .IOP_TABLE [ XLO_ADR ] = NXM_LO_ADR;        ! WRITE LOW ADDRESS
;  4       .IOP_TABLE [ XHI_ADR ] = NXM_HI_ADR;        ! WRITE HIGH ADDRESS
;  4
;  4       DELAY ( 2 );
;  4       GETPRI ( TEMP1 );
;  4       TEMP1 = .TEMP1 + ( - 5 );
;  4
;  5       IF ( .INTERRUPT_FLG EQLU WORD_LIMIT ) AND ( .PRIORITY GTRU PRIO3 )
;  4         THEN                                      ! INTERRUPT SHOULD NOT OCCUR
;  5           BEGIN
;  5             PRINTB ( E0601, .TMP_IOP_ADR, .TEMP1, .COUNTER );
;  5             ERRDF ( 0601, E0001, E1$REPORT );
;  5             DODU ( DEQNA_NO );
;  5             DOCLN;
;  4           END;
;  4
;  5       IF ( .INTERRUPT_FLG EQLU ZERO ) AND ( .PRIORITY LEQU PRIO3 )
;  4         THEN                                      ! INTERRUPT SHOULD OCCUR
;  5           BEGIN
;  5             PRINTB ( E0601, .TMP_IOP_ADR, .TEMP1, .COUNTER );
;  5             ERRDF ( 0601, E0001, E1$REPORT );
;  5             DODU ( DEQNA_NO );
;  5             DOCLN;
;  4           END;
;  4
;  4       RESET_DEQNA ( );
;  4       COUNTER = .COUNTER + 1;
;  3     END;
;  3
;  3   SETPRI ( PRIO4 );                               ! SET PROCESSOR PRI LEVEL
;  3
;  1   ENDTST;

                                        .SBTTL  $T5 TEST 5 - INTERRUPT SANITY TEST
000000  004137  000000G         $T5:    JSR     R1,$SAVE2                    ;        2075
```

```
000004  005746                                  TST     -(SP)
000006  004737  000000G                         JSR     PC,RESET.DEQNA          ;                               2088
000012  012746  000000G                         MOV     @PRI07, (SP)            ;                               2089
000016  012746  000000G                         MOV     @NXM.INT,-(SP)
000022  013700  000000G                         MOV     HWP.TABLE,R0
000026  016046  000002                          MOV     2(R0),-(SP)
000032  012746  000003                          MOV     @3,-(SP)
000036  104437                                  TRAP    37
000040  013700  000000G                         MOV     HWP.TABLE,R0            ;                               2090
000044  016077  000002  000014G                 MOV     2(R0),@IOP.TABLE+14
000052  017737  000000G  000000G                 MOV     @HWP.TABLE,TMP.IOP.ADR  ;                               2091
000060  005037  000000G                         CLR     COUNTER                 ;                               2092
000064  005037  000000G                         CLR     INTERRUPT.FLG           ;                               2093
000070  012702  000000G                         MOV     @PRI00,R2               ; *,PRIORITY                    2095
000074  000550                                  BR      12$
000076  005037  000000G             1$:         CLR     INTERRUPT.FLG           ;                               2097
000102  010200                                  MOV     R2,R0                   ; PRIORITY,*                    2098
000104  104441                                  TRAP    41
000106  013700  000000G                         MOV     REG.ADR,R0              ;                               2099
000112  152760  000100  000016                 BISB    @100,16(R0)
000120  012701  000062                          MOV     @62,R1                  ; *,$$TMP2                      2100
000124  001410                      2$:         BEQ     5$
000126  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000132  001403                                  BEQ     4$
000134  005066  000010              3$:         CLR     10(SP)                  ; $$TMP
000140  077003                                  SOB     R0,3$                   ; $$TMP1,*
000142  005301                      4$:         DEC     R1                      ; $$TMP2
000144  000767                                  BR      2$
000146  012777  160000  000010G     5$:         MOV     @-20000,@IOP.TABLE+10   ;                               2102
000154  012777  000077  000012G                 MOV     @77,@IOP.TABLE+12       ;                               2103
000162  012701  000002                          MOV     @2,R1                   ; *,$$TMP2                      2105
000166  001410                      <$:         BEQ     9$
000170  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000174  001403                                  BEQ     8$
000176  005066  000010              7$:         CLR     10(SP)                  ; $$TMP
000202  077003                                  SOB     R0,7$                   ; $$TMP1,*
000204  005301                      8$:         DEC     R1                      ; $$TMP2
000206  000767                                  BR      6$
000210  104440                      9$:         TRAP    40                      ;                               2106
000212  072027  177773                          ASH     @-5,R0                  ;                               2107
000216  010037  000000G                         MOV     R0,TEMP1
000222  023727  000000G  177777                 CMP     INTERRUPT.FLG,@-1       ;                               2109
000230  001030                                  BNE     10$
000232  020227  000000G                         CMP     R2,@PRI03               ; PRIORITY,*
000236  101425                                  BLOS    10$
000240  013716  000000G                         MOV     COUNTER,(SP)            ;                               2112
000244  010046                                  MOV     R0,-(SP)                ; TEMP1,*
000246  013746  000000G                         MOV     TMP.IOP.ADR,-(SP)
000252  012746  000000G                         MOV     @E0601,-(SP)
000256  012746  000004                          MOV     @4,-(SP)
000262  010600                                  MOV     SP,R0                   ; SP,*
000264  104414                                  TRAP    14
000266  104455                                  TRAP    55                      ;                               2113
```

```
000270  001131                                  .WORD   1131
000272  000000G                                 .WORD   E0001
000274  000000G                                 .WORD   E1$REPORT
000276  012700  000000G                         MOV     #DEQNA.NO,R0            ;                               2114
000302  104451                                  TRAP    51
000304  104444                                  TRAP    44
000306  062706  000010                          ADD     #10,SP                  ;                               2111
000312  005737  000000G             10$:        TST     INTERRUPT.FLG           ;                               2118
000316  001031                                  BNE     11$
000320  020227  000000G                         CMP     R2,#PRI03               ; PRIORITY,*
000324  101026                                  BHI     11$
000326  013716  000000G                         MOV     COUNTER,(SP)            ;                               2121
000332  013746  000000G                         MOV     TEMP1,-(SP)
000336  013746  000000G                         MOV     TMP.IOP.ADR,-(SP)
000342  012746  000000G                         MOV     #E0601,-(SP)
000346  012746  000004                          MOV     #4,-(SP)
000352  010600                                  MOV     SP,R0                   ; SP,*
000354  104414                                  TRAP    14
000356  104455                                  TRAP    55                      ;                               2122
000360  001131                                  .WORD   1131
000362  000000G                                 .WORD   E0001
000364  000000G                                 .WORD   E1$REPORT
000366  012700  000000G                         MOV     #DEQNA.NO,R0            ;                               2123
000372  104451                                  TRAP    51
000374  104444                                  TRAP    44
000376  062706  000010                          ADD     #10,SP                  ;                               2120
000402  004737  000000G             11$:        JSR     PC,RESET.DEQNA          ;                               2127
000406  005237  000000G                         INC     COUNTER                 ;                               2128
000412  062702  000040                          ADD     #40,R2                  ; *,PRIORITY                    2095
000416  020227  000000G             12$:        CMP     R2,#PRI07               ; PRIORITY,*
000422  003625                                  BLE     1$
000424  012700  000000G                         MOV     #PRI04,R0               ;                               2131
000430  104441                                  TRAP    41
000432  062706  000012                          ADD     #12,SP                  ;                               2075
000436  000207                                  RTS     PC

; Routine Size:  144 words,     Routine Base:  AB$CODE$ + 2576
; Maximum stack depth per invocation:  14 words



                                                .SBTTL  T5 TEST 5 - INTERRUPT SANITY TEST
000000  004737  002576'          T5::
000000                           1$:        JSR     PC,$T5                  ;                               2131
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 3236
; Maximum stack depth per invocation:  2 words
```

```
;    2135  1     %SBTTL 'TEST 6 - ETHERNET CARRIER SENSE TEST'
;    2136  1     !++
;    2137  1     !
;    2138  1     !  TEST 6:      ETHERNET CARRIER SENSE TEST
;    2139  1     !
;    2140  1     !  DESCRIPTION:
;    2141  1     !
;    2142  1
```

```
;    2143  3      BGNTST;
;    2144  3
;    2145  3      RESET_DEQNA ( );
;    2146  3
;    2147  3      !++
;    2148  3      !  FORMAT XMIT BUFFER INTO THE ETHERNET TRANSMIT PACKET
;    2149  3      !--
;    2150  3
;    2151  3      CLR_RBUFFER ( );
;    2152  3      CLR_XBUFFER ( );
;    2153  3      TMP_IOP_ADR = .HWP_TABLE [ ADDR ];
;    2154  3
;    2155  3      INCR INDEX FROM 0 TO 5 DO
;    2156  4        BEGIN
;    2157  4          ETH_STATION_ADR [ .INDEX ] = ..TMP_IOP_ADR;
;    2158  4          TMP_IOP_ADR = .TMP_IOP_ADR + 2;
;    2159  3        END;
;    2160  3
;    2161  3      INCR INDEX FROM 0 TO 5 DO
;    2162  4        BEGIN
;    2163  4          TBYTE1 = .REG_ADR [ .INDEX, ST_ADDR ];
;    2164  4          XMIT_BUFFER [ .INDEX ] = .TBYTE1;
;    2165  4          XMIT_BUFFER [ .INDEX + 6 ] = .XMIT_BUFFER [ .INDEX ];
;    2166  3        END;
;    2167  3
;    2168  3      XMIT_BUFFER [ PKT_TYPE ]     = LPB_PKT;
;    2169  3      XMIT_BUFFER [ PKT_TYPE + 1 ] = SKIP_CNT;
;    2170  3      XMIT_BUFFER [ PKT_TYPE + 2 ] = RFC;
;    2171  3
;    2172  3      INCR INDEX FROM 0 TO PKT_LENGTH - 1 DO
;    2173  3        XMIT_BUFFER [ PKT_DATA + .INDEX ] = .INDEX;
;    2174  3
;    2175  3      !++
;    2176  3      !  CONVERT SETUP PACKET SIZE FROM BYTE COUNT TO WORD COUNT
;    2177  3      !--
;    2178  3
;    2179  3      RBUF_LENGTH = PKT_LENGTH + 14;
;    2180  3      XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    2181  3
;    2182  3      SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2183  3      SET_XDESCR_LIST ( .XBUF_LENGTH, VE );
;    2184  3
;    2185  3      PUT_BIT ( CSR, LB, EXT_LOOPBACK );
;    2186  3      PUT_BIT ( CSR, IE, SET_IT );
;    2187  3
;    2188  3      .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;    2189  3      .IOP_TABLE [ RHI_ADR ] = 0;
;    2190  3
;    2191  3      .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;    2192  3      .IOP_TABLE [ XHI_ADR ] = 0;
;    2193  3
;    2194  3      INCR INDEX FROM 0 TO 999 DO
;    2195  3        IF GET_BIT [ CSR, CA ] EQLU ONE
```

```
;  2196  3          THEN
;  2197  4            BEGIN
;  2198  4    !         PRINTB ( DBM40, .MWP_TABLE [ ADDR ], GET_BIT [ CSR_ALL ]);
;  2199  4              TEMP2 = GET_BIT [ CSR_ALL ];
;  2200  4              EXITLOOP;
;  2201  3            END;
;  2202  3
;  2203  3      IF .TEMP4 EQLU 999
;  2204  3        THEN
;  2205  3          PRINTB ( MSG19, GET_BIT [ CSR_ALL ] );
;  2206  3
;  2207  3      INCR INDEX FROM 0 TO 2000 DO
;  2208  3        IF GET_BIT [ CSR, CA ] EQLU ZERO
;  2209  3          THEN
;  2210  4            BEGIN
;  2211  4              TEMP3 = GET_BIT [ CSR_ALL ];
;  2212  4              EXITLOOP;
;  2213  3            END;
;  2214  3
;  2215  3      IF GET_BIT [ CSR, CA ] EQLU ONE
;  2216  3        THEN
;  2217  3          PRINTB ( MSG21, GET_BIT [ CSR_ALL ] );
;  2218  3
;  2219  3      IF .TEMP5 EQLU 2000
;  2220  3        THEN
;  2221  3          PRINTB ( MSG20, GET_BIT [ CSR_ALL ] );
;  2222  3
;  2223  3      INCR INDEX FROM 0 TO 1000 DO
;  2224  3        IF GET_BIT [ CSR, RI ] EQLU ONE
;  2225  3          THEN
;  2226  4            BEGIN
;  2227  4              TEMP3 = GET_BIT [ CSR_ALL ];
;  2228  4              EXITLOOP;
;  2229  3            END;
;  2230  3
;  2231  3      IF .TEMP6 EQLU 1000
;  2232  3        THEN
;  2233  3          PRINTB ( MSG22, GET_BIT [ CSR_ALL ] );
;  2234  3
;  2235  3      CHK_CSR_STATUS  ( CSR_STATUS, CSR_STATUS );          ! 0'100220', 0'100220'
;  2236  3      CHK_XMIT_STATUS ( XFLG_STATUS, XWD11_STATUS );       ! 0'140000', 0'000000'
;  2237  3      CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS );        ! 0'140000', 0'020000'
;  2238  3      COMPARE_PACKETS ( );
;  2239  3
;  2240  1      ENDTST;


                                        .SBTTL  $T6 TEST 6 - ETHERNET CARRIER SENSE TEST
000000  010146                  $T6:    MOV     R1,-(SP)                ;                       2133
000002  162706  000030                  SUB     #30,SP
000006  004737  000000G                 JSR     PC,RESET.DEQNA          ;                       2145
000012  004737  000000G                 JSR     PC,CLR.RBUFFER          ;                       2151
000016  004737  000000G                 JSR     PC,CLR.XBUFFER          ;                       2152
```

```
000022  017737  000000G 000000G          MOV     @#MP.TABLE,TMP.IOP.ADR          ;                                       2153
000030  005000                           CLR     R0                              ; INDEX                                 2155
000032  017760  000000G 000000G    1$:   MOV     @TMP.IOP.ADR,ETH.STATION.ADR(R0); *,*(INDEX)                            2157
000040  062737  000002  000000G          ADD     #2,TMP.IOP.ADR                  ;                                       2158
000046  062700  000002                   ADD     #2,R0                           ; *,INDEX                               2155
000052  020027  000012                   CMP     R0,#12                          ; INDEX,*
000056  003765                           BLE     1$
000060  005000                           CLR     R0                              ; INDEX                                 2161
000062  010001                    2$:    MOV     R0,R1                           ; INDEX,*                               2163
000064  006301                           ASL     R1
000066  063701  000000G                  ADD     REG.ADR,R1
000072  011116                           MOV     (R1),(SP)                       ; *,TMP.LOCATION
000074  111637  000000G                  MOVB    (SP),TBYTE1
000100  111660  000000G                  MOVB    (SP),XMIT.BUFFER(R0)            ; *,*(INDEX)                            2164
000104  111660  000006G                  MOVB    (SP),XMIT.BUFFER+6(R0)          ; *,*(INDEX)                            2165
000110  005200                           INC     R0                              ; INDEX                                 2161
000112  020027  000005                   CMP     R0,#5                           ; INDEX,*
000116  003761                           BLE     2$
000120  112737  000220  000014G          MOVB    #220,XMIT.BUFFER+14             ;                                       2168
000126  105037  000015G                  CLRB    XMIT.BUFFER+15                  ;                                       2169
000132  112737  000001  000016G          MOVB    #1,XMIT.BUFFER+16               ;                                       2170
000140  005000                           CLR     R0                              ; INDEX                                 2172
000142  110060  000017G          3$:     MOVB    R0,XMIT.BUFFER+17(R0)           ; INDEX,*(INDEX)                        2173
000146  005200                           INC     R0                              ; INDEX                                 2172
000150  020027  002733                   CMP     R0,#2733                        ; INDEX,*
000154  003772                           BLE     3$
000156  012737  002752  000000G          MOV     #2752,RBUF.LENGTH               ;                                       2179
000164  012700  002752                   MOV     #2752,R0                        ;                                       2180
000170  006200                           ASR     R0
000172  005400                           NEG     R0
000174  010037  000000G                  MOV     R0,XBUF.LENGTH
000200  010046                           MOV     R0,-(SP)                        ; XBUF.LENGTH,*                         2182
000202  012746  120000                   MOV     #-60000,-(SP)
000206  004737  000000G                  JSR     PC,SET.RDESCR.LIST
000212  013716  000000G                  MOV     XBUF.LENGTH,(SP)                ;                                       2183
000216  012746  120000                   MOV     #-60000,-(SP)
000222  004737  000000G                  JSR     PC,SET.XDESCR.LIST
000226  013700  000000G                  MOV     REG.ADR,R0                      ;                                       2185
000232  012701  000016                   MOV     #16,R1
000236  060001                           ADD     R0,R1
000240  052711  001500                   BIS     #1500,(R1)                      ;                                       2186
000244  012777  000000G 000004G          MOV     #RCV.D.LIST,@IOP.TABLE+4        ;                                       2188
000252  005077  000006G                  CLR     @IOP.TABLE+6                    ;                                       2189
000256  012777  000000G 000010G          MOV     #XMIT.D.LIST,@IOP.TABLE+10      ;                                       2191
000264  005077  000012G                  CLR     @IOP.TABLE+12                   ;                                       2192
000270  012700  001750                   MOV     #1750,R0                        ; *,INDEX                               2194
000274  011166  000010            4$:    MOV     (R1),10(SP)                     ; *,TMP.LOCATION                        2195
000300  032711  020000                   BIT     #20000,(R1)                     ; *,TMP.LOCATION
000304  001405                           BEQ     5$
000306  011166  000012                   MOV     (R1),12(SP)                     ; *,TMP.LOCATION                        2199
000312  011137  000000G                  MOV     (R1),TEMP2                      ; TMP.LOCATION,*
000316  000401                           BR      6$                              ;                                       2197
000320  077013                    5$:    SOB     R0,4$                           ; INDEX,*                               2194
```

```
000322  023727  000000G 001747          6$:     CMP     TEMP4,#1747                     ;                               2203
000330  001012                                  BNE     7$
000332  011166  000014                          MOV     (R1),14(SP)                     ; *,TMP.LOCATION                2205
000336  011116                                  MOV     (R1),(SP)                       ; TMP.LOCATION,*
000340  012746  000000G                         MOV     #MSG19,-(SP)
000344  012746  000002                          MOV     #2,-(SP)
000350  010600                                  MOV     SP,R0                           ; SP,*
000352  104414                                  TRAP    14
000354  022626                                  CMP     (SP)+,(SP)+
000356  013700  000000G                 7$:     MOV     REG.ADR,R0                      ;                               2208
000362  012701  003721                          MOV     #3721,R1                        ; *,INDEX                       2207
000366  016066  000016  000016          8$:     MOV     16(R0),16(SP)                   ; *,TMP.LOCATION                2208
000374  032766  020000  000016                  BIT     #20000,16(SP)                   ; *,TMP.LOCATION
000402  001007                                  BNE     9$
000404  016666  000016  000020                  MOV     16(SP),20(SP)                   ; *,TMP.LOCATION                2211
000412  016637  000020  000000G                 MOV     20(SP),TEMP3                    ; TMP.LOCATION,*
000420  000401                                  BR      10$                             ;                               2210
000422  077117                          9$:     SOB     R1,8$                           ; INDEX,*                       2207
000424  016066  000016  000022          10$:    MOV     16(R0),22(SP)                   ; *,TMP.LOCATION                2215
000432  032766  020000  000022                  BIT     #20000,22(SP)                   ; *,TMP.LOCATION
000440  001414                                  BEQ     11$
000442  016666  000022  000024                  MOV     22(SP),24(SP)                   ; *,TMP.LOCATION                2217
000450  016616  000024                          MOV     24(SP),(SP)                     ; TMP.LOCATION,*
000454  012746  000000G                         MOV     #MSG21,-(SP)
000460  012746  000002                          MOV     #2,-(SP)
000464  010600                                  MOV     SP,R0                           ; SP,*
000466  104414                                  TRAP    14
000470  022626                                  CMP     (SP)+,(SP)+
000472  023727  000000G 003720          11$:    CMP     TEMP5,#3720                     ;                               2219
000500  001016                                  BNE     12$
000502  013700  000000G                         MOV     REG.ADR,R0                      ;                               2221
000506  016066  000016  000026                  MOV     16(R0),26(SP)                   ; *,TMP.LOCATION
000514  016616  000026                          MOV     26(SP),(SP)                     ; TMP.LOCATION,*
000520  012746  000000G                         MOV     #MSG20,-(SP)
000524  012746  000002                          MOV     #2,-(SP)
000530  010600                                  MOV     SP,R0                           ; SP,*
000532  104414                                  TRAP    14
000534  022626                                  CMP     (SP)+,(SP)+
000536  013700  000000G                 12$:    MOV     REG.ADR,R0                      ;                               2224
000542  012701  001751                          MOV     #1751,R1                        ; *,INDEX                       2223
000546  016066  000016  000030          13$:    MOV     16(R0),30(SP)                   ; *,TMP.LOCATION                2224
000554  100007                                  BPL     14$
000556  016666  000030  000032                  MOV     30(SP),32(SP)                   ; *,TMP.LOCATION                2227
000564  016637  000032  000000G                 MOV     32(SP),TEMP3                    ; TMP.LOCATION,*
000572  000401                                  BR      15$                             ;                               2226
000574  077114                          14$:    SOB     R1,13$                          ; INDEX,*                       2223
000576  023727  000000G 001750          15$:    CMP     TEMP6,#1750                     ;                               2231
000604  001014                                  BNE     16$
000606  016066  000016  000034                  MOV     16(R0),34(SP)                   ; *,TMP.LOCATION                2233
000614  016616  000034                          MOV     34(SP),(SP)                     ; TMP.LOCATION,*
000620  012746  000000G                         MOV     #MSG22,-(SP)
000624  012746  000002                          MOV     #2,-(SP)
000630  010600                                  MOV     SP,R0                           ; SP,*
```

```
000632  104414                          TRAP    14
000634  022626                          CMP     (SP)+,(SP)+
000636  012716  100220          16$:    MOV     #77560,(SP)             ;                       2235
000642  011646                          MOV     (SP),-(SP)
000644  004737  000000G                 JSR     PC,CHK.CSR.STATUS
000650  012716  140000                  MOV     #-40000,(SP)            ;                       2236
000654  005046                          CLR     -(SP)
000656  004737  000000G                 JSR     PC,CHK.XMIT.STATUS
000662  012716  140000                  MOV     #-40000,(SP)            ;                       2237
000666  012746  020000                  MOV     #20000,-(SP)
000672  004737  000000G                 JSR     PC,CHK.RCV.STATUS
000676  004737  000000G                 JSR     PC,COMPARE.PACKETS      ;                       2238
000702  062706  000044                  ADD     #44,SP                  ;                       2133
000706  012601                          MOV     (SP)+,R1
000710  000207                          RTS     PC

; Routine Size:  229 words,     Routine Base:  AB$CODE$ + 3252
; Maximum stack depth per invocation:  20 words



                                        .SBTTL  T6 TEST 6 - ETHERNET CARRIER SENSE TEST
000000  004737  003252'          T6::
000000                          1$:     JSR     PC,$T6                  ;                       2238
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,       Routine Base:  AB$CODE$ + 4164
; Maximum stack depth per invocation:  2 words

;    2241  1
;    2242  1
```

```
;    2243  1     %SBTTL 'TEST 7   STATION ADDRESS RAM TEST'
;    2244  1     !++
;    2245  1     !
;    2246  1     !  TEST 7:      STATION ADDRESS RAM TEST
;    2247  1     !
;    2248  1     !  DESCRIPTION:
;    2249  1     !
;    2250  1
```

```
;    2251  3     BGNTST;
;    2252  3
;    2253  3     RBUF_LENGTH = 128;
;    2254  3     XBUF_LENGTH = - ( .RBUF_LENGTH + -1 );
;    2255  3     DECR INDEX1 FROM 7 TO 0 DO
;    2256  4       BEGIN
;    2257  4         RESET_DEQNA ( );
;    2258  4         SET_RDESCR_LIST ( .XBUF_LENGTH, VE );
;    2259  4         SET_XDESCR_LIST ( .XBUF_LENGTH, VSE );
;    2260  4         CLR_RBUFFER ( );
;    2261  4
;    2262  4         INCR INDEX2 FROM 0 TO 127 DO
;    2263  4           XMIT_BUFFER [ .INDEX2 ] = .PTRN_TABLE [ .INDEX1 ];
;    2264  4
;    2265  4         PUT_BIT ( CSR, IE, SET_IT );
;    2266  4
;    2267  4         .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;    2268  4         .IOP_TABLE [ XHI_ADR ] = 0;
;    2269  4
;    2270  4         .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;    2271  4         .IOP_TABLE [ RHI_ADR ] = 0;
;    2272  4
;    2273  4         DELAY ( 300 );
;    2274  4
;    2275  4         IF GET_BIT [ CSR, RI ] NEQU ONE
;    2276  4           THEN
;    2277  5             BEGIN
;    2278  5               ERRDF ( 0801, E0001, ERROR$REPORT );
;    2279  5               DODU ( DEQNA_NO );
;    2280  5               DOCLN;
;    2281  4             END;
;    2282  4
;    2283  4         CHK_CSR_STATUS  ( CSR_STATUS, CSR_STATUS );        ! 0'100220', 0'100220'
;    2284  4         CHK_XMIT_STATUS ( XFLG_STATUS, XWD12_STATUS );     ! 0'140000', 0'000000'
;    2285  4         CHK_RCV_STATUS  ( RFLG_STATUS, RWD1_STATUS );      ! 0'140000', 0'020000'
;    2286  4         COMPARE_PACKETS ( );
;    2287  4
;    2288  3       END;
;    2289  3
;    2290  1     ENDTST;


                                        .SBTTL  $T7 TEST 7 - STATION ADDRESS RAM TEST
000000  004137  000000G           $T7:  JSR     R1,$SAVE3                       ;                       2240
000004  024646                        CMP     -(SP),-(SP)
000006  012737  000200  000000G       MOV     #200,RBUF.LENGTH                ;                       2253
000014  012700  000200                MOV     #200,R0                         ;                       2254
000020  006200                        ASR     R0
000022  005400                        NEG     R0
000024  010037  000000G               MOV     R0,XBUF.LENGTH
000030  012703  000007                MOV     #7,R3                           ; *,INDEX1              2255
000034  004737  000000G           1$: JSR     PC,RESET.DEQNA                  ;                       2257
000040  013746  000000G               MOV     XBUF.LENGTH,-(SP)               ;                       2258
```

```
000044  012746  120000                          MOV     #-60000,-(SP)
000050  004737  000000G                         JSR     PC,SET.RDESCR.LIST
000054  013716  000000G                         MOV     XBUF.LENGTH,(SP)                ;                               2259
000060  012746  130000                          MOV     #-50000,-(SP)
000064  004737  000000G                         JSR     PC,SET.XDESCR.LIST
000070  004737  000000G                         JSR     PC,CLR.RBUFFER                  ;                               2260
000074  005000                                  CLR     R0                              ; INDEX2                        2262
000076  116360  000000G 000000G         2$:     MOVB    PTRN.TABLE(R3),XMIT.BUFFER(R0)  ; *(INDEX1),*(INDEX2)           2263
000104  005200                                  INC     R0                              ; INDEX2                        2262
000106  020027  000177                          CMP     R0,#177                         ; INDEX2,*
000112  003771                                  BLE     2$
000114  013700  000000G                         MOV     REG.ADR,R0                      ;                               2265
000120  152760  000100  000016                  BISB    #100,16(R0)
000126  012777  000000G 000010G                 MOV     #XMIT.D.LIST,@IOP.TABLE+10      ;                               2267
000134  005077  000012G                         CLR     @IOP.TABLE+12                   ;                               2268
000140  012777  000000G 000004G                 MOV     #RCV.D.LIST,@IOP.TABLE+4        ;                               2270
000146  005077  000006G                         CLR     @IOP.TABLE+6                    ;                               2271
000152  012702  000454                          MOV     #454,R2                         ; *,$$TMP2                      2273
000156  001410                          3$:     BEQ     6$
000160  013701  000000G                         MOV     L$DLY,R1                        ; *,$$TMP1
000164  001403                                  BEQ     5$
000166  005066  000010                  4$:     CLR     10(SP)                          ; $$TMP
000172  077103                                  SOB     R1,4$                           ; $$TMP1,*
000174  005302                          5$:     DEC     R2                              ; $$TMP2
000176  000767                                  BR      3$
000200  016066  000016  000006          6$:     MOV     16(R0),6(SP)                    ; *,TMP.LOCATION                2275
000206  100410                                  BMI     7$
000210  104455                                  TRAP    55                              ;                               2278
000212  001441                                  .WORD   1441
000214  000000G                                 .WORD   E0001
000216  000000G                                 .WORD   ERROR$REPORT
000220  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                               2279
000224  104451                                  TRAP    51
000226  104444                                  TRAP    44
000230  012716  100220                  7$:     MOV     #-77560,(SP)                    ;                               2283
000234  011646                                  MOV     (SP),-(SP)
000236  004737  000000G                         JSR     PC,CHK.CSR.STATUS
000242  012716  140000                          MOV     #-40000,(SP)                    ;                               2284
000246  012746  000400                          MOV     #400,-(SP)
000252  004737  000000G                         JSR     PC,CHK.XMIT.STATUS
000256  012716  140000                          MOV     #-40000,(SP)                    ;                               2285
000262  012746  020000                          MOV     #20000,-(SP)
000266  004737  000000G                         JSR     PC,CHK.RCV.STATUS
000272  004737  000000G                         JSR     PC,COMPARE.PACKETS              ;                               2286
000276  062706  000014                          ADD     #14,SP                          ;                               2256
000302  005303                                  DEC     R3                              ; INDEX1                        2255
000304  002253                                  BGE     1$
000306  022626                                  CMP     (SP)+,(SP)+                     ;                               2240
000310  000207                                  RTS     PC

; Routine Size:  101 words,    Routine Base:  AB$CODE$ + 4200
; Maximum stack depth per invocation:  13 words
```

```
                                        .SBTTL  T7 TEST 7 - STATION ADDRESS RAM TEST
000000  004737  004200'        T7::
000000                          1$:     JSR     PC,$T7                          ;                       2288
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,      Routine Base:  AB$CODE$ + 4512
; Maximum stack depth per invocation:  2 words


;   2291  1
;   2292  1
;   2293  1     END
;   2294  0     ELUDOM


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


;                               PSECT SUMMARY
;
;       Psect Name              Words     Attributes
;        AB$CODE$                1195       RO ,  I  ,  LCL,  REL,  CON
;
;                                       Library Statistics
;
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[MAZURCZYK.SDC]QNALIB.L16;2
;                                        157        97        61          12        00:00.0


;                                               COMMAND QUALIFIERS
;       BLISS/PDP11 ZQNA3.BLI/LIST=ZQNA3.LIS/OBJECT=ZQNA3.OBJ/SOURCE=PAGE:53
; Size:          1195 code + 0 data words
```

```
; Run Time:          00:43.3
; Elapsed Time:      01:51.5
; Lines/CPU Min:      3176
; Lexemes/CPU Min: 26402
; Memory Used:  256 pages
; Compilation Complete
```

```
;    0001  0     MODULE ZQNA4 (%TITLE 'CZQNAAO 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      !++
;    1506  1      !       EXTERNAL DATA USED BY THIS MODULE
;    1507  1      ! -
;    1508  1
;    1509  1      EXTERNAL
;    1510  1
;    1511  1      !++
;    1512  1      !       COMMUNICATION AREA DECLARATIONS
;    1513  1      !--
;    1514  1
;    1515  1          RCV_D_LIST      : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1516  1          XMIT_D_LIST     : BLOCK  [ D_SIZE, WORD ] FIELD ( DL_FIELDS ),
;    1517  1          DESCR_LIST      : BLOCK  [ DESCR_SIZE, WORD] FIELD ( DL_FIELDS ),
;    1518  1          RCV_BUFFER      : VECTOR [ B_SIZE, BYTE ],
;    1519  1          XMIT_BUFFER     : VECTOR [ B_SIZE, BYTE ],
;    1520  1          DATA_BUFFER     : VECTOR [ BUF_SIZE, BYTE ],
;    1521  1          SETUP_BUFFER    : VECTOR [ SETUB_SIZE, WORD ],
;    1522  1          IOP_TABLE       : VECTOR [ 8, WORD ],
;    1523  1          BD_PROM_DESCR   : VECTOR [ BD_D_SIZE, WORD ],
;    1524  1          STATION_ADR     : VECTOR [ 4, WORD ],
;    1525  1          TARGET_ADR      : VECTOR [ T_SIZE, BYTE ],
;    1526  1
;    1527  1      !++
;    1528  1      !       HARDWARE P-TABLE STORAGE DECLARATIONS
;    1529  1      !--
;    1530  1
;    1531  1          HWP_TABLE   : REF BLOCK [ HWP_SIZE, WORD ] FIELD ( HWP_FIELDS ),
;    1532  1
;    1533  1
;    1534  1      !++
;    1535  1      !       SOFTWARE P-TABLE STORAGE DECLARATIONS
;    1536  1      !--
;    1537  1
;    1538  1          SWP_TABLE   : REF BLOCK [ SWP_SIZE, WORD ] FIELD ( SWP_FIELDS ),
;    1539  1
;    1540  1
;    1541  1      !++
;    1542  1      !       SYSTEM CLOCK STORAGE DECLARATIONS
;    1543  1      !--
;    1544  1
;    1545  1          TICKS               : WORD,             ! CLOCK RATE
;    1546  1          SECONDS             : WORD,             ! STORE SECONDS
;    1547  1          MINUTES             : WORD,             ! STORE MINUTES
;    1548  1          CANCEL_TIMER        : WORD,             !
;    1549  1          CLK_VEC             : WORD,             ! CLOCK INTERRUPT VECTOR ADR
;    1550  1          CLK_CSR             : WORD,             ! STORE CSR ADDR FOR CLOCK HERE
;    1551  1          CLK_START           : WORD,             ! STORE CLOCK START VALUE
;    1552  1          CLK_HERTZ           : WORD,             ! TOTAL # OF CLOCK INTERRUPTS
;    1553  1          CLK_ADR             : WORD,             ! LOC. TO RETURN CLOCK ADDR.
;    1554  1          CLK_TYPE            : WORD,             ! TYPE OF CLOCK ON SYSTEM
```

```
;    1555  1                                                    ! (0=NONE, 1=L CLOCK,1=P_CLOCK)
;    1556  1      !++
;    1557  1      !       MISCELLANEOUS DATA DECLARATIONS
;    1558  1      !--
;    1559  1
;    1560  1          XBUF_LENGTH         : WORD,               ! XMIT BUFFER LENGTH IN WORDS
;    1561  1          RBUF_LENGTH         : WORD,               ! RCV BUFFER LENGTH IN BYTES
;    1562  1          FREE_MEM_ADR        : WORD,               ! FREE MEMORY BEGIN ADR
;    1563  1          MEM_SIZE            : WORD,               ! FREE MEMORY SIZE
;    1564  1          INTERRUPT_FLG       : WORD,               ! 1 = INTERRUPT OCCURED
;    1565  1          COUNTER             : WORD,               !
;    1566  1          CHECKSUM            : WORD,               !
;    1567  1          CSR_WORD            : WORD,
;    1568  1          PRI07,
;    1569  1          DEQNA_NO,
;    1570  1
;    1571  1          REG_ADR             : REF REG_STR FIELD ( IOP_FIELDS ),
;    1572  1          GET_ADR             : REF ADR_STR FIELD ( IOP_FIELDS ),
;    1573  1          IOP_DATA            : REF REG_STR FIELD ( IOP_FIELDS ),
;    1574  1
;    1575  1      !++
;    1576  1      !   TEMPORARY STORAGE DATA DECLARATIONS
;    1577  1      !--
;    1578  1
;    1579  1          TMP_IOP_ADR         : WORD,               ! I/O PAGE REGISTER ADDRESS
;    1580  1          TMP_REG_DATA        : WORD,               ! I/O PAGE REG CONTENTS
;    1581  1          TEMP1               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1582  1          TEMP2               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1583  1          TEMP3               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1584  1          TEMP4               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1585  1          TEMP5               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1586  1          TEMP6               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1587  1          TEMP7               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1588  1          TEMP8               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1589  1          TEMP9               : WORD,               ! TEMPORARY STORAGE LOCATION
;    1590  1          TBYTE1              : BYTE,               ! TEMPORARY STORAGE LOC
;    1591  1          TBYTE2              : BYTE,               ! TEMPORARY STORAGE LOC
;    1592  1          TBYTE3              : BYTE,               ! TEMPORARY STORAGE LOC
;    1593  1          TBYTE4              : BYTE,               ! TEMPORARY STORAGE LOC
;    1594  1          TADR1               : WORD,               ! TEMPORARY STORAGE LOC
;    1595  1          P1                  : WORD,               ! PARAMETER # 1
;    1596  1          P2                  : WORD,               ! PARAMETER # 2
;    1597  1          P3                  : WORD,               ! PARAMETER # 3
;    1598  1          P4                  : WORD,               ! PARAMETER # 4
;    1599  1          P5                  : WORD,               ! PARAMETER # 5
;    1600  1
```

```
;    1601  1      !++
;    1602  1      !       DEBUG MESSAGES DECLARED EXTERMALLY
;    1603  1      !--
;    1604  1
;    1605  1          DBM40,DBM45,
;    1606  1
;    1607  1      !++
;    1608  1      !       DIAGNOSTIC ERROR MESSAGES DECLARED EXTERNALLY
;    1609  1      !--
;    1610  1
;    1611  1          MSG_1TDR, MSG_2TDR, MSG_3TDR, MSG_4TDR,
;    1612  1          MSG02, MSG03, MSG04, MSG05, MSG06, MSG07, MSG08, MSG09, MSG10, MSG11,
;    1613  1          MSG12, MSG13, MSG14, MSG15, MSG16, MSG17,
;    1614  1          E0001,
;    1615  1          ERR04, ERR05;
;    1616  1
```

```
;    1617  1     %SBTTL 'GLOBAL ROUTINE - ERROR$REPORT ( )'
;    1618  1
;    1619  1     !++
;    1620  1     !
;    1621  1     !  GLOBAL ROUTINE :     ERROR$REPORT
;    1622  1     !
;    1623  1     !  DESCRIPTION:
;    1624  1     !
;    1625  1     !       This routine reports errors to the operator
;    1626  1     !
;    1627  1     !--
;    1628  1
;    1629  1     %SBTTL 'GLOBAL ROUTINE - ERROR$REPORT ( )'
;    1630  1
;    1631  1     BGNMSG (ERROR$REPORT);

                                        .TITLE  ZQNA4 CZQNAAO 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, HWP.TABLE
                                        .GLOBL  SWP.TABLE, TICKS, SECONDS, MINUTES
                                        .GLOBL  CANCEL.TIMER, CLK.VEC, CLK.CSR
                                        .GLOBL  CLK.START, CLK.HERTZ, CLK.ADR
                                        .GLOBL  CLK.TYPE, XBUF.LENGTH, RBUF.LENGTH
                                        .GLOBL  FREE.MEM.ADR, MEM.SIZE, INTERRUPT.FLG
                                        .GLOBL  COUNTER, CHECKSUM, CSR.WORD, PRIO7
                                        .GLOBL  DEQNA.NO, REG.ADR, GET.ADR, IOP.DATA
                                        .GLOBL  TMP.IOP.ADR, TMP.REG.DATA, TEMP1
                                        .GLOBL  TEMP2, TEMP3, TEMP4, TEMP5, TEMP6
                                        .GLOBL  TEMP7, TEMP8, TEMP9, TBYTE1, TBYTE2
                                        .GLOBL  TBYTE3, TBYTE4, TADR1, P1, P2
                                        .GLOBL  P3, P4, P5, D8M40, D8M45, MSG.1TDR
                                        .GLOBL  MSG.2TDR, MSG.3TDR, MSG.4TDR, MSG02
                                        .GLOBL  MSG03, MSG04, MSG05, MSG06, MSG07
                                        .GLOBL  MSG08, MSG09, MSG10, MSG11, MSG12
                                        .GLOBL  MSG13, MSG14, MSG15, MSG16, MSG17
                                        .GLOBL  E0001, ERR04, ERR05

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

000000  004737  000000V       ERROR$REPORT::
                                        JSR     PC,M$ERROR$REPORT               ;                       1631
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  AC$CODE$ + 0000
```

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

;   1632  2
;   1633  2          PRINTB ( MSG03 );
;   1634  2          PRINTB ( MSG04, .XMIT_D_LIST [ FLGWD ], .RCV_D_LIST [ FLGWD ] );
;   1635  2          PRINTB ( MSG05, .XMIT_D_LIST [ DBITS ], .RCV_D_LIST [ DBITS ] );
;   1636  2          PRINTB ( MSG06, .XMIT_D_LIST [ LOADR ], .RCV_D_LIST [ LOADR ] );
;   1637  2          PRINTB ( MSG07, .XMIT_D_LIST [ TWDL ], .RCV_D_LIST [ TWDL ] );
;   1638  2          PRINTB ( MSG08, .XMIT_D_LIST [ STWD1 ], .RCV_D_LIST [ STWD1 ] );
;   1639  2          PRINTB ( MSG09, .XMIT_D_LIST [ STWD2 ], .RCV_D_LIST [ STWD2 ] );
;   1640  2          PRINTB ( MSG10, GET_BIT [ CSR_ALL ] );
;   1641  2          PRINTB ( MSG11, .HWP_TABLE [ ADDR ] );
;   1642  2
;   1643  1      ENDMSG;


                                .SBTTL  M$ERROR$REPORT GLOBAL ROUTINE - ERROR$REPORT ( )
000000  005746          M$ERROR$REPORT:
                        TST     -(SP)                   ;                               1631
000002  012746  000000G MOV     #MSG03,-(SP)            ;                               1633
000006  012746  000001  MOV     #1,-(SP)
000012  010600          MOV     SP,R0                   ; SP,*
000014  104414          TRAP    14
000016  013716  000000G MOV     RCV.D.LIST,(SP)         ;                               1634
000022  013746  000000G MOV     XMIT.D.LIST,-(SP)
000026  012746  000000G MOV     #MSG04,-(SP)
000032  012746  000003  MOV     #3,-(SP)
000036  010600          MOV     SP,R0                   ; SP,*
000040  104414          TRAP    14
000042  013716  000002G MOV     RCV.D.LIST+2,(SP)       ;                               1635
000046  013746  000002G MOV     XMIT.D.LIST+2,-(SP)
000052  012746  000000G MOV     #MSG05,-(SP)
000056  012746  000003  MOV     #3,-(SP)
000062  010600          MOV     SP,R0                   ; SP,*
000064  104414          TRAP    14
000066  013716  000004G MOV     RCV.D.LIST+4,(SP)       ;                               1636
000072  013746  000004G MOV     XMIT.D.LIST+4,-(SP)
000076  012746  000000G MOV     #MSG06,-(SP)
000102  012746  000003  MOV     #3,-(SP)
000106  010600          MOV     SP,R0                   ; SP,*
000110  104414          TRAP    14
000112  013716  000006G MOV     RCV.D.LIST+6,(SP)       ;                               1637
000116  013746  000006G MOV     XMIT.D.LIST+6,-(SP)
000122  012746  000000G MOV     #MSG07,-(SP)
000126  012746  000003  MOV     #3,-(SP)
000132  010600          MOV     SP,R0                   ; SP,*
000134  104414          TRAP    14
000136  013716  000010G MOV     RCV.D.LIST+10,(SP)      ;                               1638
000142  013746  000010G MOV     XMIT.D.LIST+10,-(SP)
000146  012746  000000G MOV     #MSG08,-(SP)
000152  012746  000003  MOV     #3,-(SP)
000156  010600          MOV     SP,R0                   ; SP,*
```

```
000160  104414                          TRAP    14
000162  013716  000012G                MOV     RCV.D.LIST+12,(SP)          ;                                   1639
000166  013746  000012G                MOV     XMIT.D.LIST+12, (SP)
000172  012746  000000G                MOV     #MSG09,-(SP)
000176  012746  000003                  MOV     #3,-(SP)
000202  010600                          MOV     SP,R0                       ; SP,*
000204  104414                          TRAP    14
000206  013700  000000G                MOV     REG.ADR,R0                  ;                                   1640
000212  016066  000016  000050         MOV     16(R0),50(SP)               ; *,TMP.LOCATION
000220  016616  000050                  MOV     50(SP),(SP)                 ; TMP.LOCATION,*
000224  012746  000000G                MOV     #MSG10,-(SP)
000230  012746  000002                  MOV     #2,-(SP)
000234  010600                          MOV     SP,R0                       ; SP,*
000236  104414                          TRAP    14
000240  017716  000000G                MOV     @MAP.TABLE,(SP)             ;                                   1641
000244  012746  000000G                MOV     #MSG11,-(SP)
000250  012746  000002                  MOV     #2,-(SP)
000254  010600                          MOV     SP,R0                       ; SP,*
000256  104414                          TRAP    14
000260  062706  000062                  ADD     #62,SP                      ;                                   1631
000264  000207                          RTS     PC

; Routine Size:  91 words,      Routine Base:  AC$CODE$ + 0010
; Maximum stack depth per invocation:  27 words


;   1644  1
;   1645  1
```

```
;   1646  1     %SBTTL 'GLOBAL ROUTINE   E1$REPORT ( )
;   1647  1
;   1648  1     !++
;   1649  1     !
;   1650  1     !  GLOBAL ROUTINE :     E1$REPORT
;   1651  1     !
;   1652  1     !  DESCRIPTION:
;   1653  1     !
;   1654  1     !       This routine reports errors to the operator
;   1655  1     !
;   1656  1     !--
;   1657  1
;   1658  1     %SBTTL 'GLOBAL ROUTINE   E1$REPORT ( )
;   1659  1
;   1660  1     BGNMSG ( E1$REPORT );


                                        .SBTTL  E1$REPORT GLOBAL ROUTINE   E1$REPORT ( )
000000  004737  000000V          E1$REPORT::
                                        JSR     PC,M$E1$REPORT                  ;                                       1660
000004  104423                         TRAP    23
000006  000207                         RTS     PC

; Routine Size:  4 words,       Routine Base:  AC$CODE$ + 0276
; Maximum stack depth per invocation:  2 words


;   1661  2
;   1662  2       TEMP1 = 1;
;   1663  2
;   1664  1     ENDMSG;


                                        .SBTTL  M$E1$REPORT GLOBAL ROUTINE   E1$REPORT ( )
000000  012737  000001  000000G   M$E1$REPORT:
                                        MOV     #1,TEMP1                        ;                                       1662
000006  000207                          RTS     PC                              ;                                       1660

; Routine Size:  4 words,       Routine Base:  AC$CODE$ + 0306
; Maximum stack depth per invocation:  0 words


;   1665  1
;   1666  1
```

```
;    1667  1      %SBTTL 'GLOBAL ROUTINE   RESET DEQNA ( )
;    1668  1
;    1669  1      GLOBAL ROUTINE RESET_DEQNA : NOVALUE =
;    1670  1
;    1671  1      !++
;    1672  1      !
;    1673  1      !  GLOBAL ROUTINE :     RESET_DEQNA
;    1674  1      !
;    1675  1      !  DESCRIPTION:
;    1676  1      !
;    1677  1      !       This routine verifies that DEQNA can be reset by setting bit 1 in the
;    1678  1      !       CSR register. After the reset, CSR is checked for nominal
;    1679  1      !       status.
;    1680  1      !
;    1681  1      !       Hardware tested:          Q-Bus DMA Interface
;    1682  1      !
;    1683  1      !       Processing:
;    1684  1      !
;    1685  1      !            BEGIN
;    1686  1      !
;    1687  1      !               set Software Reset (SR) bit in CSR and check for
;    1688  1      !                   expected CSR status
;    1689  1      !               IF error
;    1690  1      !               THEN
;    1691  1      !                  print error message if not inhibited
;    1692  1      !               ENDIF
;    1693  1      !
;    1694  1      !               clear SR bit in CSR and check for expected CSR status
;    1695  1      !               IF error
;    1696  1      !               THEN
;    1697  1      !                  print error message if not inhibited
;    1698  1      !               ENDIF
;    1699  1      !
;    1700  1      !            END
;    1701  1      !--
```

```
;    1702  1
;    1703  1      !++
;    1704  1      !
;    1705  1      !  RESET THE DEVICE AND CHECK CONTENTS OF CSR FOR NOMINAL STATUS
;    1706  1      !
;    1707  1      !
;    1708  1
;    1709  2      BEGIN
;    1710  2
;    1711  2      PUT_BIT ( CSR, ALL_BITS, ZERO );
;    1712  2      PUT_BIT ( CSR, SR, SET_IT );
;    1713  2      DELAY ( 50 );
;    1714  2      IF GET_BIT ( CSR, ALL_BITS ) NEQU CSR_1_STATUS
;    1715  2        THEN
;    1716  3          BEGIN
;    1717  3            PRINTB ( ERR05, .HWP_TABLE [ ADDR ], GET_BIT [ CSR, ALL_BITS ] );
;    1718  3            ERRDF ( 05, E0001, E1$REPORT );
;    1719  3            DODU (DEQNA_NO);
;    1720  3            DOCLN;
;    1721  2          END;
;    1722  2
;    1723  2      !++
;    1724  2      !
;    1725  2      !  CLEAR SOFTWARE RESET BIT IN THE CSR AND CHECK FOR EXPECTED STATUS
;    1726  2      !
;    1727  2      !--
;    1728  2
;    1729  2      PUT_BIT ( CSR, SR, CLR_IT );
;    1730  2      DELAY ( 50 );
;    1731  2      IF GET_BIT ( CSR, ALL_BITS ) NEQU CSR_2_STATUS
;    1732  2        THEN
;    1733  3          BEGIN
;    1734  3            PRINTB ( ERR05, .HWP_TABLE [ ADDR ], GET_BIT [ CSR, ALL_BITS ] );
;    1735  3            ERRDF ( 05, E0001, E1$REPORT );
;    1736  3            DODU (DEQNA_NO);
;    1737  3            DOCLN;
;    1738  2          END;
;    1739  2
;    1740  1      END;


                                        .GLOBL  L$DLY


                                        .SBTTL  RESET.DEQNA GLOBAL ROUTINE - RESET_DEQNA ( )
000000  004137  000000G          RESET.DEQNA::
                                        JSR     R1,$SAVE2                       ;                       1669
000004  162706  000012                  SUB     #12,SP
000010  013700  000000G                 MOV     REG.ADR,R0                      ;                       1711
000014  012702  000016                  MOV     #16,R2
000020  060002                          ADD     R0,R2
000022  005012                          CLR     (R2)
000024  152712  000002                  BISB    #2,(R2)                         ;                       1712
```

```
000030  012701  000062                          MOV     #62,R1                  ; #,$$TMP2                      1713
000034  001410                          1$:     BEQ     4$
000036  013700  000000G                         MOV     L$DLY,R0                ; #,$$TMP1
000042  001403                                  BEQ     3$
000044  005066  000010                  2$:     CLR     10(SP)                  ; $$TMP
000050  077003                                  SOB     R0,2$                   ; $$TMP1,#
000052  005301                          3$:     DEC     R1                      ; $$TMP2
000054  000767                                  BR      1$
000056  011216                          4$:     MOV     (R2),(SP)               ; #,TMP.LOCATION                1714
000060  021627  010062                          CMP     (SP),#10062
000064  001425                                  BEQ     5$
000066  011666  000002                          MOV     (SP),2(SP)              ; #,TMP.LOCATION                1717
000072  011646                                  MOV     (SP),-(SP)
000074  017746  000000G                         MOV     @#WP.TABLE,-(SP)
000100  012746  000000G                         MOV     #ERR05,-(SP)
000104  012746  000003                          MOV     #3,-(SP)
000110  010600                                  MOV     SP,R0                   ; SP,#
000112  104414                                  TRAP    14
000114  104455                                  TRAP    55                      ;                               1718
000116  000005                                  .WORD   5
000120  000000G                                 .WORD   E0001
000122  000276'                                 .WORD   E1$REPORT
000124  012700  000000G                         MOV     #DEQNA.NO,R0            ;                               1719
000130  104451                                  TRAP    51
000132  104444                                  TRAP    44
000134  062706  000010                          ADD     #10,SP                  ;                               1716
000140  013700  000000G                 5$:     MOV     REG.ADR,R0              ;                               1729
000144  142760  000000?  000016                BICB    #2,16(R0)
000152  012702  000062                          MOV     #62,R2                  ; #,$$TMP2                      1730
000156  001410                          6$:     BEQ     9$
000160  013701  000000G                         MOV     L$DLY,R1                ; #,$$TMP1
000164  001403                                  BEQ     8$
000166  005066  000010                  7$:     CLR     10(SP)                  ; $$TMP
000172  077103                                  SOB     R1,7$                   ; $$TMP1,#
000174  005302                          8$:     DEC     R2                      ; $$TMP2
000176  000767                                  BR      6$
000200  016066  000016  000004          9$:     MOV     16(R0),4(SP)            ; #,TMP.LOCATION                1731
000206  026627  000004  010060                  CMP     4(SP),#10060            ; TMP.LOCATION,#
000214  001427                                  BEQ     10$
000216  016666  000004  000006                  MOV     4(SP),6(SP)             ; #,TMP.LOCATION                1734
000224  016646  000006                          MOV     6(SP),-(SP)             ; TMP.LOCATION,#
000230  017746  000000G                         MOV     @#WP.TABLE,-(SP)
000234  012746  000000G                         MOV     #ERR05,-(SP)
000240  012746  000003                          MOV     #3,-(SP)
000244  010600                                  MOV     SP,R0                   ; SP,#
000246  104414                                  TRAP    14
000250  104455                                  TRAP    55                      ;                               1735
000252  000005                                  .WORD   5
000254  000000G                                 .WORD   E0001
000256  000276'                                 .WORD   E1$REPORT
000260  012700  000000G                         MOV     #DEQNA.NO,R0            ;                               1736
000264  104451                                  TRAP    51
000266  104444                                  TRAP    44
```

```
000270  062706  000010                          ADD     #10,SP          ;                       1733
000274  062706  000012                  10$:    ADD     #12,SP          ;                       1669
000300  000207                          RTS     PC

; Routine Size:  97 words,      Routine Base:  AC$CODE$ + 0316
; Maximum stack depth per invocation:  14 words


;   1741  1
```

```
;    1742  1     %SBTTL 'GLOBAL ROUTINE   VER DESCR STATUS ( )'
;    1743  1
;    1744  1     GLOBAL ROUTINE VER_DESCR STATUS : NOVALUE =
;    1745  1
;    1746  1     !++
;    1747  1     !
;    1748  1     !  GLOBAL ROUTINE :     VER_DESCR_STATUS
;    1749  1     !
;    1750  1     !  DESCRIPTION:
;    1751  1     !
;    1752  1     !       This routine compares actual to original descriptor.
;    1753  1     !--
;    1754  1
;    1755  1
;    1756  2     BEGIN
;    1757  2
;    1758  2     INCR INDEX FROM 0 TO BD_D_SIZE - 1 DO
;    1759  3       BEGIN
;    1760  3         TEMP1 = .DESCR_LIST [ .INDEX, W_LEN ];
;    1761  4         IF ( .TEMP1 NEQU -1 ) OR ( .TEMP1 NEQU .BD_PROM_DESCR [ .INDEX ] )
;    1762  3           THEN
;    1763  4             BEGIN
;    1764  4             PRINTB ( ERR04, .INDEX, .TEMP1, .BD_PROM_DESCR [ .INDEX ]);
;    1765  4             ERRDF ( 04, E0001, ERROR$REPORT );
;    1766  4             DODU ( DEQNA_NO );
;    1767  4             DOCLN;
;    1768  3           END;
;    1769  2       END;
;    1770  2
;    1771  1     END;


                                .SBTTL  VER.DESCR.STATUS GLOBAL ROUTINE - VER_DESCR_STATUS ( )
000000  004137  000000G    VER.DESCR.STATUS::
                            JSR     R1,$SAVE2                       ;                       1744
000004  005002               CLR     R2                              ; INDEX                 1758
000006  010200    1$:       MOV     R2,R0                           ; INDEX,*               1760
000010  006300               ASL     R0
000012  016037  000000G 000000G    MOV     DESCR.LIST(R0),TEMP1
000020  016001  000000G            MOV     DESCR.LIST(R0),R1       ;                       1761
000024  020127  177777             CMP     R1,#-1
000030  001003                     BNE     2$
000032  020160  000000G            CMP     R1,BD.PROM.DESCR(R0)
000036  001424                     BEQ     3$
000040  016046  000000G    2$:     MOV     BD.PROM.DESCR(R0),-(SP) ;                       1764
000044  010146                     MOV     R1,-(SP)
000046  010246                     MOV     R2,-(SP)                ; INDEX,*
000050  012746  000000G            MOV     #ERR04,-(SP)
000054  012746  000004             MOV     #4,-(SP)
000060  010600                     MOV     SP,R0                   ; SP,*
000062  104414                     TRAP    14
000064  104455                     TRAP    55                      ;                       1765
000066  000004                     .WORD   4
```

```
000070  000000G                                 .WORD   E0001
000072  000000'                                 .WORD   ERROR$REPORT
000074  012700  000000G                         MOV     #DEQNA.NO,R0            ;                                       1766
000100  104451                                  TRAP    51
000102  104444                                  TRAP    44
000104  062706  000012                          ADD     #12,SP                  ;                                       1763
000110  005202                          3$:     INC     R2                      ; INDEX                                 1758
000112  020227  000017                          CMP     R2,#17                  ; INDEX,*
000116  003733                                  BLE     1$
000120  000207                                  RTS     PC                      ;                                       1744

; Routine Size:  41 words,      Routine Base:  AC$CODE$ + 0620
; Maximum stack depth per invocation:  10 words


;   1772  1
```

```
;    1773  1     %SBTTL 'GLOBAL ROUTINE   CLR_XBUFFER ( )'
;    1774  1
;    1775  1     GLOBAL ROUTINE CLR_XBUFFER : NOVALUE =
;    1776  1
;    1777  1     !++
;    1778  1     !
;    1779  1     !  GLOBAL ROUTINE :     CLR_XBUFFER
;    1780  1     !
;    1781  1     !  DESCRIPTION:
;    1782  1     !
;    1783  1     !       This routine initializes transmit buffer to zeros.
;    1784  1     !--
;    1785  1
;    1786  1
;    1787  2     BEGIN
;    1788  2
;    1789  2     INCR INDEX FROM 0 TO B_SIZE - 1 DO
;    1790  2       XMIT_BUFFER [ .INDEX ] = 0;
;    1791  2
;    1792  1     END;


                                        .SBTTL  CLR.XBUFFER GLOBAL ROUTINE   CLR_XBUFFER ( )
000000  005000                 CLR.XBUFFER::
                                        CLR     R0                      ; INDEX                         1789
000002  105060  000000G        1$:     CLRB    XMIT.BUFFER(R0)         ; *(INDEX)                      1790
000006  005200                         INC     R0                      ; INDEX                         1789
000010  020027  007777                 CMP     R0,#07777               ; INDEX,*
000014  003772                         BLE     1$
000016  000207                         RTS     PC                      ;                               1775

; Routine Size:  8 words,       Routine Base:  AC$CODE$ + 0742
; Maximum stack depth per invocation:  0 words


;    1793  1
;    1794  1
```

```
;    1795  1     %SBTTL 'GLOBAL ROUTINE - CLR_RBUFFER ( )'
;    1796  1
;    1797  1     GLOBAL ROUTINE CLR_RBUFFER : NOVALUE =
;    1798  1
;    1799  1     !++
;    1800  1     !
;    1801  1     !  GLOBAL ROUTINE :     CLR_RBUFFER
;    1802  1     !
;    1803  1     !  DESCRIPTION:
;    1804  1     !
;    1805  1     !       This routine initializes RECEIVE buffer to zeros.
;    1806  1     !--
;    1807  1
;    1808  1
;    1809  2     BEGIN
;    1810  2
;    1811  2     INCR INDEX FROM 0 TO B_SIZE - 1 DO
;    1812  2       RCV_BUFFER [ .INDEX ] = 0;
;    1813  2
;    1814  1     END;


                                        .SBTTL  CLR.RBUFFER GLOBAL ROUTINE - CLR_RBUFFER ( )
000000  005000                  CLR.RBUFFER::
                                        CLR     R0                      ; INDEX                         1811
000002  105060  000000G         1$:     CLRB    RCV.BUFFER(R0)          ; .(INDEX)                      1812
000006  005200                          INC     R0                      ; INDEX                         1811
000010  020027  007777                  CMP     R0,#7777                ; INDEX,#
000014  003772                          BLE     1$
000016  000207                          RTS     PC                      ;                               1797

; Routine Size:  8 words,      Routine Base:  AC$CODE$ + 0762
; Maximum stack depth per invocation:  0 words

;    1815  1
```

```
;    1816  1      %SBTTL 'GLOBAL ROUTINE   CLR_BUFFERS ( )'
;    1817  1
;    1818  1      GLOBAL ROUTINE CLR_BUFFERS : NOVALUE =
;    1819  1
;    1820  1      !++
;    1821  1      !
;    1822  1      !  GLOBAL ROUTINE :     CLR_BUFFERS
;    1823  1      !
;    1824  1      !  DESCRIPTION:
;    1825  1      !
;    1826  1      !       This routine initializes rcv and xmit buffers to zero.
;    1827  1      !--
;    1828  1
;    1829  1
;    1830  2      BEGIN
;    1831  2
;    1832  2      INCR INDEX FROM 0 TO BUF_SIZE - 1 DO
;    1833  2        DATA_BUFFER [ .INDEX ] = 0;
;    1834  2
;    1835  1      END;


                                        .SBTTL  CLR.BUFFERS GLOBAL ROUTINE - CLR_BUFFERS ( )
000000  005000                  CLR.BUFFERS::
                                        CLR     R0                      ; INDEX                         1832
000002  105060  000000G         1$:     CLRB    DATA.BUFFER(R0)         ; *(INDEX)                      1833
000006  005200                          INC     R0                      ; INDEX                         1832
000010  020027  017777                  CMP     R0,#17777               ; INDEX,*
000014  003772                          BLE     1$
000016  000207                          RTS     PC                      ;                               1818

; Routine Size:  8 words,      Routine Base:  AC$CODE$ + 1002
; Maximum stack depth per invocation:  0 words

;    1836  1
;    1837  1
```

```
;    1838  1      %SBTTL 'GLOBAL ROUTINE   CLR DESCRIPTORS ( )
;    1839  1
;    1840  1      GLOBAL ROUTINE CLR_DESCRIPTORS : NOVALUE =
;    1841  1
;    1842  1      !++
;    1843  1      !
;    1844  1      !  GLOBAL ROUTINE :     CLR DESCRIPTORS
;    1845  1      !
;    1846  1      !  DESCRIPTION:
;    1847  1      !
;    1848  1      !       This routine initializes descriptor lists to zero.
;    1849  1      !--
;    1850  1
;    1851  1
;    1852  2      BEGIN
;    1853  2
;    1854  2      INCR INDEX FROM 0 TO DESCR_SIZE - 1 DO
;    1855  2        DESCR_LIST [ .INDEX, W_LEN ] = 0;
;    1856  2
;    1857  1      END;


                                        .SBTTL  CLR.DESCRIPTORS GLOBAL ROUTINE - CLR_DESCRIPTORS ( )
000000  005000                 CLR.DESCRIPTORS::
                                        CLR     R0                      ; INDEX                         1854
000002  005060  000000G        1$:     CLR     DESCR.LIST(R0)          ; =(INDEX)                      1855
000006  062700  000002                  ADD     #2,R0                   ; =,INDEX                       1854
000012  020027  000176                  CMP     R0,#176                 ; INDEX,=
000016  003771                          BLE     1$
000020  000207                          RTS     PC                      ;                               1840

; Routine Size:  9 words,      Routine Base:  AC$CODE$ + 1022
; Maximum stack depth per invocation:  0 words

;    1858  1
;    1859  1
;    1860  1
```

```
;    1861  1     %SBTTL 'GLOBAL ROUTINE   CLR RDESCR ( )'
;    1862  1
;    1863  1     GLOBAL ROUTINE CLR_RDESCR : NOVALUE =
;    1864  1
;    1865  1     !++
;    1866  1     !
;    1867  1     !  GLOBAL ROUTINE :     CLR_RDESCR
;    1868  1     !
;    1869  1     !  DESCRIPTION:
;    1870  1     !
;    1871  1     !       This routine initializes receive descriptor buffer to zero.
;    1872  1     !--
;    1873  1
;    1874  2     BEGIN
;    1875  2
;    1876  2     INCR INDEX FROM 0 TO D_SIZE - 1 DO
;    1877  2       RCV_D_LIST [ .INDEX, W_LEN ] = 0;
;    1878  2
;    1879  1     END;


                                        .SBTTL  CLR.RDESCR GLOBAL ROUTINE - CLR_RDESCR ( )
000000  005000                  CLR.RUESCR::
                                        CLR     R0                      ; INDEX                         1876
000002  005060  000000G         1$:     CLR     RCV.D.LIST(R0)          ; @(INDEX)                      1877
000006  062700  000002                  ADD     #2,R0                   ; @,INDEX                       1876
000012  020027  000076                  CMP     R0,#76                  ; INDEX,@
000016  003771                          BLE     1$
000020  000207                          RTS     PC                      ;                               1863

; Routine Size:  9 words,       Routine Base:  AC$CODE$ + 1044
; Maximum stack depth per invocation:  0 words

;    1880  1
;    1881  1
```

```
;    1882  1      %SBTTL 'GLOBAL ROUTINE   CLR XDESCR ( )'
;    1883  1
;    1884  1      GLOBAL ROUTINE CLR XDESCR : NOVALUE =
;    1885  1
;    1886  1      !++
;    1887  1      !
;    1888  1      !  GLOBAL ROUTINE :     CLR_XDESCR
;    1889  1      !
;    1890  1      !  DESCRIPTION:
;    1891  1      !
;    1892  1      !       This routine initializes transmit descriptor buffer to zero.
;    1893  1      !--
;    1894  1
;    1895  2      BEGIN
;    1896  2
;    1897  2      INCR INDEX FROM 0 TO D_SIZE - 1 DO
;    1898  2        XMIT_D_LIST [ .INDEX, W_LEN ] = 0;
;    1899  2
;    1900  1      END;


                                        .SBTTL  CLR.XDESCR GLOBAL ROUTINE - CLR_XDESCR ( )
000000  005000                  CLR.XDESCR::
                                        CLR     R0                      ; INDEX                         1897
000002  005060  000000G         1$:     CLR     XMIT.D.LIST(R0)         ; *(INDEX)                      1898
000006  062700  000002                    ADD     #2,R0                   ; *,INDEX                       1897
000012  020027  000076                    CMP     R0,#76                  ; INDEX,*
000016  003771                          BLE     1$
000020  000207                          RTS     PC                      ;                               1884

; Routine Size:  9 words,       Routine Base:  AC$CODE$ + 1066
; Maximum stack depth per invocation:  0 words

;    1901  1
;    1902  1
```

```
;    1903  1     %SBTTL 'GLOBAL ROUTINE   CLR_SETUP BUFFER ( )'
;    1904  1
;    1905  1     GLOBAL ROUTINE CLR_SETUP BUFFER : NOVALUE =
;    1906  1
;    1907  1     !++
;    1908  1     !
;    1909  1     !  GLOBAL ROUTINE :     CLR_SETUP_BUFFER
;    1910  1     !
;    1911  1     !  DESCRIPTION:
;    1912  1     !
;    1913  1     !       This routine clears setup mode buffer.
;    1914  1     !--
;    1915  1
;    1916  1
;    1917  2     BEGIN
;    1918  2
;    1919  2     INCR INDEX FROM 0 TO SETUB_SIZE - 1 DO
;    1920  2       SETUP_BUFFER [ .INDEX ] = 0;
;    1921  2
;    1922  1     END;


                                        .SBTTL  CLR.SETUP.BUFFER GLOBAL ROUTINE - CLR_SETUP_BUFFER ( )
000000  005000                  CLR.SETUP.BUFFER::
                                        CLR     R0                      ; INDEX                         1919
000002  005060  000000G         1$:     CLR     SETUP.BUFFER(R0)        ; @(INDEX)                      1920
000006  062700  000002                  ADD     #2,R0                   ; @,INDEX                       1919
000012  020027  000776                  CMP     R0,#776                 ; INDEX,@
000016  003771                          BLE     1$
000020  000207                          RTS     PC                      ;                               1905

; Routine Size:  9 words,      Routine Base:  AC$CODE$ + 1110
; Maximum stack depth per invocation:  0 words

;    1923  1
```

```
;    1924  1     %SBTTL 'GLOBAL ROUTINE   CHK CSR STATUS ( P1, P2 )'
;    1925  1
;    1926  1     GLOBAL ROUTINE CHK_CSR_STATUS ( P1, P2 ) : NOVALUE  =
;    1927  1
;    1928  1     !++
;    1929  1     !
;    1930  1     !  GLOBAL ROUTINE :     CHK_CSR_STATUS
;    1931  1     !
;    1932  1     !  DESCRIPTION:
;    1933  1     !
;    1934  1     !       This routine checks CSR status words for expected status.
;    1935  1     !
;    1936  1     !  INPUT PARAMETERS:
;    1937  1     !
;    1938  1     !        P1 - expected CSR status
;    1939  1     !        P2 - CSR mask
;    1940  1     !--
;    1941  1
;    1942  2     BEGIN
;    1943  2
;    1944  2     !++
;    1945  2     !  SAVE CSR, RESET TRANSMIT AND RECEIVE REQUEST BITS IN THE CSR
;    1946  2     !--
;    1947  2
;    1948  2     CSR_WORD = GET_BIT [ CSR, ALL_BITS ];
;    1949  2
;    1950  2     PUT_BIT [ CSR, RI, ONE ];
;    1951  2     PUT_BIT [ CSR, XI, ONE ];
;    1952  2
;    1953  2     !++
;    1954  2     !  MASK OUT DON'T CARE BITS IN THE CSR REGISTER AND COMPARE TO EXPECTED
;    1955  2     !  CSR STATUS. IF STATUS NOT EQUAL THEN PRINT 'BAD CSR STATUS WORD'
;    1956  2     !--
;    1957  2
;    1958  2     TEMP1 = .CSR_WORD AND .P2;
;    1959  2
;    1960  2     IF .TEMP1 NEQU .P1
;    1961  2       THEN
;    1962  3         BEGIN
;    1963  3           PRINTB ( MSG12, .TEMP1, .P1 );
;    1964  3           ERRDF ( 0001, E0001, ERROR%REPORT );
;    1965  3           DODU ( DEQNA_NO );
;    1966  3           DOCLN;
;    1967  2         END;
;    1968  1     END;


                                        .SBTTL  CHK.CSR.STATUS GLOBAL ROUTINE - CHK_CSR_STATUS ( P1, P2 )
000000  013700  000000G       CHK.CSR.STATUS::
                                        MOV     REG.ADR,R0                      ;                               1948
000004  062700  000016                  ADD     #16,R0
000010  011046                          MOV     (R0),-(SP)                      ; *,TMP.LOCATION
000012  011637  000000G                 MOV     (SP),CSR.WORD
```

```
000016  052710  100200                          BIS     #100200,(R0)                    ;                               1951
000022  011637  000000G                         MOV     (SP),TEMP1                      ; CSR.WORD,*                    1958
000026  016600  000004                          MOV     4(SP),R0                        ; P2,*
000032  005100                                  COM     R0
000034  040037  000000G                         BIC     R0,TEMP1
000040  023766  000000G 000006                  CMP     TEMP1,6(SP)                     ; *,P1                          1960
000046  001424                                  BEQ     1$
000050  016646  000006                          MOV     6(SP),-(SP)                     ; P1,*                          1963
000054  013746  000000G                         MOV     TEMP1,-(SP)
000060  012746  000000G                         MOV     #MSG12,-(SP)
000064  012746  000003                          MOV     #3,-(SP)
000070  010600                                  MOV     SP,R0                           ; SP,*
000072  104414                                  TRAP    14
000074  104455                                  TRAP    55                              ;                               1964
000076  000001                                  .WORD   1
000100  000000G                                 .WORD   E0001
000102  000000'                                 .WORD   ERROR$REPORT
000104  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                               1965
000110  104451                                  TRAP    51
000112  104444                                  TRAP    44
000114  062706  000010                          ADD     #10,SP                          ;                               1962
000120  005726                          1$:     TST     (SP)+                           ;                               1926
000122  000207                                  RTS     PC

; Routine Size:  42 words,      Routine Base:  AC$CODE$ + 1132
; Maximum stack depth per invocation:  7 words


;   1969  1
;   1970  1
```

```
;    1971  1     %SBTTL 'GLOBAL ROUTINE - CHK_XMIT STATUS ( P1, P2 )'
;    1972  1
;    1973  1     GLOBAL ROUTINE CHK_XMIT_STATUS ( P1, P2 ) : NOVALUE  =
;    1974  1
;    1975  1     !++
;    1976  1     !
;    1977  1     !  GLOBAL ROUTINE :     CHK_XMIT_STATUS
;    1978  1     !
;    1979  1     !  DESCRIPTION:
;    1980  1     !
;    1981  1     !       This routine checks transmit status words for expected status.
;    1982  1     !
;    1983  1     !  INPUT PARAMETERS:
;    1984  1     !
;    1985  1     !       P1 - XMIT flag word
;    1986  1     !       P2 - expected XMIT status word 1
;    1987  1     !
;    1988  1     !
;    1989  1     !-
;    1990  1
;    1991  2     BEGIN
;    1992  2
;    1993  2     !++
;    1994  2     !  MASK OUT DON'T CARE BITS IN THE XMIT FLAG WORD AND COMPARE TO EXPECTED
;    1995  2     !  XMIR FLAG STATUS. IF STATUS NOT EQUAL THEN PRINT 'BAD XMIT FLAG WORD
;    1996  2     !  STATUS'
;    1997  2     !--
;    1998  2
;    1999  2     TEMP2 = .XMIT_D_LIST [ FLGWD ] AND XFLG_MASK;
;    2000  2
;    2001  2     IF .TEMP2 NEQU .P1
;    2002  2       THEN
;    2003  3         BEGIN
;    2004  3           PRINTB ( MSG13, .GET_ADR [ CSR_ALL ], .TEMP2 );
;    2005  3           ERRDF ( 0802, E0001, ERROR$REPORT );
;    2006  3           DODU ( DEQNA_NO );
;    2007  3           DOCLN;
;    2008  2         END;
;    2009  2
;    2010  2     !++
;    2011  2     !  MASK OUT DON'T CARE BITS IN THE XMIT STATUS WD1 AND COMPARE TO EXPECTED
;    2012  2     !  XMIT STATUS WD1. IF STATUS NOT EQUAL THEN PRINT 'BAD XMIT STATUS WORD 1'
;    2013  2     !--
;    2014  2
;    2015  2     IF .XMIT_D_LIST [ STWD1 ] GTRU ZERO
;    2016  2       THEN
;    2017  2         TEMP3 = .XMIT_D_LIST [ STWD1 ] AND XWD1_MASK
;    2018  2       ELSE
;    2019  2         TEMP3 = .XMIT_D_LIST [ STWD1 ] AND X1_MASK;
;    2020  2
;    2021  2     IF .TEMP3 NEQU .P2
;    2022  2       THEN
;    2023  3         BEGIN
```

```
;    2024  3          PRINTB ( MSG14, .GET_ADR [ CSR_ALL ], .TEMP3 );
;    2025  3          ERRDF ( 0803, E0001, ERROR$REPORT );
;    2026  3          DODU ( DEQNA_NO );
;    2027  3          DOCLN;
;    2028  2        END;
;    2029  2
;    2030  2      !++
;    2031  2      !  CHECK THE TDR VALUE IN THE XMIT STATUS WORD 2
;    2032  2      !
;    2033  2
;    2034  2      !IF .XMIT_D_LIST [ STWD1 ] EQLU ZERO
;    2035  2      !  THEN
;    2036  2      !    PRINTB ( MSG_1TDR, .HWP_TABLE [ ADDR ], .ZERO )        ! THERE IS NO TDR
;    2037  2      !  ELSE
;    2038  2      !    BEGIN
;    2039  2      !      TEMP4 = .XMIT_D_LIST [ STWD2 ] AND XWD2_MASK;
;    2040  2      !      IF .TEMP4 GTRU ZERO
;    2041  2      !        THEN
;    2042  2      !          PRINTB ( MSG_2TDR, .HWP_TABLE [ ADDR ], .TEMP4 )          ! TDR > 0
;    2043  2      !        ELSE
;    2044  2      !          IF .TEMP4 EQLU ZERO
;    2045  2      !            THEN
;    2046  2      !              PRINTB ( MSG_3TDR, .HWP_TABLE [ ADDR ], .TEMP4 )     ! TDR = 0
;    2047  2      !            ELSE
;    2048  2      !              PRINTB ( MSG_4TDR, .HWP_TABLE [ ADDR ], .TEMP4 );    ! TDR < 0
;    2049  2      !    END;
;    2050  1      END;


                                        .SBTTL  CHK.XMIT.STATUS GLOBAL ROUTINE - CHK_XMIT_STATUS ( P1, P2 )
000000  024646                    CHK.XMIT.STATUS::
                                  CMP     -(SP),-(SP)                     ;                                       1973
000002  013737  000000G 000000G   MOV     XMIT.D.LIST,TEMP2               ;                                       1999
000010  042737  037777  000000G   BIC     #37777,TEMP2
000016  023766  000000G 000010    CMP     TEMP2,10(SP)                    ; *,P1                                  2001
000024  001432                    BEQ     1$
000026  013746  000000G           MOV     TEMP2,-(SP)                     ;                                       2004
000032  013766  000000G 000002    MOV     GET.ADR,2(SP)                   ; *,TMP.LOCATION
000040  062766  000016  000002    ADD     #16,2(SP)                       ; *,TMP.LOCATION
000046  016646  000002            MOV     2(SP),-(SP)                     ; TMP.LOCATION,*
000052  012746  000000G           MOV     #MSG13,-(SP)
000056  012746  000003            MOV     #3,-(SP)
000062  010600                    MOV     SP,R0                           ; SP,*
000064  104414                    TRAP    14
000066  104455                    TRAP    55                              ;                                       2005
000070  001442                    .WORD   1442
000072  000000G                   .WORD   E0001
000074  000000'                   .WORD   ERROR$REPORT
000076  012700  000000G           MOV     #DEQNA.NO,R0                    ;                                       2006
000102  104451                    TRAP    51
000104  104444                    TRAP    44
000106  062706  000010            ADD     #10,SP                          ;                                       2003
000112  013700  000010G   1$:     MOV     XMIT.D.LIST+10,R0               ;                                       2015
```

```
000116  001406                                  BEQ     2$
000120  010037  000000G                         MOV     R0,TEMP3                ;                                       2017
000124  042737  020017  000000G                 BIC     #20017,TEMP3
000132  000405                                  BR      3$                      ;                                       2015
000134  010037  000000G               2$:       MOV     R0,TEMP3                ;                                       2019
000140  042737  077777  000000G                 BIC     #77777,TEMP3
000146  023766  000000G 000006        3$:       CMP     TEMP3,6(SP)             ; *,P2                                  2021
000154  001432                                  BEQ     4$
000156  013746  000000G                         MOV     TEMP3,-(SP)             ;                                       2024
000162  013766  000000G 000004                 MOV     GET.ADR,4(SP)           ; *,TMP.LOCATION
000170  062766  000016  000004                 ADD     #16,4(SP)               ; *,TMP.LOCATION
000176  016646  000004                         MOV     4(SP),-(SP)             ; TMP.LOCATION,*
000202  012746  000000G                         MOV     #MSG14,-(SP)
000206  012746  000003                         MOV     #3,-(SP)
000212  010600                                  MOV     SP,R0                   ; SP,*
000214  104414                                  TRAP    14
000216  104455                                  TRAP    55                      ;                                       2025
000220  001443                                  .WORD   1443
000222  000000G                                 .WORD   E0001
000224  000000'                                 .WORD   ERROR$REPORT
000226  012700  000000G                         MOV     #DEQNA.NO,R0            ;                                       2026
000232  104451                                  TRAP    51
000234  104444                                  TRAP    44
000236  062706  000010                         ADD     #10,SP                  ;                                       2023
000242  022626                        4$:       CMP     (SP)+,(SP)+             ;                                       1973
000244  000207                                  RTS     PC

; Routine Size:  83 words,      Routine Base:  AC$CODE$ + 1256
; Maximum stack depth per invocation:  8 words


;   2051  1
;   2052  1
```

```
;    2053  1     %SBTTL 'GLOBAL ROUTINE - CHK_RCV_STATUS ( P1, P2 )'
;    2054  1
;    2055  1     GLOBAL ROUTINE CHK_RCV_STATUS ( P1, P2 ) : NOVALUE  =
;    2056  1
;    2057  1     !++
;    2058  1     !
;    2059  1     !  GLOBAL ROUTINE :     CHK_RCV_STATUS
;    2060  1     !
;    2061  1     !  DESCRIPTION:
;    2062  1     !
;    2063  1     !       This routine checks receive status words for expected status.
;    2064  1     !
;    2065  1     !  INPUT PARAMETERS:
;    2066  1     !
;    2067  1     !       P1 - expected RCV flag word
;    2068  1     !       P2 - expected RCV status word 1
;    2069  1     !
;    2070  1     !--
;    2071  1
;    2072  2     BEGIN
;    2073  2
;    2074  2     !++
;    2075  2     !  MASK OUT DON'T CARE BITS IN THE RCV FLAG WORD AND COMPARE TO EXPECTED
;    2076  2     !  RCV FLAG STATUS. IF STATUS NOT EQUAL THEN PRINT 'BAD RCV FLAG WORD
;    2077  2     !  STATUS'
;    2078  2     !--
;    2079  2
;    2080  2     TEMP1 = .RCV_D_LIST [ FLGWD ] AND RFLG_MASK;
;    2081  2
;    2082  2     IF .TEMP1 NEQU .P1
;    2083  2       THEN
;    2084  3         BEGIN
;    2085  3           PRINTB ( MSG15, .GET_ADR [ CSR_ALL ], .TEMP1 );    ! BAD RCV FLAG WD
;    2086  3           ERRDF ( 0804, E0001, ERROR$REPORT );
;    2087  3           DODU ( DEQNA_NO );
;    2088  3           DOCLN;
;    2089  2         END;
;    2090  2
;    2091  2     !++
;    2092  2     !  MASK OUT DON'T CARE BITS IN THE RCV STATUS WD1 AND COMPARE TO EXPECTED
;    2093  2     !  RCV STATUS WD1. IF STATUS NOT EQUAL THEN PRINT 'BAD RCV STATUS WORD 1'
;    2094  2     !--
;    2095  2
;    2096  2     IF .RCV_D_LIST [ STWD1 ] GTRU ZERO
;    2097  2       THEN
;    2098  2         TEMP2 = .RCV_D_LIST [ STWD1 ] AND R2_MASK
;    2099  2       ELSE
;    2100  2         TEMP2 = .RCV_D_LIST [ STWD1 ] AND R1_MASK;
;    2101  2
;    2102  2     IF .TEMP2 NEQU .P2
;    2103  2       THEN
;    2104  3         BEGIN
;    2105  3           PRINTB ( MSG16, .GET_ADR [ CSR_ALL ], .TEMP2 );    ! BAD RCV STATUS WD 1
```

```
;    2106  3          ERRDF ( 0805, E0001, ERROR$REPORT );
;    2107  3          DODU ( DEQNA_NO );
;    2108  3          DOCLN;
;    2109  2        END;
;    2110  2
;    2111  1    END;


                                        .SBTTL  CHK.RCV.STATUS GLOBAL ROUTINE - CHK_RCV_STATUS ( P1, P2 )
000000  024646                    CHK.RCV.STATUS::
                                        CMP     -(SP),-(SP)                     ;                               2055
000002  013737  000000G 000000G         MOV     RCV.D.LIST,TEMP1                ;                               2080
000010  042737  037777  000000G         BIC     #37777,TEMP1
000016  023766  000000G 000010          CMP     TEMP1,10(SP)                    ; *,P1                          2082
000024  001432                          BEQ     1$
000026  013746  0..0000G                MOV     TEMP1,-(SP)                     ;                               2085
000032  013766  000000G 000002          MOV     GET.ADR,2(SP)                   ; *,TMP.LOCATION
000040  062766  000016  000002          ADD     #16,2(SP)                       ; *,TMP.LOCATION
000046  016646  000002                  MOV     2(SP),-(SP)                     ; TMP.LOCATION,*
000052  012746  000000G                 MOV     #MSG15,-(SP)
000056  012746  000003                  MOV     #3,-(SP)
000062  010600                          MOV     SP,R0                           ; SP,*
000064  104414                          TRAP    14
000066  104455                          TRAP    55                              ;                               2086
000070  001444                          .WORD   1444
000072  000000G                         .WORD   E0001
000074  000000'                         .WORD   ERROR$REPORT
000076  012700  000000G                 MOV     #DEQNA.NO,R0                    ;                               2087
000102  104451                          TRAP    51
000104  104444                          TRAP    44
000106  062706  000010                  ADD     #10,SP                          ;                               2084
000112  013700  000010G         1$:     MOV     RCV.D.LIST+10,R0                ;                               2096
000116  001406                          BEQ     2$
000120  010037  000000G                 MOV     R0,TEMP2                        ;                               2098
000124  042737  003760  000000G         BIC     #3760,TEMP2
000132  000405                          BR      3$                              ;                               2096
000134  010037  000000G         2$:     MOV     R0,TEMP2                        ;                               2100
000140  042737  077777  000000G         BIC     #77777,TEMP2
000146  023766  000000G 000006  3$:     CMP     TEMP2,6(SP)                     ; *,P2                          2102
000154  001432                          BEQ     4$
000156  013746  000000G                 MOV     TEMP2,-(SP)                     ;                               2105
000162  013766  000000G 000004          MOV     GET.ADR,4(SP)                   ; *,TMP.LOCATION
000170  062766  000016  000004          ADD     #16,4(SP)                       ; *,TMP.LOCATION
000176  016646  000004                  MOV     4(SP),-(SP)                     ; TMP.LOCATION,*
000202  012746  000000G                 MOV     #MSG16,-(SP)
000206  012746  000003                  MOV     #3,-(SP)
000212  010600                          MOV     SP,R0                           ; SP,*
000214  104414                          TRAP    14
000216  104455                          TRAP    55                              ;                               2106
000220  001445                          .WORD   1445
000222  000000G                         .WORD   E0001
000224  000000'                         .WORD   ERROR$REPORT
000226  012700  000000G                 MOV     #DEQNA.NO,R0                    ;                               2107
```

```
000232  104451                                  TRAP    51
000234  104444                                  TRAP    44
000236  062706  000010                          ADD     #10,SP                  ;                                       2104
000242  022626                          4$:     CMP     (SP)+,(SP)+             ;                                       2055
000244  000207                                  RTS     PC

; Routine Size:  83 words,      Routine Base:  AC$CODE$ + 1524
; Maximum stack depth per invocation:  8 words


;   2112  1
```

```
;    2113  1     %SBTTL 'GLOBAL ROUTINE   COMPARE_PACKETS ( )'
;    2114  1
;    2115  1     GLOBAL ROUTINE COMPARE PACKETS : NOVALUE =
;    2116  1
;    2117  1     !++
;    2118  1     !
;    2119  1     !  GLOBAL ROUTINE :     COMPARE_PACKETS
;    2120  1     !
;    2121  1     !  DESCRIPTION:
;    2122  1     !
;    2123  1     !       This routine compares contents of transmit packet to the contents
;    2124  1     !       of receive packet and prints an error message if the don't compare.
;    2125  1     !--
;    2126  1
;    2127  2     BEGIN
;    2128  2
;    2129  2     !++
;    2130  2     !  GET RECEIVE BYTE LENGTH ( RBL ) FROM RCV DISCRIPTOR AND CUMPUTE WORD
;    2131  2     !  LENGTH. THEN COMPARE ACTUAL TO EXPECTED RCV WORD LENGTH.
;    2132  2     !--
;    2133  2
;    2134  2     TEMP3 = 0;
;    2135  2
;    2136  2     IF .XMIT_D_LIST [ SBIT ] EQLU ZERO
;    2137  2       THEN
;    2138  2         TEMP3 = .RCV_D_LIST [ STWD1 ] AND RHL_MASK;
;    2139  2
;    2140  2     TEMP3 = .TEMP3 + ( .RCV_D_LIST [ STWD2 ] AND RLL_MASK );
;    2141  2
;    2142  2     IF .TEMP3 NEQU .RBUF_LENGTH
;    2143  2       THEN
;    2144  3         BEGIN
;    2145  3           PRINTB ( MSG17, .GET_ADR [ CSR_ALL ], .TEMP3 );    ! BAD RBL (RCV LENGTH)
;    2146  3           ERRDF ( 0805, E0001, ERROR$REPORT );
;    2147  3           DODU ( DEQNA_NO );
;    2148  3           DOCLN;
;    2149  2         END;
;    2150  2
;    2151  2     INCR INDEX FROM 0 TO .TEMP3 - 1 DO
;    2152  2       IF .XMIT_BUFFER [ .INDEX ] NEQU .RCV_BUFFER [ .INDEX ]
;    2153  2         THEN
;    2154  3           BEGIN
;    2155  3             PRINTB ( DBM45, .XMIT_BUFFER [ .INDEX ], .RCV_BUFFER [ .INDEX ] );
;    2156  3             ERRDF ( 0806, E0001, ERROR$REPORT );
;    2157  3             DODU ( DEQNA_NO );
;    2158  3             DOCLN;
;    2159  2           END;
;    2160  2
;    2161  1     END;


                                        .SBTTL  COMPARE.PACKETS GLOBAL ROUTINE - COMPARE_PACKETS ( )
000000  004137  000000G              COMPARE.PACKETS::
```

```
                                                JSR     R1,$SAVE2                       ;                               2115
000004  005746                                  TST     (SP)
000006  005037  000000G                         CLR     TEMP3                           ;                               2134
000012  032737  010000  000002G                 BIT     #10000,XMIT.D.LIST+2            ;                               2136
000020  001006                                  BNE     1$
000022  013737  000010G 000000G                 MOV     RCV.D.LIST+10,TEMP3             ;                               2138
000030  042737  174377  000000G                 BIC     #174377,TEMP3
000036  005000                          1$:     CLR     R0                              ;                               2140
000040  153700  000012G                         BISB    RCV.D.LIST+12,R0
000044  060037  000000G                         ADD     R0,TEMP3
000050  023737  000000G 000000G                 CMP     TEMP3,RBUF.LENGTH               ;                               2142
000056  001432                                  BEQ     2$
000060  013746  000000G                         MOV     TEMP3,-(SP)                     ;                               2145
000064  013766  000000G 000002                 MOV     GET.ADR,2(SP)                   ; *,TMP.LOCATION
000072  062766  000016  000002                 ADD     #16,2(SP)                       ; *,TMP.LOCATION
000100  016646  000002                         MOV     2(SP),-(SP)                     ; TMP.LOCATION,*
000104  012746  000000G                         MOV     #MSG17, (SP)
000110  012746  000003                         MOV     #3,-(SP)
000114  010600                                  MOV     SP,R0                           ; SP,*
000116  104414                                  TRAP    14
000120  104455                                  TRAP    55                              ;                               2146
000122  001445                                  .WORD   1445
000124  000000G                                 .WORD   E0001
000126  000000'                                 .WORD   ERROR$REPORT
000130  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                               2147
000134  104451                                  TRAP    51
000136  104444                                  TRAP    44
000140  062706  000010                         ADD     #10,SP                          ;                               2144
000144  013702  000000G                 2$:     MOV     TEMP3,R2                        ;                               2151
000150  005001                                  CLR     R1                              ; INDEX
000152  000433                                  BR      5$
000154  126161  000000G 000000G         3$:     CMPB    XMIT.BUFFER(R1),RCV.BUFFER(R1)  ; *(INDEX),*(INDEX)             2152
000162  001426                                  BEQ     4$
000164  005046                                  CLR     -(SP)                           ;                               2155
000166  116116  000000G                         MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
000172  005046                                  CLR     -(SP)
000174  116116  000000G                         MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
000200  012746  000000G                         MOV     #DBM45,-(SP)
000204  012746  000003                         MOV     #3,-(SP)
000210  010600                                  MOV     SP,R0                           ; SP,*
000212  104414                                  TRAP    14
000214  104455                                  TRAP    55                              ;                               2156
000216  001446                                  .WORD   1446
000220  000000G                                 .WORD   E0001
000222  000000'                                 .WORD   ERROR$REPORT
000224  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                               2157
000230  104451                                  TRAP    51
000232  104444                                  TRAP    44
000234  062706  000010                         ADD     #10,SP                          ;                               2154
000240  005201                          4$:     INC     R1                              ; INDEX                         2151
000242  020102                          5$:     CMP     R1,R2                           ; INDEX,*
000244  002743                                  BLT     3$
000246  005726                                  TST     (SP)+                           ;                               2115
```

C13


```
000250  000207                                  RTS     PC

; Routine Size:  85 words,      Routine Base:  AC$CODE$ + 1772
; Maximum stack depth per invocation:  10 words


;   2162  1
;   2163  1
```

D13

```
;    2164  1     %SBTTL 'GLOBAL ROUTINE - SET_RDESCR LIST ( P1, P2)'
;    2165  1
;    2166  1     GLOBAL ROUTINE SET_RDESCR_LIST ( P1, P2 ) : NOVALUE  =
;    2167  1
;    2168  1     !++
;    2169  1     !
;    2170  1     !  GLOBAL ROUTINE :     SET_RDESCR LIST
;    2171  1     !
;    2172  1     !  DESCRIPTION:
;    2173  1     !
;    2174  1     !       This routine initializes receive descriptor list.
;    2175  1     !
;    2176  1     !  INPUT PARAMETERS:
;    2177  1     !
;    2178  1     !       P1 - expected Ethernet packet length in words
;    2179  1     !       P2 - expected RCV Descriptor List settings
;    2180  1     !
;    2181  1     !--
;    2182  1
;    2183  2     BEGIN
;    2184  2
;    2185  2       RCV_D_LIST [ FLGWD ]  = NEWB;
;    2186  2       RCV_D_LIST [ DBITS ]  = .P2;
;    2187  2       RCV_D_LIST [ LOADR ]  = RCV_BUFFER;
;    2188  2       RCV_D_LIST [ TWDL  ]  = .P1;
;    2189  2       RCV_D_LIST [ STWD1 ]  = 0;
;    2190  2       RCV_D_LIST [ STWD2 ]  = 0;
;    2191  2       RCV_D_LIST [ DLINK ]  = NEWB;
;    2192  2       RCV_D_LIST [ BSTAT ]  = E;
;    2193  2
;    2194  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                     ;                               2185
000006  016637  000002  000002G         MOV     2(SP),RCV.D.LIST+2                      ; P2.*                          2186
000014  012737  000000G 000004G         MOV     #RCV.BUFFER,RCV.D.LIST+4                ;                               2187
000022  016637  000004  000006G         MOV     4(SP),RCV.D.LIST+6                      ; P1.*                          2188
000030  005037  000010G                 CLR     RCV.D.LIST+10                           ;                               2189
000034  005037  000012G                 CLR     RCV.D.LIST+12                           ;                               2190
000040  012737  100000  000014G         MOV     #-100000,RCV.D.LIST+14                  ;                               2191
000046  012737  020000  000016G         MOV     #20000,RCV.D.LIST+16                    ;                               2192
000054  000207                         RTS     PC                                      ;                               2166

; Routine Size:  23 words,      Routine Base:  AC$CODE$ + 2244
; Maximum stack depth per invocation:  0 words

;    2195  1
```

```
;   2196  1     %SBTTL 'GLOBAL ROUTINE   SET XDESCR LIST ( P1, P2 )'
;   2197  1
;   2198  1     GLOBAL ROUTINE SET_XDESCR LIST ( P1, P2 ) : NOVALUE  =
;   2199  1
;   2200  1     !++
;   2201  1     !
;   2202  1     !  GLOBAL ROUTINE :     SET_XDESCR_LIST
;   2203  1     !
;   2204  1     !  DESCRIPTION:
;   2205  1     !
;   2206  1     !       This routine initializes transmit descriptor list.
;   2207  1     !
;   2208  1     !  INPUT PARAMETERS:
;   2209  1     !
;   2210  1     !       P1 - expected Ethernet packet length in words
;   2211  1     !       P2 - expected XMIT Descriptor List settings
;   2212  1     !
;   2213  1     !--
;   2214  1
;   2215  2     BEGIN
;   2216  2
;   2217  2       XMIT_D_LIST [ FLGWD ] = NEWB;
;   2218  2       XMIT_D_LIST [ DBITS ] = .P2;
;   2219  2       XMIT_D_LIST [ LOADR ] = XMIT_BUFFER;
;   2220  2       XMIT_D_LIST [ TWDL  ] = .P1;
;   2221  2       XMIT_D_LIST [ STWD1 ] = 0;
;   2222  2       XMIT_D_LIST [ STWD2 ] = 0;
;   2223  2       XMIT_D_LIST [ DLINK ] = NEWB;
;   2224  2       XMIT_D_LIST [ BSTAT ] = E;
;   2225  2
;   2226  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            ;                       2217
000006  016637  000002  000002G         MOV     2(SP),XMIT.D.LIST+2             ; P2,*                  2218
000014  012737  000000G 000004G         MOV     #XMIT.BUFFER,XMIT.D.LIST+4      ;                       2219
000022  016637  000004  000006G         MOV     4(SP),XMIT.D.LIST+6             ; P1,*                  2220
000030  005037  000010G                 CLR     XMIT.D.LIST+10                  ;                       2221
000034  005037  000012G                 CLR     XMIT.D.LIST+12                  ;                       2222
000040  012737  100000  000014G         MOV     #-100000,XMIT.D.LIST+14         ;                       2223
000046  012737  020000  000016G         MOV     #20000,XMIT.D.LIST+16           ;                       2224
000054  000207                         RTS     PC                              ;                       2198

; Routine Size:  23 words,      Routine Base:  AC$CODE$ + 2322
; Maximum stack depth per invocation:  0 words


;   2227  1
```

```
;    2228  1     %SBTTL 'GLOBAL ROUTINE   WALKING BIT ( P1, P2)'
;    2229  1
;    2230  1     GLOBAL ROUTINE WALKING_BIT ( P1, P2) : NOVALUE  =
;    2231  1
;    2232  1     !++
;    2233  1     !
;    2234  1     !  GLOBAL ROUTINE :     WALKING_BIT
;    2235  1     !
;    2236  1     !  DESCRIPTION:
;    2237  1     !
;    2238  1     !       This routine sets a b t in specified bit position.
;    2239  1     !
;    2240  1     !  INPUT PARAMETERS:
;    2241  1     !
;    2242  1     !       P1 - bit ( 0 or 1 )
;    2243  1     !       P2 - bit position from base address
;    2244  1     !
;    2245  1     !--
;    2246  1
;    2247  2     BEGIN
;    2248  2
;    2249  2     IF .P1 EQLU ZERO
;    2250  2       THEN
;    2251  3         BEGIN
;    2252  3           TBYTE1 = %B'00000001';
;    2253  3           TBYTE2 = %B'00000000';
;    2254  3         END
;    2255  2       ELSE
;    2256  3         BEGIN
;    2257  3           TBYTE1 = %B'11111110';
;    2258  3           TBYTE2 = %B'11111111';
;    2259  2         END;
;    2260  2
;    2261  2     SELECTONE .P2 OF
;    2262  2       SET
;    2263  2       [  0 TO  7 ]:
;    2264  2                     TEMP1 = 0;
;    2265  2       [  8 TO 47 ]:
;    2266  2                     TEMP1 = .P2 / 8;
;    2267  2       TES;
;    2268  2
;    2269  2     TEMP2 = .P2 MOD 8;
;    2270  2
;    2271  2     INCR INDEX FROM 0 TO .TEMP1 DO
;    2272  3       BEGIN
;    2273  3         IF .INDEX EQLU ZERO
;    2274  3           THEN
;    2275  3             EXITLOOP;
;    2276  3         XMIT_BUFFER [ .INDEX ] = .TBYTE2;
;    2277  3         PRINTB ( D8M45, .XMIT_BUFFER [ .INDEX ], .RCV_BUFFER [ .INDEX ] );
;    2278  2       END;
;    2279  2
;    2280  2     XMIT_BUFFER [ .TEMP1 + 1 ] = .TBYTE1 ^ ( .TEMP2 );
```

```
;   2281  2
;   2282  2     PRINTB ( DBM45, .XMIT BUFFER [ .TEMP1 + 1 ], .RCV BUFFER [ .TEMP1 + 1 ] );
;   2283  2
;   2284  1     END;


                                        .SBTTL  WALKING.BIT GLOBAL ROUTINE   WALKING BIT ( P1, P2)
000000  004137  000000G           WALKING.BIT::
                                    JSR     R1,$SAVE2                       ;                       2230
000004  005766  000012              TST     12(SP)                          ; P1                    2249
000010  001006                      BNE     1$
000012  112737  000001  000000G     MOVB    #1,TBYTE1                       ;                       2252
000020  105037  000000G             CLRB    TBYTE2                          ;                       2253
000024  000406                      BR      2$                              ;                       2249
000026  112737  000376  000000G 1$: MOVB    #376,TBYTE1                     ;                       2257
000034  112737  000377  000000G     MOVB    #377,TBYTE2                     ;                       2258
000042  016602  000010          2$: MOV     10(SP),R2                       ; P2,*                  2261
000046  002406                      BLT     3$                              ;                       2263
000050  020227  000007              CMP     R2,#7
000054  003003                      BGT     3$
000056  005037  000000G             CLR     TEMP1                           ;                       2264
000062  000414                      BR      4$                              ;                       2261
000064  020227  000010          3$: CMP     R2,#10                          ;                       2265
000070  002411                      BLT     4$
000072  020227  000057              CMP     R2,#57
000076  003006                      BGT     4$
000100  010201                      MOV     R2,R1                           ;                       2266
000102  006700                      SXT     R0
000104  071027  000010              DIV     #10,R0
000110  010037  000000G             MOV     R0,TEMP1
000114  010201              4$:     MOV     R2,R1                           ;                       2269
000116  006700                      SXT     R0
000120  071027  000010              DIV     #10,R0
000124  010137  000000G             MOV     R1,TEMP2
000130  013702  000000G             MOV     TEMP1,R2                        ;                       2271
000134  005001                      CLR     R1                              ; INDEX
000136  000424                      BR      6$
000140  005701              5$:     TST     R1                              ; INDEX                 2273
000142  001424                      BEQ     7$                              ;                       2275
000144  113761  000000G 000000G     MOVB    TBYTE2,XMIT.BUFFER(R1)          ; *,*(INDEX)            2276
000152  005046                      CLR     -(SP)                           ;                       2277
000154  116116  000000G             MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
000160  005046                      CLR     -(SP)
000162  116116  000000G             MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
000166  012746  000000G             MOV     #DBM45,-(SP)
000172  012746  000003              MOV     #3,-(SP)
000176  010600                      MOV     SP,R0                           ; SP,*
000200  104414                      TRAP    14
000202  062706  000010              ADD     #10,SP                          ;                       2272
000206  005201                      INC     R1                              ; INDEX                 2271
000210  020102              6$:     CMP     R1,R2                           ; INDEX,*
000212  003752                      BLE     5$
000214  013701  000000G     7$:     MOV     TEMP1,R1                        ;                       2280
```

H13


```
000220  005000                          CLR     R0
000222  153700  000000G                 BISB    TBYTE1,R0
000226  072037  000000G                 ASH     TEMP2,R0
000232  110061  000001G                 MOVB    R0,XMIT.BUFFER+1(R1)
000236  005046                          CLR     -(SP)                           ;                       2282
000240  116116  000001G                 MOVB    RCV.BUFFER+1(R1),(SP)
000244  005046                          CLR     -(SP)
000246  110016                          MOVB    R0,(SP)
000250  012746  000000G                 MOV     #DBM45,-(SP)
000254  012746  000003                  MOV     #3,-(SP)
000260  010600                          MOV     SP,R0                           ; SP,*
000262  104414                          TRAP    14
000264  062706  000010                  ADD     #10,SP                          ;                       2247
000270  000207                          RTS     PC                              ;                       2230

; Routine Size:  93 words,    Routine Base:  AC$CODE$ + 2400
; Maximum stack depth per invocation:  9 words


;   2285  1
```

```
;    2286  1     %SBTTL 'GLOBAL ROUTINE   WRT STATION ADR ( P1, P2 )'
;    2287  1
;    2288  1     GLOBAL ROUTINE WRT STATION ADR ( P1, P2 ): NOVALUE  =
;    2289  1
;    2290  1     !++
;    2291  1     !
;    2292  1     !  GLOBAL ROUTINE :     WRT_STATION ADR
;    2293  1     !
;    2294  1     !  DESCRIPTION:
;    2295  1     !
;    2296  1     !       This routine sets a b't in specified bit position.
;    2297  1     !
;    2298  1     !  INPUT PARAMETERS:
;    2299  1     !
;    2300  1     !       P1 - Ethernet Station Address index (1:14) in Station Address RAM
;    2301  1     !       P2 - Ethernet Station Address index ( 0:19 ) in the TARGET_ADR table
;    2302  1     !
;    2303  1     !--
;    2304  1
;    2305  2     BEGIN
;    2306  2
;    2307  2     TEMP3 = 8;
;    2308  2     COUNTER = 0;
;    2309  2     TEMP1 = ( .P2 - 1 ) * 6;
;    2310  2
;    2311  2     IF .P2 EQLU ZERO
;    2312  2       THEN
;    2313  2         TEMP1 = .P2;
;    2314  2
;    2315  2     SELECTONE .P1 OF
;    2316  2       SET
;    2317  2       [  0 TO  7 ]:
;    2318  3                     BEGIN
;    2319  3                       TEMP2 = 0;
;    2320  3                       TEMP3 = 1;
;    2321  2                     END;
;    2322  2       [  8 TO 14 ]:
;    2323  3                     BEGIN
;    2324  3                       TEMP2 = 40;
;    2325  3                       P1 = .P1 + 64;
;    2326  2                     END;
;    2327  2       TES;
;    2328  2
;    2329  2     INCR INDEX FROM .P1 TO .P1 + .TEMP2 BY .TEMP3 DO
;    2330  3       BEGIN
;    2331  3         XMIT_BUFFER [ .INDEX ] = .TARGET_ADR [ .COUNTER + .TEMP1 ];
;    2332  3         COUNTER = .COUNTER + 1;
;    2333  3         PRINTB ( DBM45, .XMIT_BUFFER [ .INDEX ], .RCV_BUFFER [ .INDEX ] );
;    2334  2       END;
;    2335  2
;    2336  1     END;
```

```
                                        .SBTTL  WRT.STATION.ADR GLOBAL ROUTINE - WRT_STATION_ADR ( P1, P2 )
000000  004137  000000G           WRT.STATION.ADR::
                                        JSR     R1,$SAVE3                       ;                                       2288
000004  012737  000010  000000G           MOV     #10,TEMP3                       ;                                       2307
000012  005037  000000G                   CLR     COUNTER                         ;                                       2308
000016  016600  000012                    MOV     12(SP),R0                       ; P2,*                                  2309
000022  010001                            MOV     R0,R1
000024  005301                            DEC     R1
000026  070127  000006                    MUL     #6,R1
000032  010137  000000G                   MOV     R1,TEMP1
000036  005700                            TST     R0                              ;                                       2311
000040  001002                            BNE     1$
000042  010037  000000G                   MOV     R0,TEMP1                        ;                                       2313
000046  016600  000014          1$:       MOV     14(SP),R0                       ; P1,*                                  2315
000052  002411                            BLT     2$                              ;                                       2317
000054  020027  000007                    CMP     R0,#7
000060  003006                            BGT     2$
000062  005037  000000G                   CLR     TEMP2                           ;                                       2319
000066  012737  000001  000000G           MOV     #1,TEMP3                        ;                                       2320
000074  000414                            BR      3$                              ;                                       2315
000076  020027  000010          2$:       CMP     R0,#10                          ;                                       2322
000102  002411                            BLT     3$
000104  020027  000016                    CMP     R0,#16
000110  003006                            BGT     3$
000112  012737  000050  000000G           MOV     #50,TEMP2                       ;                                       2324
000120  062766  000100  000014           ADD     #100,14(SP)                     ; *,P1                                  2325
000126  016602  000014          3$:       MOV     14(SP),R2                       ; P1,*                                  2329
000132  063702  000000G                   ADD     TEMP2,R2
000136  013703  000000G                   MOV     TEMP3,R3
000142  016601  000014                    MOV     14(SP),R1                       ; P1,INDEX
000146  000430                            BR      5$
000150  013700  000000G          4$:       MOV     COUNTER,R0                      ;                                       2331
000154  063700  000000G                   ADD     TEMP1,R0
000160  116061  000000G 000000G           MOVB    TARGET.ADR(R0),XMIT.BUFFER(R1)  ; *,*(INDEX)
000166  005237  000000G                   INC     COUNTER                         ;                                       2332
000172  005046                            CLR     -(SP)                           ;                                       2333
000174  116116  000000G                   MOVB    RCV.BUFFER(R1),(SP)             ; *(INDEX),*
000200  005046                            CLR     -(SP)
000202  116116  000000G                   MOVB    XMIT.BUFFER(R1),(SP)            ; *(INDEX),*
000206  012746  000000G                   MOV     #DBM45,-(SP)
000212  012746  000003                    MOV     #3,-(SP)
000216  010600                            MOV     SP,R0                           ; SP,*
000220  104414                            TRAP    14
000222  062706  000010                    ADD     #10,SP                          ;                                       2330
000226  060301                            ADD     R3,R1                           ; *,INDEX                               2329
000230  020102          5$:               CMP     R1,R2                           ; INDEX,*
000232  003746                            BLE     4$
000234  000207                            RTS     PC                              ;                                       2288

; Routine Size:  79 words,      Routine Base:  AC$CODE$ + 2672
; Maximum stack depth per invocation:  10 words
```

```
;    2338  1     %SBTTL 'GLOBAL ROUTINE   XMIT AND RCV PACKET '
;    2339  1
;    2340  1     GLOBAL ROUTINE XMIT_AND RCV_PACKET : NOVALUE  =
;    2341  1
;    2342  1     !++
;    2343  1     !
;    2344  1     !  GLOBAL ROUTINE :     XMIT_AND RCV PACKET
;    2345  1     !
;    2346  1     !  DESCRIPTION:
;    2347  1     !
;    2348  1     !       This routine initiates transm't and receive operations.
;    2349  1     !
;    2350  1     !  INPUT PARAMETERS:
;    2351  1     !
;    2352  1     !
;    2353  1     !
;    2354  1     !
;    2355  1     !--
;    2356  1
;    2357  2     BEGIN
;    2358  2
;    2359  2          .IOP_TABLE [ XLO_ADR ] = XMIT_D_LIST;
;    2360  2          .IOP_TABLE [ XHI_ADR ] = 0;
;    2361  2
;    2362  2          .IOP_TABLE [ RLO_ADR ] = RCV_D_LIST;
;    2363  2          .IOP_TABLE [ RHI_ADR ] = 0;
;    2364  2
;    2365  2          DELAY ( 300 );
;    2366  2
;    2367  2          IF GET_BIT [ CSR, RI ] NEQU ONE
;    2368  2            THEN
;    2369  3              BEGIN
;    2370  3                ERRDF ( 0801, E0001, ERROR$REPORT );
;    2371  3                DODU ( DEQNA_NO );
;    2372  3                DOCLN;
;    2373  2              END;
;    2374  1     END;


                                        .SBTTL  XMIT.AND.RCV.PACKET GLOBAL ROUTINE - XMIT_AND RCV_PACKET
000000  010146                  XMIT.AND.RCV.PACKET::
                                        MOV     R1,-(SP)                        ;                       2340
000002  024646                          CMP     -(SP),-(SP)
000004  012777  000000G 000010G        MOV     #XMIT.D.LIST,@IOP.TABLE+10      ;                       2359
000012  005077  000012G                 CLR     @IOP.TABLE+12                   ;                       2360
000016  012777  000000G 000004G        MOV     #RCV.D.LIST,@IOP.TABLE+4        ;                       2362
000024  005077  000006G                 CLR     @IOP.TABLE+6                    ;                       2363
000030  012701  000454                  MOV     #454,R1                         ; .,$$TMP2              2365
000034  001410                  1$:     BEQ     4$
000036  013700  000000G                 MOV     L$DLY,R0                        ; .,$$TMP1
000042  001403                          BEQ     3$
000044  005066  000002          2$:     CLR     2(SP)                           ; $$TMP
000050  077003                          SOB     R0,2$                           ; $$TMP1,.
```

```
000052  005301                          3$:     DEC     R1                              ; $$TMP2
000054  000767                                  BR      1$
000056  013700  000000G                 4$:     MOV     REG.ADR,R0                      ;                                       2367
000062  016016  000016                          MOV     16(R0),(SP)                     ; *,TMP.LOCATION
000066  100410                                  BMI     5$
000070  104455                                  TRAP    55                              ;                                       2370
000072  001441                                  .WORD   1441
000074  000000G                                 .WORD   E0001
000076  000000'                                 .WORD   ERROR$REPORT
000100  012700  000000G                         MOV     #DEQNA.NO,R0                    ;                                       2371
000104  104451                                  TRAP    51
000106  104444                                  TRAP    44
000110  022626                          5$:     CMP     (SP)+,(SP)+                     ;                                       2340
000112  012601                                  MOV     (SP)+,R1
000114  000207                                  RTS     PC

; Routine Size:  39 words,      Routine Base:  AC$CODE$ + 3130
; Maximum stack depth per invocation:  5 words


;   2375  1
;   2376  1     END
;   2377  0     ELUDOM



;                                   OTS external references
                                            .GLOBL  $SAVE3, $SAVE2

;                                   PSECT SUMMARY
;
;       Psect Name                  Words     Attributes
;        AC$CODE$                     851       RO ,  I  ,  LCL,  REL,  CON


;                               Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[MAZURCZYK.SDC]QNALIB.L16;2
;                                         157        86        54        12          00:00.1


;                                   COMMAND QUALIFIERS
```

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

; Size:          851 code + 0 data words
; Run Time:        00:33.4
; Elapsed Time:    01:48.1
; Lines/CPU Min:    4273
; Lexemes/CPU-Min: 30149
; Memory Used:  194 pages
; Compilation Complete
```

```
;    0001  0     MODULE ZQNA5 (%TITLE 'CZQNAA0 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
```

```
;     1502  2      LASTAD
;     1503  2      BGNSETUP(1);                                      ! NUMBER OF P TABLES
;  P  1504  2      BGNPTAB
;     1505  2      ENDPTAB
;     1506  1      ENDSETUP


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

000000                                         .PSECT  $XYZ$,  RO
000000  000004'                       BL$LAS::.WORD    T$FREE
0000C2  00000C                                .WORD    <<T$FREE-<BL$LAS+4>>/2>
000004  000000                        T$FREE::.WORD    0

        000004'                       L$LAST==             BL$LAS+4
        000001                        T$PTHV==             1

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

; Routine Size:  1 word,        Routine Base:  $XYZ$ + 0006
; Maximum stack depth per invocation:  0 words


;     1507  1
;     1508  1      END
;     1509  0      ELUDOM


;                                     PSECT SUMMARY
;
;       Psect Name                    Words     Attributes
;        $XYZ$                           4        RO ,  I  ,  LCL,  REL,  CON


;                            Library Statistics
;
;                                     -------- Symbols --------          Pages        Processing
;       File                          Total    Loaded    Percent        Mapped       Time
;
;  DISK$USER2:[MAZURCZYK.SDC]QNALIB.L16;2
;                                      157        1          0            12          00:00.1
```

```
;                                       COMMAND QUALIFIERS

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

; Size:          1 code + 3 data words
; Run Time:        00:06.3
; Elapsed Time:    00:09.6
; Lines/CPU Min:   14463
; Lexemes/CPU-Min: 76130
; Memory Used:  100 pages
; Compilation Complete
```

```
;    0001  0     !++
;    0002  0     !
;    0003  0     !  DEFINE DATA STRUCTURES IN THIS SECTION
;    0004  0     !
;    0005  0     !--
;    0006  0
;    0007  0     STRUCTURE                                       ! DEFINE ACCESS ALGORITHM
;    0008  0        REG_STR [ O, P, S, E ]=
;    0009  1           BEGIN
;    0010  1              LOCAL TMP_LOCATION;
;    0011  1              TMP_LOCATION = .(REG_STR + %UPVAL * O) <0,%BPVAL,0>;
;    0012  1              TMP_LOCATION
;    0013  0           END < P, S, E >;
;    0014  0
;    0015  0
;    0016  0     STRUCTURE                                       ! DEFINE ACCESS ALGORITHM
;    0017  0        ADR_STR [ O, P, S, E ]=
;    0018  1           BEGIN
;    0019  1              LOCAL TMP_LOCATION;
;    0020  1              TMP_LOCATION = (ADR_STR + %UPVAL * O) <0,%BPVAL,0>;
;    0021  1              TMP_LOCATION
;    0022  0           END < P, S, E >;
;    0023  0
;    0024  0
```

```
;    0025  0      !++
;    0026  0      !
;    0027  0      !   MACRO DEFINITIONS
;    0028  0      !
;    0029  0      !--
;    0030  0
;    0031  0      MACRO
;    0032  0
; M  0033  0          TST_BIT ( ADDR, EXPECTED ) =
; M  0034  0              ( IF ( .ADDR AND EXPECTED ) EQLU EXPECTED
; M  0035  0                THEN
; M  0036  0                   TRUE
; M  0037  0                ELSE
;    0038  0                   FALSE )%,
;    0039  0
;    0040  0
; M  0041  0          PUT_BIT ( OFFSET, POSITION, IMAGE ) =
; M  0042  0              BEGIN
; M  0043  0              ( .REG_ADR + %UPVAL * OFFSET )< %FIELDEXPAND ( POSITION ) > = IMAGE;
;    0044  0              END%,
;    0045  0
; M  0046  0          GET_STATION_ADR ( OFFSET, POSITION, IMAGE ) =
; M  0047  0              BEGIN
; M  0048  0              ( .STATION_ADR + OFFSET )< %FIELDEXPAND ( POSITION ) > = IMAGE;
;    0049  0              END%,
;    0050  0
;    0051  0
;    0052  0      !++
;    0053  0      !
;    0054  0      !  THIS MACRO GETS BITS SPECIFIED BY THE FIELD NAME " POSITION "
;    0055  0      !  AND MEMORY LOC SPECIFIED BY (.REG_ADR + %UPVAL * OFFSET)
;    0056  0      !
;    0057  0      !-
;    0058  0
; M  0059  0          GET_BIT ( OFFSET, POSITION ) =
; M  0060  0
;    0061  0                  .REG_ADR [ OFFSET, POSITION ] %;
;    0062  0
```

```
;  0063  0
;  0064  0     !++
;  0065  0     !
;  0066  0     !   PROGRAM LITERALS
;  0067  0     !
;  0068  0     ! -
;  0069  0
;  0070  0     LITERAL
;  0071  0        FALSE          = 0,          !
;  0072  0        TRUE           = 1,          !
;  0073  0        ZERO           = 0,          !
;  0074  0        ONE            = 1,          !
;  0075  0
;  0076  0        P_CLOCK        = 1,          !
;  0077  0        L_CLOCK        = 1,          !
;  0078  0        NO_CLOCK       = 0,          !
;  0079  0        CLEAR_FLG      = 0,          !
;  0080  0        SET_FLG        = 1,          !
;  0081  0        PWR_DELAY      = 10000,      !
;  0082  0        M1_DELAY       = 10,         !
;  0083  0        M2_DELAY       = 20,         !
;  0084  0        M3_DELAY       = 30,         !
;  0085  0        M4_DELAY       = 40,         !
;  0086  0        M5_DELAY       = 50,         !
;  0087  0
;  0088  0        RLO_ADR        = 2,          !
;  0089  0        RHI_ADR        = 3,          !
;  0090  0        XLO_ADR        = 4,          !
;  0091  0        XHI_ADR        = 5,          !
;  0092  0        IOP_LO_ADR     = 2,          !
;  0093  0        IOP_HI_ADR     = 3,          !
;  0094  0        IOP_SIZE       = %O'1E',     ! I/O PAGE REGISTER SIZE
;  0095  0        IOP_ADR        = 0,          ! OFFSET TO DEVICE ADDRESS
;  0096  0        IOP_VEC        = 2,          ! OFFSET TO DEVICE VECTOR ADDRESS
;  0097  0        IOP_BRL        = 4,          ! OFFSET TO DEVICE BR LEVEL
;  0098  0        INT_VEC        = 6,          !
;  0099  0
;  0100  0        CSR            = 7,          !
;  0101  0        WORD_LIMIT     = %O'177777', !
;  0102  0
;  0103  0        PATRN1         = %O'001411', ! CSR STATIC BITS
;  0104  0        PATRN2         = %O'011471', ! CSR STATIC BITS
;  0105  0
;  0106  0     !++
;  0107  0     !
;  0108  0     !  DESCRIPTOR LIST BUFFER OFFSETS
;  0109  0     !
;  0110  0     !--
;  0111  0
;  0112  0         D_FLAG_WD     = 0,          ! STATUS WORD 0, FLAG WORD
;  0113  0         D_DESCR_BITS  = 1,          !
;  0114  0         D_HI_ADR      = 1,          !
;  0115  0         D_LO_ADR      = 2,          !
```

```
;   0116  0          D_WD_COUNT    = 3,            !
;   0117  0          D_WD1_STATUS  = 4,            !
;   0118  0          D_WD2_STATUS  = 5,            !
;   0119  0
;   0120  0          T_SIZE        = 120,
;   0121  0          DESCR_SIZE    = 64,
;   0122  0          D_SIZE        = DESCR_SIZE / 2,
;   0123  0          BD_D_SIZE     = 16,
;   0124  0          BUF_SIZE      = 4096 * 2,
;   0125  0          B_SIZE        = BUF_SIZE / 2,
;   0126  0          SETUB_SIZE    = 256,
;   0127  0          BYTE_COUNT    = - ( BUF_SIZE / 4 ),
;   0128  0          PROM_SIZE     = 4096,
;   0129  0          CHSUM_OFFSET  = 6,
;   0130  0
;   0131  0          NXM_LO_ADR    = %O'160000',          ! NXM ADDRESS - LOW ORDER BITS
;   0132  0          NXM_HI_ADR    = %O'000077',          ! NXM ADDRESS - HIGH ORDER BITS
;   0133  0
;   0134  0
;   0135  0          XFLG_MASK     = %O'140000',          ! TRANSMIT FLAG WORD MASK BITS
;   0136  0          X1_MASK       = %O'100000',          ! TRANSMIT STATUS WD 1 MASK BITS
;   0137  0          XWD1_MASK     = %O'157760',          ! TRANSMIT STATUS WD 1 MASK BITS
;   0138  0          XWD2_MASK     = %O'140000',          ! TRANSMIT STATUS WD 2 MASK BITS
;   0139  0          XFLG_STATUS   = %O'140000',          ! EXPECTED TRANSMIT FLAG WORD
;   0140  0          XWD12_STATUS  = %O'000400',          ! EXPECTED TRANSMIT STATUS WD 1
;   0141  0                                               !  BIT 8 IS SET IN INTERNAL LOOOPBACK MODES
;   0142  0          XWD11_STATUS  = %O'000000',          !  BIT 8 IS RESET IN EXTERNAL LOOOPBACK MODES
;   0143  0
;   0144  0
;   0145  0          RFLG_MASK     = %O'140000',          ! RECEIVE FLAG WORD MASK BITS
;   0146  0          R1_MASK       = %O'100000',          ! RECEIVE STATUS WD 1 MASK BITS
;   0147  0          R2_MASK       = %O'174017',          ! RECEIVE STATUS WD 1 MASK BITS
;   0148  0          RWD1_MASK     = %O'140000',          ! RECEIVE STATUS WD 1 MASK BITS
;   0149  0          RWD1_STATUS   = %O'020000',          ! EXPECTED RECEIVE STATUS WD 1
;   0150  0          RFLG_STATUS   = %O'140000',          ! EXPECTED RECEIVE FLAG WORD
;   0151  0
;   0152  0
;   0153  0          RHL_MASK      = %O'003400',          ! RCV HIGH ORDER LENGTH BITS
;   0154  0          RLL_MASK      = %O'000377',          ! RCV LOW ORDER LENGTH BITS
;   0155  0
;   0156  0          SA_RBL        = %O'177775',          ! STATION ADR RCV BUF LENGTH - 3 WDS
;   0157  0
;   0158  0          PKT_LENGTH    = %DECIMAL'1500',      ! PACKET LENGTH
;   0159  0          LPB_PKT       = %O'0220',            ! LOOPBACK PACKET
;   0160  0          PKT_TYPE      = %DECIMAL'12',        ! PACKET TYPE
;   0161  0          SKIP_CNT      = 0,                   !
;   0162  0          RFC           = 1,                   !
;   0163  0          PKT_DATA      = %DECIMAL'15',        !
;   0164  0          LSPL          = %DECIMAL'1514',      ! LONGEST SETUP PACKET LENGTH
;   0165  0
;   0166  0      !++
;   0167  0      !  BUFFER DESCRIPTOR / CHAIN DESCRIPTOR BIT DEFINITIONS
;   0168  0      !--
```

```
;  0169  0
;  0170  0         V     = %O'100000',                ! VALID ADDRESS IF 1
;  0171  0         C     = %O'040000',                ! CHAIN ADDRESS IF 1
;  0172  0         E     = %O'020000',                ! END OF MESSAGE IF 1
;  0173  0         S     = %O'010000',                ! SETUP MODE PACKET IF 1
;  0174  0
;  0175  0         NEWB  = %O'100000',                ! BUFFER NOT USED IF 1
;  0176  0         LASTD= %O'100000',                 ! LAST DESCRIPTOR IN CHAIN
;  0177  0         VE    = %O'120000',                !
;  0178  0         VSE   = %O'130000',                !
;  0179  0
;  0180  0
;  0181  0      !++
;  0182  0      !  STATION ADDRESS CONSTANTS
;  0183  0      !--
;  0184  0
;  0185  0         SADR1 = 0,                        ! HIGH STATION ADDRESS BITS
;  0186  0         SADR2 = 1,                        ! MIDDLE BITS
;  0187  0         SADR3 = 2,                        ! LOW STATION ADDRESS BITS
;  0188  0         CHSUM = 3,                        ! ACTUAL CHECKSUM INDEX
;  0189  0
;  0190  0      !++
;  0191  0      !   HARDWARE AND SOFTWARE P-TABLE EQUATES
;  0192  0      !--
;  0193  0
;  0194  0          SWP_SIZE     = 1,                ! SOFTWARE P-TABLE SIZE ( WORDS )
;  0195  0          HWP_SIZE     = 2,                ! HARDWARE P-TABLE SIZE ( WORDS )
;  0196  0
;  0197  0      !++
;  0198  0      !  EQUATES
;  0199  0      !--
;  0200  0
;  0201  0              XLRL_SET         = %B'11',    ! XMIT AND RCV LISTS INVALID
;  0202  0              ILEL_SET         = %B'11',    ! INTERNAL AND EXTERNAL LOOPBACK BITS
;  0203  0              ILEL_CLR         = %B'00',    ! INTERNAL AND EXTERNAL LOOPBACK BITS
;  0204  0
;  0205  0
;  0206  0      !++
;  0207  0      !   EQUATES FOR DEQNA LOOPBACK MODES
;  0208  0      !--
;  0209  0
;  0210  0              INT_LOOPBACK     = %B'00',    ! INTERNAL LOOPBACK MODE
;  0211  0              INX_LOOPBACK     = %B'01',    ! INTERNAL/EXTENDED LOOPBACK MODE
;  0212  0              EXT_LOOPBACK     = %B'11',    ! EXTERNAL LOOPBACK MODE
;  0213  0
;  0214  0      !++
;  0215  0      !  STATUS WORD
;  0216  0      !--
;  0217  0
;  0218  0          CSR_1_STATUS     = %O'010062',    !
;  0219  0          CSR_2_STATUS     = %O'010060',    !
;  0220  0          CSR_STATUS       = %O'100220',    !
;  0221  0          CSR_MASK         = %O'010376',    !
```

```
;    0222  0
;    0223  0      !++
;    0224  0      !  SET CSR BITS
;    0225  0      !--
;    0226  0
;    0227  0              SET_IT  = 1,
;    0228  0              CLR_IT  = 0;
;    0229  0
```

```
;  0230  0     !++
;  0231  0     !
;  0232  0     !   THE CONTROL AND STATUS REGISTER BIT DEFINITIONS
;  0233  0     !
;  0234  0     !--
;  0235  0
;  0236  0     FIELD
;  0237  0         IOP_FIELDS  =
;  0238  0             SET
;  0239  0             RE      = [ 0, 1, 0 ],  ! RECEIVER ENABLE          R/W ( ACTIVE HIGH )
;  0240  0             SR      = [ 1, 1, 0 ],  ! SOFTWARE RESET           R/W ( ACTIVE HIGH )
;  0241  0             NI      = [ 2, 1, 0 ],  ! NXM INTERRUPT            R   ( ACTIVE HIGH )
;  0242  0             BD      = [ 3, 1, 0 ],  ! BOOT/DIAGNOSTIC ROM      R/W ( ACTIVE HIGH )
;  0243  0             XL      = [ 4, 1, 0 ],  ! XMIT LIST INVALID        R   ( ACTIVE HIGH )
;  0244  0             RL      = [ 5, 1, 0 ],  ! RCV LIST INVALID         R   ( ACTIVE HIGH )
;  0245  0             IE      = [ 6, 1, 0 ],  ! INTERRUPT ENABLE         R/W ( ACTIVE HIGH )
;  0246  0             XI      = [ 7, 1, 0 ],  ! XMIT INTERRUPT REQUEST   R/W ( ACTIVE HIGH )
;  0247  0             IL      = [ 8, 1, 0 ],  ! INTERNAL LOOPBACK MODE   R/W ( ACTIVE LOW  )
;  0248  0             EL      = [ 9, 1, 0 ],  ! EXTERNAL LOOPBACK MODE   R/W ( ACTIVE HIGH )
;  0249  0             SE      = [10, 1, 0 ],  ! SANITY TIMER ENABLE      R/W ( ACTIVE HIGH )
;  0250  0             X1      = [11, 1, 0 ],  ! RESERVED, UNUSABLE
;  0251  0             XC      = [12, 1, 0 ],  ! TRANSCEIVER PWR          R   ( ACTIVE HIGH )
;  0252  0             CA      = [13, 1, 0 ],  ! CARRIER                  R   ( ACTIVE HIGH )
;  0253  0             X2      = [14, 1, 0 ],  ! RESERVED, UNUSABLE
;  0254  0             RI      = [15, 1, 0 ],  ! RCV INTERRUPT REQUEST    R/W ( ACTIVE HIGH )
;  0255  0
;  0256  0             LB      = [ 8, 2, 0 ],          ! LOOPBACK BITS
;  0257  0             XLRL    = [ 4, 2, 0 ],          ! XMIT AND RCV LISTS INVALID BITS
;  0258  0             ALL_BITS= [ 0,16, 0 ],          ! FETCH WHOLE WORD
;  0259  0
;  0260  0             LO_NIBBLE = [ 0,  0, 0 ],       !
;  0261  0             HI_NIBBLE = [ 0,  4, 0 ],       !
;  0262  0             LO_BYTE   = [ 0,  8, 0 ],       !
;  0263  0             HI_BYTE   = [ 0, 16, 0 ],       ! GET WORD, ALL BITS
;  0264  0             ST_ADDR   = [ 0,  8, 0 ],       ! STATION ADDRESS LOW BYTE
;  0265  0             ST_WORD   = [ 0, 16, 0 ],       ! GET WORD, ALL BITS
;  0266  0
;  0267  0             RCV_LO  = [ 2, 0, 16, 0 ],      ! RCV BUFFER DESCRIPTOR LIST LOW ADDRESS
;  0268  0             RCV_HI  = [ 3, 0, 8, 0 ],       ! RCV BUFFER DESCRIPTOR LIST HIGH ADDRESS
;  0269  0             XMIT_LO = [ 4, 0, 16, 0 ],      ! XMIT BUFFER DESCRIPTOR LIST LOW ADDRESS
;  0270  0             XMIT_HI = [ 5, 0, 8, 0 ],       ! XMIT BUFFER DESCRIPTOR LIST HIGH ADDRESS
;  0271  0             VEC_ADR = [ 2, 8, 0 ],          ! INTERRUPT VECTOR ADDRESS
;  0272  0             VEC_ALL = [ 6, 0, 16, 0 ],      ! INTERRUPT VECTOR ADDRESS
;  0273  0             CSR_ALL = [ 7, 0, 16, 0 ]       ! CONTROL AND STATUS REGISTER
;  0274  0             TES;
```

```
;  0275  0     FIELD
;  0276  0         DL_FIELDS  =
;  0277  0             SET
;  0278  0             FLGWD   = [ 0, 0, 16, 0 ],      !
;  0279  0             DBITS   = [ 1, 0, 16, 0 ],      !
;  0280  0             SBIT    = [ 1, 12, 1, 0 ],      !
;  0281  0             LOADR   = [ 2, 0, 16, 0 ],      !
;  0282  0             TWDL    = [ 3, 0, 16, 0 ],      ! TRANSMIT WORD LENGTH
;  0283  0             STWD1   = [ 4, 0, 16, 0 ],      !
;  0284  0             STWD2   = [ 5, 0, 16, 0 ],      !
;  0285  0             DLINK   = [ 6, 0, 16, 0 ],      !
;  0286  0             BSTAT   = [ 7, 0, 16, 0 ],      !
;  0287  0             B_LEN   = [ 0, 8, 0 ],          !
;  0288  0             W_LEN   = [ 0,16, 0 ]           !
;  0289  0             TES;
;  0290  0
;  0291  0     !++
;  0292  0     !
;  0293  0     !  LOOPBACK FIELD DEFINITIONS
;  0294  0     !
;  0295  0     !--
;  0296  0
;  0297  0     FIELD
;  0298  0         LPBK_FIELDS  =
;  0299  0             SET
;  0300  0             SET_LOOPBACK    = [8, 2, 0 ]    ! LOOPBACK MACRO DEFINITION
;  0301  0             TES;
;  0302  0
;  0303  0     !++
;  0304  0     !
;  0305  0     !  HARDWARE P-TABLE FIELD DEFINITIONS
;  0306  0     !
;  0307  0     !--
;  0308  0
;  0309  0     FIELD
;  0310  0         HWP_FIELDS =
;  0311  0             SET
;  0312  0             ADDR    = [ 0, 0, 16, 0 ],      ! I/O PAGE BASE ADDRESS
;  0313  0             VEC     = [ 1, 0, 16, 0 ],      ! INTERRUPT VECTOR ADDRESS
;  0314  0             BRL     = [ 2, 0, 16, 0 ]       ! BR LEVEL
;  0315  0             TES;
;  0316  0
;  0317  0
;  0318  0     !++
;  0319  0     !
;  0320  0     !  SOFTWARE P-TABLE FIELD DEFINITIONS
;  0321  0     !
;  0322  0     !--
;  0323  0
;  0324  0     FIELD
;  0325  0         SWP_FIELDS =
;  0326  0             SET
;  0327  0             ERR_CNT = [0,0,16,0]            ! # OF ERRORS BEFORE DROPPING DEQNA
```

```
;   0328  0              TES;
;   0329  0
;   0330  0



;                                   COMMAND QUALIFIERS

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

; Run Time:          00:02.7
; Elapsed Time:      00:23.7
; Lines/CPU Min:      7443
; Lexemes/CPU-Min:  31443
; Memory Used:  33 pages
; Library Precompilation Complete
```

```
Partition name : DUMMY
Identification : V01.0
Task  UIC      : [202,22]
Task attributes: -HD
Total address windows: 1.
Task  image  size  : 8544. words
Task address limits: 002000 043257
R-W disk blk limits: 000002 000043 000042 00034.

*** Root segment: ZQNA1


R/W mem  limits: 002000 043257 041260 17072.
Disk blk limits: 000002 000043 000042 00034.


Memory allocation synopsis:

Section                                          Title  Ident  File
-------                                          -----  ----   ----
$CODE$:(RO,I,LCL,REL,CON)      002000 000274 00188.
                               002000 000176 00126.  ZQNA1  V01.0  ZQNA1.OBJ;2
                               002176 000076 00062.  ZQNA2  V01.0  ZQNA2.OBJ;2
$GLOB$:(RW,D,LCL,REL,CON)      002274 021664 09140.
                               002274 021664 09140.  ZQNA1  V01.0  ZQNA1.OBJ;2
$PLIT$:(RO,D,LCL,REL,CON)      024160 006056 03118.
                               024160 006056 03118.  ZQNA1  V01.0  ZQNA1.OBJ;2
AA$COD:(RO,I,LCL,REL,CON)      032236 000706 00454.
                               032236 000706 00454.  ZQNA2  V01.0  ZQNA2.OBJ;2
AB$COD:(RO,I,LCL,REL,CON)      033144 004526 02390.
                               033144 004526 02390.  ZQNA3  V01.0  ZQNA3.OBJ;2
AC$COD:(RO,I,LCL,REL,CON)      037672 003246 01702.
                               037672 003246 01702.  ZQNA4  V01.0  ZQNA4.OBJ;2
. BLK.:(RW,I,LCL,REL,CON)      043140 000000 00000.
$XYZ$ :(RO,I,LCL,REL,CON)      043140 000116 00078.
                               043140 000106 00070.  CZQNAA 2.4    B16SAV.OBJ;2
                               043246 000010 00008.  ZQNA5  V01.0  ZQNA5.OBJ;2


Global symbols:

ADR     000020     BIT1    000002     BIT8    000400     CLK.IN 033076-R  CSR.WO 024100-R  DBM38  025124-R  D$PCNT 002122-R
BD.PRO  023766-R   BIT10   002000     BIT9    001000     CLK.ST 024042-R  DATA.B 002474-R  DBM39  025206-R  EF.CON 000036
BIT0    000001     BIT11   004000     BL$LAS  043246-R   CLK.TY 024034-R  DBM01  024414-R  DBM40  025252-R  EF.NEW 000035
BIT00   000001     BIT12   010000     BOE     000400     CLK.VE 024036-R  DBM02  024452-R  DBM41  025316-R  EF.PWR 000034
BIT01   000002     BIT13   020000     BUF.LE  024076-R   CLR.BU 040674-R  DBM03  024476-R  DBM42  025402-R  EF.RES 000037
BIT02   000004     BIT14   040000     CANCEL  024054-R   CLR.DE 040714-R  DBM04  024524-R  DBM43  025430-R  EF.STA 000040
BIT03   000010     BIT15   100000     CHECKS  024074-R   CLR.RB 040654-R  DBM05  024554-R  DBM44  025514-R  ERRBLK 002150-R
BIT04   000020     BIT2    000004     CHK.CS  041024-R   CLR.RD 040736-R  DBM06  024610-R  DBM45  025602-R  ERRMSG 002146-R
BIT05   000040     BIT3    000010     CHK.RC  041416-R   CLR.SE 041002-R  DBM33  024644-R  DBM46  025640-R  ERRNBR 002144-R
BIT06   000100     BIT4    000020     CHK.XM  041150-R   CLR.XB 040634-R  DBM34  024704-R  DEQNA. 024070-R  ERROR$ 037672-R
BIT07   000200     BIT5    000040     CLK.AD  024032-R   CLR.XD 040760-R  DBM35  024742-R  DESCR. 002274-R  ERRTYP 002142-R
BIT08   000400     BIT6    000100     CLK.CS  024040-R   COMPAR 041664-R  DBM36  025014-R  DFHTBL 002164-R  ERR01  025702-R
BIT09   001000     BIT7    000200     CLK.HE  024044-R   COUNTE 024072-R  DBM37  025060-R  DFSTBL 002154-R  ERR02  025750-R
```

ERR03	026010-R	IOP.DA	024104-R	L$EXP5	002066-R	L$STA	002030-R	MSG12	030434-R	QST02	024210-R	TEMP5	024124-R		
ERR04	026044-R	IOP.TA	023522-R	L$HARD	002246-R	L$SW	002164-R	MSG13	030512-R	QST03	024240-R	TEMP6	024126-R		
ERR05	026120-R	ISR	000100	L$HIME	002120-R	L$SWLE	002162-R	MSG14	030574-R	QST04	024270-R	TEMP7	024130-R		
ERR06	026202-R	IXE	004000	L$HPCP	002016-R	L$TEST	002114-R	MSG15	030660-R	QST05	024342-R	TEMP8	024132-R		
ERR07	026206-R	LOE	040000	L$HPTP	002022-R	L$TIML	002014-R	MSG16	030742-R	QST06	024404-R	TEMP9	024134-R		
ERR08	026212-R	LOT	000010	L$HRDL	002244-R	L$UNIT	002012-R	MSG17	031026-R	QST07	024406-R	TICKS	024046-R		
ETH.ST	023542-R	L$ACP	002110-R	L$HW	002154-R	MEM.SI	024064-R	MSG18	031114-R	QST08	024410-R	TMP.IO	024110-R		
EVL	000004	L$APT	002036-R	L$HWLE	002152-R	MINUTE	024052-R	MSG19	031140-R	QST09	024412-R	TMP.RE	024112-R		
E0001	026216-R	L$AU	033050-R	L$ICP	002104-R	MSG.BR	031746-R	MSG20	031226-R	RBUF.L	024060-R	T$FREE	043252-R		
E0101	026252-R	L$AUT	002070-R	L$INIT	032770-R	MSG.CL	032102-R	MSG21	031316-R	RCV.BU	002474-R	T$PTHV	000001		
E0201	026334-R	L$AUTO	033002-R	L$LADP	002026-R	MSG.CS	031706-R	MSG22	031376-R	RCV.D.	002274-R	T1	033552-R		
E0202	026420-R	L$CCP	002106-R	L$LAST	043252-R	MSG.EN	032056-R	MSG23	031462-R	REG.AD	024102-R	T2	034122-R		
E0301	026504-R	L$CLEA	033024-R	L$LOAD	002100-R	MSG.IN	031774-R	NXM.IN	033060-R	RESET.	040210-R	T3	035016-R		
E0401	026566-R	L$CO	002032-R	L$LUN	002074-R	MSG.LE	032204-R	PHYS.A	022474-R	SECOND	024050-R	T4	035726-R		
E0501	026652-R	L$DEPO	002011-R	L$MREV	002050-R	MSG.NX	032142-R	PNT	001000	SETUP.	022522-R	T5	036402-R		
E0502	026742-R	L$DESC	002214-R	L$NAME	002000-R	MSG.PW	031660-R	PRI	002000	SET.RD	042136-R	T6	037330-R		
E0503	027030-R	L$DESP	002076-R	L$NDHR	002266-R	MSG.1T	031470-R	PRI00	000000	SET.XD	042214-R	T7	037656-R		
E0601	027126-R	L$DEVP	002060-R	L$NDHW	002160-R	MSG.2T	031520-R	PRI01	000040	STATIO	023556-R	UAM	000200		
E0801	027212-R	L$DISP	002124-R	L$NDSF	002272-R	MSG.3T	031560-R	PRI02	000100	SWPTBL	002164-R	VER.DE	040512-R		
E1$REP	040170-R	L$DLY	002116-R	L$NDSW	002166-R	MSG.4T	031620-R	PRI03	000140	SWP.TA	024030-R	WALKIN	042272-R		
FREE.M	024062-R	L$DTP	002040-R	L$PRIO	002042-R	MSG01	027276-R	PRI04	000200	TADR1	024154-R	WRT.ST	042564-R		
GET.AD	024106-R	L$DTYP	002034-R	L$PROT	002170-R	MSG02	027360-R	PRI05	000240	TADR2	024156-R	XBUF.L	024056-R		
GP$1	002246-R	L$DU	033036-R	L$PRT	002112-R	MSG03	027446-R	PRI06	000300	TARGET	023576-R	XMIT.A	043022-R		
GP$2	002256-R	L$DUT	002072-R	L$REPP	002062-R	MSG04	027552-R	PRI07	000340	TBYTE1	024150-R	XMIT.B	012474-R		
HOE	100000	L$DVTY	002176-R	L$REV	002010-R	MSG05	027644-R	PTRN.T	023566-R	TBYTE2	024151-R	XMIT.D	002374-R		
HWPTBL	002154-R	L$EF	002052-R	L$RPT	032246-R	MSG06	027736-R	P1	024136-R	TBYTE3	024152-R	$END.L	043254-R		
HWP.TA	024026-R	L$ENVI	002044-R	L$SFTL	002270-R	MSG07	030030-R	P2	024140-R	TBYTE4	024153-R	$SAVE2	043140-R		
IBE	010000	L$ERRT	002142-R	L$SOFT	002272-R	MSG08	030122-R	P3	024142-R	TEMP1	024114-R	$SAVE3	043154-R		
IDU	000040	L$ETP	002102-R	L$SPC	002056-R	MSG09	030214-R	P4	024144-R	TEMP2	024116-R	$SAVE4	043172-R		
IER	020000	L$EXP1	002046-R	L$SPCP	002020-R	MSG10	030306-R	P5	024146-R	TEMP3	024120-R	$SAVE5	043212-R		
INTERR	024066-R	L$EXP4	002064-R	L$SPTP	002024-R	MSG11	030360-R	QST01	024160-R	TEMP4	024122-R				

*** Task builder statistics:

Total work file references: 41365.
Work file reads: 0.
Work file writes: 0.
Size of core pool: 4016. words (15. pages)
Size of work file: 3328. words (13. pages)

Elapsed time:00:00:17

ZQNAAO CREATED BY TKB ON 2 FEB 84 AT 14:50 PAGE 1

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...			
ADR	000020	# ZQNA1	# ZQNA2		
BD.PRO	023766-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	043246-R	# ZQNA5			
BOE	000400	# ZQNA1	# ZQNA2		
BUF.LE	024076-R	# ZQNA1			
CANCEL	024054-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
CHECKS	024074-R	# ZQNA1	ZQNA3	ZQNA4	
CHK.CS	041024-R	ZQNA3	# ZQNA4		
CHK.RC	041416-R	ZQNA3	# ZQNA4		
CHK.XM	041150-R	ZQNA3	# ZQNA4		
CLK.AD	024032-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
CLK.CS	024040-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
CLK.HE	024044-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
CLK.IN	033076-R	# ZQNA2	ZQNA3		
CLK.ST	024042-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
CLK.TY	024034-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
CLK.VE	024036-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
CLR.BU	040674-R	ZQNA3	# ZQNA4		
CLR.DE	040714-R	ZQNA3	# ZQNA4		
CLR.RB	040654-R	ZQNA3	# ZQNA4		
CLR.RD	040736-R	# ZQNA4			
CLR.SE	041002-R	# ZQNA4			
CLR.XB	040634-R	ZQNA3	# ZQNA4		
CLR.XD	040760-R	# ZQNA4			
COMPAR	041664-R	ZQNA3	# ZQNA4		
COUNTE	024072-R	# ZQNA1	ZQNA3	ZQNA4	

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...			
CSR.WO	024100-R	# ZQNA1	ZQNA4		
DATA.B	002474-R	# ZQNA1	ZQNA3	ZQNA4	
DBM01	024414-R	# ZQNA1	ZQNA2		
DBM02	024452-R	# ZQNA1	ZQNA2		
DBM03	024476-R	# ZQNA1	ZQNA2		
DBM04	024524-R	# ZQNA1	ZQNA2		
DBM05	024554-R	# ZQNA1	ZQNA2		
DBM06	024610-R	# ZQNA1	ZQNA2		
DBM33	024644-R	# ZQNA1	ZQNA2	ZQNA3	
DBM34	024704-R	# ZQNA1	ZQNA2	ZQNA3	
DBM35	024742-R	# ZQNA1	ZQNA2	ZQNA3	
DBM36	025014-R	# ZQNA1	ZQNA2	ZQNA3	
DBM37	025060-R	# ZQNA1	ZQNA2	ZQNA3	
DBM38	025124-R	# ZQNA1	ZQNA2	ZQNA3	
DBM39	025206-R	# ZQNA1	ZQNA2	ZQNA3	
DBM40	025252-R	# ZQNA1	ZQNA3	ZQNA4	
DBM41	025316-R	# ZQNA1	ZQNA3		
DBM42	025402-R	# ZQNA1	ZQNA2	ZQNA3	
DBM43	025430-R	# ZQNA1	ZQNA3		
DBM44	025514-R	# ZQNA1	ZQNA3		
DBM45	025602-R	# ZQNA1	ZQNA3	ZQNA4	
DBM46	025640-R	# ZQNA1	ZQNA3		
DEQNA.	024070-R	# ZQNA1	ZQNA3	ZQNA4	
DESCR.	002274-R	# ZQNA1	ZQNA3	ZQNA4	
DFHTBL	002164-R	# ZQNA1			
DFSTBL	002154-R	# ZQNA1			
D$PCNT	002122-R	# ZQNA1			
EF.CON	000036	# ZQNA1	# ZQNA2		
EF.NEW	000035	# ZQNA1	# ZQNA2		
EF.PWR	000034	# ZQNA1	# ZQNA2		
EF.RES	000037	# ZQNA1	# ZQNA2		
EF.STA	000040	# ZQNA1	# ZQNA2		
ERRBLK	002150-R	# ZQNA1			
ERRMSG	002146-R	# ZQNA1			
ERRNBR	002144-R	# ZQNA1			
ERROR$	037672-R	ZQNA3	# ZQNA4		
ERRTYP	002142-R	# ZQNA1			
ERR01	025702-R	# ZQNA1	ZQNA2		
ERR02	025750-R	# ZQNA1	ZQNA2		
ERR03	026010-R	# ZQNA1	ZQNA2		
ERR04	026044-R	# ZQNA1	ZQNA4		
ERR05	026120-R	# ZQNA1	ZQNA3	ZQNA4	
ERR06	026202-R	# ZQNA1	ZQNA3		
ERR07	026206-R	# ZQNA1	ZQNA3		
ERR08	026212-R	# ZQNA1	ZQNA3		
ETH.ST	023542-R	# ZQNA1	ZQNA3		
EVL	000004	# ZQNA1	# ZQNA2		
E0001	026216-R	# ZQNA1	ZQNA3	ZQNA4	
E0101	026252-R	# ZQNA1	ZQNA3		
E0201	026334-R	# ZQNA1	ZQNA3		
E0202	026420-R	# ZQNA1	ZQNA3		
E0301	026504-R	# ZQNA1	ZQNA3		

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...			
E0401	026566-R	# ZQNA1	ZQNA3		
E0501	026652 R	# ZQNA1	ZQNA3		
E0502	026742-R	# ZQNA1	ZQNA3		
E0503	027030-R	# ZQNA1	ZQNA3		
E0601	027126-R	# ZQNA1	ZQNA3		
E0801	027212-R	# ZQNA1			
E1$REP	040170-R	ZQNA3	# ZQNA4		
FREE.M	024062-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
GET.AD	024106-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
GP$1	002246-R	# ZQNA2			
GP$2	002256-R	# ZQNA2			
HOE	100000	# ZQNA1	# ZQNA2		
HWPTBL	002154-R	# ZQNA1			
HWP.TA	024026-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
IBE	010000	# ZQNA1	# ZQNA2		
IDU	000040	# ZQNA1	# ZQNA2		
IER	020000	# ZQNA1	# ZQNA2		
INTERR	024066-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
IOP.DA	024104-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
IOP.TA	023522-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
ISR	000100	# ZQNA1	# ZQNA2		
IXE	004000	# ZQNA1	# ZQNA2		
LOE	040000	# ZQNA1	# ZQNA2		
LOT	000010	# ZQNA1	# ZQNA2		
L$ACP	002110-R	# ZQNA1			
L$APT	002036-R	# ZQNA1			
L$AU	033050-R	ZQNA1	# ZQNA2		
L$AUT	002070-R	# ZQNA1			
L$AUTO	033002-R	ZQNA1	# ZQNA2		
L$CCP	002106-R	# ZQNA1			
L$CLEA	033024-R	ZQNA1	# ZQNA2		
L$CO	002032-R	# ZQNA1			
L$DEPO	002011-R	# ZQNA1			
L$DESC	002214-R	ZQNA1	# ZQNA2		
L$DESP	002076-R	# ZQNA1			
L$DEVP	002060-R	# ZQNA1			
L$DISP	002124-R	# ZQNA1			
L$DLY	002116-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
L$DTP	002040-R	# ZQNA1			
L$DTYP	002034-R	# ZQNA1			
L$DU	033036-R	ZQNA1	# ZQNA2		
L$DUT	002072-R	# ZQNA1			
L$DVTY	002176-R	ZQNA1	# ZQNA2		
L$EF	002052-R	# ZQNA1			
L$ENVI	002044-R	# ZQNA1			
L$ERRT	002142-R	# ZQNA1			
L$ETP	002102-R	# ZQNA1			
L$EXP1	002046-R	# ZQNA1			
L$EXP4	002064-R	# ZQNA1			
L$EXP5	002066-R	# ZQNA1			
L$HARD	002246-R	ZQNA1	# ZQNA2		
L$HIME	002120-R	# ZQNA1			

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...			
L$HPCP	002016-R	# ZQNA1			
L$HPTP	002022-R	# ZQNA1			
L$HRDL	002244-R	# ZQNA2			
L$HW	002154-R	# ZQNA1			
L$HWLE	002152-R	# ZQNA1			
L$ICP	002104-R	# ZQNA1			
L$INIT	032770-R	ZQNA1	# ZQNA2		
L$LADP	002026-R	# ZQNA1			
L$LAST	043252-R	ZQNA1	# ZQNA5		
L$LOAD	002100-R	# ZQNA1			
L$LUN	002074-R	# ZQNA1			
L$MREV	002050-R	# ZQNA1			
L$NAME	002000-R	# ZQNA1			
L$NDHR	002266-R	# ZQNA2			
L$NDHW	002160-R	# ZQNA1			
L$NDSF	002272-R	# ZQNA2			
L$NDSW	002166-R	# ZQNA1			
L$PRIO	002042-R	# ZQNA1			
L$PROT	002170-R	# ZQNA1			
L$PRT	002112-R	# ZQNA1			
L$REPP	002062-R	# ZQNA1			
L$REV	002010-R	# ZQNA1			
L$RPT	032246-R	ZQNA1	# ZQNA2		
L$SFTL	002270-R	# ZQNA2			
L$SOFT	002272-R	ZQNA1	# ZQNA2		
L$SPC	002056-R	# ZQNA1			
L$SPCP	002020-R	# ZQNA1			
L$SPTP	002024-R	# ZQNA1			
L$STA	002030-R	# ZQNA1			
L$SW	002164-R	# ZQNA1			
L$SWLE	002162-R	# ZQNA1			
L$TEST	002114-R	# ZQNA1			
L$TIML	002014-R	# ZQNA1			
L$UNIT	002012-R	# ZQNA1			
MEM.SI	024064-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
MINUTE	024052-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4
MSG.BR	031746-R	# ZQNA1			
MSG.CL	032102-R	# ZQNA1			
MSG.CS	031706-R	# ZQNA1			
MSG.EN	032056-R	# ZQNA1			
MSG.IN	031774-R	# ZQNA1			
MSG.LE	032204-R	# ZQNA1			
MSG.NX	032142-R	# ZQNA1			
MSG.PW	031660-R	# ZQNA1	ZQNA2		
MSG.1T	031470-R	# ZQNA1	ZQNA4		
MSG.2T	031520-R	# ZQNA1	ZQNA4		
MSG.3T	031560-R	# ZQNA1	ZQNA4		
MSG.4T	031620-R	# ZQNA1	ZQNA4		
MSG01	027276-R	# ZQNA1	ZQNA3		
MSG02	027360-R	# ZQNA1	ZQNA4		
MSG03	027446-R	# ZQNA1	ZQNA4		
MSG04	027552-R	# ZQNA1	ZQNA4		

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...				
MSG05	027644-R	# ZQNA1	ZQNA4			
MSG06	027736-R	# ZQNA1	ZQNA4			
MSG07	030030-R	# ZQNA1	ZQNA4			
MSG08	030122-R	# ZQNA1	ZQNA4			
MSG09	030214-R	# ZQNA1	ZQNA4			
MSG10	030306-R	# ZQNA1	ZQNA4			
MSG11	030360-R	# ZQNA1	ZQNA4			
MSG12	030434-R	# ZQNA1	ZQNA4			
MSG13	030512-R	# ZQNA1	ZQNA4			
MSG14	030574-R	# ZQNA1	ZQNA4			
MSG15	030660-R	# ZQNA1	ZQNA4			
MSG16	030742-R	# ZQNA1	ZQNA4			
MSG17	031026-R	# ZQNA1	ZQNA4			
MSG18	031114-R	# ZQNA1	ZQNA3			
MSG19	031140-R	# ZQNA1	ZQNA3			
MSG20	031226-R	# ZQNA1	ZQNA3			
MSG21	031316-R	# ZQNA1	ZQNA3			
MSG22	031376-R	# ZQNA1	ZQNA3			
MSG23	031462-R	# ZQNA1	ZQNA3			
NXM.IN	033060-R	# ZQNA2	ZQNA3			
PHYS.A	022474-R	# ZQNA1	ZQNA3			
PNT	001000	# ZQNA1	# ZQNA2			
PRI	002000	# ZQNA1	# ZQNA2			
PRI00	000000	# ZQNA1	# ZQNA2	ZQNA3		
PRI01	000040	# ZQNA1	# ZQNA2	ZQNA3		
PRI02	000100	# ZQNA1	# ZQNA2	ZQNA3		
PRI03	000140	# ZQNA1	# ZQNA2	ZQNA3		
PRI04	000200	# ZQNA1	# ZQNA2	ZQNA3		
PRI05	000240	# ZQNA1	# ZQNA2	ZQNA3		
PRI06	000300	# ZQNA1	# ZQNA2	ZQNA3		
PRI07	000340	# ZQNA1	# ZQNA2	ZQNA3	ZQNA4	
PTRN.T	023566-R	# ZQNA1	ZQNA3			
P1	024136-R	# ZQNA1	ZQNA4			
P2	024140-R	# ZQNA1	ZQNA4			
P3	024142-R	# ZQNA1	ZQNA4			
P4	024144-R	# ZQNA1	ZQNA4			
P5	024146-R	# ZQNA1	ZQNA4			
QST01	024160-R	# ZQNA1	ZQNA2			
QST02	024210-R	# ZQNA1	ZQNA2			
QST03	024240-R	# ZQNA1	ZQNA2			
QST04	024270-R	# ZQNA1	ZQNA2			
QST05	024342-R	# ZQNA1	ZQNA2			
QST06	024404-R	# ZQNA1				
QST07	024406-R	# ZQNA1	ZQNA2			
QST08	024410-R	# ZQNA1	ZQNA2			
QST09	024412-R	# ZQNA1	ZQNA2			
RBUF.L	024060-R	# ZQNA1	ZQNA3	ZQNA4		
RCV.BU	002474-R	# ZQNA1	ZQNA3	ZQNA4		
RCV.D.	002274-R	# ZQNA1	ZQNA3	ZQNA4		
REG.AD	024102-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
RESET.	040210-R	ZQNA2	ZQNA3	# ZQNA4		
SECOND	024050-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...				
SETUP.	022522-R	# ZQNA1	ZQNA3	ZQNA4		
SET.RD	042136-R	ZQNA3	# ZQNA4			
SET.XD	042214-R	ZQNA3	# ZQNA4			
STATIO	023556-R	# ZQNA1	ZQNA3	ZQNA4		
SWPTBL	002164-R	# ZQNA1				
SWP.TA	024030-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TADR1	024154-R	# ZQNA1	ZQNA3	ZQNA4		
TADR2	024156-R	# ZQNA1	ZQNA3			
TARGET	023576-R	# ZQNA1	ZQNA4			
TBYTE1	024150-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE2	024151-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE3	024152-R	# ZQNA1	ZQNA3	ZQNA4		
TBYTE4	024153-R	# ZQNA1	ZQNA3	ZQNA4		
TEMP1	024114-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP2	024116-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP3	024120-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP4	024122-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP5	024124-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP6	024126-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP7	024130-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP8	024132-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TEMP9	024134-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TICKS	024046-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TMP.IO	024110-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
TMP.RE	024112-R	# ZQNA1	ZQNA2	ZQNA3	ZQNA4	
T$FREE	043252-R	# ZQNA5				
T$PTHV	000001	ZQNA1	# ZQNA5			
T1	033552-R	ZQNA1	# ZQNA3			
T2	034122-R	ZQNA1	# ZQNA3			
T3	035016-R	ZQNA1	# ZQNA3			
T4	035726-R	ZQNA1	# ZQNA3			
T5	036402-R	ZQNA1	# ZQNA3			
T6	037330-R	ZQNA1	# ZQNA3			
T7	037656-R	ZQNA1	# ZQNA3			
UAM	000200	# ZQNA1	# ZQNA2			
VER.DE	040512-R	ZQNA3	# ZQNA4			
WALKIN	042272-R	ZQNA3	# ZQNA4			
WFT.ST	042564-R	ZQNA3	# ZQNA4			
XBUF.L	024056-R	# ZQNA1	ZQNA3	ZQNA4		
XMIT.A	043022-R	ZQNA3	# ZQNA4			
XMIT.B	012474-R	# ZQNA1	ZQNA3	ZQNA4		
XMIT.D	002374-R	# ZQNA1	ZQNA3	ZQNA4		
$END.L	043254-R	# ZQNA5				
$SAVE2	043140-R	# CZQNAA	ZQNA3	ZQNA4		
$SAVE3	043154-R	# CZQNAA	ZQNA3	ZQNA4		
$SAVE4	043172-R	# CZQNAA	ZQNA2	ZQNA3		
$SAVE5	043212-R	# CZQNAA				