

RC25

RC25 DISK FORMATTER
CZRCHBO

COPYRIGHT (c) 1983-84
AH-T275B-MC
FICHE 01 OF 02

JUL 1984
digital
Made In USA

This microfiche card contains a grid of frames. The frames are organized into columns and rows. The first four columns contain text, likely representing a header or metadata. The remaining columns contain data, which appears to be a list of records or entries. Each frame contains a small amount of text, possibly a single record or a portion of a larger record. The text is too small to read clearly, but it seems to be structured in a way that suggests a database or a list of files. The frames are arranged in a regular grid pattern, typical of microfiche cards.

RC25

RC25 DISK FORMATTER
CZRCHBO

COPYRIGHT (c) 1983-84
RH-T275B-MC
FICHE 02 OF 02

JUL 1984
digital
Made In USA

[Faded text from the microfiche grid, including labels like 'FIELD NO.', 'DATE', 'TIME', and 'LOCATION']

11

ZRCHB1

ZRCHB0 RC25 DISK FORMATTER

5-Apr-1984 13:43:03
22-Feb-1984 12:03:49

VAX-11 B1100-16 V3-555
SPIDER#USERS:(NEALE.AZTEC)ZRCHB1.B16;7 (1)

SEQ 0001

Page 1

: 0001 MODULE ZRCHB1 (TITLE ZRCHB0 RC25 DISK FORMATTER'
: 0002 IDENT = REV B PATCH 00') .
:

: 0004 \$SBTTL 'USER DOCUMENTATION
: C 0005 \$(

: C 0006
: C 0007

: C 0008

: C 0009

: C 0010

: C 0011

: C 0012

: C 0013

: C 0014

: C 0015

: C 0016

: C 0017

: C 0018

: C 0019

: C 0020

: C 0021

: C 0022

: C 0023

: C 0024

: C 0025

: C 0026

: C 0027

: C 0028

: C 0029

: C 0030

: C 0031

: C 0032

: C 0033

: C 0034

: C 0035

: C 0036

: C 0037

: C 0038

: C 0039

: C 0040

: C 0041

: C 0042

: C 0043

: C 0044

: C 0045

: C 0046

: C 0047

: C 0048

: C 0049

: C 0050

IDENTIFICATION

PRODUCT CODE: AC-T274B-MC
PRODUCT NAME: CZRCHB0 RC25 DISK FORMATTER
PRODUCT DATE: 23-January 1984
MAINTAINER : Disk Engineering
AUTHOR : D.W.Neale

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1983, 1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL PDP UNIBUS MASSBUS
DEC DECUS DECTAPE

TABLE OF CONTENTS

:	C	0051	
:	C	0052	
:	C	0053	
:	C	0054	
:	C	0055	
:	C	0056	0.0 MAINTENANCE HISTORY
:	C	0057	1.0 GENERAL INFORMATION
:	C	0058	1.1 PROGRAM ABSTRACT
:	C	0059	1.2 PERFORMANCE GOALS
:	C	0060	1.3 SYSTEM REQUIREMENTS
:	C	0061	1.4 RELATED DOCUMENTS AND STANDARDS
:	C	0062	1.5 DIAGNOSTIC HIERARCHY PREREQUISITES
:	C	0063	1.6 ASSUMPTIONS
:	C	0064	1.7 PRODUCT USERS AND USES
:	C	0065	1.8 RESTRICTIONS
:	C	0066	1.9 BAD BLOCK DEFINITION
:	C	0067	1.10 SOURCE MODULE DEFINITION
:	C	0068	
:	C	0069	2.0 OPERATING INSTRUCTIONS
:	C	0070	2.1 COMMANDS
:	C	0071	2.2 SWITCHES
:	C	0072	2.3 FLAGS
:	C	0073	2.4 HARDWARE QUESTIONS
:	C	0074	2.5 SOFTWARE QUESTIONS
:	C	0075	2.6 EXTENDED P-TABLE DIALOGUE
:	C	0076	2.7 QUICK STARTUP PROCEDURE
:	C	0077	2.8 PID (PROCESS INDICATOR) WORDS
:	C	0078	2.9 HOST/DM FORMATTER RUN TIME MESSAGES
:	C	0079	
:	C	0080	3.0 ERROR INFORMATION
:	C	0081	3.1 TYPES OF ERRORS MESSAGES
:	C	0082	3.2 SPECIFIC ERROR MESSAGES
:	C	0083	
:	C	0084	4.0 PERFORMANCE AND PROGRESS REPORTS
:	C	0085	
:	C	0086	5.0 DEVICE INFORMATION TABLES
:	C	0087	
:	C	0088	6.0 DM FORMATTER TEST SUMMARIES
:	C	0089	6.1 REFORMAT MODE PROCEDURE
:	C	0090	6.2 RESTORE MODE PROCEDURE
:	C	0091	6.3 RECONSTRUCT MODE PROCEDURE
:	C	0092	
:	C	0093	7.0 HOST TEST SUMMARIES
:	C	0094	7.1 ATTENDED MODE
:	C	0095	7.2 UN-ATTENDED MODE
:	C	0096	
:	C	0097	8.0 APPENDIX
:	C	0098	APPENDIX A GLOSSARY
:	C	0099	APPENDIX B BAD BLOCK REPLACEMENT
:	C	0100	APPENDIX C FCT
:	C	0101	APPENDIX D RCT
:	C	0102	APPENDIX E DATA PATTERNS
:	C	0103	APPENDIX F PING PONG ALGORITHM
:	C	0104	APPENDIX G CREATING ZRCHB DM CODE
:	C	0105	APPENDIX H CREATING ZRCHB HOST CODE

01

ZRCHB1
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
USER DOCUMENTATION

5 Apr-1984 13:43:03
22 Feb-1984 12:03:49

VAX-11 B119-16 V3-555
SPIDER#USERS:(NEALE.AZTEC)ZRCHB1.B16;7 (3)

SEQ 0003
Page 3

: C 0106
: C 0107
: C 0108
: C 0109
: C 0110
: C 0111
: C 0112
: C 0113
: C 0114

0.0 MAINTENANCE HISTORY

Modified By: D.W.Neale Date: 13 Jul-83 Version: 1.0
Original release

Modified By: D.W.Neale Date: 23 Jan 84 Version: 2.0
Modified DM formatter to format inner DBN tracks with offset.
Other minor host code changes marked by the notation "V2.0".

```

: C 0115      1.0 GENERAL INFORMATION
: C 0116
: C 0117      1.1 PROGRAM ABSTRACT
: C 0118      This program will prepare RC25 media for use as addressable storage by
: C 0119      providing headers and replacing of bad blocks. This formatter will be
: C 0120      composed of two sections: a host resident section, and a
: C 0121      controller-resident DM code section.
: C 0122
: C 0123      The host section will serially format up to sixteen RC25 subsystems by
: C 0124      downline-loading the DM code, and monitor the task. The downline
: C 0125      section will consist of overlays of DM code. The first of these will be
: C 0126      down-line loaded to the drive itself, and accomplish the actual business
: C 0127      of formatting, calling in additional overlays as needed, asking software
: C 0128      parameters questions and printing formatter error and informational
: C 0129      messages.
: C 0130      There are three general modes of DM formatter operation and they are:
: C 0131
: C 0132      o REFORMAT - This mode is used to format a medium which has been
: C 0133      previously formatted, and is being reformatted to clear
: C 0134      existing data or to change the mode of the medium to 512 bytes
: C 0135      per sector. It assumes that the FCT is still intact.
: C 0136      o RESTORE - This mode will only be run by DIGITAL Manufacturing
: C 0137      personnel. It provides an external copy of the FCT, produced
: C 0138      when the disk was manufactured and stored offline, to the
: C 0139      RC25 formatter.
: C 0140      o RECONSTRUCT - This mode is used when none of the other modes
: C 0141      is possible. It detects bad blocks by performing repetitive
: C 0142      read checks of each sector. For this reason, a RECONSTRUCT
: C 0143      run takes considerably longer than the other modes.
: C 0144
: C 0145      There are two general modes of HOST operation and they are:
: C 0146      o ATTENDED MODE- This is where an operator must be present at the
: C 0147      console terminal to respond to DM formatter software parameter
: C 0148      questions. In this HOST mode the operator can choose any of the
: C 0149      three DM formatting modes.
: C 0150      o UN-ATTENDED MODE- This is where the HOST will automatically answer
: C 0151      DM formatter software parameter questions to perform a REFORMAT
: C 0152      mode to all units selected units via the hardware P_Tables.
: C 0153
: C 0154      1.2 Performance Goals
: C 0155      Before initiating the format process, simple checks will be made to
: C 0156      assure the object drive exists, and that communications are possible
: C 0157      between host and downline program sections.
: C 0158
: C 0159      To maximize throughput:
: C 0160      o For full Track reads a half Track skew will be employed. That
: C 0161      is, for even physical LBN track numbers each block number in
: C 0162      Track N+1 will occur 16 block times later than in Track N.
: C 0163      o During bad block replacement, primary replacement blocks (RBN)
: C 0164      will be used first. A second pass will then use secondary
: C 0165      RBN's for any additional bad blocks.
: C 0166      All bad blocks, whether induced by the manufacturing process or
: C 0167      wear-caused, will be revectorred to Replacement Blocks (RBN). This will
: C 0168      assure the full compliment of 25,451 blocks per surface for host
: C 0169      applications use.
: C 0170      Product reliability is enhanced through use of four copies of the FCT
: C 0171      and RCT.

```

: C 0172
: C 0173
: C 0174
: C 0175
: C 0176
: C 0177
: C 0178
: C 0179
: C 0180
: C 0181
: C 0182
: C 0183
: C 0184
: C 0185
: C 0186
: C 0187
: C 0188
: C 0189
: C 0190
: C 0191
: C 0192
: C 0193
: C 0194
: C 0195
: C 0196
: C 0197
: C 0198
: C 0199
: C 0200
: C 0201
: C 0202
: C 0203
: C 0204
: C 0205
: C 0206
: C 0207
: C 0208
: C 0209
: C 0210
: C 0211
: C 0212
: C 0213
: C 0214
: C 0215
: C 0216

1.3 SYSTEM REQUIREMENTS

The following are required to run the RC25 Formatter on PDP-11 Systems:

- * A PDP 11 series CPU
- * A minimum of 28K words of main memory
- * An RC25 subsystem
- * An XXDP+ load medium
- * A console terminal
- * Diagnostic Supervisor

1.4 RELATED DOCUMENTS AND STANDARDS

1. DUP.V05 Diagnostic/Utilities Protocol version 0.5
2. UQSSP.DOC Unibus/Q-Bus Storage Systems port V1.5
4. MSCP.DOC Mass Storage Control Protocol
5. AZTEC.DOC RC25 Microcode Documentation
6. DSDFV11.DOC Standard Disk Format Specification version 1.1
7. SUPPRGC.DOC PDP-11 Diagnostic Supervisor Programming guide
8. CHQUSD.SEQ XXDP+/SUPR user manual
9. BLISS-16 Language Guide
10. BLISS-1E User's Guide

1.5 DIAGNOSTIC HIERARCHY PREREQUISITES

A fully functional CPU, main memory and RC25 subsystem are required.

1.6 ASSUMPTIONS

Prior to the first formatting of a media, an exhaustive surface analysis is assumed to have taken place to detect bad spots caused by the manufacturing process. The results of this analysis is assumed to have been written as two FCT images per FCT track on the RC25 media.

1.7 PRODUCT USERS AND USES

This program is intended for use on media which has undergone a surface analysis process, resulting in the writing of Factory Control Tables (FCT) on the media.

The program will be used by Engineering and Manufacturing Groups as a product development tool, and by Field Service personnel for media recovery.

This program is a utility and is not intended for use as a diagnostic. For future use, the DM formatter code portion of this program will be used by operating systems as a format utility program.

: C 0217 1.8 RESTRICTIONS
: C 0218
: C 0219 This program will be host loaded.
: C 0220
: C 0221 With the exception of the FCT and inner DBN tracks, all information previously
: C 0222 recorded on the media will be destroyed. This includes all LBN, RBN and RCT
: C 0223 blocks.
: C 0224
: C 0225 The format process will reset the forced error indicator (EDC field) of
: C 0226 each LBN, DBN, and used RBN and set the forced error indicators of all
: C 0227 unused REN's.
: C 0228
: C 0229 The FCT preamble will indicate zero for both the size of the 576 byte
: C 0230 replacement table, and the size of the controller scratch area.
: C 0231
: C 0232 All sectors will have 512 byte data fields.
: C 0233
: C 0234 This Host program will run on only PDP 11 family CPUs which support the
: C 0235 Diagnostic Supervisors.
: C 0236

: C 0237 1.9 Bad Block Definition
: C 0238
: C 0239 The Formatters bad block replacement convention is as follows:
: C 0240
: C 0241 For Read operations:
: C 0242
: C 0243 Any ECC, read or compare oata failure will result in four retries.
: C 0244
: C 0245 Failure of any retries will be considered a Hard error and
: C 0246 the block in error will be replaced.
: C 0247
: C 0248 The block in question will be considered good and no revectoring
: C 0249 will be done if all retries are successful.
: C 0250
: C 0251 For Write operations:
: C 0252
: C 0253 A Write failure will result in four retries.
: C 0254
: C 0255 Failure of any retries will be considered a Hard error and
: C 0256 the block in error will be replaced.
: C 0257
: C 0258 The block in question will be considered good and no revectoring
: C 0259 will be done if all retries are successful.

- : C 0260 1.10 SOURCE MODULE DEFINITION
- : C 0261
- : C 0262 The following list all source module comprising ZRCHB0 and a description
- : C 0263 of their contents.
- : C 0264
- : C 0265 1. ZRCHB0.R16 Host source bliss library
- : C 0266 2. ZRCHB1.B16 Program source document module
- : C 0267 3. ZRCHB2.B16 Global data module
- : C 0268 4. ZRCHB3.B16 Init code source module
- : C 0269 5. ZRCHB4.B16 Test source module
- : C 0270 6. ZRCHB5.B16 Global routine source module
- : C 0271 7. ZRCHB6.B16 DM formatter down-line load executable module
- : C 0272 8. ZRCHB7.B16 DRS> Last address source module
- : C 0273 9. ZRCHB8.EXE DMCONV.EXE Dmconv executable code
- : C 0274 10. ZRCHB9.BP2 DMCONV.BP2 Dmconv source module
- : C 0275 11. ZRCHB10.LST AZFMTR.LST DM formatter list file
- : C 0276 12. ZRCHB11.MAC AZFMTR.MAC DM formatter source macro module
- : C 0277 13. ZRCHB12.SAV AZFMTR.SAV DM formatter executable code
- : C 0278 14. ZRCHB.COM Project indirect command file
- : C 0279 15. ZRCHB.DOC Program document listing file
- : C 0280 16. ZRCHB.SEQ Host sequence listing
- : C 0281 17. ZRCHB.BIN Program XXDP*.bin file
- : C 0282

```

: C 0283      2.0 OPERATING INSTRUCTIONS
: C 0284
: C 0285      This section contains a brief description of the runtime services.
: C 0286      for detailed information, refer to the XXDP+ user's manual (CHQUS).
: C 0287
: C 0288      2.1 COMMANDS
: C 0289
: C 0290      There are eleven legal commands for the diagnostic runtime services
: C 0291      (supervisor). This section lists the commands and gives a very
: C 0292      brief description of them. The XXDP+ user's manual has more details.
: C 0293
: C 0294      COMMAND      EFFECT
: C 0295      -----
: C 0296      START          Start the diagnostic from an initial state
: C 0297
: C 0298      RESTART        Start the diagnostic without initializing
: C 0299
: C 0300      CONTINUE       Continue at test that was interrupted (after ^C)
: C 0301      Attempts to continue after a control c will result
: C 0302      in the present units formatting to be aborted and
: C 0303      the next logical units formatting to commence.
: C 0304
: C 0305      PROCEED         Continue from an error halt. This host program
: C 0306      is coded to be a utility and not a diagnostic,
: C 0307      therefor no error macros are used. For this
: C 0308      reason this DRS> command has no meaning.
: C 0309
: C 0310      EXIT            Return to XXDP+ monitor (XXDP+ operation only!)
: C 0311
: C 0312      ADD             Activate a unit for testing (all units are
: C 0313      considered to be active at start time
: C 0314
: C 0315      DROP            Deactivate a unit
: C 0316
: C 0317      PRINT           Print statistical information. Report summary
: C 0318      coding is remote DM program driven and is not
: C 0319      implemented in the host.
: C 0320
: C 0321      ^ISPLAY        Type a list of all device information
: C 0322
: C 0323      FLAGS           Type the state of all flags (see section 2.3)
: C 0324
: C 0325      ZFLAGS         Clear all flags (see section 2.3)
: C 0326
: C 0327      A command can be recognized by the first three characters.
: C 0328      So you may, for example, type 'STA' instead of 'START'.
: C 0329
: C 0330      2.2 SWITCHES
: C 0331
: C 0332      There are several switches which are used to modify supervisor
: C 0333      operation. These switches are appended to the legal commands.
: C 0334      All of the legal switches are tabulated below with a brief
: C 0335      description of each. In the descriptions below, a decimal number
: C 0336      is designated by "DDDD".
: C 0337
: C 0338      SWITCH          EFFECT
: C 0339

```

```

: C 0340           /TESTS:LIST      Execute only those tests specified in
: C 0341           the list. List is a string of test
: C 0342           numbers, for example - /TESTS:1:5:7-10.
: C 0343           This list will cause tests 1,5,7,8,9,10 to
: C 0344           be run. All other tests will not be run.
: C 0345
: C 0346           /PASS:DDDDD      Execute DDDDD passes (DDDDD = 1 to 64000)
: C 0347           The host code will perform only one pass.
: C 0348           At completion of pass one the program will
: C 0349           be terminated regardless of this switch.
: C 0350
: C 0351           /FLAGS:FLGS      Set specified flags. flags are described
: C 0352           in section 2.3.
: C 0353
: C 0354           /EOP:DDDDD      Report end of pass message after every
: C 0355           DDDDD passes only. (DDDDD = 1 to 64000)
: C 0356           The host code will perform only one pass.
: C 0357           At completion of pass one the program will
: C 0358           be terminated regardless of this switch.
: C 0359
: C 0360           /UNITS:LIST      TEST/ADD/DROP only those units specified
: C 0361           in the list. List example /UNITS:0:5:10 12
: C 0362           use units 0,5,10,11,12 (unit numbers = 0-63)
: C 0363

```

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 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
FLAGS					
ZFLAGS					
EXIT					

C 0396

```

: C 0397
: C 0398      2.3  FLAGS
: C 0399
: C 0400      Flags are used to set up certain operational parameters such as
: C 0401      looping on error. All flags are cleared at startup and remain
: C 0402      cleared until explicitly set using the flags switch. Flags
: C 0403      are also cleared after a start command unless set using the
: C 0404      flag switch. The ZFLAGS command may also be used to clear
: C 0405      all flags, with the exception of the START and ZFLAGS commands.
: C 0406      No commands affect the state of the flags; they remain set or
: C 0407      cleared as specified by the last flag switch.
: C 0408
: C 0409      FLAG      EFFECT
: C 0410      - - - - -
: C 0411      HOE      Halt on error - control is returned to
: C 0412      runtime services command mode
: C 0413
: C 0414      LOE      Loop on error
: C 0415
: C 0416      IER*     Inhibit all error reports
: C 0417
: C 0418      IBR*     Inhibit all error reports except first
: C 0419      level (first level contains error type,
: C 0420      number, PC, test and unit)
: C 0421
: C 0422      IXR*     Inhibit extended error reports (those
: C 0423      called by PRINTX macro's)
: C 0424
: C 0425      PRI      Direct messages to line printer
: C 0426
: C 0427      PNT      Print test number as test executes
: C 0428
: C 0429      BOE      "BELL" on error
: C 0430
: C 0431      UAM      Unattended mode (no manual intervention)
: C 0432
: C 0433      ISR      Inhibit statistical reports (does not apply
: C 0434      to diagnostics which do not support statis-
: C 0435      tical reporting)
: C 0436
: C 0437      IDR      Inhibit program dropping of units
: C 0438
: C 0439      ADR      Execute autodrop code
: C 0440
: C 0441      LOT      Loop on test
: C 0442
: C 0443      EVL      Execute evaluation (on diagnostics which
: C 0444      have evaluation support)
: C 0445
: C 0446      *error messages are described in section 3.1
: C 0447
: C 0448
: C 0449      See the XXDP* user's manual for more details on flags. You may
: C 0450      specify more than one flag with the flag switch. For example,
: C 0451      to cause the program to loop on error, inhibit error reports
: C 0452      and type a "BELL" on error, you may use the following string:
: C 0453

```

/FLAGS:LOE:IER:BOE

2.4 HARDWARE QUESTIONS

When a diagnostic is started, the runtime services will prompt the user for hardware information by typing "CHANGE HW (L) ?". You must answer "Y" after a start command unless the hardware information has been "preloaded" using the setup utility (see Chapter 6 of the XXDP+ user's Manual). When you answer this question with a "Y", the runtime services will ask for the number of units (in decimal). You will then be asked the following questions for each unit.

UNITS (D) ?

Answer with the number of units to be tested (no default). This answer will determine how many times the following questions are asked. A unit is a logical disk (single platter) on an RC25. A maximum of sixteen units will be accepted by the init code.

RC25 IP REGISTER ADDRESS (D) 172150 ?

Answer with the address of the IP Register of one RC25 controller as addressed by the processor with memory management turned off (i.e., An even 16 bit address in the range of 160000 to 177774).

RC25 INTERRUPT VECTOR ADDRESS (D) 154 ?

Answer with the interrupt vector address of the RC25 controller. A vector address in the range of 4 to 774 may be specified.

RC25 BUS REQUEST LEVEL (D) 5 ?

Answer with the interrupt priority used by the RC25. Levels 4 to 7 are accepted.

UNIT NUMBER TO BRING ONLINE (D) 0 ?

Answer with the physical platter number you wish to bring online. The removable platter is an even number and the fixed platter is the sequentially following odd number.

2.5 SOFTWARE QUESTIONS

The supervisor will ask the question "CHANGE SW (L) ?". This question is to be answered with 'Y'. The following message will be printed to the console terminal:

FORMAT IN UNATTENDED REFORMAT MODE (L) YES ?

A 'YES' response (the default) will cause the host to run in UN ATTENDED reformat mode (see section 7.2 for description).

A 'NO' response will cause the host to run in ATTENDED mode (see section 7.1 for description). In this mode an operator must be present at the

```

: C 0511 console terminal to answer the following DM formatter software questions:
: C 0512
: C 0513 The DM will ask the following questions. The question numbers are those
: C 0514 used as the DUP question numbers.
: C 0515
: C 0516 0. Enter date:
: C 0517 Enter the current date in MM-DD-YYYY.
: C 0518
: C 0519 1. Enter unit number to format:
: C 0520 Enter the number of the unit to format.
: C 0521
: C 0522 2. Enter sector size to be used (512/576):
: C 0523 This question is for compatibility with other formatters.
: C 0524 The RC25 formatter will skip this question.
: C 0525
: C 0526 3. Enter mode to be used (slow/normal/fast):
: C 0527 This question is for compatibility with other formatters.
: C 0528 The RC25 formatter will skip this question.
: C 0529
: C 0530 4. Use existing bad block information (y/n):
: C 0531 If the answer is YES, a REFORMAT mode format is done. If
: C 0532 the answer is NO, the mode of format is determined by
: C 0533 question 5. If the answer is YES, question 5 is skipped.
: C 0534
: C 0535 5. DOWN-LINE LOAD bad block information (y/n):
: C 0536 An answer of NO will cause a RECONSTRUCT mode format to be
: C 0537 performed. If an answer of YES is given, a RESTORE mode
: C 0538 format will be performed. An answer of NO will skip
: C 0539 question 6.
: C 0540
: C 0541 NOTE: It is the responsibility of the host program to
: C 0542 obtain the name of the file to be used for the DOWN-LINE
: C 0543 LOAD.
: C 0544
: C 0545 6. Continue if bad block information is inaccessible (y/n):
: C 0546 An answer of YES will cause a RECONSTRUCT mode format to be
: C 0547 done if the formatter cannot read the FCT on the disk or if
: C 0548 a RESTORE mode format was attempted and the bad block data
: C 0549 file was unavailable. An answer of NO will cause the format
: C 0550 to abort if the bad block information is inaccessible. An
: C 0551 answer of NO will terminate the question sequence.
: C 0552
: C 0553 FCT blocks will be requested via the special DUP message
: C 0554 code. The first data word in this message will be the FCT
: C 0555 block (relative to zero) requested by the formatter. All
: C 0556 DUP sequence numbers for FCT block requests will be 1. The
: C 0557 response is a 3 word block from the host. The first word is
: C 0558 0 if the block was successfully retrieved and non-zero if it
: C 0559 was not. The next 2 words contain the UNIBUS address of a
: C 0560 buffer containing the 512 byte block from the FCT file.
: C 0561
: C 0562 7. Enter serial number:
: C 0563 Enter the 64-bit decimal serial number of the disk to be
: C 0564 formatted. In the RC25 formatter this number is used only
: C 0565 if a RECONSTRUCT mode format is used.
: C 0566

```

2.6 EXTENDED P TABLE DIALOGUE

When you answer the hardware questions, you are building entries in a table that describes the devices under test. The simplest way to build this table is to answer all questions for each unit to be tested. If you have a multiplexed device such as a mass storage controller with several drives or a communication device with several lines, this becomes tedious since most of the answers are repetitious.

To illustrate a more efficient method, suppose you are testing a fictional device, the XY11. Suppose this device consists of a control module with eight units (sub-devices) attached to it. These units are described by the octal numbers 0 through 7. There is one hardware parameter that can vary among units called the Q-FACTOR. This Q-FACTOR may be 0 or 1. Below is a simple way to build a table for one xy11 with eight units.

• UNITS (0) ? 8<CR>

UNIT 1

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 0<CR>

Q-FACTOR (0) 0 ? 1<CR>

UNIT 2

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 1<CR>

Q-FACTOR (0) 1 ? 0<CR>

UNIT 3

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 2<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 4

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 3<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 5

CSR ADDRESS (0) ? 160000<CR>

SUB DEVICE # (0) ? 4<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 6

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 5<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 7

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 6<CR>

Q FACTOR (0) 0 ? 1<CR>

UNIT 8

CSR ADDRESS (0) ? 160000<CR>

: C 0567
: C 0568
: C 0569
: C 0570
: C 0571
: C 0572
: C 0573
: C 0574
: C 0575
: C 0576
: C 0577
: C 0578
: C 0579
: C 0580
: C 0581
: C 0582
: C 0583
: C 0584
: C 0585
: C 0586
: C 0587
: C 0588
: C 0589
: C 0590
: C 0591
: C 0592
: C 0593
: C 0594
: C 0595
: C 0596
: C 0597
: C 0598
: C 0599
: C 0600
: C 0601
: C 0602
: C 0603
: C 0604
: C 0605
: C 0606
: C 0607
: C 0608
: C 0609
: C 0610
: C 0611
: C 0612
: C 0613
: C 0614
: C 0615
: C 0616
: C 0617
: C 0618
: C 0619
: C 0620
: C 0621
: C 0622
: C 0623


```

: C 0624          SUB-DEVICE # (0) ? 7<CR>
: C 0625          Q-FACTOR (0) 1 ? <CR>
: C 0626
: C 0627          Notice that the default value for the Q-FACTOR changes when a
: C 0628          non default response is given. Be careful when specifying
: C 0629          multiple units!
: C 0630
: C 0631          As you can see from the above example, the hardware parameters
: C 0632          do not vary significantly from unit to unit. The procedure shown
: C 0633          is not very efficient.
: C 0634
: C 0635          The runtime services can take multiple unit specifications however.
: C 0636          Let's build the same table using the multiple specification feature.
: C 0637
: C 0638          # UNITS (0) ? 8<CR>
: C 0639
: C 0640          UNIT 1
: C 0641          CSR ADDRESS (0) ? 160000<CR>
: C 0642          SUB DEVICE # (0) ? 0,1<CR>
: C 0643          Q-FACTOR (0) 0 ? 1,0<CR>
: C 0644
: C 0645          UNIT 3
: C 0646          CSR ADDRESS (0) ? 160000<CR>
: C 0647          SUB-DEVICE # (0) ? 2-5<CR>
: C 0648          Q-FACTOR (0) 0 ? 0<CR>
: C 0649
: C 0650          UNIT 7
: C 0651          CSR ADDRESS (0) ? 160000<CR>
: C 0652          SUB-DEVICE # (0) ? 6,7<CR>
: C 0653          Q-FACTOR (0) 0 ? 1<CR>
: C 0654
: C 0655          As you can see in the above dialogue, the runtime services will
: C 0656          build as many entries as it can with the information given in any
: C 0657          one pass through the questions. In the first pass, two entries
: C 0658          are built since two sub-devices and Q-FACTORS were specified. The
: C 0659          services assume that the CSR address is 160000 for both since it
: C 0660          was specified only once. In the second pass, four entries were
: C 0661          built. This is because four sub-devices were specified. The
: C 0662          "-" construct tells the runtime services to increment the data
: C 0663          from the first number to the second. In this case, sub-devices
: C 0664          2, 3, 4 and 5 were specified. (If the sub-device were specified
: C 0665          by addresses, the increment would be by 2 since addresses must
: C 0666          be on an even boundary.) The CSR addresses and Q-FACTORS for
: C 0667          the four entries are assumed to be 160000 and 0 respectively
: C 0668          since they were only specified once. The last two units are
: C 0669          specified in the third pass.
: C 0670
: C 0671          The whole process could have been accomplished in one pass as shown
: C 0672          below.
: C 0673
: C 0674          # UNITS (0) ? 8<CR>
: C 0675
: C 0676          UNIT 1
: C 0677          CSR ADDRESS (0) ? 160000<CR>
: C 0678          SUB DEVICE # (0) ? 0 7<CR>
: C 0679          Q-FACTOR (0) 0 ? 0,1,0,...,1,1<CR>
: C 0680

```

```

: C 0681      As you can see from this example, null replies (commas enclosing
: C 0682      a null field) tell the runtime services to repeat the last reply.
: C 0683
: C 0684      2.7 QUICK START UP PROCEDURE (XXDP.)
: C 0685
: C 0686      To start-up this program:
: C 0687
: C 0688          1. Boot XXDP.
: C 0689
: C 0690          2. Give the date
: C 0691
: C 0692          3. Type 'R ZRCHB0', where name is the name of the bin or bic
: C 0693             file for this program
: C 0694
: C 0695          4. Type 'START'
: C 0696
: C 0697          5. Answer the "CHANGE HW" question with "Y"
: C 0698
: C 0699          6. Answer all the hardware questions
: C 0700
: C 0701          7. Answer the "CHANGE SW" question with "N"
: C 0702
: C 0703      When you follow this procedure you will be using only the
: C 0704      defaults for flags and software parameters. These defaults
: C 0705      are described in sections 2.3 and 2.5.
: C 0706

```

: C 0707
 : C 0708
 : C 0709
 : C 0710
 : C 0711
 : C 0712
 : C 0713
 : C 0714
 : C 0715
 : C 0716
 : C 0717
 : C 0718
 : C 0719
 : C 0720
 : C 0721
 : C 0722
 : C 0723
 : C 0724
 : C 0725
 : C 0726
 : C 0727
 : C 0728
 : C 0729
 : C 0730
 : C 0731
 : C 0732
 : C 0733
 : C 0734
 : C 0735
 : C 0736
 : C 0737
 : C 0738
 : C 0739

2.8 PID (PROCESS INDICATOR) WORDS

The process indicator words (PID) are maintained by the remote DM program and indicates to the host the DM programs progress running in the controller.

These PID words are then obtained by the host via the DUP GET_DUST_STATUS command. The newly obtained PID are compared to previous PID words. If the PID has not increased since the last GET_DUST_STATUS then the remote program is to be considered dead and the connection to the controller broken.

However if the PID value has increased since the last GET_DUST_STATUS then the remote program is to be considered still running. The new PID words are saved and command time out waits reinitiated.

This host code will print out to the console terminal the PID values after each GET_DUST_STATUS while the formatter is running in the controller. The printing format is as follows:

```
FORMATTER PROGRESSING: PID HI=xxxxxxx(0) PID LO=xxxxxxx(0)
```

To further demonstrate to the host the progress of the DM formatter the following is performed:

Each time the DM formatter calls in a new overlay from host memory the low PID word is cleared and the high PID word is incremented. The operator can then be observed when new overlays are called into DM memory.

2.9 HOST/DM FORMATTER RUN TIME MESSAGES

All host: messages, fatal errors or operator prompts are printed in UPPER case characters while all DM formatter: messages, fatal errors, completion information or operator prompts are printed in lower case characters.

```

: C 0740      3.0 ERROR INFORMATION
: C 0741
: C 0742      3.1 TYPES OF ERROR MESSAGES
: C 0743
: C 0744      Due to the fact that this is an utility and not a diagnostic error macro
: C 0745      calls will not be used by this host code to report errors. Instead all
: C 0746      errors will be printed to the operator via the PRINTF macro using the
: C 0747      prefix "$FTLERR- error message text" indicating this to be a fatal error.
: C 0748
: C 0749      3.2 SPECIFIC ERROR MESSAGES
: C 0750
: C 0751      DM Code Error Messages
: C 0752
: C 0753      All errors returned are fatal. Error messages returned by the RC25
: C 0754      formatter are as follows. The error numbers given are the DUP error
: C 0755      message numbers.
: C 0756
: C 0757          1 "GET STATUS failure"
: C 0758          2 "LESI send error"
: C 0759          3 "Unsuccessful LESI command"
: C 0760          4 "LESI receive error"
: C 0761          5 "UNIBUS I/O error"
: C 0762          6 "Formatter initialization error"
: C 0763          7 "Nonexistent unit number"
: C 0764          8 "DBN/XBN format error (drive FORMAT command failed)"
: C 0765          9 "FCT does not have enough good copies of each block"
: C 0766         10 "SEEK error"
: C 0767         11 "RCT does not have enough good copies of each block"
: C 0768         12 "LBN format error (drive FORMAT command failed)"
: C 0769         13 "FCT write error"
: C 0770         14 "RCT read error"
: C 0771         15 "RCT write error"
: C 0772         16 "RCT full"
: C 0773         17 "FCT read error"
: C 0774         18 "FCT nonexistent"
: C 0775         19 "FCT downline-load error"
: C 0776         20 "Drive init timeout"
: C 0777         21 "Illegal response to start-up quest'on"
: C 0778         22 "WARNING possible head addressing problems run diagnostics"
: C 0779         23 "INPUT Error"
: C 0780
: C 0781

```

ZRCHB1
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
USER DOCUMENTATION

5-Apr-1984 13:43:03
22-Feb-1984 12:03:49

SEQ 0019
Page 19
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE,AZTEC]ZRCHB1.B16;7 (12)

```

:      C 0782      Host Error Messages
:      C 0783
:      C 0784      :
:      C 0785      ! Generic self-detected fatal port/controller errors
:      C 0786      !
:      C 0787      $FTLERR- UNRECOGNIZABLE ERROR CODE
:      C 0788      $FTLERR- ENVELOPE/PACKET READ (PARITY OR TIMEOUT)
:      C 0789      $FTLERR- ENVELOPE/PACKET WRITE (PARITY OR TIMEOUT)
:      C 0790      $FTLERR- CONTROLLER ROM AND RAM PARITY
:      C 0791      $FTLERR- CONTROLLER RAM PARITY
:      C 0792      $FTLERR- CONTROLLER ROM PARITY
:      C 0793      $FTLERR- RING READ (PARITY OR TIMEOUT)
:      C 0794      $FTLERR- RING WRITE (PARITY OR TIMEOUT)
:      C 0795      $FTLERR- INTERRUPT MASTER
:      C 0796      $FTLERR- HOST ACCESS TIMEOUT
:      C 0797      $FTLERR- CREDIT LIMIT EXCEEDED
:      C 0798      $FTLERR- BUS MASTER ERROR
:      C 0799      $FTLERR- DIAGNOSTIC CONTROLLER FATAL ERROR
:      C 0800      $FTLERR- INSTRUCTION LOOP TIMEOUT
:      C 0801      $FTLERR- INVALID CONNECTION IDENTIFIER
:      C 0802      $FTLERR- INTERRUPT WRITE
:      C 0803      $FTLERR- MAINTENANCE READ/WRITE INVALID REGION IDENTIFIER
:      C 0804      $FTLERR- MAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER
:      C 0805      $FTLERR- CONTROLLER RAM ERROR (NON-PARITY)
:      C 0806      $FTLERR- INIT SEQUENCE ERROR
:      C 0807      $FTLERR- HIGH LEVEL PROTOCOL INCOMPATIBILITY ERROR
:      C 0808      $FTLERR- PURGE/POLL HARDWARE FAILURE
:      C 0809      $FTLERR- MAPPING REGISTER READ ERROR (PARITY OR TIMEOUT)

```

```

: C 0810      :
: C 0811      : RC25 Self-detected fatal port/controller errors
: C 0812      :
: C 0813      : $FTLERR- VAX READ/WRITE ERROR ON INTERRUPT
: C 0814      : $FTLERR- INCONSISTENCY AT U.BFIL
: C 0815      : $FTLERR- INCONSISTENCY AT U.BMTY
: C 0816      : $FTLERR- INCONSISTENCY AT U.ALOC
: C 0817      : $FTLERR- INCONSISTENCY AT SERVO ENTRY (PIP SET)
: C 0818      : $FTLERR- INCONSISTENCY AT SERVO ENTRY (ERR SET)
: C 0819      : $FTLERR- INCONSISTENCY AT U.SEND
: C 0820      : $FTLERR- INCONSISTENCY AT U.RECV
: C 0821      : $FTLERR- INCONSISTENCY AT U.ATTN
: C 0822      : $FTLERR- INCONSISTENCY AT U.ONLN
: C 0823      : $FTLERR- ILLEGAL D REQUEST (U.QDRQ)
: C 0824      : $FTLERR- FENCE-POST ERROR AT PROTAB
: C 0825      : $FTLERR- BAD PACKET DEQUEUED AT U.DONE
: C 0826      : $FTLERR- UNEXPLAINED D-PROC SUSPENSION (U..TDS)
: C 0827      : $FTLERR- DUP PACKET D-Q FAILED (XFC 34/35)
: C 0828      : $FTLERR- INCONSISTENCY AT U.HTST
: C 0829      : $FTLERR- INCONSISTENCY AT U.SEKO
: C 0830      : $FTLERR- INCONSISTENCY AT U.CKSV
: C 0831      : $FTLERR- D.OPCD FOUND ILLEGAL OPCODE
: C 0832      : $FTLERR- D.CSF FOUND ILLEGAL OPCODE
: C 0833      : $FTLERR- UNKNOWN BAD DRIVE STATUS AT D.DSTS
: C 0834      : $FTLERR- ILLEGAL XFC EXECUTED BY DM
: C 0835      : $FTLERR- D PICKED UP A ZERO SCB.DB
: C 0836      : $FTLERR- INCONSISTENCY AT D IDLE LOOP
: C 0837      : $FTLERR- DM WORD COUNT ERROR ON HOST DMA/SEND/RECV
: C 0838      : $FTLERR- UNKNOWN DISPLAY FAULT CODE AT D.DFLT
: C 0839      : $FTLERR- DRIVE NOT FAULTING IN P.OFLN STATE
: C 0840      : $FTLERR- U POWER UP DIAGNOSTICS FAILED
: C 0841      : $FTLERR- D POWER UP DIAGNOSTICS FAILED
: C 0842      : $FTLERR- ADAPTER CARD FAILURE
: C 0843      : $FTLERR- EC.TMR TIMED OUT
: C 0844      : $FTLERR- U.SEND/U.RECV RING READ INCONSISTENCY
: C 0845      : $FTLERR- UNKNOWN WAITRV REASON AT D.RVCT
: C 0846      : $FTLERR- D.ARCS DID NOT FIND CLOSEST UNDONE ZONE
: C 0847      : $FTLERR- U.SEEK FOUND SEEK TO ILLEGAL TRACK
: C 0848      : $FTLERR- U.HTST INIT DIAG DMA WRITE FAILED
: C 0849      : $FTLERR- U.HTST INIT DIAG DMA COMPARE FAILED
: C 0850      : $FTLERR- U.SYDR FOUND SS.DER SET AND SS.SPN NOT SET
: C 0851      : $FTLERR- MASTER DRIVES ACLO ASSERTED

```

ZRCHB1
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
USER DOCUMENTATION

5-Apr-1984 13:43:03
22-Feb-1984 12:03:49

VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB1.B16;7 (14)

SEQ 0021

Page 21

```

:      C 0852      :
:      C 0853      : Host runtime fatal error messages
:      C 0854      :
:      C 0855      : $FTLERR- NON-EXISTENT RC25 REGISTER ADRS xxxxxx
:      C 0856      : $FTLERR- DUP SERVER ACTIVE AFTER INITIALIZATION
:      C 0857      : $FTLERR- DUP SERVER INACTIVE AFTER EX_SUP_PROG COMMAND
:      C 0858      : $FTLERR- RESPONSE STATUS ERROR: actual status error message
:      C 0859      : $FTLERR- HOST/CONTROLLER OUT OF SEQ
:      C 0860      : $FTLERR- REMOTE PROG NOT RUNNING
:      C 0861      : $FTLERR- UNKNOWN RETURN STATUS CODE
:      C 0862      : $FTLERR- COM AREA INIT ERROR
:      C 0863      : $FTLERR- PORT/HOST SYNC ERROR
:      C 0864      : $FTLERR- MESSAGE LENGTH ERROR
:      C 0865      : $FTLERR- UNKNOWN ENCODE RECEIVED
:      C 0866      : $FTLERR- ADAPTER PURGE ERROR
:      C 0867      : $FTLERR- UNKNOWN INTERRUPT
:      C 0868      : $FTLERR- INIT SEQ STEP TIMED OUT
:      C 0869      : $FTLERR- INIT SEQ COMPARE ERROR
:      C 0870      : $FTLERR- UNEXPECTED ATTENTION END MESSAGE RECEIVED
:      C 0871      : $FTLERR- UNEXPECTED COMMAND OPCODE IN END MESSAGE RECEIVED
:      C 0872      : $FTLERR- UNEXPECTED SERIOUS EXCEPTION END MESSAGE RECEIVED
:      C 0873      : $FTLERR- INVALID COMMAND END MESSAGE RECEIVED
:      C 0874      : $FTLERR- UNKNOWN MESSAGE TYPE RECEIVED
:      C 0875      : $FTLERR- OUTSTANDING COMMAND BUFFER FULL
:      C 0876      : $FTLERR- OUTSTANDING COMMAND BUFFER OUT OF SYNC ERROR
:      C 0877      : $FTLERR- UNKNOWN MESSAGE NUMBER RECEIVED
:      C 0878      : $FTLERR- FILE READ ERROR
:      C 0879      : $FTLERR- PORT/CONTROLLER TIMEOUT ERROR
:      C 0880      : $FTLERR- ILLEGAL FCT FILE LENGTH

```

```

: C 0881      :
: C 0882      : Init code fatal error messages
: C 0883      :
: C 0884      : $FTLERR- NO ADDITIONAL UNITS TO FORMAT - ABORTING
: C 0885      : $FTLERR- INIT CODE RE-ENTERED DUE TO PWR FAIL
: C 0886      : $FTLERR- ABORTING HOST AND REMOTE PROGRAMS
: C 0887      : $FTLERR- ILLEGAL NUMBER OF UNITS SELECTED
: C 0888      : $FTLERR- LIMIT OF SIXTEEN UNITS PER FORMATTING SESSION
: C 0889      : $FTLERR- RC25 CONTROLLER INITIALIZATION ERROR
: C 0890      : $FTLERR- PROTOCOL VIOLATION ERROR
: C 0891      : $FTLERR- COMMUNICATION AREA INIT ERROR
: C 0892      :
: C 0893      : Dup return status codes
: C 0894      :
: C 0895      : SUCCESSFUL
: C 0896      : INVALID COMMAND
: C 0897      : NO REGION AVAILABLE
: C 0898      : NO REGION SUITABLE
: C 0899      : PROGRAM NOT KNOWN
: C 0900      : LOAD FAILURE
: C 0901      : STANDALONE
: C 0902      :
: C 0903      : MSCP return status codes
: C 0904      :
: C 0905      : SUCCESS
: C 0906      : INVALID COMMAND
: C 0907      : COMMAND ABORTED
: C 0908      : UNIT-OFFLINE
: C 0909      : UNIT-AVAILABLE
: C 0910      : MEDIA FORMAT ERROR
: C 0911      : WRITE PROTECTED
: C 0912      : COMPARE ERROR
: C 0913      : DATA ERROR
: C 0914      : HOST BUFFER ACCESS ERROR
: C 0915      : CONTROLLER ERROR
: C 0916      : DRIVE ERROR
: C 0917      : MESSAGE FROM AN INTERNAL DIAGNOSTIC
: C 0918      :

```



```

: C 0919      4.0 PERFORMANCE AND PROGRESS REPORTS
: C 0920
: C 0921      The formatter issues the following messages upon normal completion. All
: C 0922      but the last are sent as DUP informational messages. The last is sent
: C 0923      as a DUP termination message.
: C 0924
: C 0925      1. "format completed"
: C 0926
: C 0927      2. "n revectorred LBNs"
: C 0928      Where n is the number of LBNs revectorred in the user data
: C 0929      area.
: C 0930
: C 0931      3. "n primary revectorred LBNs"
: C 0932      Where n is the number of LBNs in message #2 which were
: C 0933      primary revectorred.
: C 0934
: C 0935      4. "n secondary/tertiary revectorred LBNs"
: C 0936      Where n is the number of the LBNs in message #2 which were
: C 0937      secondary or tertiary revectorred.
: C 0938
: C 0939      5. "n bad block' in the RCT area due to data errors"
: C 0940      Where n is the number of blocks in the total RCT area which
: C 0941      were bad due to errors in the data portion of their sectors.
: C 0942
: C 0943      6. "n bad blocks in the RCT due to header or timing errors"
: C 0944      Where n is the number of blocks in the total RCT area which
: C 0945      were bad due to errors in the header or timing areas of
: C 0946      their sectors.
: C 0947
: C 0948      7. "n bad blocks in the DBN area due to data errors"
: C 0949      Where n is the number of blocks in the DBN area which were
: C 0950      bad due to errors in the data area of their sectors.
: C 0951
: C 0952      8. "n bad blocks in the DBN area due to header or timing
: C 0953      errors"
: C 0954      Where n is the number of blocks in the DBN area which were
: C 0955      bad due to errors in the header or timing area of their
: C 0956      sectors.
: C 0957
: C 0958      9. "n bad blocks in the XBN area due to oata errors"
: C 0959      Where n is the number of blocks in the XBN area which were
: C 0960      bad due to errors in the data area of their sectors.
: C 0961
: C 0962      10. 'n bad blocks in the XBN area due to header or timing
: C 0963      errors"
: C 0964      Where n is the number of blocks in the XBN area which were
: C 0965      bad due to errors in the header or timing area of their
: C 0966      sectors.
: C 0967
: C 0968      11. "n bad RBNS"
: C 0969      Where n is the number of blocks in the RBN area which were
: C 0970      bad due to errors in the data area of their sectors.
: C 0971
: C 0972      12. "n blocks retried on the check pass"
: C 0973      Where n is the number of blocks which had an error on the
: C 0974      first read attempt after formatting.
: C 0975

```

```

:      C 0976      13. "FCT used successfully" or
:      C 0977
:      C 0978      14. "FCT was not used"
:      C 0979      Depending on the answers to the start-up questions and the
:      C 0980      availability of the bad block information (FCT). This
:      C 0981      message has the DUP termination message code.
:      C 0982
:      C 0983      5.0 DEVICE INFORMATION TABLES
:      C 0984
:      C 0985      :
:      C 0986      : Hardware parameter coding questions
:      C 0987      :
:      C 0988      HW_Q1_IP      = IP REGISTER ADDRESS
:      C 0989      HW_Q2_VECTOR = INTERRUPT VECTOR ADDRESS
:      C 0990      HW_Q3_BR      = BUS REQUEST LEVEL
:      C 0991      HW_Q4_UNIT    = UNIT NUMBER TO FORMAT
:      C 0992

```

```

: C 0993      6.0 TEST SUMMARIES
: C 0994
: C 0995      6.1 Reformat Mode Procedure
: C 0996
: C 0997      Reformat mode is accomplished in four sections:
: C 0998
: C 0999      * Table Setup
: C 1000      * Scanning
: C 1001      * Replacement
: C 1002      * Check Pass
: C 1003
: C 1004
: C 1005      6.1.1 Table Setup
: C 1006
: C 1007      This section will build an RCT skeleton and perform part of the primary
: C 1008      revectoring.
: C 1009
: C 1010      A. The presence of a good FCT will be determined by the ability
: C 1011      to read each block in an FCT copy, and by verifying the data
: C 1012      contained in the FCT Information Block.
: C 1013
: C 1014      If none of the four FCT copies is good, the Formatter will
: C 1015      either:
: C 1016
: C 1017      1. Abort the program if DM software question #6 was
: C 1018      answered with a 'NO'.
: C 1019
: C 1020      2. Will do a Reconstruct mode if software question #6
: C 1021      was answered with a 'YES'.
: C 1022
: C 1023      (Software question #6 is: Continue if bad block
: C 1024      information is inaccessible (Y/N):).
: C 1025
: C 1026      B. The RCT tracks will be formatted. Bad blocks in the RCT
: C 1027      area will be marked with a header code of 11. If a bad
: C 1028      block occurs in the same position in each copy of the RCT,
: C 1029      the format will be aborted, as the media is unusable.
: C 1030
: C 1031      C. Both outer and inner DBN areas will be formatted with here.
: C 1032      Two of these inner DBN tracks will be formatted with +20.
: C 1033      and -20. offset for diagnostic purposes.
: C 1034
: C 1035      D. A partial RCT will be built from the FCT: The RCT
: C 1036      Information Block will be assembled. FCT Bad Block
: C 1037      Descriptors coded for primary revectoring will be
: C 1038      transformed into RCT entries. This is done by converting
: C 1039      the PBN address into an LBN address, and entering it into
: C 1040      the corresponding RBN descriptor slot for the particular
: C 1041      track, with a code of 02. Other RBN descriptors will be
: C 1042      designated as unallocated (code 00) if they have a
: C 1043      corresponding RBN, or as nulls (code 10) if they do not.
: C 1044
: C 1045      Two copies of the RCT will be written to each of the two RCT
: C 1046      tracks.
: C 1047
: C 1048      All LBN's marked for secondary revectoring will be stored in
: C 1049      an "R2" Table. Replacement RBN's for these LBN's will be

```

resolved after primary revectoring is completed.

6.1.2 Scanning

NOTE:

The following section 6.1.2 generally describes the scanning philosophy used by this formatter. Specific scanning philosophies for each formatting mode is as follows:

REFORMAT mode:

In this mode the FCT file is accessible. The RCT file is built from the FCT and the LBN area is scanned for additional bad blocks. This mode will scan only for data pattern W.

RESTORE mode:

In this mode an external FCT file is brought in from the Host and written to the media's FCT track. After the FCT file has been restored a normal Reformat mode is performed scanning for data pattern W only.

RECONSTRUCT mode:

In this mode the FCT file is inaccessible. The FCT I.D. block is not formatted but rather read/write scanned with all data patterns. The I.D. is then rebuilt indicating zero FCT entries. The Reformat mode overlay is then called in to DM space and will read/write scan the LBN area with all data patterns.

This section writes headers throughout the LBN area. It scans for additional bad blocks, and completes the primary revectoring.

Scanning will start at LBN Track 0 (Top Surface OGB) and progress to LBN Track Max (Bottom Surface OGB) following logical ordering of LBN tracks.

E. An LBN track is written with headers and Data Pattern W, using the Format Track on Index XFC. A failure of this XFC will result in 10 retries. If all retries fail, the format will be aborted.

The header codes connote block status. They will be determined by the FCT Bad Block Descriptors as in the following table:

FCT Descriptor Code	Header Code
Not listed	00 - Good LBN
12 or 14	05 - Primary revector
02 or 04	03 - Secondary revector
01	11 - Secondary revector
	header problem
Any for RBN	11 - Jusable RBN

: C 1050
:
: C 1051
:
: C 1052
:
: C 1053
:
: C 1054
:
: C 1055
:
: C 1056
:
: C 1057
:
: C 1058
:
: C 1059
:
: C 1060
:
: C 1061
:
: C 1062
:
: C 1063
:
: C 1064
:
: C 1065
:
: C 1066
:
: C 1067
:
: C 1068
:
: C 1069
:
: C 1070
:
: C 1071
:
: C 1072
:
: C 1073
:
: C 1074
:
: C 1075
:
: C 1076
:
: C 1077
:
: C 1078
:
: C 1079
:
: C 1080
:
: C 1081
:
: C 1082
:
: C 1083
:
: C 1084
:
: C 1085
:
: C 1086
:
: C 1087
:
: C 1088
:
: C 1089
:
: C 1090
:
: C 1091
:
: C 1092
:
: C 1093
:
: C 1094
:
: C 1095
:
: C 1096
:
: C 1097
:
: C 1098
:
: C 1099
:
: C 1100
:
: C 1101
:
: C 1102
:
: C 1103
:
: C 1104
:
: C 1105
:
: C 1106

NOTE

: C 1107
 : C 1108
 : C 1109
 : C 1110
 : C 1111
 : C 1112
 : C 1113
 : C 1114
 : C 1115
 : C 1116
 : C 1117
 : C 1118
 : C 1119
 : C 1120
 : C 1121
 : C 1122
 : C 1123
 : C 1124
 : C 1125
 : C 1126
 : C 1127
 : C 1128
 : C 1129
 : C 1130
 : C 1131
 : C 1132
 : C 1133
 : C 1134
 : C 1135
 : C 1136
 : C 1137
 : C 1138
 : C 1139
 : C 1140
 : C 1141
 : C 1142
 : C 1143
 : C 1144
 : C 1145
 : C 1146
 : C 1147
 : C 1148
 : C 1149
 : C 1150
 : C 1151
 : C 1152
 : C 1153
 : C 1154
 : C 1155
 : C 1156
 : C 1157
 : C 1158
 : C 1159
 : C 1160
 : C 1161
 : C 1162
 : C 1163

If the RBN is not listed in the FCT, a single block per track may be designated for primary revectoring with an FCT descriptor code of 12 or 14.

If there are no bad blocks in the track, the forced error indicator will be set in the RBN at format XFC time by complimenting its expected EDC field.

- F. All blocks in the track will be read without revectoring enabled, and compared to Pattern W. A write-read-compare sequence will then be performed on each block in the track using Pattern M, which is the compliment of Pattern W.

Error-free blocks will be written with LBN-unique Pattern 1. If the track has a primary revector block, its data field will be written with 128 copies of the tracks RBN address, and the RBN will be written with LBN-unique Pattern 1 for the replaced block.

Each block, except used RBN and secondary replacement blocks, is read and its data compared to the expected value. This will be done with revectoring enabled for a primary replacement block, and disabled for the others.

This process is repeated using LBN-unique patterns 2, 3 and 4.

- G. Any error, including correctable errors, will result in 4 retries. Any block failing these will be considered a bad block. If no primary revector block already exists in the track, the first to fail will be primary revectorized by making an RCT entry in its RBN descriptor, reheadering it with a code of 05, and resetting the RBN forced error flag.

Other new bad blocks will be reheadered with secondary revector codes of 03 or 11, depending on the presence of header errors.

If any new bad block(s) is found, steps D and E will be partially repeated to include the additional bad block information, and verify the format write. The write-read-compare sequences with patterns 2, 3 and 4 will not be repeated, however.

- H. Addresses of blocks to be secondary revectorized will be accrued in an "R2 Table" within the program, until 127 entries are made. At this time, 8 copies of the table will be saved on outer DBN tracks starting at DBN block 0 top surface.

6.1.3 Replacement

This section will complete the RCT and resolve all secondary bad block replacement.

```

: C 1164
: C 1165
: C 1166
: C 1167
: C 1168
: C 1169
: C 1170
: C 1171
: C 1172
: C 1173
: C 1174
: C 1175
: C 1176
: C 1177
: C 1178
: C 1179
: C 1180
: C 1181
: C 1182
: C 1183
: C 1184
: C 1185
: C 1186
: C 1187
: C 1188
: C 1189
: C 1190
: C 1191
: C 1192
: C 1193
: C 1194
: C 1195
: C 1196
: C 1197
: C 1198
: C 1199
: C 1200
: C 1201
: C 1202
: C 1203
: C 1204
: C 1205
: C 1206
: C 1207
: C 1208
: C 1209
: C 1210
: C 1211
: C 1212
: C 1213
: C 1214
: C 1215
: C 1216
: C 1217
: C 1218
: C 1219
: C 1220

```

I. When all tracks have undergone the scanning section, The "R2 Tables" will be read from the media. The secondary revectorors will be resolved in the RCT, finding RBN Bad Block Descriptors through the Ping-pong algorithm.

Tracks whose RBN is assigned to be used for secondary revectoring will be written to reset the forced error indicator in the RBN.

6.1.4 Check pass

This section makes write and read passes of the media as a final check to verify revectoring capabilities.

J. Starting with the LBN track pair nearest the OGB, the data fields of all blocks on the media will be written as follows:

Block Type	Data
Good	Pattern 3
Primary revectored	Pattern 3 of replaced block
Secondary revectored	Pattern 3 of replaced block
Used RBN	Pattern 3 of replaced block

Starting with the top surface LBN track nearest the OGB all Host Application Blocks will be read to test for revectoring capabilities. Failure of any read or write will result in 10 retries. Failure of all retries will result in a fatal error message sent to the host and the format aborted.

6.2 Restore Mode Procedure

Restore mode will only be run by DIGITAL Manufacturing personnel. It requires an external copy of the FCT produced when the platter was manufactured and is stored on the System Boot Media. This procedure is as follows:

A. The FCT tracks will be formatted using the Format Track on Index XFC. A failure of this XFC will result in 10 retries. If all retries fail, the format will be aborted.

The header code of each block will be 12, indicating good FCT blocks. The data will be pattern W.

B. Each block will be read and its data verified. Write-read-compare sequences will then be performed on each block, using patterns M, 1, 2, 3 and 4.

Any errors will result in 10 retries. Any block failing a retry will be considered a bad block.

```

: C 1221      C. The tracks will be reformatted, if necessary, to change bad
: C 1222      block header codes to 11.
: C 1223
: C 1224      D. The operator will be queried for the original FCT file name.
: C 1225      The FCT file will then be inputted from the boot device and
: C 1226      sent to the DU when requested.
: C 1227
: C 1228      E. Two copies of the FCT will be written to each FCT track.
: C 1229      These will include bad blocks found in step B.
: C 1230
: C 1231      F. This mode will be aborted if any FCT sector fails in all 4
: C 1232      copies. The RECONSTRUCT mode will then be called in if the
: C 1233      option to continue if the FCT file is inaccessible else the
: C 1234      formatter will be aborted.
: C 1235
: C 1236      G. The program will automatically enter the Format Mode, and
: C 1237      proceed as in 4.1 above.
: C 1238
: C 1239
: C 1240      6.3 Reconstruct Mode Procedure
: C 1241
: C 1242      This mode will be performed if:
: C 1243
: C 1244      1. The operator has explicitly requested this as the formatting
: C 1245      mode by answering 'NO' to the following software questions:
: C 1246          . "Use existing Bad Block information?"
: C 1247          . "Down-line load Bad Block information?"
: C 1248
: C 1249      2. The Operator has responded with a 'YES' to the software
: C 1250      question:
: C 1251          . "Continue if Bad Block information is inaccessible?
: C 1252          and the Bad Block file is discovered inaccessible
: C 1253          during a Reformat or Restore Formatting Mode.
: C 1254
: C 1255      Reconstruct formatting procedure is as follows:
: C 1256
: C 1257      1. Sector 0 of each FCT copy will be read/write scanned.
: C 1258
: C 1259      The Formatting process will be aborted if all four copies of
: C 1260      Sector 0 are bad.
: C 1261
: C 1262      2. The Volume Information Block (Sector 0) will be
: C 1263      reconstructed using the following information:
: C 1264          . Media Mode                = 512 Byte Format
: C 1265          . Formatting Instance Number = 0
: C 1266          . Volume Serial Number      = Operator Supplied S.N.
: C 1267          . Time of First Formatting  = Operator Supplied Date
: C 1268          (VAX/VMS Format)
: C 1269          . Time of Most Recent Formatting = Operator Supplied Date
: C 1270          (VAX/VMS Format)
: C 1271
: C 1272
: C 1273
: C 1274
: C 1275
: C 1276
: C 1277

```

ZRCHB1
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
USER DOCUMENTATION

5-Apr-1984 13:43:03
22 Feb-1984 12:03:49

VAX-11 Bliss 16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB1.B16;7 (17)

SEQ 0030

Page 30

: C 1278
: C 1279
: C 1280
: C 1281
: C 1282
: C 1283
: C 1284
: C 1285
: C 1286
: C 1287
: C 1288
: C 1289
: C 1290
: C 1291
: C 1292
: C 1293
: C 1294
: C 1295
: C 1296
: C 1297
: C 1298
: C 1299

. Number of Used 512 Table Entries = 0

. Number of Used 576 Table Entries = 0

. Remaining Words Zeroed

3. The Volume Information Block is then written to each FCT copy and read back for data integrity.

Read Failures at all four copies will cause the formatting process to abort.

NOTE

Remaining FCT Sectors 1 thru 15 are not altered.

4. The formatting process will then perform the Reformat Mode Procedure.

: C 1300 7.0 HOST TEST SUMMARIES
: C 1301
: C 1302 7.1 ATTENDED FORMAT MODE
: C 1303
: C 1304 In this HOST mode the operator must be present at the console terminal
: C 1305 to respond to DM formatter software questions. Via these software questions
: C 1306 the operator can choose from any of the three formatting modes (REFORMAT,
: C 1307 RESTORE or RECONSTRUCT) and can specify the unit number to be formatted.
: C 1308 The unit number given to the DM formatter however must be the same unit
: C 1309 the host reported bringing on-line.
: C 1310
: C 1311 7.2 UN-ATTENDED REFORMAT MODE
: C 1312
: C 1313 In this HOST mode a REFORMAT mode will automatically be performed on all
: C 1314 physical unit numbers contained within operator built P_Tables.
: C 1315
: C 1316 When this mode is selected (which is the hosts default mode) the host
: C 1317 will ask the operator for the date in the same format as the DM formatter
: C 1318 would. This date is then automatically given to the DM formatter and the
: C 1319 physical unit number within the current P_Table is given to the DM as the
: C 1320 unit to format.
: C 1321
: C 1322 In this mode the operator can select up to 16 units (a unit being a physical
: C 1323 platter) to format and be free to leave the console terminal while the
: C 1324 units are being formatted.
: C 1325

:	C 1326	8.0 APPENDIX	
:	C 1327		APPENDIX A
:	C 1328		
:	C 1329		GLOSSARY
:	C 1330		
:	C 1331	DBN	Diagnostic Block Number. A number of per surface are reserved for diagnostic use. These tracks are called the DBN area.
:	C 1332		
:	C 1333		
:	C 1334	DM	Diagnostic Machine. An interpreter built into the RC25 firmware which is used to execute programs downline loaded into the RC25's buffer memory.
:	C 1335		
:	C 1336		
:	C 1337		
:	C 1338	EDC	Error Detecting Code. A means of verifying correct controller operation.
:	C 1339		
:	C 1340		
:	C 1341	FCT	Factory Control Table. A 16 sector table containing volume ID information and a map of bad blocks on the media. The FCT is written as part of the initial media formatting.
:	C 1342		
:	C 1343		
:	C 1344		
:	C 1345	FRU	Field Replaceable Unit. An item which may be exchanged in the field.
:	C 1346		
:	C 1347		
:	C 1348	IGB	Inner Guard Bands. The guard bands closer to the spindle.
:	C 1349		
:	C 1350	IP	Initialization and Polling Register. This register is used for port control.
:	C 1351		
:	C 1352		
:	C 1353	LBN	Logical Block Number. Most visible blocks including the Host Application Area and the RCT tracks.
:	C 1354		
:	C 1355		
:	C 1356	MSCP	Mass Storage Command Protocol. The method of communication used between the host and the RC25 over the bus.
:	C 1357		
:	C 1358		
:	C 1359	OGB	Outer Guard Bands. The guard bands further from the spindle.
:	C 1360		
:	C 1361	PBN	Physical Block Number. The absolute address of a block on RC25 media.
:	C 1362		
:	C 1363		
:	C 1364	RBN	Replacement Block Number. RC25 has one RBN per track for bad block replacement.
:	C 1365		
:	C 1366		
:	C 1367	RCT	Replacement and Caching Table. A 16 sector table containing media ID information and a map of the current bad block locations.
:	C 1368		
:	C 1369		
:	C 1370		
:	C 1371	RC	RC25 product name.
:	C 1372		
:	C 1373	SA	Status, Address and Purge Register for port control.
:	C 1374		
:	C 1375	XFC	Extended Function Call. Instructions which interface Diagnostic Machine code to the RC25 microcode.
:	C 1376		

APPENDIX B

BAD BLOCK REPLACEMENT

: C 1377
: C 1378
: C 1379
: C 1380
: C 1381
: C 1382
: C 1383 Bad block replacement is an addressing technique for replacing a bad
: C 1384 block with a good one. The block which replaces it is called a
: C 1385 replacement block, or RBN.
: C 1386
: C 1387 Each RC25 track has 32 blocks. In the host-addressable area, the first
: C 1388 thirty-one are for normal data storage, and are called Logical Blocks,
: C 1389 or LBNs. The last block on each of these tracks is an RBN.
: C 1390
: C 1391 If a single bad block exists on a track, it can be replaced by the RBN
: C 1392 of that track. This is called Primary replacement.
: C 1393
: C 1394 If other bad blocks are found in the same track, they must be replaced
: C 1395 by unused RBNs of other tracks. This is called Secondary replacement.
: C 1396
: C 1397 Two types of tables record bad block information for the media. One is
: C 1398 the Factory Control Table (FCT), which is written on the media at the
: C 1399 factory. It is a map of the bad blocks caused by the manufacturing
: C 1400 process. The other is the Replacement and Caching Table (RCT), which
: C 1401 includes both the FCT information, and bad block information uncovered
: C 1402 by the latest formatting of the media. There are four copies of each
: C 1403 table per unit.
: C 1404

APPENDIX C

FCT

```

: C 1405
: C 1406
: C 1407
: C 1408
: C 1409
: C 1410
: C 1411 The basic unit of the FCT is the Bad Block Descriptor. This descriptor
: C 1412 gives the Physical Block Address of the bad block, and a four bit code
: C 1413 indicating why it was retired.
: C 1414
: C 1415 The Physical Block Address, or PBN, is the absolute address of a block
: C 1416 anywhere on the media, without the distinction of host-accessibility.
: C 1417
: C 1418 There are a variable number of Bad Block Descriptors in the FCT. These
: C 1419 descriptors are sorted in descending track order, and in ascending PBN
: C 1420 order within tracks.

```

APPENDIX D

RCT

The basic unit of the RCT is the double-word RBN Descriptor. There are 1584 of these, one for each RBN in the host-accessible LBN area of the media.

The position of each RBN Descriptor in the RCT is fixed. The first RBN Descriptor is associated with the RBN block in the first host-visible track. The second descriptor with the RBN of the second host-visible track, etc..

In the format process, the FCT is given. Four copies of it are found on the media; two on each FCT track. The RCT is rebuilt from the FCT entries and additional bad block information is gathered in a scan pass. Primary replacement may be possible for one block per track if identified as bad by either of these sources. The tracks RBN must itself be a good, unused block.

Primary replacement is accomplished by:

- * Making the bad blocks header code 05.
- * Writing the bad blocks data field as 128 copies of its tracks RBN address.
- * Writing the address of the bad block to the associated RBN Descriptor in the RCT (with a code of 02).

Secondary replacement is needed for bad blocks which do not have a good, unused RBN block.

Secondary replacement is accomplished similarly to Primary Replacement, but first an unused RBN of another track must be found.

- * Find an unused RBN Descriptor in the RCT, using the Ping-pong algorithm.
- * Make the bad blocks header code 03 or 11, depending on error type.
- * Write the bad blocks data field with 128 copies of the target RBN address.
- * Write the address of the bad block to the associated RBN Descriptor in the RCT (with a code of 03).

After replacement an attempt by the host to access a replaced block will result in access to the block with which it had been replaced.

Good blocks are headered with codes of 00.

To help detect addressing problems, unused RBNs will have a forced error indication set by complimenting their EDC fields.

APPENDIX E
DATA PATTERNS

```

: C 1473
: C 1474
: C 1475
: C 1476
: C 1477
: C 1478
: C 1479 Six patterns will be used during the scan pass.
: C 1480
: C 1481 Pattern W is a pseudo-random pattern, with the LBN address as the first
: C 1482 word. Pattern M is its compliment. Its use will assure that all data
: C 1483 bits can toggle.
: C 1484
: C 1485 Patterns 1 through 4 are made up of single word elements. These
: C 1486 elements are as follows:
: C 1487
: C 1488     Element A = 071311.      This causes peak shift in data encoding.
: C 1489
: C 1490     Element B = 106466.    This causes single transitions spaced by
: C 1491     1, 2, 3 and 4 zeros.
: C 1492
: C 1493     Element C = 043146.    This causes both the highest and lowest
: C 1494     frequency of transitions in the encoding
: C 1495     logic.
: C 1496
: C 1497     Element D = 134631.    This also causes both the highest and
: C 1498     lowest frequency of transitions in the
: C 1499     encoding logic.
: C 1500
: C 1501     Element L = 17NXXX.    Where XXXX is the LBN address to which the
: C 1502     pattern is written.
: C 1503
: C 1504
: C 1505 These elements are assembled into patterns 1 through 4 as follows:
: C 1506
: C 1507     Pattern 1 = LBBAABBBBBAAAAABBBBBBAAAAAAABBBBBBBBB etc.
: C 1508
: C 1509     Pattern 2 = BLBBBBAAAAABBBBBBAAAAABBBBBBAAAAAAA etc.
: C 1510
: C 1511     Pattern 3 = LDDCCDDDDCCCCDDDDDDCCCCCCCCDDDDDDDD etc.
: C 1512
: C 1513     Pattern 4 = DLCDDCCCCDDDDDDCCCCCCCCDDDDDDDDCCCCCCCC etc.
: C 1514
: C 1515
: C 1516

```

: C 1517 Embedding the LBN address within the patterns gives them LBN-uniqueness.
: C 1518 This feature is taken advantage of in validating revectoring.
: C 1519
: C 1520 Three steps are taken to insure the data will not interfere with
: C 1521 Secondary Revectoring by matching the "Compare 128" algorithm. These
: C 1522 are:
: C 1523
: C 1524 * The high four bits in every word contain invalid header codes.
: C 1525
: C 1526 * Before secondary revectoring is attempted, blocks not involved
: C 1527 with revectoring will be written with data fields which
: C 1528 contain no 32-bit repetitive patterns.
: C 1529
: C 1530 * When the format is complete, no unrevectorized blocks will
: C 1531 contain data which has 32-bit repetitive patterns.
: C 1532

APPENDIX F

Ping Pong algorithm

The search begins at the primary replacement block descriptor. If the descriptor is not empty then a ping-pong search of the sector containing the primary replacement block descriptor ensues. If an empty descriptor is not encountered then a linear scan of the remaining RCT blocks and descriptors within blocks, (with wrap-around at the end of the RCT) ensues until one of two things occur:

1. An unallocated replacement block descriptor is encountered in an overflow location - a secondary.
2. The entire RCT is searched without success - a failure.

The search operates at two levels:

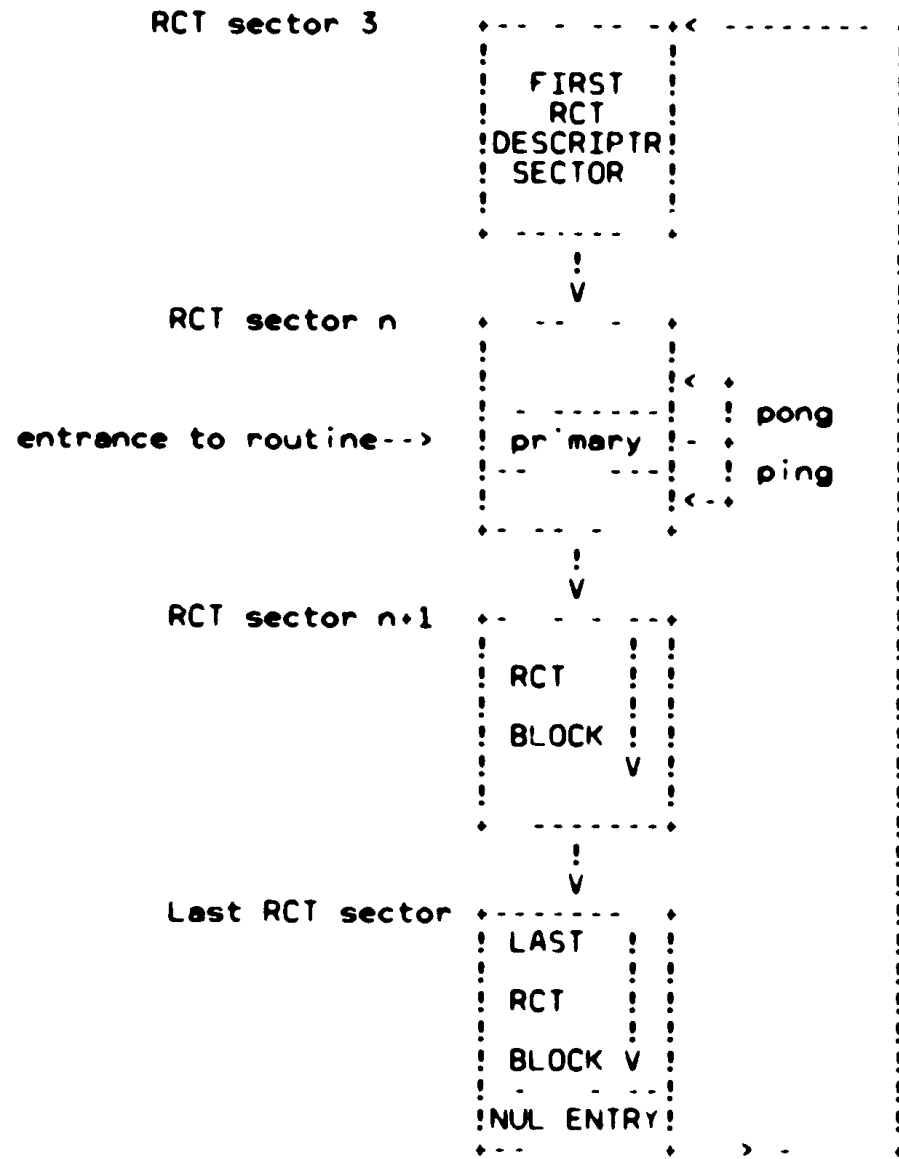
1. Within the primary descriptor RCT sector, outward from the primary descriptor searched (starting with the next highest RBN descriptor). Note that this degenerates to a linear search once the first or last descriptor is encountered.
2. Then a linear search, starting with the next highest RCT sector address, once the initial sector has been completely searched. Each new sector is searched in a linear fashion starting at the lowest RBN descriptor and scanning until the highest RBN descriptor in the sector is encountered.

If at any time during the linear search a null (not an empty) entry is encountered, the search resumes at the first entry in the third RCT sector (the first with descriptors). The search is terminated when it is certain that all the RCT entries have been searched.

: C 1533
:
: C 1534
:
: C 1535
:
: C 1536
:
: C 1537
:
: C 1538
:
: C 1539
:
: C 1540
:
: C 1541
:
: C 1542
:
: C 1543
:
: C 1544
:
: C 1545
:
: C 1546
:
: C 1547
:
: C 1548
:
: C 1549
:
: C 1550
:
: C 1551
:
: C 1552
:
: C 1553
:
: C 1554
:
: C 1555
:
: C 1556
:
: C 1557
:
: C 1558
:
: C 1559
:
: C 1560
:
: C 1561
:
: C 1562
:
: C 1563
:
: C 1564
:
: C 1565
:
: C 1566
:
: C 1567

: C 1568
: C 1569
: C 1570
: C 1571
: C 1572
: C 1573
: C 1574
: C 1575
: C 1576
: C 1577
: C 1578
: C 1579
: C 1580
: C 1581
: C 1582
: C 1583
: C 1584
: C 1585
: C 1586
: C 1587
: C 1588
: C 1589
: C 1590
: C 1591
: C 1592
: C 1593
: C 1594
: C 1595
: C 1596
: C 1597
: C 1598
: C 1599
: C 1600
: C 1601
: C 1602
: C 1603
: C 1604
: C 1605
: C 1606
: C 1607
: C 1608
: C 1609
: C 1610
: C 1611
: C 1612

ALGORITHM FIGURE -



: C 1613
: C 1614
: C 1615
: C 1616
: C 1617
: C 1618
: C 1619
: C 1620
: C 1621
: C 1622
: C 1623
: C 1624
: C 1625
: C 1626
: C 1627
: C 1628
: C 1629
: C 1630
: C 1631
: C 1632
: C 1633
: C 1634
: C 1635
: C 1636
: C 1637
: C 1638
: C 1639
: C 1640
: C 1641
: C 1642
: C 1643
: C 1644
: C 1645
: C 1646
: C 1647
: C 1648
: C 1649
: C 1650
: C 1651
: C 1652
: C 1653
: C 1654
: C 1655
: C 1656
: C 1657
: C 1658
: C 1659
: C 1660
: C 1661
: C 1662
: C 1663
: C 1664
: C 1665
: C 1666
: C 1667
: C 1668
: C 1669

APPENDIX G

CREATING ZRCMB0 DM CODE

G.1 FILES REQUIRED

A. RTEM (RT-11 emulator) including:

- a. DMACRO.SAV The DM macro assembler
- b. DMACRO.MAC The DM macro library
- c. DTX.SAV The XXDP, UPD2 emulator including:
- d. DTX DRIVERS including:
 - 1. IZDDAO.DTX
 - 2. IZDKAO.DTX
 - 3. IZDLAO.DTX
 - 4. IZDMAO.DTX
 - 5. IZDTAO.DTX
 - 6. IZDXAO.DTX
 - 7. IZDYAO.DTX
 - 8. IZMMAO.DTX
 - 9. IZMSAO.DTX
 - 10. IZMTAO.DTX

- e. STARTM.COM RTEM start up com file. See below assembling DM source files

B. DMCONV.EXE

Converts DMACRO output .SAV files into a Bliss ascii array. This ascii array is then built into a separate Bliss module and compiled. The object code from this module is then linked with other Bliss object modules resulting in a APT compatible, contiguous object file including both host and DM code.

G.1 BUILDING DM SOURCE CODE

DM source code can be built using any of the text editors available under any operating system. Guide lines for its creation can be found in the AZTEC.DOC manual.

G.2 ASSEMBLING DM SOURCE CODE

The DM macro assembler (DMACRO.SAV) is used to assemble DM code modules and is written to be run under RT-11 (or RTEM emulator). The following example demonstrated the RTEM startup com file used to create the released DM formatter .SAV file. This start up command file includes: allocating more virtual storage, bringing the DM source code into the RTEM system, assembling the DM source code, linking and moving the output .SAV file back to the user directory. The following example is performed on VMS operating system.

```

$ <VMS>
$ set def [neale.rtem]           !Set default directory to the RTEM sub d'rectory
$ mcr rtem                       !Run the RT 11 emulator
. RTE>/vs                        !Type /vs to prompt RTE>

RTEM 11 (VAX/VMS) V01.00

RT 11FB (S) V04.00

```

```

: C 1670
: C 1671
: C 1672
: C 1673
: C 1674
: C 1675
: C 1676
: C 1677
: C 1678
: C 1679
: C 1680
: C 1681
: C 1682
: C 1683
: C 1684
: C 1685
: C 1686
: C 1687
: C 1688
: C 1689
: C 1690
: C 1691
: C 1692
: C 1693
: C 1694
: C 1695
: C 1696
: C 1697
: C 1698
: C 1699
: C 1700
: C 1701
: C 1702
: C 1703
: C 1704
: C 1705
: C 1706
: C 1707
: C 1708
: C 1709
: C 1710
: C 1711
: C 1712
: C 1713
: C 1714
: C 1715
: C 1716
: C 1717
: C 1718
: C 1719
: C 1720
: C 1721
: C 1722
: C 1723
: C 1724
: C 1725
: C 1726

```

```

! The following is performed by the startm.com start-up command file. Before
! running the command file the virtual storage disk VS1: must have already
! been generated using the JOAT utility.
!Set term characteristics
!Enter JOAT (Jack of all trades) utility
!Allocate virtual device VS1: to RTEM system
!Exit JOAT
!Make room in VS1: for this assembly
!Enter File interchange program
!Bring in DM source code from user d'rectory
!Exit FIP
!Run the DM macro assembler
!Assemble DM source and save in VS1:
!Exit the DM assembler
!Run the RTEM linker
!Link the .obj file and save in VS1:
!Exit the linker
!Run FIP
!Output .LST file to user directory
!Output .SAV file to user directory
!Exit FIP
!Return to VMS

```

G.3 BUILDING THE BLISS ASCII ARRAY

The next step is to run the DMCONV program to build the Bliss ascii array from the DMACRO output .SAV file. To do this follow this example:

```

$ run DMCONV.EXE
INPUT FILE NAME (I.E. XXXXXX.YYY) AZFMTR.SAV
OUTPUT FILE NAME (I.E. XXXXXX.YYY) AZFMTR.VEC
BLOCK VECTOR NAME (I.E. XXXXXX.YYY) AZFMTR

```

Note:

In this example it is assumed that DMCONV.EXE and the .SAV file are both in the default directory.

G.4 BUILDING THE DM BLISS MODULE

The output from the DMCONV program (AZFMTR.VEC) is then included into a Bliss module (for the purpose of ZRCMB0 this module is ZRCMB6.B16) and compiled. To compile this module takes a VERY long time so it is suggested that it is submitted to the batch queue after hours.

The entire procedure to compile and link all Bliss source modules (including the DM bliss module) is demonstrated by the following command procedure (azcor.com):

```

: C 1727 ; @azcom !Execute command procedure
: C 1728 $ set verify
: C 1729 $ bliss/pdp11/list/library ZRCHB0.R16
: C 1730 $ bliss/pdp11/listing=ZRCHB.DOC ZRCHB1.B16
: C 1731 $ bliss/pdp11/list ZRCHB2.B16
: C 1732 $ bliss/pdp11/list ZRCHB3.B16
: C 1733 $ bliss/pdp11/list ZRCHB4.B16
: C 1734 $ bliss/pdp11/list ZRCHB5.B16
: C 1735 $ set process/name="Killer
: C 1736 $ bliss/pdp11/list ZRCHB6.B16
: C 1737 $ set process/name="Fisher_man
: C 1738 $ bliss/pdp11/list ZRCHB7.B16
: C 1739 $ MCR TKB
: C 1740 ZRCHB/NOHD/NOHM,ZRCHB/CR/-SP-ZRCHB2,ZRCHB3,ZRCHB4,ZRCHB5,ZRCHB6,ZRCHB7,neis1b/1b
: C 1741 /
: C 1742 PAR=DUMMY:2000:176000
: C 1743 STACK=0
: C 1744 //
: C 1745 $ !*****
: C 1746 $ !*
: C 1747 $ !* Now run the TKBBIN program to create a .bin file *
: C 1748 $ !*
: C 1749 $ !*****
: C 1750 $ set noverify
: C 1751 $

```

The TKBBIN utility is then run on the output from the TKB linker to produce an XXDP* executable .BIN file. The following is an example of the procedure:

```

: C 1756 $ RUN TKBBIN
: C 1757 Input filename ? ZRCHB
: C 1758 ;load Addr: 2000
: C 1759 ;xfer Addr: 1
: C 1760 ;Nr char : 74600
: C 1761 ;End of Job
: C 1762 $

```

G.5 GETTING .BIN FILES ONTO XXDP* MEDIA

The RT-11 emulator is used to move XXDP* .BIN files from a VMS user directory onto XXDP* media. The following is an example of this procedure.

```

: C 1769 $ set def [neale.rtem] !Set default directory to RTEM sub directory
: C 1770 $ mcr rtem !Run the RT 11 emulator
: C 1771 .RTE>/vs !Type /vs to prompt
: C 1772 !
: C 1773 ! The startm.com Command file will automatically
: C 1774 ! begin to execute when entering RTEM. Control
: C 1775 ! c the command procedure at this time and do the
: C 1776 ! following:
: C 1777 !
: C 1778 .R JOAT !Run JOAT utility
: C 1779 .DLO:/A !Allocate XXDP* device to pip .BIN file to
: C 1780 *+C !Exit JOAT
: C 1781 !
: C 1782 .R FIP !Run FIP utility
: C 1783 *VS1:=[NEALE.AZTEC]ZRCHB.BIN/F/I !Bring .BIN file from user directory into VS1:

```

E4

ZRCHB1
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
USER DOCUMENTATION

5-Apr-1984 13:43:03
22 Feb-1984 12:03:49

VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB1.B16;7 (27)

SEQ 0043
Page 43

:	C 1784	^C	!Exit FIP
:	C 1785		
:	C 1786	.R DTX	!Run the XXDP+ UPD2 emulator
:	C 1787	*DEL DLO:ZRCHB.BIN	!Delete from XXDP+ media old .BIN file
:	C 1788	*PIP DLO:ZRCHB.BIN=VS1:ZRCHB.BIN	!PIP new .BIN file to XXDP+ media
:	C 1789	*LOAD DLO:ZRCHB.BIN	!Load the file to verify the check-sum
:	C 1790	XFR: 000001 CORE:002000,76600	
:	C 1791	^C	!Exit DTX
:	C 1792		
:	C 1793	.BOO SY	!Boot back to VMS
:	C 1794	\$ <VMS>	!Back at VMS again

ZRCHB1
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
USER DOCUMENTATION

5-Apr-1984 13:43:03
22-Feb-1984 12:03:49

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB1.B16;7 (28)

SEQ 0044
Page 44

```

:           C 1795           APPENDIX H
:           C 1796
:           C 1797           CREATING ZRCHB0 HOST CODE
:           C 1798
:           C 1799           H.I FILES REQUIRED
:           C 1800           A. Text editor
:           C 1801           B. Bliss16 compiler
:           C 1802           C. TKB linker
:           C 1803           D. TKBBIN Utility This converts Bliss16 .EXE files into XXPD+
:           C 1804                       .BIN files.
:           C 1805
:           C 1806
:           C 1807           H.1 SOURCE FILE CREATING
:           C 1808
:           C 1809           Using any text editor, follow PDP-11 diagnostic design guidelines demonstrated
:           C 1810           in SUPPRGC.DOC and CHQUSD.SEQ.
:           C 1811
:           C 1812           H.2 COMPILING BLISS SOURCE MODULES
:           C 1813
:           C 1814           Follow steps starting at G.4 to compile Bliss source modules and
:           C 1815           to get the .BIN file onto XXDP+ media.
:           1816           )%
:           1817           ELUDDM

```

```

.TITLE ZRCHB1 ZRCHB0 RC25 DISK FORMATTER
.IDENT /REV B /

```

```

:           PSECT SUMMARY
:
: Psect Name           Words           Attributes

```

```

:           COMMAND QUALIFIERS

```

```

: BLISS /PDP11/LISTING=ZRCHB.DOC ZRCHB1.B16

```

```

: Size:           0 code + 0 data words
: Run Time:           00:08.6
: Elapsed Time:      00:49.4
: Memory Used:      6 pages
: Compilation Complete

```

```

: 0001 MODULE ZRCHB2 (#TITLE 'ZRCHB0 RC25 DISK FORMATTER
: 0002 IDENT = 'REV B PATCH 00',
: 0003 ADDRESSING MODE (RELATIVE) ,
: 0004 ENVIRONMENT (NOEIS)
: 0005 ) =
: 0006 BEGIN
: 0007 #sbttl 'PROGRAM HEADER'
: 0008 !
: 0009 ! Pretty Declarations
: 0010 !
: 0011 ! <BLF/LOWERCASE_KEY>
: 0012 !
: 0013 !
: 0014 library 'ZRCHB0'; !Define RC25 Formatter Library
: 0015
: 0016 require 'BLSMAC.REQ'; !Define Bliss Macro Require file
: 1505
: 1506 !*
: 1507 ! The psect named "code or $code$" is redefined here
: 1508 ! to be called "aa$code". This is done to force the TKB
: 1509 ! linker to place the programs header information starting
: 1510 ! at absolute address 2000.
: 1511 !-
: 1512 psect
: 1513 code = aa$code;
: 1514
: 1515 literal
: 1516 DS#NBR_OF_TESTS = 1; !Indicates number f test in Diag
: 1517
: 1518 !*
: 1519 ! The structure of a diagnostic program may contain any or all of the
: 1520 ! ten optional sections. But five of the optional sections require a
: 1521 ! pointer that is derived by and for the supervisor, and is located in
: 1522 ! the header block. Therefore, in relation to the effective use of
: 1523 ! these five pointers, the optional sections call must be coded to re-
: 1524 ! flect usage (i.e., any,all,or none).
: 1525 !
: 1526 ! The following coding possibilities exist:
: 1527 !
: 1528 ! POINTER (BGNRPT,BGNSW,BGNSFT,BGNAU,BGNDU,ERRTBL,BGNSETUP)
: 1529 !
: 1530 ! (or any subset of the args)
: 1531 !
: 1532 ! POINTER (ALL) ; All provides pointers for all five
: 1533 ! ; sections
: 1534 ! POINTER (NONE) ; None indicates to supervisor that no
: 1535 ! ; pointers are required.
: 1536 ! ; this is the default
: 1537 !
: 1538 ! No pointers are optional using bliss. Make sure the following
: 1539 ! sections of code are in place (in the correct skels),even if
: 1540 ! the sections are blank.
: 1541 !
: 1542 ! ARGUMENT FUNCTION
: 1543 ! - - - -
: 1544 ! RPT REPORT CODE
: 1545 ! SW SOFTWARE TABLE

```

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
PROGRAM HEADER

5 Apr-1984 13:19:09
5-Apr-1984 13:16:08

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (1)

SEQ 0046
Page 2

```

:      1546 :      SFT          SOFTWARE TABLE QUESTIONS
:      1547 :      AU           ADD CODE
:      1548 :      DU           DROP CODE
:      1549 :      TBL          ERROR TABLE
:      1550 :      SETUP        ASSEMBLED P-TABLES
:      1551 :      -
:      1552 POINTER (ALL);
:      1553 !*
:      1554 ! The program header section contains general information which des-
:      1555 ! cribes the major characteristics of the diagnostic program. This in-
:      1556 ! cludes, the program name, and revision and patch-order levels. The
:      1557 ! header also provides space for an event flag register, and for the
:      1558 ! storage of pointers, through which the supervisor may find access to
:      1559 ! other key sections of the program(e.g., dispatch table, initialize and
:      1560 ! clean-up code, etc.). An argument on the header gives the device type
:      1561 ! if it is an XXDP+ bootable device. This enables the supervisor to pro-
:      1562 ! vide load medium protection when necessary.
:      1563 !-
:      1564 HEADER (ASCII'CZRCH ', ASCII'B', ASCII'O', 1200, 0, PRI00);

```


ZRCH82 ZRCH80 RC25 DISK FORMATTER
REV B PATCH 00 DISPATCH TABLE

5-Apr 1984 13:19:09
5 Apr 1984 13:16:08

VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCH82.B16;5 (2)

```

:      1565 #sbttl 'DISPATCH TABLE'
:      1566 !*
:      1567 ! The dispatch table section contains address pointers to the various
:      1568 ! tests contained within the diagnostic program. This section requires
:      1569 ! the coding of only the dispatch macro.
:      1570 !-
:      1571 DISPATCH (DS#NBR_OF TESTS);
:      1572 ERR_TBL;                                !Define Supervisor Error table storage

```

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
DEFAULT HARDWARE P TABLE

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (3)

SEQ 0048
Page 4

```

:      1573 #sbttl 'DEFAULT HARDWARE P-TABLE'
:      1574 !,
:      1575 ! The default hardware P-Table contains default values of
:      1576 ! the test-device parameters. The structure of this table
:      1577 ! is identical to the structure of the hardware P-Tables,
:      1578 ! and is used as a "template" for building the P-Tables.
:      1579 !-
:      1580 BGNHW (DFPTBL);
:      1581
:      1582 global
:      1583     HW_IP_ADRS : word initial (#'172150'),      !Define RC25 Controller IP reg
:      1584     HW_VECTOR : word initial (#'154'),      !Define RC25 interrupt vector addrs
:      1585     HW_BR_LEVEL : word initial (5),           !Define RC25 bus request level
:      1586     HW_UNIT_NO : word initial (0);          !Define RC25 unit no. to format
:      1587
:      1588 ENDHW;

```

ZRCH82
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
SOFTWARE P-TABLE

5-Apr-1984 13:19:09
5 Apr-1984 13:16:08

VAX-11 B119-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCH82.B16;5 (4)

```

:      1589 #sbttl 'SOFTWARE P-TABLE
:      1590 !*
:      1591 ! The software table contains various data used by the
:      1592 ! program as operational parameters. These parameters are
:      1593 ! set up at assembly time and may be varied by the operator
:      1594 ! at run time.
:      1595 !-
:      1596 BGNSW (SFPTBL);
:      1597
:      1598 global
:      1599     SW_UNATT : word initial (1);           !Format in unattended refmt mode
:      1600
:      1601 ENDSW;

```

ZRCH82
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
PROTECTION TABLE

5 Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0050
Page 6
VAX-11 Bliss 16 v3 555
SPIDER#USERS:[NEALE.AZTEC]ZRCH82.B16;5 (5)

```
: 1602 #sbttl 'PROTECTION TABLE
: 1603 !'
: 1604 ! This table is used by the runtime
: 1605 ! services to protect the load media.
: 1606 !
: 1607 ! 1st arg = Offset into P Table for csr address
: 1608 ! 2nd arg = Offset into P_Table for massbus address
: 1609 ! 3rd arg = Offset into P Table for drive number
: 1610 !
: 1611 BGNPROT (0, 1, 6);
: 1612 ENDPROT;
```

```

:      1613 #sbttl 'MODULE DECLARATIONS'
:      1614 !*
:      1615 ! Within BLSMAC.REQ the psect names, plit global and own,
:      1616 ! are redefined to be asicode. This is done to force the
:      1617 ! tkb linker to link the header information starting at
:      1618 ! at absolute address 2000. Redefine these psect names
:      1619 ! back to their original names for house keeping purposes.
:      1620 !
:      1621 ! Also change the attributes for the psect "global" so that
:      1622 ! global data will not be linked starting at absolute address
:      1623 ! 2000.
:      1624 !-
:      1625 psect
:      1626     plit = $plit$( global),
:      1627     global = $glob$(nowrite, noexecute, global, concatenate),
:      1628     own = $own$;
:      1629
:      1630 !*
:      1631 ! Structure declarations used within this
:      1632 ! module.
:      1633 !-
:      1634
:      1635 structure
:      1636
:      1637     !*
:      1638     ! RC25 register accessing structure. This
:      1639     ! structure allows RC25 register accessing
:      1640     ! to be transportable between the PDP-11 and
:      1641     ! VAX Diagnostic Supervisors.
:      1642     !
:      1643     ! This also defines an access algorithm for
:      1644     ! VAX to allow field reference to MBA address
:      1645     ! space without generating machine checks.
:      1646     !-
:      1647
:      1648     RC25 [0, P, S, E] =
:      1649         begin
:      1650
:      1651             local
:      1652                 RC$S_REG;
:      1653
:      1654                 RC$S_REG = .(RC25 + $upval*0)<0, $bpval, 0>;
:      1655                 RC$S_REG
:      1656             end
:      1657         <P, S, E>;
:      1658

```

```

: 1659 #sbt1 'GLOBAL DATA SECTION'
: 1660 !.
: 1661 ! The global data section contains data that are used
: 1662 ! in more than one test or module.
: 1663 !.
: 1664
: 1665 global
: 1666 !
: 1667 ! Communication area Declarations
: 1668 !
: 1669 COM_AREA : blockvector [REC_ALLOCATE + SND_ALLOCATE + HDR_SIZE, 2, word],
: 1670 HEAD_AREA : ref block [4, word] field (HDR_FIELD),
: 1671 RECEIVE_RING : ref blockvector [REC_ALLOCATE, 2, word] field (DSC_FIELD),
: 1672 SEND_RING : ref blockvector [SND_ALLOCATE, 2, word] field (DSC_FIELD),
: 1673 REC_ENVELOPE : blockvector [REC_ALLOCATE, RB_SIZE + 2, word] field (ENV_FIELD),
: 1674 SND_ENVELOPE : blockvector [SND_ALLOCATE, SB_SIZE + 2, word] field (ENV_FIELD),
: 1675 FCT_BUF : block [256, word],
: 1676 REC_BUF : block [RECB_SIZE, word] field (RECB_FIELD),
: 1677 SND_BUF : vector [SNDB_SIZE, word],
: 1678 OUT$STD_BUF : blockvector [REC_ALLOCATE, 2, word] field (OUT$FIELD),
: 1679 RET_EN$AD : ref block [RB_SIZE + 2, word] field (ENV_FIELD);
: 1680
: 1681 global bind
: 1682 !
: 1683 ! Diagnostic supervisor printing ascii format strings.
: 1684 !
: 1685 FMT1 = uplit (#asciz'#N#T'), !Print one ascii string pointer
: 1686 FMT2 = uplit (#asciz'#N#N#A#INITIALIZING RC25 CONTROLLER ADDRESS:#S#06#A(0)#N'),
: 1687 FMT3 = uplit (#asciz'#N#/#LOGICAL UNIT #D2#A(D)#S#APHYSICAL UNIT #D2#A(D)#S#AFORMAT ABORTED'),
: 1688 FMT4 = uplit (#asciz'#N#AMICRO CODES MOD 16 VERSION NUMBER =#S#02#A(0)#N'),
: 1689 FMT5 = uplit (#asciz'#N#ABRINGING ONLINE: LOGICAL UNIT #D2#A(D)#S#APHYSICAL UNIT #D2#A(D)#N'),
: 1690 FMT6 = uplit (#asciz'#N#A$FTLERR- NON-EXISTENT RC25 CONTROLLER ADDRESS #06#A(0)#N'),
: 1691 FMT7 = uplit (#asciz'#A#AFORMATTING PHYSICAL UNIT #D2#A(D)#N'),
: 1692 PID_FMT = uplit (#asciz'#N#AFORMATTER PROGRESSING: PID HI=#06#A(0)#S#S#APID LO=#06#A(0)'),
: 1693 CRLF = uplit (#asciz'#N'),
: 1694 XCRLF = uplit (#asciz'#N#N#N#N#N#N#N#N#N#N'),
: 1695 !
: 1696 ! Ring base address declarat'on
: 1697 !
: 1698 RINGBASE = COM_AREA [REC_BASE],
: 1699 !
: 1700 !
: 1701 !
: 1702 MSGADR = REC_BUF [MSG_TXT];
: 1703
: 1704 global
: 1705 !
: 1706 ! Miscellaneous data declarations
: 1707 !
: 1708 DATETXT : vector [12, byte], !Operator data input storage
: 1709 FLG_WRD : bitvector [16], !Global flag word
: 1710 NEX_FLAG : word, !Non existent RC25 register flag
: 1711 OVSA : word, !Overlay section starting adrs
: 1712 NXT_CRN : byte, !Stores next cmd ref number
: 1713 RET_STATUS : word initial ('000000'), !Saves various return status codes
: 1714 LUN : word, !Stores logical unit number being formatted
: 1715 PID_SAVE : vector [2, word], !Saves proces indicator word

```

```

:      1716      NSD_SLOT : word,      !Next send Descriptor slot
:      1717      NRD_SLOT : word,      !Next receive Descriptor slot
:      1718      !
:      1719      ! Hardware P_Table storage declarations
:      1720      !
:      1721      RC25_ADDR : ref RC25 field (ISD_FIELD),      !Controller register access structure
:      1722      VEC_ADDR : word,      !Interrupt vector address storage
:      1723      BR_LEVEL : word,      !Bus request level storage
:      1724      UNIT_NO : word,      !Unit number to format storage
:      1725      PTBL_PTR : ref vector [4, word],      !Stores P Table base address
:      1726      !
:      1727      ! Dup Protocol data structures
:      1728      !
:      1729      !
:      1730      !Reserved field mask structure declaration
:      1731      !
:      1732      RSVD_STRUCT : vector [4, word] preset (      !Reserved SA reg fields definitions
:      1733      [0] = #x'07FF',      !V2.0 Step one rsvd field
:      1734      [1] = #x'0000',      !Step two rsvd field
:      1735      [2] = #x'0700',      !Step three rsvd field
:      1736      [3] = #x'07FF'),      !Step four rsvd & ucode field
:      1737      !
:      1738      !
:      1739      ! Init Sequence Data_Structure declaration
:      1740      !
:      1741      !
:      1742      ISD_STRUCT : blockvector [4, 2, word] field (ISD_FIELD) preset (
:      1743      !
:      1744      ! Step one read SA register field declaration
:      1745      !
:      1746      [BLKO, WRDO, ERR_BIT] = 0,      !Error bit
:      1747      [BLKO, WRDO, STP_FIELD] = #b'0001',      !All step bit fields
:      1748      [BLKO, WRDO, S1R_NV] = 1,      !V2.0 Mask out host inter vec settable adrs
:      1749      [BLKO, WRDO, S1R_QB] = 1,      !Mask out 22 bit addressing
:      1750      [BLKO, WRDO, S1R_DI] = 1,      !Mask enhanced diag implementation
:      1751      [BLKO, WRDO, S1R_OD] = 1,      !Mask out port allows odd addressing
:      1752      [BLKO, WRDO, S1R_MP] = 1,      !Mask out Port supported adr
:      1753      [BLKO, WRDO, S1R_RSVD] = #o'77',      !V2.0 Reserved field
:      1754      !
:      1755      ! Step one write SA register field declaration
:      1756      !
:      1757      [BLKO, WRD1, ERR_BIT] = 1,      !Error bit
:      1758      [BLKO, WRD1, S1W_WR] = 0,      !Diag wrap around
:      1759      [BLKO, WRD1, S1W_CRING] = SND_SIZ,      !Number of Send-ring slots 'pwr of 2
:      1760      [BLKO, WRD1, S1W_RRING] = REC_SIZ,      !Number of Receive ring slots pwr of 2
:      1761      [BLKO, WRD1, S1W_IE] = 0,      !Init Sequence interrupt request
:      1762      [BLKO, WRD1, S1W_VADR] = #o 33,      !Interrupt vector address
:      1763      !
:      1764      ! Step two read SA register field declaration
:      1765      !
:      1766      [BLK1, WRDO, ERR_BIT] = 0,      !Error bit
:      1767      [BLK1, WRDO, STP_FIELD] = #b 0010',      !All step bit fields
:      1768      [BLK1, WRDO, S2R_PTYP] = 0,      !Port type number
:      1769      [BLK1, WRDO, S2R_BIT7] = 1,      !Echoed IE bit from step one write
:      1770      [BLK1, WRDO, S2R_WR] = 0,      !Echoed bit 14 from step one write
:      1771      [BLK1, WRDO, S2R_CRING] = SND_SIZ,      !Echoed bits 3 5 from step one write
:      1772      [BLK1, WRDO, S2R_RRING] = REC_SIZ,      !Echoed bits 0 2 from step one write

```

```

: 1773
: 1774 : Step two write SA register field declaration
: 1775
: 1776 [BLK1, WRD1, S2W_LRBASE]= RINGBASE, !Ring base lower address
: 1777
: 1778 : NOTE:
: 1779 : The adapter purge interrupt is loaded within
: 1780 : the bgninit code due to the inability to field
: 1781 : select bits <1, 15, 0> from the ringbase adrs.
: 1782 : [BLK1, WRD1, S2W_PI]= 0, !Adapter purge interrupt request
: 1783
: 1784 : Step three read SA register field declaration
: 1785
: 1786 [BLK2, WRD0, ERR_BIT] = 0, !Error bit
: 1787 [BLK2, WRD0, STP_FIELD] = #b'0100', !All step bit fields
: 1788 [BLK2, WRD0, S3R_RSVD]= #o'7', !Reserved
: 1789 [BLK2, WRD0, S3R_IE]= 0, !Echoed IE bit from step one write
: 1790 [BLK2, WRD0, S3R_VADR]= #o'33', !Echoed VADR from step one write
: 1791
: 1792 : Step three write SA register field declaration
: 1793
: 1794 [BLK2, WRD1, S3W_PP]= 0, !Purge & Poll test request
: 1795 [BLK2, WRD1, S3W_HRBASE]= 0, !Ring base high address
: 1796
: 1797 : Step four read SA register field declaration
: 1798
: 1799 [BLK3, WRD0, ERR_BIT] = 0, !Error bit
: 1800 [BLK3, WRD0, STP_FIELD] = #b'1000', !All step bit fields
: 1801 [BLK3, WRD0, S4R_RSVD]= #o'7', !Reserved
: 1802 [BLK3, WRD0, S4R_MOD] = #x'f', !Mask it out
: 1803 [BLK3, WRD0, S4R_VER] = #x'f', !Mask it out
: 1804
: 1805 : Step four write SA register field declaration
: 1806
: 1807 [BLK3, WRD1, S4W_RSVD]= #o'377', !Reserved
: 1808 [BLK3, WRD1, S4W_BURST]= 0, !Max number longwords per NPR xfer
: 1809 [BLK3, WRD1, S4W_LF]= 0, !Last fail request
: 1810 [BLK3, WRD1, S4W_GO]= 0); !Go bit
: 1811

```



```

:      1812 %sbttl 'GLOBAL TEXT SECTION'
:      1813 !*
:      1814 ! The global text section contains format statements,
:      1815 ! messages, and ASCII information that are used in
:      1816 ! all modules.
:      1817 !-
:
:      1818
:      1819 global bind
:      1820 !
:      1821 ! Self-detected fatal port/controller errors
:      1822 !
:      1823 PFE_STRUCT = uplit (
:      1824     uplit (%asciz' %NSA$FTLERR- UNRECOGNIZABLE ERROR CODE'),
:      1825     uplit (%asciz' %NSA$FTLERR- ENVELOPE/PACKET READ (PARITY OR TIMEOUT)'),
:      1826     uplit (%asciz' %NSA$FTLERR- ENVELOPE/PACKET WRITE (PARITY OR TIMEOUT)'),
:      1827     uplit (%asciz' %NSA$FTLERR- CONTROLLER ROM AND RAM PARITY'),
:      1828     uplit (%asciz' %NSA$FTLERR- CONTROLLER RAM PARITY'),
:      1829     uplit (%asciz' %NSA$FTLERR- CONTROLLER ROM PARITY'),
:      1830     uplit (%asciz' %NSA$FTLERR- RING READ (PARITY OR TIMEOUT)'),
:      1831     uplit (%asciz' %NSA$FTLERR- RING WRITE (PARITY OR TIMEOUT)'),
:      1832     uplit (%asciz' %NSA$FTLERR- INTERRUPT MASTER'),
:      1833     uplit (%asciz' %NSA$FTLERR- HOST ACCESS TIMEOUT'),
:      1834     uplit (%asciz' %NSA$FTLERR- CREDIT LIMIT EXCEEDED'),
:      1835     uplit (%asciz' %NSA$FTLERR- BUS MASTER ERROR'),
:      1836     uplit (%asciz' %NSA$FTLERR- DIAGNOSTIC CONTROLLER FATAL ERROR'),
:      1837     uplit (%asciz' %NSA$FTLERR- INSTRUCTION LOOP TIMEOUT'),
:      1838     uplit (%asciz' %NSA$FTLERR- INVALID CONNECTION IDENTIFIER'),
:      1839     uplit (%asciz' %NSA$FTLERR- INTERRUPT WRITE'),
:      1840     uplit (%asciz' %NSA$FTLERR- MAINTENANCE READ/WRITE INVALID REGION IDENTIFIER'),
:      1841     uplit (%asciz' %NSA$FTLERR- MAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER),
:      1842     uplit (%asciz' %NSA$FTLERR- CONTROLLER RAM ERROR (NON-PARITY)'),
:      1843     uplit (%asciz' %NSA$FTLERR- INIT SEQUENCE ERROR'),
:      1844     uplit (%asciz' %NSA$FTLERR- HIGH LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
:      1845     uplit (%asciz' %NSA$FTLERR- PURGE/POLL HARDWARE FAILURE '),
:      1846     uplit (%asciz' %NSA$FTLERR- MAPPING REGISTER READ ERROR (PARITY OR TIMEOUT) )
:      1847         ) : vector [23],
:      1848 ! <BLF/PAGE >

```

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0056
Page 12
VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB2.B16;5(9)

```

:      1849      :
:      1850      : Error message structure
:      1851      :
:      1852      : MSG_STRUCT = uplit (
:      1853      : uplit ('%asciz'%N%$FTLERR- RESPONSE STATUS ERROR:%S'),
:      1854      : uplit ('%asciz'%N%$FTLERR- HOST/CONTROLLER OUT OF SEQ'),
:      1855      : uplit ('%asciz'%N%$FTLERR- REMOTE PROG NOT RUNNING'),
:      1856      : uplit ('%asciz'%N%$FTLERR- UNKNOWN RETURN STATUS CODE'),
:      1857      : uplit ('%asciz'%N%$FTLERR- COM AREA INIT ERROR'),
:      1858      : uplit ('%asciz'%N%$FTLERR- PORT/HOST SYNC ERROR'),
:      1859      : uplit ('%asciz'%N%$FTLERR- MESSAGE LENGTH ERROR'),
:      1860      : uplit ('%asciz'%N%$FTLERR- UNKNOWN ENCODE RECEIVED'),
:      1861      : uplit ('%asciz'%N%$FTLERR- ADAPTOR PURGE ERROR'),
:      1862      : uplit ('%asciz'%N%$FTLERR- UNKNOWN INTERRUPT'),
:      1863      : uplit ('%asciz'%N%$FTLERR- INIT SEQ STEP TIMED OUT'),
:      1864      : uplit ('%asciz'%N%$FTLERR- INIT SEQ COMPARE ERROR'),
:      1865      : uplit ('%asciz'%N%$FTLERR- UNEXPECTED ATTENTION END MESSAGE RECEIVED'),
:      1866      : uplit ('%asciz'%N%$FTLERR- UNEXPECTED COMMAND OPCODE IN END MESSAGE RECEIVED'),
:      1867      : uplit ('%asciz'%N%$FTLERR- UNEXPECTED SERIOUS EXCEPTION END MESSAGE RECEIVED'),
:      1868      : uplit ('%asciz'%N%$FTLERR- INVALID COMMAND END MESSAGE RECEIVED'),
:      1869      : uplit ('%asciz'%N%$FTLERR- UNKNOWN MESSAGE TYPE RECEIVED'),
:      1870      : uplit ('%asciz'%N%$FTLERR- OUTSTANDING COMMAND BUFFER FULL'),
:      1871      : uplit ('%asciz'%N%$FTLERR- OUT STANDING COMMAND BUFFER OUT OF SYNC ERROR'),
:      1872      : uplit ('%asciz'%N%$FTLERR- UNKNOWN MESSAGE NUMBER RECEIVED'),
:      1873      : uplit ('%asciz'%N%$FTLERR- FILE READ ERROR'),
:      1874      : uplit ('%asciz'%N%$FTLERR- PORT/CONTROLLER TIMEOUT ERROR'),
:      1875      : uplit ('%asciz'%N%$FTLERR- ILLEGAL FCT FILE LENGTH')) : vector [23],
:      1876      : !<blf/page>

```

```

:      1877      :
:      1878      : Self detected fatal port/controller errors
:      1879      :
:      1880      : RC_STRUCTURE = uplit (
:      1881      : uplit ('%asciz'%N%$FTLERR- VAX READ/WRITE ERROR ON INTERRUPT'),
:      1882      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.BFIL'),
:      1883      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.BMTY'),
:      1884      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.ALOC'),
:      1885      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT SERVO ENTRY (PIP SET)'),
:      1886      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT SERVO ENTRY (ERR SET)'),
:      1887      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.SEND'),
:      1888      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.RECV'),
:      1889      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.ATTN'),
:      1890      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.ONLN'),
:      1891      : uplit ('%asciz'%N%$FTLERR- ILLEGAL D REQUEST (U.QDRQ)'),
:      1892      : uplit ('%asciz'%N%$FTLERR- FENCE-POST ERROR AT PROTAB'),
:      1893      : uplit ('%asciz'%N%$FTLERR- BAD PACKET DEQUEUED AT U.DONE'),
:      1894      : uplit ('%asciz'%N%$FTLERR- UNEXPLAINED D-PROC SUSPENSION (U..TDS)'),
:      1895      : uplit ('%asciz'%N%$FTLERR- DUP PACKET D-Q FAILED (XFC 34/35)'),
:      1896      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.HTST'),
:      1897      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.SEK0'),
:      1898      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT U.CKSV'),
:      1899      : uplit ('%asciz'%N%$FTLERR- D.OPCD FOUND ILLEGAL OPCODE'),
:      1900      : uplit ('%asciz'%N%$FTLERR- D.CSF FOUND ILLEGAL OPCODE'),
:      1901      : uplit ('%asciz'%N%$FTLERR- UNKNOWN BAD DRIVE STATUS AT D.DSTS'),
:      1902      : uplit ('%asciz'%N%$FTLERR- ILLEGAL XFC EXECUTED BY DM'),
:      1903      : uplit ('%asciz'%N%$FTLERR- D PICKED UP A ZERO SCB.DB'),
:      1904      : uplit ('%asciz'%N%$FTLERR- INCONSISTENCY AT D IDLE LOOP'),
:      1905      : uplit ('%asciz'%N%$FTLERR- DM WORD COUNT ERROR ON HOST DMA/SEND/RECV'),
:      1906      : uplit ('%asciz'%N%$FTLERR- UNKNOWN DISPLAY FAULT CODE AT D.DFLT'),
:      1907      : uplit ('%asciz'%N%$FTLERR- DRIVE NOT FAULTING IN P.OFLN STATE'),
:      1908      : uplit ('%asciz'%N%$FTLERR- U POWER UP DIAGNOSTICS FAILED'),
:      1909      : uplit ('%asciz'%N%$FTLERR- D POWER UP DIAGNOSTICS FAILED'),
:      1910      : uplit ('%asciz'%N%$FTLERR- ADAPTER CARD FAILURE'),
:      1911      : uplit ('%asciz'%N%$FTLERR- EC.TMR TIMED OUT'),
:      1912      : uplit ('%asciz'%N%$FTLERR- U.SEND/U.RECV RING READ INCONSISTENCY'),
:      1913      : uplit ('%asciz'%N%$FTLERR- UNKNOWN WAITRV REASON AT D.RVCT'),
:      1914      : uplit ('%asciz'%N%$FTLERR- D.ARCS DID NOT FIND CLOSEST UNDONE ZONE'),
:      1915      : uplit ('%asciz'%N%$FTLERR- U.SEEK FOUND SEEK TO ILLEGAL TRACK'),
:      1916      : uplit ('%asciz'%N%$FTLERR- U.HTST INIT DIAG DMA WRITE FAILED'),
:      1917      : uplit ('%asciz'%N%$FTLERR- U.HTST INIT DIAG DMA COMPARE FAILED'),
:      1918      : uplit ('%asciz'%N%$FTLERR- U.SYDR FOUND SS.DER SET AND SS.SPN NOT SET'),
:      1919      : uplit ('%asciz'%N%$FTLERR- MASTER DRIVES ACLO ASSERTED )
:      1920      : ) : vector [39],
:      1921      : !<blf/page>

```

G5

ZRCH82
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0058
Page 14
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:{NEALE,AZTEC}ZRCH82.B16;5 (11)

```
: 1922      !  
: 1923      ! Dup return status codes  
: 1924      !  
: 1925      SDUP_STRUCT = uplit (  
: 1926      uplit (#asciz'#A SUCCESSFUL#N'),  
: 1927      uplit (#asciz'#AINVALID COMMAND#N'),  
: 1928      uplit (#asciz'#ANO REGION AVAILABLE#N'),  
: 1929      uplit (#asciz'#ANO REGION SUITABLE#N'),  
: 1930      uplit (#asciz'#APROGRAM NOT KNOWN#N'),  
: 1931      uplit (#asciz'#ALOAD FAILURE#N'),  
: 1932      uplit (#asciz'#ASTANDALONE#N')  
: 1933      ) : vector [7],  
: 1934      !<blf/page>
```

ZRCMB2
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION5-Apr-1984 13:19:09
5-Apr-1984 13:16:08VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCMB2.B16;5 (12)

SEQ 0059

Page 15

```
:
: 1935      :
: 1936      : MSCP return status codes
: 1937      :
: 1938      SMSCP_STRUCT = uplit (
: 1939 uplit (%esciz'ASUCCESS'N'),
: 1940 uplit (%esciz'AINVALID COMMAND'N'),
: 1941 uplit (%esciz'ACOMMAND ABORTED'N'),
: 1942 uplit (%esciz'AAUNIT-OFFLINE'N'),
: 1943 uplit (%esciz'AAUNIT-AVAILABLE'N'),
: 1944 uplit (%esciz'AMEDIA FORMAT ERROR'N'),
: 1945 uplit (%esciz'AAWRITE PROTECTED'N'),
: 1946 uplit (%esciz'ACOMPARE ERROR'N'),
: 1947 uplit (%esciz'ADATA ERROR'N'),
: 1948 uplit (%esciz'AAHOST BUFFER ACCESS ERROR'N'),
: 1949 uplit (%esciz'ACONTROLLER ERROR'N'),
: 1950 uplit (%esciz'ADRIVE ERROR'N'),
: 1951 uplit (%esciz'AMESSAGE FROM AN INTERNAL DIAGNOSTIC'N')
: 1952      ) : vector [13],
: 1953      !<BLF/PAGE>
```

```

: 1954      :
: 1955      : Init code error and informational messages
: 1956      :
: 1957      : DATMSG = uplit (%asciz'ENTER DATE <MM-DD-YYYY> '),
: 1958      : NO_ADD_UNITS = uplit (%asciz'%N%AND ADDITIONAL UNITS TO FORMAT - ABORTING'),
: 1959      : PWR_MSG = uplit (%asciz'%N%A$FTLERR- INIT CODE RE-ENTERED DUE TO PWR FAIL'),
: 1960      : ABO_MSG = uplit (%asciz'%N%A$FTLERR- ABORTING HOST AND REMOTE PROGRAMS'),
: 1961      : TO_MANY_UNITS = uplit (%asciz'%N%A$FTLERR- ILLEGAL NUMBER OF UNITS SELECTED'),
: 1962      : GOOD_NUM_UNITS = uplit (%asciz'%N%A$FTLERR- LIMIT OF SIXTEEN UNITS PER FORMATING SESSION'),
: 1963      : BOOT_FAILURE = uplit (%asciz'%N%A$FTLERR- RC25 CONTROLLER INITIALIZATION ERROR'),
: 1964      : PROTO_VIOLATION = uplit (%asciz'%N%A$FTLERR- PROTOCOL VIOLATION ERROR'),
: 1965      : PORT_INIT_ERR = uplit (%asciz'%N%A$FTLERR- COMMUNICATION AREA INIT ERROR'),
: 1966      : ACTIVE_DUP_SERVER = uplit (%asciz'%N%A$FTLERR- DUP SERVER ACTIVE AFTER INITIALIZING'),
: 1967      : INACTIVE_DUP_SERVER = uplit (%asciz'%N%A$FTLERR- DUP SERVER INACTIVE AFTER EX_SUP_PROG COMMAND'),
: 1968      :
: 1969      : Local load media DM module file name.ext
: 1970      :
: 1971      : !DM_FN$EXT = UPLIT (%ASCIZ'AZFMTR.SAV'),
: 1972      : FCT_REQ_MSG = uplit (%asciz'ENTER FCT FILE NAME TO RESTORE (FILENAME.EXT): '),
: 1973      :
: 1974      : Hardware parameter coding questions
: 1975      :
: 1976      : HW_Q1_IP = uplit (%asciz'RC25 IP REGISTER ADDRESS'),
: 1977      : HW_Q2_VECTOR = uplit (%asciz'RC25 INTERRUPT VECTOR ADDRESS'),
: 1978      : HW_Q3_BR = uplit (%asciz'RC25 BUS REQUEST LEVEL'),
: 1979      : HW_Q4_UNIT = uplit (%asciz'UNIT NUMBER TO BRING ONLINE'),
: 1980      :
: 1981      : Software parameter coding questions
: 1982      :
: 1983      : SW_Q1_UNATT = uplit (%asciz'FORMAT IN UNATTENDED REFORMAT MODE'),
: 1984      : SW_Q2_NOTICE = uplit (%asciz'%N%A***** NOTICE *****'),
: 1985      : SW_Q3_OPER = uplit (%asciz'%N%A OPERATOR MUST BE PRESENT IN ATTENDED MODE '),
: 1986      : SW_Q4_UNATT = uplit (%asciz'%N%A RUNNING IN UN-ATTENDED REFORMAT MODE');
: 1987      :
: 1988      : end
: 1989      :
: 1990      : eludom

```

```

.TITLE ZRCHB2 ZRCHB0 RC25 DISK FORMATTER
.IDENT /REV B /

```

Address	Offset	Value	Label	Field	Value
000000			.PSECT	AA\$CODE.	RO
000000	103	132	L\$NAME::	.ASCII	/CZR/
000003	103	110		.ASCII	/CH /
000006	000			.BYTE	0
000007	000			.BYTE	0
000010			L\$REV::		
000010	102			.ASCII	/B/
000011	060			.ASCII	/O/
000012	000000G		L\$UNIT::	.WORD	T\$PTHV
000014	002260		L\$TIML::	.WORD	2260
000016	000000G		L\$HPCP::	.WORD	L\$HARD
000020	000000G		L\$SPCP::	.WORD	L\$SOFT
000022	000140'		L\$HPTP::	.WORD	L\$HW
000024	000154'		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	000126'	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	000160'	L\$PRT::	.WORD	L\$PROT	
000114	000000	L\$TEST::	.WORD	0	
000116	000000	L\$DLY::	.WORD	0	
000120	000000	L\$HIME::	.WORD	0	
000122	000001	D\$PCNT::	.WORD	1	
000124	000000G	L\$DISPATCH::			
			.WORD	T1	
000126		ERRTYP::	.BLKW	1	
000130		ERRNBR::	.BLKW	1	
000132		ERRMSG::	.BLKW	1	
000134		ERRBLK::	.BLKW	1	
000136	000000C	L\$HWLEN::			
			.WORD	<<L\$NDHW L\$HWLEN>/2>	
000140	172150	HW.IP.ADRS::			
			.WORD	-5630	
000142	000154	HW.VECTOR::			
			.WORD	154	
000144	000005	HW.BR.LEVEL::			
			.WORD	5	
000146	000000	HW.UNIT.NO::			
			.WORD	0	
000150		L\$NDHW::	.BLKW	1	
000152	000000C	L\$SWLEN::			
			.WORD	<<L\$NDSW L\$SWLEN>/2>	
000154	000001	SW.UNATT::			
			.WORD	1	
000156		L\$NDSW::	.BLKW	1	
000160	000000	L\$PROT::	.WORD	0	
000162	177777		.WORD	1	

000164 000006

.WORD 6

000000					.PSECT \$PLIT\$, RO , D , GBL
000000	045	116	045	P.AAA:	.ASCII /#N#/
000003	124	000	000		.ASCII /T/<00><00>
000006	045	116	045	P.AAB:	.ASCII /#N#/
000011	116	045	101		.ASCII /N#A/
000014	111	116	111		.ASCII /INI/
000017	124	111	101		.ASCII /TIA/
000022	114	111	132		.ASCII /LIZ/
000025	111	116	107		.ASCII /ING/
000030	040	122	103		.ASCII / RC/
000033	062	065	040		.ASCII /25 /
000036	103	117	116		.ASCII /CON/
000041	124	122	117		.ASCII /TRO/
000044	114	114	105		.ASCII /LLE/
000047	122	040	101		.ASCII /R A/
000052	104	104	122		.ASCII /DDR/
000055	105	123	123		.ASCII /ESS/
000060	072	045	123		.ASCII /:#S/
000063	045	117	066		.ASCII /#06/
000066	045	101	050		.ASCII /#A(/
000071	117	051	045		.ASCII /0)#/
000074	116	000			.ASCII /N/<00>
000076	045	116	045	P.AAC:	.ASCII /#N#/
000101	101	114	117		.ASCII /ALO/
000104	107	111	103		.ASCII /GIC/
000107	101	114	040		.ASCII /AL /
000112	125	116	111		.ASCII /UNI/
000115	124	040	045		.ASCII /T #/
000120	104	062	045		.ASCII /D2#/
000123	101	050	104		.ASCII /A(D/
000126	051	045	123		.ASCII /)#S/
000131	045	101	120		.ASCII /#AP/
000134	110	131	123		.ASCII /HYS/
000137	111	103	101		.ASCII /ICA/
000142	114	040	125		.ASCII /L U/
000145	116	111	124		.ASCII /NIT/
000150	040	045	104		.ASCII / #D/
000153	062	045	101		.ASCII /2#A/
000156	050	104	051		.ASCII / (D)/
000161	045	123	045		.ASCII /#S#/
000164	101	106	117		.ASCII /AFO/
000167	122	115	101		.ASCII /RMA/
000172	124	040	101		.ASCII /T A/
000175	102	117	122		.ASCII /BOR/
000200	124	105	104		.ASCII /TED/
000203	000				.ASCII <00>
000204	045	116	045	P.AAD:	.ASCII /#N#/
000207	101	115	111		.ASCII /AMI/
000212	103	122	117		.ASCII /CRO/
000215	040	103	117		.ASCII / CO/
000220	104	105	123		.ASCII /DES/
000223	040	115	117		.ASCII / MO/
000226	104	040	061		.ASCII /D 1/

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr 1984 13:19:09
5-Apr 1984 13:16:08

SEQ 0063
Page 19
VAX-11 B11es-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

000231	066	040	126	.ASCII	/6 V/
000234	105	122	123	.ASCII	/ERS/
000237	111	117	116	.ASCII	/ION/
000242	040	116	125	.ASCII	/ NU/
000245	115	102	105	.ASCII	/MBE/
000250	122	040	075	.ASCII	/R =/
000253	045	123	045	.ASCII	/S#/
000256	117	062	045	.ASCII	/02#/
000261	101	050	117	.ASCII	/A(O/
000264	051	045	116	.ASCII	/)N/
000267	000			.ASCII	<00>
000270	045	116	045	P.AAE: .ASCII	/N#/
000273	101	102	122	.ASCII	/ABR/
000276	111	116	107	.ASCII	/ING/
000301	111	116	107	.ASCII	/ING/
000304	040	117	116	.ASCII	/ ON/
000307	114	111	116	.ASCII	/LIN/
000312	105	072	040	.ASCII	/E: /
000315	114	117	107	.ASCII	/LOG/
000320	111	103	101	.ASCII	/ICA/
000323	114	040	125	.ASCII	/L U/
000326	116	111	124	.ASCII	/NIT/
000331	040	045	104	.ASCII	/ #D/
000334	062	045	101	.ASCII	/2#A/
000337	050	104	051	.ASCII	/ (D)/
000342	045	123	045	.ASCII	/S#/
000345	101	120	110	.ASCII	/APH/
000350	131	123	111	.ASCII	/YSI/
000353	103	101	114	.ASCII	/CAL/
000356	040	125	116	.ASCII	/ UN/
000361	111	124	040	.ASCII	/IT /
000364	045	104	062	.ASCII	/#D2/
000367	045	101	050	.ASCII	/#A(/
000372	104	051	045	.ASCII	/D)#/
000375	116	000	000	.ASCII	/N/<00><00>
000400	045	116	045	P.AAF: .ASCII	/N#/
000403	101	044	106	.ASCII	/A#F/
000406	124	114	105	.ASCII	/TLE/
000411	122	122	055	.ASCII	/RR-/
000414	040	116	117	.ASCII	/ NO/
000417	116	055	105	.ASCII	/N-E/
000422	130	111	123	.ASCII	/XIS/
000425	124	105	116	.ASCII	/TEN/
000430	124	040	122	.ASCII	/T R/
000433	103	062	065	.ASCII	/C25/
000436	040	103	117	.ASCII	/ CO/
000441	116	124	122	.ASCII	/NTR/
000444	117	114	114	.ASCII	/OLL/
000447	105	122	040	.ASCII	/ER /
000452	101	104	104	.ASCII	/ADD/
000455	122	105	123	.ASCII	/RES/
000460	123	040	045	.ASCII	/S #/
000463	117	066	045	.ASCII	/06#/
000466	101	050	117	.ASCII	/A(O/
000471	051	045	116	.ASCII	/)N/
000474	000	000		.ASCII	<00><00>
000476	045	101	106	P.AAG: .ASCII	/#AF/

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

VAX-11 Bliss-16 V3-555
SPIDER#USERS:(NEALE.AZTEC)ZRCHB2.B16;5 (13)

000501	117	122	115	.ASCII	/ORM/	
000504	101	124	124	.ASCII	/ATT/	
000507	111	116	107	.ASCII	/ING/	
000512	040	120	110	.ASCII	/ PH/	
000515	131	123	111	.ASCII	/YSI/	
000520	103	101	114	.ASCII	/CAL/	
000523	040	125	116	.ASCII	/ UN/	
000526	111	124	040	.ASCII	/IT /	
000531	045	104	062	.ASCII	/D2/	
000534	045	101	050	.ASCII	/A(/	
000537	104	051	045	.ASCII	/D)/	
000542	116	000		.ASCII	/N/<00>	
000544	045	116	045	P.AAH:	.ASCII	/N#
000547	101	106	117	.ASCII	/AFO/	
000552	122	115	101	.ASCII	/RMA/	
000555	124	124	105	.ASCII	/TTE/	
000560	122	040	120	.ASCII	/R P/	
000563	122	117	107	.ASCII	/ROG/	
000566	122	105	123	.ASCII	/RES/	
000571	123	111	116	.ASCII	/SIN/	
000574	107	072	040	.ASCII	/G: /	
000577	040	120	111	.ASCII	/ PI/	
000602	104	040	110	.ASCII	/D H/	
000605	111	075	045	.ASCII	/I=#/	
000610	117	066	045	.ASCII	/O6#	
000613	101	050	117	.ASCII	/A(O/	
000616	051	045	123	.ASCII	/)S/	
000621	045	123	045	.ASCII	/S#	
000624	101	120	1.1	.ASCII	/API/	
000627	104	040	114	.ASCII	/D L/	
000632	117	075	045	.ASCII	/O=#/	
000635	117	066	045	.ASCII	/O6#	
000640	101	050	117	.ASCII	/A(O/	
000643	051	000	000	.ASCII	/)/<00><00>	
000646	045	116	000	P.AAI:	.ASCII	/N/<00>
000651	000			.ASCII	<00>	
000652	045	116	045	P.AAJ:	.ASCII	/N#
000655	116	045	116	.ASCII	/N#N/	
000660	045	116	045	.ASCII	/N#	
000663	116	045	116	.ASCII	/N#N/	
000666	045	116	045	.ASCII	/N#	
000671	116	000	000	.ASCII	/N/<00><00>	
000674	045	116	045	P.AAL:	.ASCII	/N#
000677	101	044	106	.ASCII	/A\$F/	
000702	124	114	105	.ASCII	/TLE/	
000705	122	122	055	.ASCII	/RR-/	
000710	040	125	116	.ASCII	/ UN/	
000713	122	105	103	.ASCII	/REC/	
000716	117	107	116	.ASCII	/OGN/	
000721	111	132	101	.ASCII	/IZA/	
000724	102	114	105	.ASCII	/BLE/	
000727	040	105	122	.ASCII	/ ER/	
000732	122	117	122	.ASCII	/ROR/	
000735	040	103	117	.ASCII	/ CO/	
000740	104	105	000	.ASCII	/DE/<00>	
000743	000			.ASCII	<00>	
000744	045	116	045	P.AAM:	.ASCII	/N#

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0065
Page 21
VAX-11 Bliss-16 V3-555
SPIDER\$USFRS:(NEALE.AZTEC)ZRCHB?.B16;5 (13)

000747	101	044	106	.ASCII	/A\$F/
000752	124	114	105	.ASCII	/TLE/
000755	122	122	055	.ASCII	/RR-/
000760	040	105	116	.ASCII	/ EN/
000763	126	105	114	.ASCII	/VEL/
000766	117	120	105	.ASCII	/OPE/
000771	057	120	101	.ASCII	<57>/PA/
000774	103	113	105	.ASCII	/CKE/
000777	124	040	122	.ASCII	/T R/
C01002	105	101	104	.ASCII	/EAD/
001005	040	050	120	.ASCII	/ (P/
001010	101	122	111	.ASCII	/ARI/
001013	124	131	040	.ASCII	/TY /
001016	117	122	040	.ASCII	/OR /
001021	124	111	115	.ASCII	/TIM/
001024	105	117	125	.ASCII	/EQU/
001027	124	051	000	.ASCII	/T)/<00>
001032	045	116	045	P.AAN: .ASCII	/N\$ /
001035	101	044	106	.ASCII	/A\$F/
001040	124	114	105	.ASCII	/TLE/
001043	122	122	055	.ASCII	/RR-/
001046	040	105	116	.ASCII	/ EN/
001051	126	105	114	.ASCII	/VEL/
001054	117	120	105	.ASCII	/OPE/
001057	057	120	101	.ASCII	<57>/PA/
001062	103	113	105	.ASCII	/CKE/
001065	124	040	127	.ASCII	/T W/
001070	122	111	124	.ASCII	/RIT/
001073	105	040	050	.ASCII	/E (/
001076	120	101	122	.ASCII	/PAR/
001101	111	124	131	.ASCII	/ITY/
001104	040	117	122	.ASCII	/ OR/
001107	040	124	111	.ASCII	/ TI/
001112	115	105	117	.ASCII	/MEO/
001115	125	124	051	.ASCII	/UT)/
001120	000	000		.ASCII	<00><00>
001122	045	116	045	P.AAO: .ASCII	/N\$ /
001125	101	044	106	.ASCII	/A\$F/
001130	124	114	105	.ASCII	/TLE/
001133	122	122	055	.ASCII	/RR-/
001136	040	103	117	.ASCII	/ CO/
001141	116	124	122	.ASCII	/NTR/
001144	117	114	114	.ASCII	/OLL/
001147	105	122	040	.ASCII	/ER /
001152	122	117	115	.ASCII	/ROM/
001155	040	101	116	.ASCII	/ AN/
001160	104	040	122	.ASCII	/D R/
001163	101	115	040	.ASCII	/AM /
001166	120	101	122	.ASCII	/PAR/
001171	111	124	131	.ASCII	/ITY/
001174	000	000		.ASCII	<00><00>
001176	045	116	045	P.AAP: .ASCII	/N\$ /
001201	101	044	106	.ASCII	/A\$F/
001204	124	114	105	.ASCII	/TLE/
001207	122	122	055	.ASCII	/RR-/
001212	040	103	117	.ASCII	/ CO/
001215	116	124	122	.ASCII	/NTR/

001220	117	114	114	.ASCII	/OLL/
001223	105	122	040	.ASCII	/ER /
001226	122	101	115	.ASCII	/RAM/
001231	040	120	101	.ASCII	/ PA/
001234	122	111	124	.ASCII	/RIT/
001237	131	000	000	.ASCII	/Y/<00><00>
001242	045	116	045	P.AAQ:	.ASCII
001245	101	044	106	.ASCII	/M# /
001250	124	114	105	.ASCII	/A#F/
001253	122	122	055	.ASCII	/TLE/
001256	040	103	117	.ASCII	/RR- /
001261	116	124	122	.ASCII	/ CO/
001264	117	114	114	.ASCII	/NTR/
001267	105	122	040	.ASCII	/OLL/
001272	122	117	115	.ASCII	/ER /
001275	040	120	101	.ASCII	/ROM/
001300	122	111	124	.ASCII	/ PA/
001303	131	000	000	.ASCII	/RIT/
001306	045	116	045	P.AAR:	.ASCII
001311	101	044	106	.ASCII	/Y/<00><00>
001314	124	114	105	.ASCII	/M# /
001317	122	122	055	.ASCII	/A#F/
001322	040	122	111	.ASCII	/TLE/
001325	116	107	040	.ASCII	/RR- /
001330	122	105	101	.ASCII	/ RI/
001333	104	040	050	.ASCII	/NG /
001336	120	101	122	.ASCII	/REA/
001341	111	124	131	.ASCII	/D (/
001344	040	117	122	.ASCII	/PAR/
001347	040	124	111	.ASCII	/ITY/
001352	115	105	117	.ASCII	/ OR/
001355	125	124	051	.ASCII	/ TI/
001360	000	000		.ASCII	/MEO/
001362	045	116	045	P.AAS:	.ASCII
001365	101	044	106	.ASCII	<00><00>
001370	124	114	105	.ASCII	/M# /
001373	122	122	055	.ASCII	/A#F/
001376	040	122	111	.ASCII	/TLE/
001401	116	107	040	.ASCII	/RR- /
001404	127	122	111	.ASCII	/ RI/
001407	124	105	040	.ASCII	/NG /
001412	050	120	101	.ASCII	/WRI/
001415	122	111	124	.ASCII	/TE /
001420	131	040	117	.ASCII	/(PA,
001423	122	040	124	.ASCII	/RIT/
001426	111	115	105	.ASCII	/Y O/
001431	117	125	124	.ASCII	/R T/
001434	051	000		.ASCII	/IME/
001436	045	116	045	P.AAT:	.ASCII
001441	101	044	106	.ASCII	/OUT/
001444	124	114	105	.ASCII	/)/<(00>
001447	122	122	055	.ASCII	/M# /
001452	040	111	116	.ASCII	/A#F/
001455	124	105	122	.ASCII	/TLE/
001460	122	125	120	.ASCII	/RR- /
001463	124	040	115	.ASCII	/ IN/
001466	101	123	124	.ASCII	/TER/
				.ASCII	/RUP/
				.ASCII	/T M/
				.ASCII	/AST/

ZRCMB2
REV B PATCH 00

ZRCMB0 RC25 DISK FORMATTER
GLOBAL TFXE SECTION

5-Apr 1984 13:19:09
5 Apr 1984 13:16:08

SEQ 0067
Page 23
VAX-11 Bliss-16 V3 555
SPIDER#USERS:(NFALE.AZTEC)ZRCMB2.B16;5 (13)

001471	105	122	000		.ASCII	/ER/<00>
001474	045	116	045	P.AAU:	.ASCII	/MNS/
001477	101	044	106		.ASCII	/A&F/
001502	124	114	105		.ASCII	/TLE/
001505	122	122	055		.ASCII	/RR-/
001510	040	110	117		.ASCII	/MO/
001513	123	124	040		.ASCII	/ST/
001516	101	103	103		.ASCII	/ACC/
001521	105	123	123		.ASCII	/ESS/
001524	040	124	111		.ASCII	/TI/
001527	115	105	117		.ASCII	/MEO/
001532	125	124	000		.ASCII	/UT/<00>
001535	000				.ASCII	<00>
001536	045	116	045	P.AAV:	.ASCII	/MNS/
001541	101	044	106		.ASCII	/A&F/
001544	124	114	105		.ASCII	/TLE/
001547	122	122	055		.ASCII	/RR-/
001552	040	103	122		.ASCII	/CR/
001555	105	104	111		.ASCII	/EDI/
001560	124	040	114		.ASCII	/T L/
001563	111	115	111		.ASCII	/IMI/
001566	124	040	105		.ASCII	/T E/
001571	130	103	105		.ASCII	/XCE/
001574	105	104	105		.ASCII	/EDE/
001577	104	000	000		.ASCII	/D/<00><00>
001602	045	116	045	P.AAW:	.ASCII	/MNS/
001605	101	044	106		.ASCII	/A&F/
001610	124	114	105		.ASCII	/TLE/
001613	122	122	055		.ASCII	/RR-/
001616	040	102	125		.ASCII	/BU/
001621	123	040	115		.ASCII	/S M/
001624	101	123	124		.ASCII	/AST/
001627	105	122	040		.ASCII	/ER/
001632	105	122	122		.ASCII	/ERR/
001635	117	122	000		.ASCII	/OR/<00>
001640	045	116	045	P.AAX:	.ASCII	/MNS/
001643	101	044	106		.ASCII	/A&F/
001646	124	114	105		.ASCII	/TLE/
001651	122	122	055		.ASCII	/RR-/
001654	040	104	111		.ASCII	/DI/
001657	101	107	116		.ASCII	/AGN/
001662	117	123	124		.ASCII	/OST/
001665	111	103	040		.ASCII	/IC/
001670	103	117	116		.ASCII	/CON/
001673	124	122	117		.ASCII	/TRO/
001676	114	114	105		.ASCII	/LLE/
001701	122	040	106		.ASCII	/R F/
001704	101	124	101		.ASCII	/ATA/
001707	114	040	105		.ASCII	/L E/
001712	122	122	117		.ASCII	/RRO/
001715	122	040	000		.ASCII	/R/<00><00>
001720	045	116	045	P.AAY:	.ASCII	/MNS/
001723	101	044	106		.ASCII	/A&F/
001726	124	114	105		.ASCII	/TLE/
001731	122	122	055		.ASCII	/RR-/
001734	040	111	116		.ASCII	/IN/
001737	123	124	122		.ASCII	/STR/

001742	125	103	124	.ASCII	/UCT/	
001745	111	117	116	.ASCII	/ION/	
001750	040	114	117	.ASCII	/ LO/	
001753	117	120	040	.ASCII	/OP /	
001756	124	111	115	.ASCII	/TIM/	
001761	105	117	125	.ASCII	/EOU/	
001764	124	000		.ASCII	/T/<00>	
001766	045	116	045	P.AAZ:	.ASCII	/N#/
001771	101	044	106	.ASCII	/A#F/	
001774	124	114	105	.ASCII	/TLE/	
001777	122	122	055	.ASCII	/RR-/	
002002	040	111	116	.ASCII	/ IN/	
002005	126	101	114	.ASCII	/VAL/	
002010	111	104	040	.ASCII	/ID /	
002013	103	117	116	.ASCII	/CON/	
002016	116	105	103	.ASCII	/NEC/	
002021	124	111	117	.ASCII	/TIO/	
002024	116	040	1:1	.ASCII	/N I/	
002027	104	105	116	.ASCII	/DEN/	
002032	124	111	106	.ASCII	/TIF/	
002035	111	105	122	.ASCII	/IER/	
002040	000	000		.ASCII	<00><00>	
002042	045	116	045	P.ABA:	.ASCII	/N#/
002045	101	044	106	.ASCII	/A#F/	
002050	124	114	105	.ASCII	/TLE/	
002053	122	122	055	.ASCII	/RR-/	
002056	040	111	116	.ASCII	/ IN/	
002061	124	105	122	.ASCII	/TER/	
002064	122	125	120	.ASCII	/RUP/	
002067	124	040	127	.ASCII	/T W/	
002072	122	111	124	.ASCII	/RIT/	
002075	105	000	000	.ASCII	/E/<00><00>	
002100	045	116	045	P.ABB:	.ASCII	/N#/
002103	101	044	106	.ASCII	/A#F/	
002106	124	114	105	.ASCII	/TLE/	
002111	122	122	055	.ASCII	/RR-/	
002114	040	115	101	.ASCII	/ MA/	
002117	111	116	124	.ASCII	/INT/	
002122	105	116	101	.ASCII	/ENA/	
002125	116	103	105	.ASCII	/NCE/	
002130	040	122	105	.ASCII	/ RE/	
002133	101	104	057	.ASCII	/AD/<57>	
002136	127	122	111	.ASCII	/WRI/	
002141	124	105	040	.ASCII	/TE /	
002144	111	116	126	.ASCII	/INV/	
002147	101	114	111	.ASCII	/ALI/	
002152	104	040	122	.ASCII	/D R/	
002155	105	107	111	.ASCII	/EGI/	
002160	117	116	040	.ASCII	/ON /	
002163	111	104	105	.ASCII	/IDE/	
002166	116	124	111	.ASCII	/NTI/	
002171	106	111	105	.ASCII	/FIE/	
002174	122	000		.ASCII	/R/<00>	
002176	045	116	045	P.ABC:	.ASCII	/N#/
002201	101	044	106	.ASCII	/A#F/	
002204	124	114	105	.ASCII	/ILE/	
002207	122	122	055	.ASCII	/RR /	

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr 1984 13:19:09
5 Apr -1984 13:16:08

SEQ 0069
Page 25
VAX-11 Bliss-16 V3 555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

002212	040	115	101	.ASCII	/ MA/
002215	111	116	124	.ASCII	/INT/
002220	105	116	101	.ASCII	/ENA/
002223	116	103	105	.ASCII	/NCE/
002226	040	127	122	.ASCII	/ WR/
002231	111	124	105	.ASCII	/ITE/
002234	040	114	117	.ASCII	/ LO/
002237	101	104	040	.ASCII	/AD /
002242	124	117	040	.ASCII	/TO /
002245	116	117	116	.ASCII	/NON/
002250	055	114	117	.ASCII	/ LO/
002253	101	104	101	.ASCII	/ADA/
002256	102	114	105	.ASCII	/BLE/
002261	040	103	117	.ASCII	/ CO/
002264	116	124	122	.ASCII	/NTR/
002267	117	114	114	.ASCII	/OLL/
002272	105	122	000	.ASCII	/ER/<00>
002275	000			.ASCII	<00>
002276	045	116	045	P.ABD:	.ASCII /#N#/
002301	101	044	106	.ASCII	/A#F/
002304	124	114	105	.ASCII	/TLE/
002307	122	122	055	.ASCII	/RR-/
002312	040	103	117	.ASCII	/ CO/
002315	116	124	122	.ASCII	/NTR/
002320	117	114	114	.ASCII	/OLL/
002323	105	122	040	.ASCII	/ER /
002326	122	101	115	.ASCII	/RAM/
002331	040	105	122	.ASCII	/ ER/
002334	122	117	122	.ASCII	/ROR/
002337	040	050	116	.ASCII	/ (N/
002342	117	116	055	.ASCII	/ON-/
002345	120	101	122	.ASCII	/PAR/
002350	111	124	131	.ASCII	/ITY/
002353	051	000	000	.ASCII	/)/<00><00>
002356	045	116	045	P.ABE:	.ASCII /#N#/
002361	101	044	106	.ASCII	/A#F/
002364	124	114	105	.ASCII	/TLE/
002367	122	122	055	.ASCII	/RR-/
002372	040	111	116	.ASCII	/ IN/
002375	111	124	040	.ASCII	/IT /
002400	123	105	121	.ASCII	/SEQ/
002403	125	105	116	.ASCII	/UEN/
002406	103	105	040	.ASCII	/CE /
002411	105	122	122	.ASCII	/ERR/
002414	117	122	000	.ASCII	/OR/<00>
002417	000			.ASCII	<00>
002420	045	116	045	P.ABF:	.ASCII /#N#/
002423	101	044	106	.ASCII	/A#F/
002426	124	114	105	.ASCII	/TLE/
002431	122	122	055	.ASCII	/RR /
002434	040	110	111	.ASCII	/ HI/
002437	107	110	040	.ASCII	/GH /
002442	114	105	126	.ASCII	/LEV/
002445	105	114	040	.ASCII	/EL /
002450	120	122	117	.ASCII	/PRO/
002453	124	117	103	.ASCII	/TOC/
002456	117	114	040	.ASCII	/OL /

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB2.816;5 (13)
Page 26

002461	111	116	103	.ASCII	/INC/
002464	117	115	120	.ASCII	/OMP/
002467	101	124	111	.ASCII	/ATI/
002472	102	111	114	.ASCII	/BIL/
002475	111	124	131	.ASCII	/ITY/
002500	040	105	122	.ASCII	/ ER/
002503	122	117	122	.ASCII	/ROR/
002506	000	000		.ASCII	<00><00>
002510	045	116	045	P.ABG: .ASCII	/N#/
002513	101	044	106	.ASCII	/A\$F/
002516	124	114	105	.ASCII	/TLE/
002521	122	122	055	.ASCII	/RR-/
002524	040	120	125	.ASCII	/ PU/
002527	122	107	105	.ASCII	/RGE/
002532	057	120	117	.ASCII	<57>/PO/
002535	114	114	040	.ASCII	/LL /
002540	110	101	122	.ASCII	/HAR/
002543	104	127	101	.ASCII	/DWA/
002546	122	105	040	.ASCII	/RE /
002551	106	101	111	.ASCII	/FAI/
002554	114	125	122	.ASCII	/LUR/
002557	105	040	000	.ASCII	/E /<00>
002562	045	116	045	P.ABH: .ASCII	/N#/
002565	101	044	106	.ASCII	/A\$F/
002570	124	114	105	.ASCII	/TLE/
002573	122	122	055	.ASCII	/RR-/
002576	040	115	101	.ASCII	/ MA/
002601	120	120	111	.ASCII	/PPI/
002604	116	107	040	.ASCII	/NG /
002607	122	105	107	.ASCII	/REG/
002612	111	123	124	.ASCII	/IST/
002615	105	122	040	.ASCII	/ER /
002620	122	105	101	.ASCII	/REA/
002623	104	040	105	.ASCII	/D E/
002626	122	122	117	.ASCII	/RRO/
002631	122	040	050	.ASCII	/R (/
002634	120	101	122	.ASCII	/PAR/
002637	111	124	131	.ASCII	/ITY/
002642	040	117	122	.ASCII	/ OR/
002645	040	124	111	.ASCII	/ TI/
002650	115	105	117	.ASCII	/MEO/
002653	125	124	051	.ASCII	/UT)/
002656	000	000		.ASCII	<00><00>
002660	000674'			P.AAK: .WORD	P.AAL
002662	000744'			.WORD	P.AAM
002664	001032'			.WORD	P.AAN
002666	001122'			.WORD	P.AAO
002670	001176'			.WORD	P.AAP
002672	001242'			.WORD	P.AAQ
002674	001306'			.WORD	P.AAR
002676	001362'			.WORD	P.AAS
002700	001436'			.WORD	P.AAT
002702	001474'			.WORD	P.AAU
002704	001536'			.WORD	P.AAV
002706	001602'			.WORD	P.AAW
002710	001640'			.WORD	P.AAX
002712	001720'			.WORD	P.AAY

002714	001766'				.WORD	P.AAZ
002716	002042'				.WORD	P.ABA
002720	002100'				.WORD	P.ABB
002722	002176'				.WORD	P.ABC
002724	002276'				.WORD	P.ABD
002726	002356'				.WORD	P.ABE
002730	002420'				.WORD	P.ABF
002732	002510'				.WORD	P.ABG
002734	002562'				.WORD	P.ABH
002736	045	116	045	P.ABJ:	.ASCII	/N# /
002741	101	044	106		.ASCII	/A#F /
002744	124	114	105		.ASCII	/TLE /
002747	122	122	055		.ASCII	/RR- /
002752	040	122	105		.ASCII	/RE /
002755	123	120	117		.ASCII	/SPO /
002760	116	123	105		.ASCII	/NSE /
002763	040	123	124		.ASCII	/ST /
002766	101	124	125		.ASCII	/ATU /
002771	123	040	105		.ASCII	/S E /
002774	122	122	117		.ASCII	/RRO /
002777	122	072	045		.ASCII	/R: # /
003002	123	000			.ASCII	/S/<00>
003004	045	116	045	P.ABK:	.ASCII	/N# /
003007	101	044	106		.ASCII	/A#F /
003012	124	114	105		.ASCII	/TLE /
003015	122	122	055		.ASCII	/RR- /
003020	040	110	117		.ASCII	/HO /
003023	123	124	057		.ASCII	/ST/<57>
003026	103	117	116		.ASCII	/CON /
003031	124	122	117		.ASCII	/TRO /
003034	114	114	105		.ASCII	/LLE /
003037	122	040	117		.ASCII	/R O /
003042	125	124	040		.ASCII	/UT /
003045	117	106	040		.ASCII	/OF /
003050	123	105	121		.ASCII	/SEQ /
003053	000				.ASCII	<00>
003054	045	116	045	P.ABL:	.ASCII	/N# /
003057	101	044	106		.ASCII	/A#F /
003062	124	114	105		.ASCII	/TLE /
003065	122	122	055		.ASCII	/RR- /
003070	040	122	105		.ASCII	/RE /
003073	115	117	124		.ASCII	/MOT /
003076	105	040	120		.ASCII	/E P /
003101	122	117	107		.ASCII	/ROG /
003104	040	116	117		.ASCII	/NO /
003107	124	040	122		.ASCII	/T R /
003112	125	116	116		.ASCII	/UNN /
003115	111	116	107		.ASCII	/ING /
003120	000	000			.ASCII	<00><00>
003122	045	116	045	P.ABM:	.ASCII	/N# /
003125	101	044	106		.ASCII	/A#F /
003130	124	114	105		.ASCII	/TLE /
003133	122	122	055		.ASCII	/RR- /
003136	040	125	116		.ASCII	/UN /
003141	113	116	117		.ASCII	/KNO /
003144	127	116	040		.ASCII	/WN /
003147	122	105	124		.ASCII	/RET /

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0072
Page 28
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

003152	125	122	116	.ASCII	/URN/
003155	040	123	124	.ASCII	/ST/
003160	101	124	125	.ASCII	/ATU/
003163	123	040	103	.ASCII	/S C/
003166	117	104	105	.ASCII	/ODE/
003171	000			.ASCII	<00>
003172	045	116	045	P.ABN:	.ASCII /#N#/
003175	101	044	106	.ASCII	/A\$F/
003200	124	114	105	.ASCII	/TLE/
003203	122	122	055	.ASCII	/RR-/
003206	040	103	117	.ASCII	/CO/
003211	115	040	101	.ASCII	/M A/
003214	122	105	101	.ASCII	/REA/
003217	040	111	116	.ASCII	/IN/
003222	111	124	040	.ASCII	/IT /
003225	105	122	122	.ASCII	/ERR/
003230	117	122	000	.ASCII	/OR/<00>
003233	000			.ASCII	<00>
003234	045	116	045	P.ABO:	.ASCII /#N#/
003237	101	044	106	.ASCII	/A\$F/
003242	124	114	105	.ASCII	/TLE/
003245	122	122	055	.ASCII	/RR-/
003250	040	120	117	.ASCII	/PO/
003253	122	124	057	.ASCII	/RT/<57>
003256	110	117	123	.ASCII	/HOS/
003261	124	040	123	.ASCII	/T S/
003264	131	116	103	.ASCII	/YNC/
003267	040	105	122	.ASCII	/ER/
003272	122	117	122	.ASCII	/ROR/
003275	000			.ASCII	<00>
003276	045	116	045	P.ABP:	.ASCII /#N#/
003301	101	044	106	.ASCII	/A\$F/
003304	124	114	105	.ASCII	/TLE/
003307	122	122	055	.ASCII	/RR-/
003312	040	115	105	.ASCII	/ME/
003315	123	123	101	.ASCII	/SSA/
003320	107	105	040	.ASCII	/GE /
003323	114	105	116	.ASCII	/LEN/
003326	107	124	110	.ASCII	/GTH/
003331	040	105	122	.ASCII	/ER/
003334	122	117	122	.ASCII	/ROR/
003337	000			.ASCII	<00>
003340	045	116	045	P.ABQ:	.ASCII /#N#/
003343	101	044	106	.ASCII	/A\$F/
003346	124	114	105	.ASCII	/TLE/
003351	122	122	055	.ASCII	/RR /
003354	040	125	116	.ASCII	/UN/
003357	113	116	117	.ASCII	/KNO/
003362	127	116	040	.ASCII	/WN /
003365	105	116	104	.ASCII	/END/
003370	103	117	104	.ASCII	/COD/
003373	105	040	122	.ASCII	/E R/
003376	105	103	105	.ASCII	/ECE/
003401	111	126	105	.ASCII	/IVE/
003404	104	000		.ASCII	/D/<00>
003406	045	116	045	P.ABR:	.ASCII /#N#/
003411	101	044	106	.ASCII	/A\$F/

ZRCH82
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr 1984 13:19:09
5 Apr 1984 13:16:08

SEQ 0073
Page 29
VAX 11 B111-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH82.B16;5 (13)

003414	124	114	105	.ASCII	/TLE/
003417	122	122	055	.ASCII	/RR-/
003422	040	101	104	.ASCII	/AD/
003425	101	120	124	.ASCII	/APT/
003430	117	122	040	.ASCII	/OR/
003433	120	125	122	.ASCII	/PUR/
003436	107	105	040	.ASCII	/GE/
003441	105	122	122	.ASCII	/ERR/
003444	117	122	000	.ASCII	/OR/<00>
003447	000			.ASCII	<00>
003450	045	116	045	P.ABS: .ASCII	/NMS/
003453	101	044	106	.ASCII	/A&F/
003456	124	114	105	.ASCII	/TLE/
003461	122	122	055	.ASCII	/RR-/
003464	040	125	116	.ASCII	/UN/
003467	113	116	117	.ASCII	/KNO/
003472	127	116	040	.ASCII	/WN/
003475	111	116	124	.ASCII	/INT/
003500	105	122	122	.ASCII	/ERR/
003503	125	120	124	.ASCII	/UPT/
003506	000	000		.ASCII	<00><00>
003510	045	116	045	P.ABT: .ASCII	/NMS/
003513	101	044	106	.ASCII	/A&F/
003516	124	114	105	.ASCII	/TLE/
003521	122	122	055	.ASCII	/RR-/
003524	040	111	116	.ASCII	/IN/
003527	111	124	040	.ASCII	/IT/
003532	123	105	121	.ASCII	/SEQ/
003535	040	123	124	.ASCII	/ST/
003540	105	120	040	.ASCII	/EP/
003543	124	111	115	.ASCII	/TIM/
003546	105	104	040	.ASCII	/ED/
003551	117	125	124	.ASCII	/OUT/
003554	000	000		.ASCII	<00><00>
003556	045	116	045	P.ABU: .ASCII	/NMS/
003561	101	044	106	.ASCII	/A&F/
003564	124	114	105	.ASCII	/TLE/
003567	122	122	055	.ASCII	/RR-/
003572	040	111	116	.ASCII	/IN/
003575	111	124	040	.ASCII	/IT/
003600	123	105	121	.ASCII	/SEQ/
003603	040	103	117	.ASCII	/CO/
003606	115	120	101	.ASCII	/MPA/
003611	122	105	040	.ASCII	/RE/
003614	105	122	122	.ASCII	/ERR/
003617	117	122	000	.ASCII	/OR/<00>
003622	045	116	045	P.ABV: .ASCII	/NMS/
003625	101	044	106	.ASCII	/A&F/
003630	124	114	105	.ASCII	/TLE/
003633	122	122	055	.ASCII	/RR-/
003636	040	125	116	.ASCII	/UN/
003641	105	130	120	.ASCII	/EXP/
003644	105	103	124	.ASCII	/ECT/
003647	105	104	040	.ASCII	/ED/
003652	101	124	124	.ASCII	/ATT/
003655	105	116	124	.ASCII	/ENT/
003660	111	117	116	.ASCII	/ION/

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr 1984 13:19:09
5 Apr-1984 13:16:08

SEQ 0074
Page 30
VAX-11 Bliss 16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCHB2.B16;5 (13)

003663	040	105	116	.ASCII	/ EN/	
003666	104	040	115	.ASCII	/D M/	
003671	105	123	123	.ASCII	/ESS/	
003674	101	107	105	.ASCII	/AGE/	
003677	040	122	105	.ASCII	/ RE/	
003702	103	105	111	.ASCII	/CEI/	
003705	126	105	104	.ASCII	/VED/	
003710	000	000		.ASCII	<00><00>	
003712	045	116	045	P.ABW:	.ASCII	/NMS/
003715	101	044	106	.ASCII	/A#F/	
003720	124	114	105	.ASCII	/TLE/	
003723	122	122	055	.ASCII	/RR /	
003726	040	125	116	.ASCII	/ UN/	
003731	105	130	120	.ASCII	/EXP/	
003734	105	103	124	.ASCII	/ECT/	
003737	105	104	040	.ASCII	/ED /	
003742	103	117	115	.ASCII	/COM/	
003745	115	101	116	.ASCII	/MAN/	
003750	104	040	117	.ASCII	/D O/	
003753	120	103	117	.ASCII	/PCO/	
003756	104	105	040	.ASCII	/DE /	
003761	111	116	040	.ASCII	/IN /	
003764	105	116	104	.ASCII	/END/	
003767	040	115	105	.ASCII	/ ME/	
003772	123	123	101	.ASCII	/SSA/	
003775	107	105	040	.ASCII	/GE /	
004000	122	105	103	.ASCII	/REC/	
004003	105	111	126	.ASCII	/EIV/	
004006	105	104	000	.ASCII	/ED/<00>	
004011	000			.ASCII	<00>	
004012	045	116	045	P.ABX:	.ASCII	/NMS/
004015	101	044	106	.ASCII	/A#F/	
004020	124	114	105	.ASCII	/TLE/	
004023	122	122	055	.ASCII	/RR /	
004026	040	125	116	.ASCII	/ UN/	
004031	105	130	120	.ASCII	/EXP/	
004034	105	103	124	.ASCII	/ECT/	
004037	105	104	040	.ASCII	/ED /	
004042	123	105	122	.ASCII	/SER/	
004045	111	117	125	.ASCII	/IOU/	
004050	123	040	105	.ASCII	/S E/	
004053	130	103	105	.ASCII	/XCE/	
004056	120	124	111	.ASCII	/PTI/	
004061	117	116	040	.ASCII	/ON /	
004064	105	116	104	.ASCII	/END/	
004067	040	115	105	.ASCII	/ ME/	
004072	123	123	101	.ASCII	/SSA/	
004075	107	105	040	.ASCII	/GE /	
004100	122	105	103	.ASCII	/REC/	
004103	105	111	126	.ASCII	/EIV/	
004106	105	104	000	.ASCII	/ED/<00>	
004111	000			.ASCII	<00>	
004112	045	116	045	P.ABY:	.ASCII	/NMS/
004115	101	044	106	.ASCII	/A#F/	
004120	124	114	105	.ASCII	/TLE/	
004123	122	122	055	.ASCII	/RR /	
004126	040	111	116	.ASCII	/ IN/	

ZRCH82
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr 1984 13:19:09
5 Apr 1984 13:16:08

SEQ 0075
Page 31
VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCH82.B16;5 (13)

004131	126	101	114	.ASCII	/VAL/	
004134	111	104	040	.ASCII	/ID /	
004137	103	117	115	.ASCII	/COM/	
004142	115	101	116	.ASCII	/MAN/	
004145	104	040	105	.ASCII	/D E/	
004150	116	104	040	.ASCII	/ND /	
004153	115	105	123	.ASCII	/MES/	
004156	123	101	107	.ASCII	/SAG/	
004161	105	040	122	.ASCII	/E R/	
004164	105	103	105	.ASCII	/ECE/	
004167	111	126	105	.ASCII	/IVE/	
004172	104	000		.ASCII	/D/<00>	
004174	045	116	045	P.ABZ:	.ASCII	/N# /
004177	101	044	106	.ASCII	/A\$F/	
004202	124	114	105	.ASCII	/TLE/	
004205	122	122	055	.ASCII	/RR-/	
004210	040	125	116	.ASCII	/ UN/	
004213	113	116	117	.ASCII	/KNO/	
004216	127	116	040	.ASCII	/WN /	
004221	115	105	123	.ASCII	/MES/	
004224	123	101	107	.ASCII	/SAG/	
004227	105	040	124	.ASCII	/E T/	
004232	131	120	105	.ASCII	/YPE/	
004235	040	122	105	.ASCII	/ RE/	
004240	103	105	111	.ASCII	/CEI/	
004243	126	105	104	.ASCII	/VED/	
004246	000	000		.ASCII	<00><00>	
004250	045	116	045	P.ACA:	.ASCII	/N# /
004253	101	044	106	.ASCII	/A\$F/	
004256	124	114	105	.ASCII	/TLE/	
004261	122	122	055	.ASCII	/RR-/	
004264	040	117	125	.ASCII	/ OU/	
004267	124	123	124	.ASCII	/TST/	
004272	101	116	104	.ASCII	/AND/	
004275	111	116	107	.ASCII	/ING/	
004300	040	103	117	.ASCII	/ CO/	
004303	115	115	101	.ASCII	/MMA/	
004306	116	104	040	.ASCII	/ND /	
004311	102	125	106	.ASCII	/BUF/	
004314	106	105	122	.ASCII	/FER/	
004317	040	106	125	.ASCII	/ FU/	
004322	114	114	000	.ASCII	/LL/<00>	
004325	000			.ASCII	<00>	
004326	045	116	045	P.ACB:	.ASCII	/N# /
004331	101	044	106	.ASCII	/A\$F/	
004334	124	114	105	.ASCII	/TLE/	
004337	122	122	055	.ASCII	/RR /	
004342	040	117	125	.ASCII	/ OU/	
004345	124	040	123	.ASCII	/T S/	
004350	124	101	116	.ASCII	/TAN/	
004353	104	111	116	.ASCII	/DIN/	
004356	107	040	103	.ASCII	/G C/	
004361	117	115	115	.ASCII	/OMM/	
004364	101	116	104	.ASCII	/AND/	
004367	040	102	125	.ASCII	/ BU/	
004372	106	106	105	.ASCII	/FFE/	
004375	122	040	117	.ASCII	/R O/	

ZRCMB2
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION5-Apr 1984 13:19:09
5 Apr 1984 13:16:08VAX 11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCMB2.B16;5 (13)
Page 32

004400	125	124	040	.ASCII	/UT /
004403	117	106	040	.ASCII	/OF /
004406	123	131	116	.ASCII	/SYN/
004411	103	040	105	.ASCII	/C E/
004414	122	122	117	.ASCII	/RRO/
004417	122	000	000	.ASCII	/R/<00><00>
004422	045	116	045	P.ACC: .ASCII	/N# /
004425	101	044	106	.ASCII	/A\$F/
004430	124	114	105	.ASCII	/TLE/
004433	122	122	055	.ASCII	/RR-/
004436	040	125	116	.ASCII	/ UN/
004441	113	116	117	.ASCII	/KNO/
004444	127	116	040	.ASCII	/WN /
004447	115	105	123	.ASCII	/MES/
004452	123	101	107	.ASCII	/SAG/
004455	105	040	116	.ASCII	/E N/
004460	125	115	102	.ASCII	/UMB/
004463	105	122	040	.ASCII	/ER /
004466	122	105	103	.ASCII	/REC/
004471	105	111	126	.ASCII	/EIV/
004474	105	104	000	.ASCII	/ED/<00>
004477	000			.ASCII	<00>
004500	045	116	045	P.ACD: .ASCII	/N# /
004503	101	044	106	.ASCII	/A\$F/
004506	124	114	105	.ASCII	/TLE/
004511	122	122	055	.ASCII	/RR /
004514	040	106	111	.ASCII	/ FI/
004517	114	105	040	.ASCII	/LE /
004522	122	105	101	.ASCII	/REA/
004525	104	040	105	.ASCII	/D E/
004530	122	122	117	.ASCII	/RRO/
004533	122	000	000	.ASCII	/R/<00><00>
004536	045	116	045	P.ACE: .ASCII	/N# /
004541	101	044	106	.ASCII	/A\$F/
004544	124	114	105	.ASCII	/TLE/
004547	122	122	055	.ASCII	/RR-/
004552	040	120	117	.ASCII	/ PO/
004555	122	124	057	.ASCII	/RT/<57>
004560	103	117	116	.ASCII	/CON/
004563	124	122	117	.ASCII	/TRO/
004566	114	114	105	.ASCII	/LLE/
004571	122	040	124	.ASCII	/R T/
004574	111	115	105	.ASCII	/IME/
004577	117	125	124	.ASCII	/OUT/
004602	040	105	122	.ASCII	/ ER/
004605	122	117	122	.ASCII	/ROR/
004610	000	000		.ASCII	<00><00>
004612	045	116	045	P.ACF: .ASCII	/N# /
004615	101	044	106	.ASCII	/A\$F/
004620	124	114	105	.ASCII	/TLE/
004623	122	122	055	.ASCII	/RR-/
004626	040	111	114	.ASCII	/ IL/
004631	114	105	107	.ASCII	/LEG/
004634	101	114	040	.ASCII	/AL /
004637	106	103	124	.ASCII	/FCT/
004642	040	106	111	.ASCII	/ FI/
004645	114	105	040	.ASCII	/LE /

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr-1984 13:19:09
5 Apr 1984 13:16:08

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB2.B16:5 (13)

004650	114	105	116		.ASCII	/LEN/
004653	107	124	110		.ASCII	/GTH/
004656	000	000			.ASCII	<00><00>
004660	002736'			P.ABI:	.WORD	P.ABJ
004662	003004'				.WORD	P.ABK
004664	003054'				.WORD	P.ABL
004666	003122'				.WORD	P.ABM
004670	003172'				.WORD	P.ABN
004672	003234'				.WORD	P.ABO
004674	003276'				.WORD	P.ABP
004676	003340'				.WORD	P.ABQ
004700	003406'				.WORD	P.ABR
004702	003450'				.WORD	P.ABS
004704	003510'				.WORD	P.ABT
004706	003556'				.WORD	P.ABU
004710	003622'				.WORD	P.ABV
004712	003712'				.WORD	P.ABW
004714	004012'				.WORD	P.ABX
004716	004112'				.WORD	P.ABY
004720	004174'				.WORD	P.ABZ
004722	004250'				.WORD	P.ACA
004724	004326'				.WORD	P.ACB
004726	004422'				.WORD	P.ACC
004730	004500'				.WORD	P.ACD
004732	004536'				.WORD	P.ACE
004734	004612'				.WORD	P.ACF
004736	045	116	045	P.ACH:	.ASCII	/N#
004741	101	044	106		.ASCII	/A\$F/
004744	124	114	105		.ASCII	/TLE/
004747	122	122	055		.ASCII	/RR-/
004752	040	126	101		.ASCII	/VA/
004755	130	040	122		.ASCII	/X R/
004760	105	101	104		.ASCII	/EAD/
004763	057	127	122		.ASCII	<57>/WR/
004766	111	124	105		.ASCII	/ITE/
004771	040	105	122		.ASCII	/ER/
004774	122	117	122		.ASCII	/ROR/
004777	040	117	116		.ASCII	/ON/
005002	040	111	116		.ASCII	/IN/
005005	124	105	122		.ASCII	/TER/
005010	122	125	120		.ASCII	/RUP/
005013	124	000	000		.ASCII	/T/<00><00>
005016	045	116	045	P.ACI:	.ASCII	/N#
005021	101	044	106		.ASCII	/A\$F/
005024	124	114	105		.ASCII	/TLE/
005027	122	122	055		.ASCII	/RR /
005032	040	111	116		.ASCII	/IN/
005035	103	117	116		.ASCII	/CON/
005040	123	111	123		.ASCII	/SIS/
005043	124	105	116		.ASCII	/TEN/
005046	103	131	040		.ASCII	/CY /
005051	101	124	040		.ASCII	/AT /
005054	125	056	102		.ASCII	/U.B/
005057	106	111	114		.ASCII	/FIL/
005062	000	000			.ASCII	<00><00>
005064	045	116	045	P.ACJ:	.ASCII	/N#
005067	101	044	106		.ASCII	/A\$F/

N6

ZRCHB2
REV R PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0078
Page 34
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

005072	124	114	105	.ASCII	/TLE/	
005075	122	122	055	.ASCII	/RR-/	
005100	040	111	116	.ASCII	/ IN/	
005103	103	117	116	.ASCII	/CON/	
005106	123	111	123	.ASCII	/SIS/	
005111	124	105	116	.ASCII	/TEN/	
005114	103	131	040	.ASCII	/CY /	
005117	101	124	040	.ASCII	/AT /	
005122	125	056	102	.ASCII	/U.B/	
005125	115	124	131	.ASCII	/MTY/	
005130	000	000		.ASCII	<00><00>	
005132	045	116	045	P.ACK:	.ASCII	/N#/
005135	101	044	106	.ASCII	/A\$F/	
005140	124	114	105	.ASCII	/TLE/	
005143	122	122	055	.ASCII	/RR-/	
005146	040	111	116	.ASCII	/ IN/	
005151	103	117	116	.ASCII	/CON/	
005154	123	111	123	.ASCII	/SIS/	
005157	124	105	116	.ASCII	/TEN/	
005162	103	131	040	.ASCII	/CY /	
005165	101	124	040	.ASCII	/AT /	
005170	125	056	101	.ASCII	/U.A/	
005173	114	117	103	.ASCII	/LOC/	
005176	000	000		.ASCII	<00><00>	
005200	045	116	045	P.ACL:	.ASCII	/N#/
005203	101	044	106	.ASCII	/A\$F/	
005206	124	114	105	.ASCII	/TLE/	
005211	122	122	055	.ASCII	/RR-/	
005214	040	111	116	.ASCII	/ IN/	
005217	103	117	116	.ASCII	/CON/	
005222	123	111	123	.ASCII	/SIS/	
005225	124	105	116	.ASCII	/TEN/	
005230	103	131	040	.ASCII	/CY /	
005233	101	124	040	.ASCII	/AT /	
005236	123	105	122	.ASCII	/SER/	
005241	126	117	040	.ASCII	/VO /	
005244	105	116	124	.ASCII	/ENT/	
005247	122	131	040	.ASCII	/RY /	
005252	050	120	111	.ASCII	/(PI/	
005255	120	040	123	.ASCII	/P S/	
005260	105	124	051	.ASCII	/ET)/	
005263	000			.ASCII	<00>	
005264	045	116	045	P.ACM:	.ASCII	/N#/
005267	101	044	106	.ASCII	/A\$F/	
005272	124	114	105	.ASCII	/TLE/	
005275	122	122	055	.ASCII	/RR-/	
005300	040	111	116	.ASCII	/ IN/	
005303	103	117	116	.ASCII	/CON/	
005306	123	111	123	.ASCII	/SIS/	
005311	124	105	116	.ASCII	/TEN/	
005314	103	131	040	.ASCII	/CY /	
005317	101	124	040	.ASCII	/AT /	
005322	123	105	122	.ASCII	/SER/	
005325	126	117	040	.ASCII	/VO /	
005330	105	116	124	.ASCII	/ENT/	
005333	122	131	040	.ASCII	/RY /	
005336	050	105	122	.ASCII	/(ER/	

ZRCH82
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr-1984 13:19:09
5 Apr-1984 13:16:08

SEQ 0079
Page 35
VAX-11 Bli 16 v3 555
SPIDER#USERS:(NEALE.AZTEC)ZRCH82.B16;5 (13)

005341	122	040	123	.ASCII	/R S/
005344	105	124	051	.ASCII	/ET)/
005347	000			.ASCII	<00>
005350	045	116	045	P.AC�:	.ASCII /#N#/
005353	101	044	106	.ASCII	/A#F/
005356	124	114	105	.ASCII	/TLE/
005361	122	122	055	.ASCII	/RR /
005364	040	111	116	.ASCII	/ IN/
005367	103	117	116	.ASCII	/CON/
005372	123	111	123	.ASCII	/SIS/
005375	124	105	116	.ASCII	/TEN/
005400	103	131	040	.ASCII	/CY /
005403	101	124	040	.ASCII	/AT /
005406	125	056	123	.ASCII	/U.S/
005411	105	116	104	.ASCII	/END/
005414	000	000		.ASCII	<00><00>
005416	045	116	045	P.ACO:	.ASCII /#N#/
005421	101	044	106	.ASCII	/A#F/
005424	124	114	105	.ASCII	/TLE/
005427	122	122	055	.ASCII	/RR-/
005432	040	111	116	.ASCII	/ IN/
005435	103	117	116	.ASCII	/CON/
005440	123	111	123	.ASCII	/SIS/
005443	124	105	116	.ASCII	/TEN/
005446	103	131	040	.ASCII	/CY /
005451	101	124	040	.ASCII	/AT /
005454	125	056	122	.ASCII	/U.R/
005457	105	103	126	.ASCII	/ECV/
005462	000	000		.ASCII	<00><00>
005464	045	116	045	P.ACP:	.ASCII /#N#/
005467	101	044	106	.ASCII	/A#F/
005472	124	114	105	.ASCII	/TLE/
005475	122	122	055	.ASCII	/RR-/
005500	040	111	116	.ASCII	/ IN/
005503	103	117	116	.ASCII	/CON/
005506	123	111	123	.ASCII	/SIS/
005511	124	105	116	.ASCII	/TEN/
005514	103	131	040	.ASCII	/CY /
005517	101	124	040	.ASCII	/AT /
005522	125	056	101	.ASCII	/U.A/
005525	124	124	116	.ASCII	/TIN/
005530	000	000		.ASCII	<00><00>
005532	045	116	045	P.ACQ:	.ASCII /#N#/
005535	101	044	106	.ASCII	/A#F/
005540	124	114	105	.ASCII	/TLE/
005543	122	122	055	.ASCII	/RR /
005546	040	111	116	.ASCII	/ IN/
005551	103	117	116	.ASCII	/CON/
005554	123	111	123	.ASCII	/SIS/
005557	124	105	116	.ASCII	/TEN/
005562	103	131	040	.ASCII	/CY /
005565	101	124	040	.ASCII	/AT /
005570	125	056	117	.ASCII	/U.O/
005573	116	114	116	.ASCII	/MLN/
005576	000	000		.ASCII	<00><00>
005600	045	116	045	P.ACR:	.ASCII /#N#/
005603	101	044	106	.ASCII	/A#F/

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr 1984 13:19:09
5-Apr 1984 13:16:08

SEQ 0080
Page 36
VAX-11 B110-16 V3 555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

005606	124	114	105	.ASCII	/TLE/	
005611	122	122	055	.ASCII	/RR-/	
005614	040	111	114	.ASCII	/ IL/	
005617	114	105	107	.ASCII	/LEG/	
005622	101	114	040	.ASCII	/AL /	
005625	104	040	122	.ASCII	/D ?/	
005630	105	121	125	.ASCII	/EQU/	
005633	105	123	124	.ASCII	/EST/	
005636	040	050	125	.ASCII	/ (U/	
005641	056	121	104	.ASCII	/.QD/	
005644	122	121	051	.ASCII	/RQ)/	
005647	000			.ASCII	<00>	
005650	045	116	045	P.ACS:	.ASCII	/N#/
005653	101	044	106	.ASCII	/A#F/	
005656	124	114	105	.ASCII	/TLE/	
005661	122	122	055	.ASCII	/RR-/	
005664	040	106	105	.ASCII	/ FE/	
005667	116	103	105	.ASCII	/NCE/	
005672	055	120	117	.ASCII	/-PO/	
005675	123	124	040	.ASCII	/ST /	
005700	105	122	122	.ASCII	/ERR/	
005703	117	122	040	.ASCII	/OR /	
005706	101	124	040	.ASCII	/AT /	
005711	120	122	117	.ASCII	/PRO/	
005714	124	101	102	.ASCII	/TAB/	
005717	000			.ASCII	<00>	
005720	045	116	045	P.ACT:	.ASCII	/N#/
005723	101	044	106	.ASCII	/A#F/	
005726	124	114	105	.ASCII	/TLE/	
005731	122	122	055	.ASCII	/RR-/	
005734	040	102	101	.ASCII	/ BA/	
005737	104	040	120	.ASCII	/D P/	
005742	101	103	113	.ASCII	/ACK/	
005745	105	124	040	.ASCII	/ET /	
005750	104	105	121	.ASCII	/DEQ/	
005753	125	105	125	.ASCII	/UEU/	
005756	105	104	040	.ASCII	/ED /	
005761	101	124	040	.ASCII	/AT /	
005764	125	056	104	.ASCII	/U.D/	
005767	117	116	105	.ASCII	/ONE/	
005772	000	000		.ASCII	<00><00>	
005774	045	116	045	P.ACU:	.ASCII	/N#/
005777	101	044	106	.ASCII	/A#F/	
006002	124	114	105	.ASCII	/TLE/	
006005	122	122	055	.ASCII	/RR-/	
006010	040	125	116	.ASCII	/ UN/	
006013	105	130	120	.ASCII	/EXP/	
006016	114	101	111	.ASCII	/LAI/	
006021	116	105	104	.ASCII	/NED/	
006024	040	104	055	.ASCII	/ D /	
006027	120	122	117	.ASCII	/PRO/	
006032	103	040	123	.ASCII	/C S/	
006035	125	123	120	.ASCII	/USP/	
006040	105	116	123	.ASCII	/ENS/	
006043	111	117	116	.ASCII	/ION/	
006046	040	050	125	.ASCII	/ (U/	
006051	056	056	124	.ASCII	/. .T/	

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr 1984 13:19:09
5 Apr 1984 13:16:08

SEQ 0081
Page 37
VAX-11 B1199-16 V3 555
SPIDER\USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

006054	104	123	051		.ASCII	/DS/
006057	000				.ASCII	<00>
006060	045	116	045	P.ACW:	.ASCII	/M#
006063	101	044	106		.ASCII	/A#F/
006066	124	114	105		.ASCII	/TLE/
006071	122	122	055		.ASCII	/RR /
006074	040	104	125		.ASCII	/ DU/
006077	120	040	120		.ASCII	/P P/
006102	101	103	113		.ASCII	/ACK/
006105	105	124	040		.ASCII	/ET /
006110	104	055	121		.ASCII	/D-Q/
006113	040	106	101		.ASCII	/ FA/
006116	111	114	105		.ASCII	/ILE/
006121	104	040	050		.ASCII	/D (/
006124	130	106	103		.ASCII	/XFC/
006127	040	063	064		.ASCII	/ 34/
006132	057	063	065		.ASCII	<57>/35/
006135	051	000	000		.ASCII	/)/<00><00>
006140	045	116	045	P.ACW:	.ASCII	/M#
006143	101	044	106		.ASCII	/A#F/
006146	124	114	105		.ASCII	/TLE/
006151	122	122	055		.ASCII	/RR -/
006154	040	111	116		.ASCII	/ IN/
006157	103	117	116		.ASCII	/CON/
006162	123	111	123		.ASCII	/SIS/
006165	124	105	116		.ASCII	/TEN/
006170	103	131	040		.ASCII	/CY /
006173	101	124	040		.ASCII	/AT /
006176	125	056	110		.ASCII	/U.M/
006201	124	123	124		.ASCII	/TST/
006204	000	000			.ASCII	<00><00>
006206	045	116	045	P.ACX:	.ASCII	/M#
006211	101	044	106		.ASCII	/A#F/
006214	124	114	105		.ASCII	/TLE/
006217	122	122	055		.ASCII	/RR -/
006222	040	111	116		.ASCII	/ IN/
006225	103	117	116		.ASCII	/CON/
006230	123	111	123		.ASCII	/SIS/
006233	124	105	116		.ASCII	/TEN/
006236	103	131	040		.ASCII	/CY /
006241	101	124	040		.ASCII	/AT /
006244	125	056	123		.ASCII	/U.S/
006247	105	113	117		.ASCII	/EKO/
006252	000	000			.ASCII	<00><00>
006254	045	116	045	P.ACY:	.ASCII	/M#
006257	101	044	106		.ASCII	/A#F/
006262	124	114	105		.ASCII	/TLE/
006265	122	122	055		.ASCII	/RR /
006270	040	111	116		.ASCII	/ IN/
006273	103	117	116		.ASCII	/CO\ /
006276	123	111	123		.ASCII	/SIS/
006301	124	105	116		.ASCII	/TEN/
006304	103	131	040		.ASCII	/CY /
006307	101	124	040		.ASCII	/AT /
006312	125	056	103		.ASCII	/U.C/
006315	113	123	126		.ASCII	/KSV/
006320	000	000			.ASCII	<00><00>

006322	045	116	045	P.ACZ:	.ASCII	/#N#/ /A#F/ /TLE/ /RR-/ /D./ /OPC/ /D F/ /OUN/ /D I/ /LLE/ /GAL/ /OP/ /COD/ /E/<00><00>
006325	101	044	106		.ASCII	
006330	124	114	105		.ASCII	
006333	122	122	055		.ASCII	
006336	040	104	056		.ASCII	
006341	117	120	103		.ASCII	
006344	104	040	106		.ASCII	
006347	117	125	116		.ASCII	
006352	104	040	111		.ASCII	
006355	114	114	105		.ASCII	
006360	107	101	114		.ASCII	
006363	040	117	120		.ASCII	
006366	103	117	104		.ASCII	
006371	105	000	000	P.ADA:	.ASCII	/#N#/ /A#F/ /TLE/ /RR-/ /D./ /CSF/ /FO/ /UND/ /IL/ /LEG/ /AL / /OPC/ /ODE/ <00>
006374	045	116	045		.ASCII	
006377	101	044	106		.ASCII	
006402	124	114	105		.ASCII	
006405	122	122	055		.ASCII	
006410	040	104	056		.ASCII	
006413	103	123	106		.ASCII	
006416	040	106	117		.ASCII	
006421	125	116	104		.ASCII	
006424	040	111	114		.ASCII	
006427	114	105	107		.ASCII	
006432	101	114	040		.ASCII	
006435	117	120	103		.ASCII	
006440	117	104	105		.ASCII	
006443	000				.ASCII	
006444	045	116	045	P.ADB:	.ASCII	/#N#/ /A#F/ /TLE/ /RR-/ /UN/ /KNO/ /WN / /BAD/ /DR/ /IVE/ /ST/ /ATU/ /S A/ /T D/ /DS/ /TS/<00>
006447	101	044	106		.ASCII	
006452	124	114	105		.ASCII	
006455	122	122	055		.ASCII	
006460	040	125	116		.ASCII	
006463	113	116	117		.ASCII	
006466	127	116	040		.ASCII	
006471	102	101	104		.ASCII	
006474	040	104	122		.ASCII	
006477	111	126	105		.ASCII	
006502	040	123	124		.ASCII	
006505	101	124	125		.ASCII	
006510	123	040	101		.ASCII	
006513	124	040	104		.ASCII	
006516	056	104	123		.ASCII	
006521	124	123	000		.ASCII	
006524	045	116	045	P.ADC:	.ASCII	/#N#/ /A#F/ /TLE/ /RR-/ /IL/ /LEG/ /AL / /XFC/ /EX/ /ECU/ /TED/ /BY/ /DM/
006527	101	044	106		.ASCII	
006532	124	114	105		.ASCII	
006535	122	122	055		.ASCII	
006540	040	111	114		.ASCII	
006543	114	105	107		.ASCII	
006546	101	114	040		.ASCII	
006551	130	106	103		.ASCII	
006554	040	105	130		.ASCII	
006557	105	103	125		.ASCII	
006562	124	105	104		.ASCII	
006565	040	102	131		.ASCII	
006570	040	104	115		.ASCII	

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0083
Page 39
VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

006573	000				.ASCII	<00>
006574	045	116	045	P.ADD:	.ASCII	/N# /
006577	101	044	106		.ASCII	/A\$F /
006602	124	114	105		.ASCII	/TLE /
006605	122	122	055		.ASCII	/RR- /
006610	040	104	040		.ASCII	/ D /
006613	120	111	103		.ASCII	/PIC /
006616	113	105	104		.ASCII	/KED /
006621	040	125	120		.ASCII	/ UP /
006624	040	101	040		.ASCII	/ A /
006627	132	105	122		.ASCII	/ZER /
006632	117	040	123		.ASCII	/O S /
006635	103	102	056		.ASCII	/CB. /
006640	104	102	000		.ASCII	/DB/<00>
006643	000				.ASCII	<00>
006644	045	116	045	P.ADE:	.ASCII	/N# /
006647	101	044	106		.ASCII	/A\$F /
006652	124	114	105		.ASCII	/TLE /
006655	122	122	055		.ASCII	/RR- /
006660	040	111	116		.ASCII	/ IN /
006663	103	117	116		.ASCII	/CON /
006666	123	111	123		.ASCII	/SIS /
006671	124	105	116		.ASCII	/TEN /
006674	103	131	040		.ASCII	/CY /
006677	101	124	040		.ASCII	/AT /
006702	104	040	111		.ASCII	/D I /
006705	104	114	105		.ASCII	/DLE /
006710	040	114	117		.ASCII	/ LO /
006713	117	120	000		.ASCII	/OP/<00>
006716	045	116	045	P.ADF:	.ASCII	/N# /
006721	101	044	106		.ASCII	/A\$F /
006724	124	114	105		.ASCII	/TLE /
006727	122	122	055		.ASCII	/RR- /
006732	040	104	115		.ASCII	/ DM /
006735	040	127	117		.ASCII	/ WO /
006740	122	104	040		.ASCII	/RD /
006743	103	117	125		.ASCII	/COU /
006746	116	124	040		.ASCII	/NT /
006751	105	122	122		.ASCII	/ERR /
006754	117	122	040		.ASCII	/OR /
006757	117	116	040		.ASCII	/ON /
006762	110	117	123		.ASCII	/HOS /
006765	124	040	104		.ASCII	/T D /
006770	115	101	057		.ASCII	/MA/<57>
006773	123	105	116		.ASCII	/SEN /
006776	104	057	122		.ASCII	/D/<57>/R /
007001	105	103	126		.ASCII	/ECV /
007004	000	000			.ASCII	<00><00>
007006	045	116	045	P.ADG:	.ASCII	/N# /
007011	101	044	106		.ASCII	/A\$F /
007014	124	114	105		.ASCII	/TLE /
007017	122	122	055		.ASCII	/RR- /
007022	040	125	116		.ASCII	/ UN /
007025	113	116	117		.ASCII	/KNO /
007030	127	116	040		.ASCII	/WN /
007033	104	111	123		.ASCII	/DIS /
007036	120	114	101		.ASCII	/PLA /

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0084
Page 40
VAX-11 B119-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB2.B16:5 (13)

007041	131	040	106	.ASCII	/Y F/	
007044	101	125	114	.ASCII	/AUL/	
007047	124	040	103	.ASCII	/T C/	
007052	117	104	105	.ASCII	/ODE/	
007055	040	101	124	.ASCII	/ AT/	
007060	040	104	056	.ASCII	/ D./	
007063	104	106	114	.ASCII	/DFL/	
007066	124	000		.ASCII	/T/<00>	
007070	045	116	045	P.ADH:	.ASCII	/N#/
007073	101	044	106	.ASCII	/A\$F/	
007076	124	114	105	.ASCII	/TLE/	
007101	122	122	055	.ASCII	/RR-/	
007104	040	104	122	.ASCII	/ DR/	
007107	111	126	105	.ASCII	/IVE/	
007112	040	116	117	.ASCII	/ NO/	
007115	124	040	106	.ASCII	/T F/	
007120	101	125	114	.ASCII	/AUL/	
007123	124	111	116	.ASCII	/TIN/	
007126	107	040	111	.ASCII	/G I/	
007131	116	040	120	.ASCII	/N P/	
007134	056	117	106	.ASCII	/.OF/	
007137	114	116	040	.ASCII	/LN /	
007142	123	124	101	.ASCII	/STA/	
007145	124	105	000	P.ADI:	.ASCII	/TE/<00>
007150	045	116	045	.ASCII	/N#/	
007153	101	044	106	.ASCII	/A\$F/	
007156	124	114	105	.ASCII	/TLE/	
007161	122	122	055	.ASCII	/RR-/	
007164	040	125	040	.ASCII	/ U /	
007167	120	117	127	.ASCII	/POW/	
007172	105	122	040	.ASCII	/ER /	
007175	125	120	040	.ASCII	/UP /	
007200	104	111	101	.ASCII	/DIA/	
007203	107	116	117	.ASCII	/GNO/	
007206	123	124	111	.ASCII	/STI/	
007211	103	123	040	.ASCII	/CS /	
007214	106	101	111	.ASCII	/FAI/	
007217	114	105	104	.ASCII	/LED/	
007222	000	000		P.ADJ:	.ASCII	<00><00>
007224	045	116	045	.ASCII	/N#/	
007227	101	044	106	.ASCII	/A\$F/	
007232	124	114	105	.ASCII	/TLE/	
007235	122	122	055	.ASCII	/RR-/	
007240	040	104	040	.ASCII	/ D /	
007243	120	117	127	.ASCII	/POW/	
007246	105	122	040	.ASCII	/ER /	
007251	125	120	040	.ASCII	/UP /	
007254	104	111	101	.ASCII	/DIA/	
007257	107	116	117	.ASCII	/GNO/	
007262	123	124	111	.ASCII	/STI/	
007265	103	123	040	.ASCII	/CS /	
007270	106	101	111	.ASCII	/FAI/	
007273	114	105	104	.ASCII	/LED/	
007276	000	000		P.ADK:	.ASCII	<00><00>
007300	045	116	045	.ASCII	/N#/	
007303	101	044	106	.ASCII	/A\$F/	
007306	124	114	105	.ASCII	/TLE/	

ZRCH82
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5 Apr-1984 13:16:08

SEQ 0085
Page 41
VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCH82.B16;5 (13)

007311	122	122	055	.ASCII	/RR-/
007314	040	101	104	.ASCII	/ AD/
007317	101	120	124	.ASCII	/APT/
007322	105	122	040	.ASCII	/ER /
007325	103	101	122	.ASCII	/CAR/
007330	104	040	106	.ASCII	/D F/
007333	101	111	114	.ASCII	/AIL/
007336	125	122	105	.ASCII	/URE/
007341	000			.ASCII	<00>
007342	045	116	045	P.ADL:	.ASCII / NS /
007345	101	044	106		.ASCII /A\$F/
007350	124	114	105		.ASCII /TLE/
007353	122	122	055		.ASCII /RR-/
007356	040	105	103		.ASCII / EC/
007361	056	124	115		.ASCII /.TM/
007364	122	040	124		.ASCII /R T/
007367	111	115	105		.ASCII /IME/
007372	104	040	117		.ASCII /D O/
007375	125	124	000	P.ADM:	.ASCII /UT/<00>
007400	045	116	045		.ASCII / NS /
007403	101	044	106		.ASCII /A\$F/
007406	124	114	105		.ASCII /TLE/
007411	122	122	055		.ASCII /RR-/
007414	040	125	056		.ASCII / U./
007417	123	105	116		.ASCII /SEN/
007422	104	057	125		.ASCII /D/<57>/U/
007425	056	122	105		.ASCII /.RE/
007430	103	126	040		.ASCII /CV /
007433	122	111	116		.ASCII /RIN/
007436	107	040	122		.ASCII /G R/
007441	105	101	104		.ASCII /EAD/
007444	040	111	116		.ASCII / IN/
007447	103	117	116		.ASCII /CON/
007452	123	111	123		.ASCII /SIS/
007455	124	105	116		.ASCII /TEN/
007460	103	131	000		.ASCII /CY/<00>
007463	000				.ASCII <00>
007464	045	116	045	P.ADN:	.ASCII / NS /
007467	101	044	106		.ASCII /A\$F/
007472	124	114	105		.ASCII /TLE/
007475	122	122	055		.ASCII /RR-/
007500	040	125	116		.ASCII / UN/
007503	113	116	117		.ASCII /KNO/
007506	127	116	040		.ASCII /WN /
007511	127	101	111		.ASCII /WAI/
007514	124	122	126		.ASCII /TRV/
007517	040	122	105		.ASCII / RE/
007522	101	123	117		.ASCII /ASO/
007525	116	040	101		.ASCII /N A/
007530	124	040	104		.ASCII /T D/
007533	056	122	126		.ASCII /.RV/
007536	103	124	000		.ASCII /CT/<00>
007541	000				.ASCII <00>
007542	045	116	045	P.ADO:	.ASCII / NS /
007545	101	044	106		.ASCII /A\$F/
007550	124	114	105		.ASCII /TLE/
007553	122	122	055		.ASCII /RR /

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr-1984 13:19:09
5 Apr 1984 13:16:08

SEQ 0086
Page 42
VAX-11 B111-16 V3 555
SPIDER#USERS:(NEALE.AZTEC)ZRCHB2.B16:5 (13)

007556	040	104	056	.ASCII	/ D /
007561	101	122	103	.ASCII	/ARC/
007564	123	040	104	.ASCII	/S D/
007567	111	104	040	.ASCII	/ID /
007572	116	117	124	.ASCII	/NOT/
007575	040	106	111	.ASCII	/ FI/
007600	116	104	040	.ASCII	/ND /
007603	103	114	117	.ASCII	/CLO/
007606	123	105	123	.ASCII	/SES/
007611	124	040	125	.ASCII	/T U/
007614	116	104	117	.ASCII	/NDO/
007617	116	105	040	.ASCII	/NE /
007622	132	117	116	.ASCII	/ZON/
007625	105	000	000	.ASCII	/E/<00><00>
007630	045	116	045	P.ADP: .ASCII	/NMS/
007633	101	044	106	.ASCII	/A#F/
007636	124	114	105	.ASCII	/TLE/
007641	122	122	055	.ASCII	/RR-/
007644	040	125	056	.ASCII	/ U /
007647	123	105	105	.ASCII	/SEE/
007652	113	040	106	.ASCII	/K F/
007655	117	125	116	.ASCII	/OUN/
007660	104	040	123	.ASCII	/D S/
007663	105	105	113	.ASCII	/EEK/
007666	040	124	117	.ASCII	/ TO/
007671	040	111	114	.ASCII	/ IL/
007674	114	105	107	.ASCII	/LEG/
007677	101	114	040	.ASCII	/AL /
007702	124	122	101	.ASCII	/TRA/
007705	103	113	000	.ASCII	/CK/<00>
007710	045	116	045	P.ADQ: .ASCII	/NMS/
007713	101	044	106	.ASCII	/A#F/
007716	124	114	105	.ASCII	/TLE/
007721	122	122	055	.ASCII	/RR-/
007724	040	125	056	.ASCII	/ U /
007727	110	124	123	.ASCII	/HTS/
007732	124	040	111	.ASCII	/T I/
007735	116	111	124	.ASCII	/NIT/
007740	040	104	111	.ASCII	/ DI/
007743	101	107	040	.ASCII	/AG /
007746	104	115	101	.ASCII	/DMA/
007751	040	127	122	.ASCII	/WR/
007754	111	124	105	.ASCII	/ITE/
007757	040	106	101	.ASCII	/ FA/
007762	111	114	105	.ASCII	/ILE/
007765	104	000	000	.ASCII	/D/<00><00>
007770	045	116	045	P.ADR: .ASCII	/NMS/
007773	101	044	106	.ASCII	/A#F/
007776	124	114	105	.ASCII	/TLE/
010001	122	122	055	.ASCII	/RR /
010004	040	125	056	.ASCII	/ U /
010007	110	124	123	.ASCII	/HTS/
010012	124	040	111	.ASCII	/T I/
010015	116	111	124	.ASCII	/NIT/
010020	040	104	111	.ASCII	/ DI/
010023	101	107	040	.ASCII	/AG /
010026	104	115	101	.ASCII	/DMA/

J7

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr 1984 13:19:09
5 Apr 1984 13:16:08

SEQ 0087
Page 43
VAX 11 Bliss 16 V3 555
SPIDER\$USERS:[NFALE,AZTEC]ZRCHB2.B16;5 (13)

010031	040	103	117
010034	115	120	101
010037	122	105	040
010042	106	101	111
010045	114	105	104
010050	000	000	
010052	045	116	045
010055	101	044	106
010060	124	114	105
010063	122	122	055
010066	040	125	056
010071	123	131	104
010074	122	040	106
010077	117	125	116
010102	104	040	123
010105	123	056	104
010110	105	122	040
010113	123	105	124
010116	040	101	116
010121	104	040	123
010124	123	056	123
010127	120	116	040
010132	116	117	124
010135	040	123	105
010140	124	000	
010142	045	116	045
010145	101	044	106
010150	124	114	105
010153	122	122	055
010156	040	115	101
010161	123	124	105
010164	122	040	104
010167	122	111	126
010172	105	123	040
010175	101	103	114
010200	117	040	101
010203	123	123	105
010206	122	124	105
010211	104	000	000
010214	004736		
010216	005016		
010220	005064		
010222	005132		
010224	005200		
010226	005264		
010230	005350		
010232	005416		
010234	005464		
010236	005532		
010240	005600		
010242	005650		
010244	005720		
010246	005774		
010250	006060		
010252	006140		
010254	006206		
010256	006254		

	.ASCII	/ CO/
	.ASCII	/MPA/
	.ASCII	/RE /
	.ASCII	/FAI/
	.ASCII	/LED/
	.ASCII	<00><00>
P.ADS:	.ASCII	/NMB/
	.ASCII	/A#F/
	.ASCII	/TLE/
	.ASCII	/RR-/
	.ASCII	/ U./
	.ASCII	/SYD/
	.ASCII	/R F/
	.ASCII	/OUN/
	.ASCII	/D S/
	.ASCII	/S.D/
	.ASCII	/ER /
	.ASCII	/SET/
	.ASCII	/ AN/
	.ASCII	/D S/
	.ASCII	/S.S/
	.ASCII	/PN /
	.ASCII	/NOT/
	.ASCII	/ SE/
	.ASCII	/T/<00>
P.ADT:	.ASCII	/NMB/
	.ASCII	/A#F/
	.ASCII	/TLE/
	.ASCII	/RR-/
	.ASCII	/ MA/
	.ASCII	/STE/
	.ASCII	/R D/
	.ASCII	/RIV/
	.ASCII	/ES /
	.ASCII	/ACL/
	.ASCII	/O A/
	.ASCII	/SSE/
	.ASCII	/RTE/
	.ASCII	/D/<00><00>
P.ACG:	.WORD	P.ACH
	.WORD	P.ACI
	.WORD	P.ACJ
	.WORD	P.ACK
	.WORD	P.ACL
	.WORD	P.ACM
	.WORD	P.ACN
	.WORD	P.ACO
	.WORD	P.ACP
	.WORD	P.ACQ
	.WORD	P.ACR
	.WORD	P.ACS
	.WORD	P.ACT
	.WORD	P.ACU
	.WORD	P.ACV
	.WORD	P.ACW
	.WORD	P.ACX
	.WORD	P.ACY

010260	006322				.WORD	P.ACZ
010262	006374				.WORD	P.ADA
010264	006444				.WORD	P.ADB
010266	006524				.WORD	P.ADC
010270	006574				.WORD	P.ADD
010272	006644				.WORD	P.ADE
010274	006716				.WORD	P.ADF
010276	007006				.WORD	P.ADG
010300	007070				.WORD	P.ADH
010302	007150				.WORD	P.ADI
010304	007224				.WORD	P.ADJ
010306	007300				.WORD	P.ADK
010310	007342				.WORD	P.ADL
010312	007400				.WORD	P.ADM
010314	007464				.WORD	P.ADN
010316	007542				.WORD	P.ADO
010320	007630				.WORD	P.ADP
010322	007710				.WORD	P.ADQ
010324	007770				.WORD	P.ADR
010326	010052				.WORD	P.ADS
010330	010142				.WORD	P.ADT
010332	045	101	040	P.ADV:	.ASCII	/#A /
010335	123	125	103		.ASCII	/SUC/
010340	103	105	123		.ASCII	/CES/
010343	123	106	125		.ASCII	/SFU/
010346	114	045	116		.ASCII	/L#N/
010351	000				.ASCII	<00>
010352	045	101	111	P.ADW:	.ASCII	/#AI/
010355	116	126	101		.ASCII	/NVA/
010360	114	111	104		.ASCII	/LID/
010363	040	103	117		.ASCII	/ CO/
010366	115	115	101		.ASCII	/MMA/
010371	116	104	045		.ASCII	/ND# /
010374	116	000			.ASCII	/N/<00>
010376	045	101	116	P.ADX:	.ASCII	/#AN/
010401	117	040	122		.ASCII	/O R/
010404	105	107	111		.ASCII	/EGI/
010407	117	116	040		.ASCII	/ON /
010412	101	126	101		.ASCII	/AVA/
010415	111	114	101		.ASCII	/ILA/
010420	102	114	105		.ASCII	/BLE/
010423	045	116	000		.ASCII	/#N/<00>
010426	045	101	116	P.ADY:	.ASCII	/#AN/
010431	117	040	122		.ASCII	/O R/
010434	105	107	111		.ASCII	/EGI/
010437	117	116	040		.ASCII	/ON /
010442	123	125	111		.ASCII	/SUI/
010445	124	101	102		.ASCII	/TAB/
010450	114	105	045		.ASCII	/LE# /
010453	116	000	000		.ASCII	/N/<00><00>
010456	045	101	120	P.ADZ:	.ASCII	/#AP/
010461	122	117	107		.ASCII	/ROG/
010464	122	101	115		.ASCII	/RAM/
010467	040	116	117		.ASCII	/ NO/
010472	124	040	113		.ASCII	/T K/
010475	116	117	127		.ASCII	/NOW/
010500	116	045	116		.ASCII	/##N/

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0089
Page 45
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

010503	000				.ASCII	<00>
010504	045	101	114	P.AEA:	.ASCII	/AL/
010507	117	101	104		.ASCII	/OAD/
010512	040	106	101		.ASCII	/FA/
010515	111	114	125		.ASCII	/ILU/
010520	122	105	045		.ASCII	/RE/
010523	116	000	000		.ASCII	/N/<00><00>
010526	045	101	123	P.AEB:	.ASCII	/AS/
010531	124	101	116		.ASCII	/TAN/
010534	104	101	114		.ASCII	/DAL/
010537	117	116	105		.ASCII	/ONE/
010542	045	116	000		.ASCII	/N/<00>
010545	000				.ASCII	<00>
010546	010332			P.ADU:	.WORD	P.ADV
010550	010352				.WORD	P.ADW
010552	010376				.WORD	P.ADX
010554	010426				.WORD	P.ADY
010556	010456				.WORD	P.ADZ
010560	010504				.WORD	P.AEA
010562	010526				.WORD	P.AEB
010564	045	101	123	P.AED:	.ASCII	/AS/
010567	125	103	103		.ASCII	/UCC/
010572	105	123	123		.ASCII	/ESS/
010575	045	116	000		.ASCII	/N/<00>
010600	045	101	111	P.AEE:	.ASCII	/AI/
010603	116	126	101		.ASCII	/NVA/
010606	114	111	104		.ASCII	/LID/
010611	040	103	117		.ASCII	/CO/
010614	115	115	101		.ASCII	/MMA/
010617	116	104	045		.ASCII	/ND/
010622	116	000			.ASCII	/N/<00>
010624	045	101	103	P.AEF:	.ASCII	/AC/
010627	117	115	115		.ASCII	/OMM/
010632	101	116	104		.ASCII	/AND/
010635	040	101	102		.ASCII	/AB/
010640	117	122	124		.ASCII	/ORT/
010643	105	104	045		.ASCII	/ED/
010646	116	000			.ASCII	/N/<00>
010650	045	101	125	P.AEG:	.ASCII	/AU/
010653	116	111	124		.ASCII	/NIT/
010656	055	117	106		.ASCII	/-OF/
010661	106	114	111		.ASCII	/FLI/
010664	116	105	045		.ASCII	/NE/
010667	116	000	000		.ASCII	/N/<00><00>
010672	045	101	125	P.AEH:	.ASCII	/AU/
010675	116	111	124		.ASCII	/NIT/
010700	055	101	126		.ASCII	/-AV/
010703	101	111	114		.ASCII	/AIL/
010706	101	102	114		.ASCII	/ABL/
010711	105	045	116		.ASCII	/EN/
010714	000	000			.ASCII	<00><00>
010716	045	101	115	P.AEI:	.ASCII	/AM/
010721	105	104	111		.ASCII	/EDI/
010724	101	040	106		.ASCII	/AF/
010727	117	122	115		.ASCII	/ORM/
010732	101	124	040		.ASCII	/AT/
010735	105	122	122		.ASCII	/ERR/

ZRCMB2
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION5-Apr-1984 13:19:09
5-Apr-1984 13:16:08SEQ 0090
Page 46
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NFALE.AZTEC]ZRCMB2.B16;5 (13)

010740	117	122	045		.ASCII	/OR#/ </td></tr>
--------	-----	-----	-----	--	--------	---------------------

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

VAX-11 Bliss-16 V3 555

SEQ 0091

Page 47

SPIDER\$USERS:[NEALE.AZTEC;ZRCHB2.B16;5 (13)

011204	103	045	116		.ASCII	/C#N/
011207	000				.ASCII	<00>
011210	010564'			P.AEC:	.WORD	P.AED
011212	010600'				.WORD	P.AEE
011214	010624'				.WORD	P.AEF
011216	010650'				.WORD	P.AEG
011220	010672'				.WORD	P.AEH
011222	010716'				.WORD	P.AEI
011224	010746'				.WORD	P.AEJ
011226	010772'				.WORD	P.AEK
011230	011014'				.WORD	P.AEL
011232	011034'				.WORD	P.AEM
011234	011072'				.WORD	P.AEN
011236	011120'				.WORD	P.AEO
011240	011140'				.WORD	P.AEP
011242	105	116	124	P.AEQ:	.ASCII	/ENT/
011245	105	122	040		.ASCII	/ER /
011250	104	101	124		.ASCII	/DAT/
011253	105	040	074		.ASCII	/E </
011256	115	115	055		.ASCII	/MM-/
011261	104	104	055		.ASCII	/DD-/
011264	131	131	131		.ASCII	/YYY/
011267	131	076	040		.ASCII	/Y> /
011272	000	000			.ASCII	<00><00>
011274	045	116	045	P.AER:	.ASCII	/#N#/
011277	101	116	117		.ASCII	/ANO/
011302	040	101	104		.ASCII	/ AD/
011305	104	111	124		.ASCII	/DIT/
011310	111	117	116		.ASCII	/ION/
011313	101	114	040		.ASCII	/AL /
011316	125	116	111		.ASCII	/UNI/
011321	124	123	040		.ASCII	/TS /
011324	124	117	040		.ASCII	/TO /
011327	106	117	122		.ASCII	/FOR/
011332	115	101	124		.ASCII	/MAT/
011335	040	055	040		.ASCII	/ - /
011340	101	102	117		.ASCII	/ABO/
011343	122	124	111		.ASCII	/RTI/
011346	116	107	000		.ASCII	/NG/<00>
011351	000				.ASCII	<00>
011352	045	116	045	P.AES:	.ASCII	/#N#/
011355	101	044	106		.ASCII	/A#F/
011360	124	114	105		.ASCII	/TLE/
011363	122	122	055		.ASCII	/RR-/
011366	040	111	116		.ASCII	/ IN/
011371	111	124	040		.ASCII	/IT /
011374	103	117	104		.ASCII	/COD/
011377	105	040	122		.ASCII	/E R/
011402	105	055	105		.ASCII	/E-E/
011405	116	124	105		.ASCII	/NTE/
011410	122	105	104		.ASCII	/RED/
011413	040	104	125		.ASCII	/ DU/
011416	105	040	124		.ASCII	/E T/
011421	117	040	120		.ASCII	/O P/
011424	127	122	040		.ASCII	/WR /
011427	106	101	111		.ASCII	/FAI/
011432	114	000			.ASCII	/L/<00>

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr-1984 13:19:09
5 Apr-1984 13:16:08

SEQ 0092
Page 48
VAX-11 B1100-16 V3-555
SPIDER\USERS:(NEALE.AZTEC)\ZRCHB2.B16;5 (13)

011434	045	116	045	P.AET:	.ASCII	/#N#/
011437	101	044	106		.ASCII	/A#F/
011442	124	114	105		.ASCII	/TLE/
011445	122	122	055		.ASCII	/RR-/
011450	040	101	102		.ASCII	/AB/
011453	117	122	124		.ASCII	/ORT/
011456	111	116	107		.ASCII	/ING/
011461	040	110	117		.ASCII	/HO/
011464	123	124	040		.ASCII	/ST /
011467	101	116	104		.ASCII	/AND/
011472	040	122	105		.ASCII	/RE/
011475	115	117	124		.ASCII	/MOT/
011500	105	040	120		.ASCII	/E P/
011503	122	117	107		.ASCII	/ROG/
011506	122	101	115		.ASCII	/RAM/
011511	123	000	000		.ASCII	/S/<00><00>
011514	045	116	045	P.AEU:	.ASCII	/#N#/
011517	101	044	106		.ASCII	/A#F/
011522	124	114	105		.ASCII	/TLE/
011525	122	122	055		.ASCII	/RR-/
011530	040	111	114		.ASCII	/IL/
011533	114	105	107		.ASCII	/LEG/
011536	101	114	040		.ASCII	/AL /
011541	116	125	115		.ASCII	/NUM/
011544	102	105	122		.ASCII	/BER/
011547	040	117	106		.ASCII	/OF/
011552	040	125	116		.ASCII	/UN/
011555	111	124	123		.ASCII	/ITS/
011560	040	123	105		.ASCII	/SE/
011563	114	105	103		.ASCII	/LEC/
011566	124	105	104		.ASCII	/TED/
011571	000				.ASCII	<00>
011572	045	116	045	P.AEV:	.ASCII	/#N#/
011575	101	044	106		.ASCII	/A#F/
011600	124	114	105		.ASCII	/TLE/
011603	122	122	055		.ASCII	/RR-/
011606	040	114	111		.ASCII	/LI/
011611	115	111	124		.ASCII	/MIT/
011614	040	117	106		.ASCII	/OF/
011617	040	123	111		.ASCII	/SI/
011622	130	124	105		.ASCII	/XTE/
011625	105	116	040		.ASCII	/EN /
011630	125	116	111		.ASCII	/UNI/
011633	124	123	040		.ASCII	/TS /
011636	120	105	122		.ASCII	/PER/
011641	040	106	117		.ASCII	/FO/
011644	122	115	101		.ASCII	/RMA/
011647	124	111	116		.ASCII	/TIN/
011652	107	040	123		.ASCII	/G S/
011655	105	123	123		.ASCII	/ESS/
011660	111	117	116		.ASCII	/ION/
011663	000				.ASCII	<00>
011664	045	116	045	P.AEW:	.ASCII	/#N#/
011667	101	044	106		.ASCII	/A#F/
011672	124	114	105		.ASCII	/TLE/
011675	122	122	055		.ASCII	/RR /
011700	040	122	103		.ASCII	/RC/

ZRCMB2
REV B PATCH 00

ZRCMB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr 1984 13:19:09
5 Apr 1984 13:16:08

SEQ 0093
Page 49
VAX-11 B110-16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCMB2.B16;5(13)

011703	062	065	040	.ASCII	/25 /	
011706	103	117	116	.ASCII	/CON/	
011711	124	122	117	.ASCII	/TRO/	
011714	114	114	105	.ASCII	/LLE/	
011717	122	040	111	.ASCII	/R I/	
011722	116	111	124	.ASCII	/NIT/	
011725	111	101	114	.ASCII	/IAL/	
011730	111	132	101	.ASCII	/IZA/	
011733	124	111	117	.ASCII	/TIO/	
011736	116	040	105	.ASCII	/N E/	
011741	122	122	117	.ASCII	/RRO/	
011744	122	000		.ASCII	/R/<00>	
011746	045	116	045	P.AEX:	.ASCII	/NNS/
011751	101	044	106	.ASCII	/A#F/	
011754	124	114	105	.ASCII	/TLE/	
011757	122	122	055	.ASCII	/RR-/	
011762	040	120	122	.ASCII	/ PR/	
011765	117	124	117	.ASCII	/OTO/	
011770	103	117	114	.ASCII	/COL/	
011773	040	126	111	.ASCII	/ VI/	
011776	117	114	101	.ASCII	/OLA/	
012001	124	111	117	.ASCII	/TIO/	
012004	116	040	105	.ASCII	/N E/	
012007	122	122	117	.ASCII	/RRO/	
012012	122	000		.ASCII	/R/<00>	
012014	045	116	045	P.AEY:	.ASCII	/NNS/
012017	101	044	106	.ASCII	/A#F/	
012022	124	114	105	.ASCII	/TLE/	
012025	122	122	055	.ASCII	/RR-/	
012030	040	103	117	.ASCII	/ CO/	
012033	115	115	125	.ASCII	/MMU/	
012036	116	111	103	.ASCII	/NIC/	
012041	101	124	111	.ASCII	/ATI/	
012044	117	116	040	.ASCII	/ON /	
012047	101	122	105	.ASCII	/ARE/	
012052	101	040	111	.ASCII	/A I/	
012055	116	111	124	.ASCII	/NIT/	
012060	040	105	122	.ASCII	/ ER/	
012063	122	117	122	.ASCII	/ROR/	
012066	000	000		.ASCII	<00><00>	
012070	045	116	045	P.AEZ:	.ASCII	/NNS/
012073	101	044	106	.ASCII	/A#F/	
012076	124	114	105	.ASCII	/TLE/	
012101	122	122	055	.ASCII	/RR-/	
012104	040	104	125	.ASCII	/ DU/	
012107	120	040	123	.ASCII	/P S/	
012112	105	122	126	.ASCII	/ERV/	
012115	105	122	040	.ASCII	/ER /	
012120	101	103	124	.ASCII	/ACT/	
012123	111	126	105	.ASCII	/IVE/	
012126	040	101	106	.ASCII	/ AF/	
012131	124	105	122	.ASCII	/TER/	
012134	040	111	116	.ASCII	/ IN/	
012137	111	124	111	.ASCII	/ITI/	
012142	101	114	111	.ASCII	/ALI/	
012145	132	111	116	.ASCII	/ZIN/	
012150	107	000		.ASCII	/G/<00>	

012152	045	116	045	P.AFA:	.ASCII	/M#
012155	101	044	106		.ASCII	/A#F
012160	124	114	105		.ASCII	/TLE
012163	122	122	055		.ASCII	/RR-
012166	040	104	125		.ASCII	/DU
012171	120	040	123		.ASCII	/P S
012174	105	122	126		.ASCII	/ERV
012177	105	122	040		.ASCII	/ER
012202	111	116	101		.ASCII	/INA
012205	103	124	111		.ASCII	/CTI
012210	126	105	040		.ASCII	/VE
012213	101	106	124		.ASCII	/AFT
012216	105	122	040		.ASCII	/ER
012221	105	130	137		.ASCII	/EX
012224	123	125	120		.ASCII	/SUP
012227	137	120	122		.ASCII	/PR
012232	117	107	040		.ASCII	/OG
012235	103	117	115		.ASCII	/COM
012240	115	101	116		.ASCII	/MAN
012243	104	000	000		.ASCII	/D/<00><00>
012246	105	116	124	P.AFB:	.ASCII	/ENT
012251	105	122	040		.ASCII	/ER
012254	106	103	124		.ASCII	/FCT
012257	040	106	111		.ASCII	/FI
012262	114	105	040		.ASCII	/LE
012265	116	101	115		.ASCII	/NAM
012270	105	040	124		.ASCII	/E T
012273	117	040	122		.ASCII	/O R
012276	105	123	124		.ASCII	/EST
012301	117	122	105		.ASCII	/ORE
012304	040	050	106		.ASCII	/ (F
012307	111	114	105		.ASCII	/ILE
012312	116	101	115		.ASCII	/NAM
012315	105	056	105		.ASCII	/E.E
012320	130	124	051		.ASCII	/XT)
012323	072	040	000		.ASCII	/: /<00>
012326	122	103	062	P.AFC:	.ASCII	/RC2
012331	065	040	040		.ASCII	/S
012334	111	120	040		.ASCII	/IP
012337	040	040	122		.ASCII	/ R
012342	105	107	111		.ASCII	/EGI
012345	123	124	105		.ASCII	/STE
012350	122	040	040		.ASCII	/R
012353	040	101	104		.ASCII	/ AD
012356	104	122	105		.ASCII	/DRE
012361	123	123	000		.ASCII	/SS/<00>
012364	122	103	062	P.AFD:	.ASCII	/RC2
012367	065	040	111		.ASCII	/S I
012372	116	124	105		.ASCII	/NTE
012375	122	122	125		.ASCII	/RRU
012400	120	124	040		.ASCII	/PT
012403	126	105	103		.ASCII	/VEC
012406	124	117	122		.ASCII	/TOR
012411	040	101	104		.ASCII	/ AD
012414	104	122	105		.ASCII	/DRE
012417	123	123	000		.ASCII	/SS/<00>
012422	122	103	062	P.AFE:	.ASCII	/RC2

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0095
Page 51
VAX-11 Bliss-16 V3 555
SPIDER#USERS:(NEALE.AZTEC)ZRCHB2.B16;5 (13)

012425	065	040	040	.ASCII	/5 /
012430	040	040	102	.ASCII	/ B/
012433	125	123	040	.ASCII	/US /
012436	040	040	122	.ASCII	/ R/
012441	105	121	125	.ASCII	/EQU/
012444	105	123	124	.ASCII	/EST/
012447	040	040	040	.ASCII	/ /
012452	114	105	126	.ASCII	/LEV/
012455	105	114	000	.ASCII	/EL/<00>
012460	125	116	111	P.AFF:	.ASCII /UNI/
012463	124	040	040	.ASCII	/T /
012466	116	125	115	.ASCII	/NUM/
012471	102	105	122	.ASCII	/BER/
012474	040	124	117	.ASCII	/ TO/
012477	040	102	122	.ASCII	/ BR/
012502	111	116	107	.ASCII	/ING/
012505	040	040	117	.ASCII	/ O/
012510	116	114	111	.ASCII	/NLI/
012513	116	105	000	.ASCII	/NE/<00>
012516	106	117	122	P.AFG:	.ASCII /FOR/
012521	115	101	124	.ASCII	/MAT/
012524	040	111	116	.ASCII	/ IN/
012527	040	125	116	.ASCII	/ UN/
012532	101	124	124	.ASCII	/ATT/
012535	105	116	104	.ASCII	/END/
012540	105	104	040	.ASCII	/ED /
012543	122	105	106	.ASCII	/REF/
012546	117	122	115	.ASCII	/ORM/
012551	101	124	040	.ASCII	/AT /
012554	115	117	104	.ASCII	/MOD/
012557	105	000	000	.ASCII	/E/<00><00>
012562	045	116	045	P.AFH:	.ASCII /#N#/
012565	101	052	052	.ASCII	/A**/
012570	052	052	052	.ASCII	/***/
012573	052	052	052	.ASCII	/***/
012576	052	052	052	.ASCII	/***/
012601	052	052	052	.ASCII	/***/
012604	052	052	052	.ASCII	/***/
012607	052	052	052	.ASCII	/***/
012612	052	040	116	.ASCII	/* N/
012615	117	124	111	.ASCII	/OTI/
012620	103	105	040	.ASCII	/CE /
012623	052	052	052	.ASCII	/***/
012626	052	052	052	.ASCII	/***/
012631	052	052	052	.ASCII	/***/
012634	052	052	052	.ASCII	/***/
012637	052	052	052	.ASCII	/***/
012642	052	052	052	.ASCII	/***/
012645	052	052	052	.ASCII	/***/
012650	000	000		.ASCII	<00><00>
012652	045	116	045	P.AFI:	.ASCII /#N#/
012655	101	040	040	.ASCII	/A /
012660	040	040	040	.ASCII	/ /
012663	117	120	105	.ASCII	/OPE/
012666	122	101	124	.ASCII	/RAT/
012671	117	122	040	.ASCII	/OR /
012674	115	125	123	.ASCII	/MUS/

ZRCHB2
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5-Apr 1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0096
Page 52
VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB2.B16;5 (13)

012677	124	040	102	.ASCII	/T B/
012702	105	040	120	.ASCII	/E P/
012705	122	105	123	.ASCII	/RES/
012710	105	116	124	.ASCII	/ENT/
012713	040	111	116	.ASCII	/ IN/
012716	040	101	124	.ASCII	/ AT/
012721	124	105	116	.ASCII	/TEN/
012724	104	105	104	.ASCII	/DED/
012727	040	115	117	.ASCII	/ MO/
012732	104	105	040	.ASCII	/DE /
012735	000			.ASCII	<00>
012736	045	116	045	P.AFJ: .ASCII	/N#/
012741	101	040	040	.ASCII	/A /
012744	040	040	040	.ASCII	/ /
012747	040	040	122	.ASCII	/ R/
012752	125	116	116	.ASCII	/UNN/
012755	111	116	107	.ASCII	/ING/
012760	040	111	116	.ASCII	/ IN/
012763	040	125	116	.ASCII	/ UN/
012766	055	101	124	.ASCII	/-AT/
012771	124	105	116	.ASCII	/TEN/
012774	104	105	104	.ASCII	/DED/
012777	040	122	105	.ASCII	/ RE/
013002	106	117	122	.ASCII	/FOR/
013005	115	101	124	.ASCII	/MAT/
013010	040	115	117	.ASCII	/ MO/
013013	104	105	000	.ASCII	/DE/<00>

000000	.PSECT	\$GLOB\$, RO , D , GBL
000000	COM.AREA::	
	.BLKW	24
000050	HEAD.AREA::	
	.BLKW	1
000052	RECEIVE.RING::	
	.BLKW	1
000054	SEND.RING::	
	.BLKW	1
000056	REC.ENVELOPE::	
	.BLKW	200
000456	SND.ENVELOPE::	
	.BLKW	130
000736	FCT.BUF::	
	.BLKW	400
001736	REC.BUF::	
	.BLKW	170
002316	SND.BUF::	
	.BLKW	45
002430	OUT\$STD.BUF::	
	.BLKW	10
002450	RET.EN\$AD::	
	.BLKW	1
002452	DATETXT::	
	.BLKW	6
002466	FLG.WRD::	
	.BLKW	1

ZRCH82
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
GLOBAL TEXT SECTION

5 Apr-1984 13:19:09
5-Apr-1984 13:16:08

SEQ 0097
Page 53
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH82.B16;5 (13)

002470		NEX.FLAG::	
		.BLKW	1
002472		OVSA::	.BLKW 1
002474		NXT.CRN::	
		.BLKB	1
		.EVEN	
002476	000000	RET.STATUS::	
		.WORD	0
002500		LUN::	.BLKW 1
002502		PID.SAVE::	
		.BLKW	2
002506		NSD.SLOT::	
		.BLKW	1
002510		NRD.SLOT::	
		.BLKW	1
002512		RC25.ADDR::	
		.BLKW	1
002514		VEC.ADDR::	
		.BLKW	1
002516		BR.LEVEL::	
		.BLKW	1
002520		UNIT.NO::	
		.BLKW	1
002522		PTBL.PTR::	
		.BLKW	1
002524	003777	RSVD.STRUCT::	
		.WORD	3777
002526	000000	.WORD	0
002530	003400	.WORD	3400
002532	003777	.WORD	3777
002534	007777	ISD.STRUCT::	
		.WORD	7777
002536	111033	.WORD	-66745
002540	010222	.WORD	10222
002542	000010	.WORD	RINGBASE
002544	023433	.WORD	23433
002546	000000	.WORD	0
002550	043777	.WORD	43777
002552	177400	.WORD	-400

.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

000126'	L\$ERRTBL==	ERRTYP
000154'	L\$SW==	L\$SWLEN+2
000140'	L\$HW==	L\$HWLEN+2
000011'	L\$DEPO==	L\$REV+1
000140'	DFPTBL==	L\$HWLEN+2
000154'	SFPTBL==	L\$SWLEN+2
000000'	FMT1==	P.AAA
000006'	FMT2==	P.AAB
000076'	FMT3==	P.AAC
000204'	FMT4==	P.AAD
000270'	FMT5==	P.AAE

000400'	FMT6==	P.AAF
000476'	FMT7==	P.AAG
000544'	PID.FMT==	P.AAH
000646'	CRLF==	P.AAI
000652'	XCRLF==	P.AAJ
000010'	RINGBASE==	COM.AREA+10
001740'	MSGADR==	REC.BUF+2
002660'	PFE.STRUCT==	P.AAK
004660'	EMSG.STRUCT==	P.ABI
010214'	RC.STRUCTURE==	P.ACG
010546'	SDUP.STRUCT==	P.ADU
011210'	SMSCP.STRUCT==	P.AEC
011242'	DATMSG==	P.AEQ
011274'	NO.ADD.UNITS==	P.AER
011352'	PWR.MSG==	P.AES
011434'	ABO.MSG==	P.AET
011514'	TO.MANY.UNITS==	P.AEU
011572'	GOOD.NUM.UNITS==	P.AEV
011664'	BOOT.FAILURE==	P.AEW
011746'	PROTO.VIOLATION==	P.AEX
012014'	PORT.INIT.ERR==	P.AEY
012070'	ACTIVE.DUP.SERVER==	P.AEZ
012152'	INACTIVE.DUP.SERVER==	P.AFA
012246'	FCT.REQ.MSG==	P.AFB
012326'	HW.Q1.IP==	P.AFC
012364'	HW.Q2.VECTOR==	P.AFD
012422'	HW.Q3.BR==	P.AFE
012460'	HW.Q4.UNIT==	P.AFF
012516'	SW.Q1.UNATT==	P.AFG
012562'	SW.Q2.NOTICE==	P.AFH
012652'	SW.Q3.OPER==	P.AFI
012736'	SW.Q4.UNATT==	P.AFJ

PSECT SUMMARY

:					
:					
:	Psect Name	Words	Attributes		
:	AA\$CODE	59	RO : I : LCL.	REL.	CON
:	\$GLOB\$	694	RO : D : GBL.	REL.	CON
:	\$PLIT\$	2823	RO : D : GBL.	REL.	CON
:					

LIBRARY STATISTICS

:					
:					
:	File	Total	Symbols Loaded	Percent	Blocks Read
:	SPIDER\$USERS:[NEALE.AZTEC]ZRCH80.L16;2	398	176	44	42
:					

I8

ZRCHB2 ZRCHB0 RC25 DISK FORMATTER
REV B PATCH 00 GLOBAL TEXT SECTION

5 Apr 1984 13:19:09
5 Apr 1984 13:16:08

SEQ 0099
Page 55
VAX-11 Bliss 16 V3 555
SPIDER#USERS:(NEALE.AZTEC)ZRCHB2.B16;5 (13)

COMMAND QUALIFIERS

:
: BLISS PDP11/LIST ZRCHB2.B16
: Size: 0 code , 3576 data words
: Run Time: 00:25.5
: Elapsed Time: 02:03.2
: Memory Used: 209 pages
: Comp lat on Complete

```

: 0001 MODULE ZRCMB3 (#TITLE 'ZRCMB0 RC25 DISK FORMATTER'
: 0002 IDENT = 'REV B PATCH 00',
: 0003 ADDRESSING MODE (RELATIVE) ,
: 0004 ENVIRONMENT (NOEIS)
: 0005 ) .
: 0006 BEGIN
: 0007 #sbt1 'MODULE DECLARATIONS'
: 0008 !
: 0009 ! Pretty Declarations
: 0010 !
: 0011 !<BLF/LOWERCASE_KEY>
: 0012 !
: 0013
: 0014 library 'ZRCMB0'; !Define RC25 Library module
: 0015
: 0016 require 'BLSMAC.REQ'; !Define Bliss Macro Library
: 1505
: 1506 !.
: 1507 ! Structure declarations used within this module.
: 1508 !-
: 1509
: 1510 structure
: 1511
: 1512 !.
: 1513 ! RC25 register accessing structure. This
: 1514 ! structure allows RC25 register accessing
: 1515 ! to be transportable between the PDP-11 and
: 1516 ! VAX Diagnostic Supervisors.
: 1517 !
: 1518 ! This also defines an access algorithm for
: 1519 ! VAX to allow field reference to MBA address
: 1520 ! space without generating machine checks.
: 1521 !-
: 1522
: 1523 RC25 (O, P, S, E) =
: 1524 begin
: 1525
: 1526 local
: 1527 RC%S_REG;
: 1528
: 1529 RC%S_REG = .(RC25 * #upval+0)<0, #bpval, 0>;
: 1530 RC%S_REG
: 1531 end
: 1532 <P, S, E>;
: 1533
: 1534 !<blf/page>

```

ZRCH83
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:33:45
5-Apr-1984 13:31:29

VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCH83.B16;5 (2)

SEQ 0101

Page 2

```
: 1535 !*
: 1536 ! The psect named "code or $code$" is redefined here
: 1537 ! to be called "aa$code". This is done to force the tkb
: 1538 ! linker to place the programs header information starting
: 1539 ! at absolute address 2000. Then for consistency "a$code"
: 1540 ! is used in place of "code or $code" across all modules.
: 1541 !
: 1542 psect
: 1543     code = aa$code;
: 1544
: 1545 !*
: 1546 ! External Routine declared outside this module.
: 1547 !-
: 1548
: 1549 external routine
: 1550     DECODE : novalue,
: 1551     LOAD_FILE;
: 1552
: 1553 !<blf/page>
```

```

: 1554 !*
: 1555 ! External Declaration of datums declared outside of this module.
: 1556 !-
: 1557
: 1558 external
: 1559 |
: 1560 | : DM load file from local media definitions
: 1561 | :
: 1562 | :DM_FN$EXT, !DM file name ascii text
: 1563 | AZFMTR : vector [8989, word], !DM host buffer adrs
: 1564 | OVSA : word, !Overlay section starting adrs
: 1565 | :
: 1566 | : Hardware question ascii string messages
: 1567 | :
: 1568 | HW_Q1_IP, !H/W question 1 for IP reg address
: 1569 | HW_Q2_VECTOR, !H/W question 2 for interrupt vector address
: 1570 | HW_Q3_BR, !H/W question 3 for bus req level
: 1571 | HW_Q4_UNIT, !H/W question 4 for unit no. to format
: 1572 | :
: 1573 | : Software question ascii string messages
: 1574 | :
: 1575 | SW_Q1_UNATT, !Unattended reformat software question
: 1576 | SW_Q2_NOTICE, !Notice message
: 1577 | SW_Q3_OPER, !Operator must be present in this mode
: 1578 | SW_Q4_UNATT, !Running in unattended mode
: 1579 | :
: 1580 | : Formatting print string
: 1581 | :
: 1582 | FMT2, !Notifies unit being formatted
: 1583 | FMT3, !Notifies format abort
: 1584 | CRLF, !<CR><LF>
: 1585 | XCRLF, !Prints ten <CR><LF>
: 1586 | :
: 1587 | : Init code error and informational messages
: 1588 | :
: 1589 | NO_ADD_UNITS,
: 1590 | PWR_MSG,
: 1591 | ABO_MSG,
: 1592 | TO_MANY_UNITS,
: 1593 | GOOD_NUM_UNITS,
: 1594 | DATMSG, !Message to ask operator for date
: 1595 | :
: 1596 | : Miscellaneous external data declarations
: 1597 | :
: 1598 | FLG_WRD : bitvector [16], !Global flag word
: 1599 | LUN : byte,
: 1600 | PTBL_PTR : ref vector [4, word],
: 1601 | DATE_TXT : vector [12, byte], !Storage for operator date input
: 1602 | :
: 1603 | : Supervisor defined data declarations
: 1604 | :
: 1605 | L$UNIT,
: 1606 | :
: 1607 | : Hardware P_Table storage declarations
: 1608 | :
: 1609 | RC25_ADDR : ref RC25 field (ISD_FIELD),
: 1610 | VEC_ADDR : word,

```


M8

ZRCHB3
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:33:45
5-Apr-1984 13:31:29

SEQ 0103
Page 4
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB3.B16;5(3)

```
:      1611      BR_LEVEL : word,  
:      1612      UNIT_NO  : word,  
:      1613      !  
:      1614      ! Software question response storage declarations  
:      1615      !  
:      1616      SW_UNATT : word,  
:      1617      !  
:      1618      ! Formatter data structures  
:      1619      !  
:      1620      ISD_STRUCT : blockvector [4, 2, word] field (ISD_FIELD);  
:      1621
```

ZRCH83
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
TYPE AND DESCRIPTION

5-Apr-1984 13:33:45
5-Apr-1984 13:31:29

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:(NEALE.AZTEC)ZRCH83.B16;5 (4)

SEQ 0104
Page 5

```
:      1622 #sbttl 'TYPE AND DESCRIPTION'
:      1623 !
:      1624 ! Two lines of text will be printed to the operator (in addition to the
:      1625 ! program name). The first will come from the "DESCRIPT" macro at start
:      1626 ! up time and will identify the diagnostics. The second will come from
:      1627 ! the "DEVTYPE" macro at hardware dialogue time and will identify the
:      1628 ! device under test. The arugments of both macros are 72 character
:      1629 ! ascii strings enclosed in parenthesis:
:      1630 !
:      1631 DESCRIPT (#asciz'RC25 DISK FORMATTER');
:      1632 DEVTYP (#asciz'RC25 DISK DRIVE SUBSYSTEM');
```

ZRCH83
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
HARDWARE PARAMETER CODING

5-Apr-1984 13:33:45
5-Apr-1984 13:31:29

VAX-11 B110-16 V3 555
SPIDER\USERS:[NEALE.AZTEC]ZRCH83.B16;5 (5)

SEQ 0105

Page 6

```

:      1633 #abttl 'HARDWARE PARAMETER CODING'
:      1634 !.
:      1635 ! The hardware parameter coding section contains macros
:      1636 ! that are used by the supervisor to build P-Tables. The
:      1637 ! macros are not executed as machine instructions but are
:      1638 ! interpreted by the supervisor as data structures. The
:      1639 ! macros allow the supervisor to establish communications
:      1640 ! with the operator.
:      1641 !.
:      1642 BGNHRD;
:      1643 GPRMA (HW_Q1_IP, #0'0', 0, #0'16000', #0'17777', YES, 1);      !Get RC25 Controller IP register
:      1644 GPRMA (HW_Q2_VECTOR, #0'2', 0, 4, #0'774', YES, 1);      !Get RC25 Interrupt Vector address
:      1645 GPRMD (HW_Q3_BR, #0'4', 0, #0'17777', 0, 7, YES, 1);      !Get RC25 Bus Request Priority
:      1646 GPRMD (HW_Q4_UNIT, #0'6', D, #0'17777', 0, #decimal'253', YES, 1);      !Get unit number to format
:      1647 ENDHRD;

```

ZRCHB3
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
SOFTWARE PARAMETER CODING SECTION

5 Apr 1984 13:33:45
5 Apr-1984 13:31:29

SEQ 0106
Page 7
VAX-11 B110-16 V3 555
SPIDER#USERS:(NEALE.AZTEC)ZRCHB3.016,5(6)

```
: 1648 #sbttl 'SOFTWARE PARAMETER CODING SECTION'
: 1649 !.
: 1650 ! The software parameter coding section contains macros
: 1651 ! that are used by the supervisor to build P Tables. The
: 1652 ! macros are not executed as machine instructions but are
: 1653 ! interpreted by the supervisor as data structures. The
: 1654 ! macros allow the supervisor to establish communications
: 1655 ! with the operator.
: 1656 !.
: 1657 BGNSFT;
: 1658 GPRML (SW_Q1_UNATT, 0, 1, YES, 1);
: 1659 ENDSFT;
```

ZRCHB3
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
REPORT CODING SECTION

5 Apr-1984 13:33:45
5 Apr-1984 13:31:29

VAX-11 B110-16 V3-555
SPIDER\$USERS:(NEALE.AZTEC)ZRCHB3.B16;5 (7)

```

:      1660 *sbt1 'REPORT CODING SECTION'
:      1661 !
:      1662 ! The statistical report coding section contains the PRINTS macros that
:      1663 ! will be used to generate statistical reports. The 'BGNRPT' and 'ENDRPT'
:      1664 ! macros are used as begining and ending directives for the coding con
:      1665 ! tained in the section. However, an externally located DORPT call, or
:      1666 ! a print command from the operator, may be used to request the execu-
:      1667 ! tion of the report coding.
:      1668 !-
:      1669 BGNRPT;
:      1670 return;
:      1671 !
:      1672 ! Report summary coding is remote program driven
:      1673 !
:      1674 ENDRPT;

```

```

.TITLE ZRCHB3 ZRCHB0 RC25 DISK FORMATTER
.IDENT /REV B /

```

```

000000          .PSECT AA$CODE, RO
000000      122      103      062      L$DESC: .ASCII /RC2/
000003      065      040      104      .ASCII /5 D/
000006      111      123      113      .ASCII /ISK/
000011      040      106      117      .ASCII / FO/
000014      122      115      101      .ASCII /RMA/
000017      124      124      105      .ASCII /TTE/
000022      122      000      .ASCII /R/<00>
000024      .BLKB 2
000026      122      103      062      L$DVTYP:
000031      065      040      104      .ASCII /RC2/
000034      111      123      113      .ASCII /5 D/
000037      040      104      122      .ASCII /ISK/
000042      111      126      105      .ASCII / DR/
000045      040      123      125      .ASCII /IVE/
000050      102      123      131      .ASCII / SU/
000053      123      124      105      .ASCII /BSY/
000056      115      000      .ASCII /STE/
000060      .BLKB 2
000062      000000C      L$HRDLN:
000064      000031      .WORD <<<L$NDHRD-L$HRDLN>/2> 1>
000066      000000G      GP$1: .WORD 31
000070      016000      .WORD HW.Q1.IP
000072      177777      .WORD 16000
000074      001031      .WORD 1
000076      000000G      GP$2: .WORD 1031
000100      000004      .WORD HW.Q2.VECTOR
000102      000774      .WORD 4
000104      002032      .WORD 774
000106      000000G      GP$3: .WORD 2032
000110      177777      .WORD HW.Q3.BR
000112      000000      .WORD 1
000114      000007      .WORD 0
000116      003052      GP$4: .WORD 7
          .WORD 3052

```

```

000120 000000G      .WORD  HW.Q4.UNIT
000122 177777      .WORD  -1
000124 000000      .WORD  0
000126 000375      .WORD  375
000130              L$NDHRD: .BLKW  1
000132 000000C      L$SFTLN: .WORD  <<<L$NDSFT-L$SFTLN>/2> 1>
000134 000130      GP$5:  .WORD  130
000136 000000G      .WORD  SW.Q1.UNATT
000140 000001      .WORD  1
000142              L$NDSFT: .BLKW  1

.GLOBAL  DECODE, LOAD.FILE, AZFMTR, OVSA
.GLOBAL  HW.Q1.IP, HW.Q2.VECTOR, HW.Q3.BR
.GLOBAL  HW.Q4.UNIT, SW.Q1.UNATT, SW.Q2.NOTICE
.GLOBAL  SW.Q3.OPER, SW.Q4.UNATT, FMT2
.GLOBAL  FMT3, CRLF, XCRLF, NO.ADD.UNITS
.GLOBAL  PWR.MSG, ABO.MSG, TO.MANY.UNITS
.GLOBAL  GOOD.NUM.UNITS, DATMSG, FLG.WRD
.GLOBAL  LUN, PTBL.PTR, DATETXT, L$UNIT
.GLOBAL  RC25.ADDR, VEC.ADDR, BR.LEVEL
.GLOBAL  UNIT.NO, SW.UNATT, ISD.STRUCT

```

```

000064'  L$HARD==  L$HRDLN*2
000134'  L$SOFT==  L$SFTLN*2

```

```

000000 000207      LRPT:  .SBTTL  LRPT REPORT CODING SECTION
                   RTS      PC
;
: Routine Size: 1 word,      Routine Base: AA$CODE * 0144
: Maximum stack depth per invocation: 0 words

```

```

000000 004767 177772  L$RPT: .SBTTL  L$RPT REPORT CODING SECTION
000004 104425          JSR      PC,LRPT
000006 000207          TRAP    25
                   RTS      PC
;
: Routine Size: 4 words,      Routine Base: AA$CODE * 0146
: Maximum stack depth per invocation: 2 words

```

```

: 1675 #sbttl 'INITIALIZE SECTION'
: 1676 BGNINIT;
: 1677
: 1678 !**
: 1679 ! The initialization code is executed at the beginning of every
: 1680 ! sub-pass and is primarily used for requesting P_Tables. Any
: 1681 ! other set-up type functions may also be performed in the init
: 1682 ! code.
: 1683
: 1684 ! The initialize code is executed under five conditions. There
: 1685 ! are supervisor event flags that are used to let the diagnostic
: 1686 ! know under which condition the execution is taking place. The
: 1687 ! event flags are read using the "READEF" macro.
: 1688
: 1689 ! The conditions under which the init code is executed and the
: 1690 ! corresponding event flags are:
: 1691
: 1692 !           START COMMAND           EF.START
: 1693 !           RESTART COMMAND         EF.RESTART
: 1694 !           CONTINUE COMMAND        EF.CONTINUE
: 1695 !           POWERDOWN/POWERUP       EF.PWR
: 1696 !           NEW PASS                 EF.NEW
: 1697
: 1698 ! Example of event flag use:
: 1699
: 1700 !           if READEF(EF.START) then
: 1701 !               START_FLAG = 1;
: 1702 ! ---
: 1703
: 1704 !**
: 1705 ! First read the event flag EF_PWR to see if this init code is being
: 1706 ! performed due to a system power fail. If it is then report the
: 1707 ! incident to the operator and abort the DM machine and further execution
: 1708 ! of this program.
: 1709 ! -
: 1710
: 1711 ! if READEF (EF_PWR)                !Is the PWR event flag set
: 1712 ! then
: 1713 !     begin                          !Report the incident and abort
: 1714 !     PRINTB (PWR_MSG);              !Power fail print message
: 1715 !     PRINTB (ABO_MSG);              !Aborting program message
: 1716 !     WRT_RC25 (RCIP, ONES);         !Abort DM code execution
: 1717 !     DOCLN;                          !Abort further program execution
: 1718 !     end;
: 1719
: 1720 !
: 1721 ! See if the DRS commands START or RESTART were used to start
: 1722 ! this formatting session. If either of them were used then
: 1723 ! start the formatting session at logical unit zero.
: 1724 !
: 1725
: 1726 ! if (READEF (EF_START)) or (READEF (EF_RESTART)) !Did start of restart get us here
: 1727 ! then
: 1728 !     begin
: 1729 !     !
: 1730 !     ! Check the operator for trying to format
: 1731 !     ! more than the defined limit. If they

```

```

:      1732      ! are then report the error and die.
:      1733      !
:      1734
:      1735      if .L$UNIT gtru L$LIMIT      !Is the formatting limit exceeded
:      1736      then
:      1737          begin
:      1738              PRINTB (TO_MANY_UNITS);      !Report the error
:      1739              PRINTB (GOOD_NUM_UNITS);      !Tell him/her the limit
:      1740              DOCLN;      !Go to cleanup and die
:      1741          end;
:      1742
:      1743      !
:      1744      ! Test to see if this host code is to run in UNATTENDED REFORMAT MODE.
:      1745      ! If it is then ask the operator for the date. If not then tell the
:      1746      ! operator that an operator must be present during this formatting.
:      1747      !
:      1748
:      1749      if .SW_UNATT
:      1750      then
:      1751          begin
:      1752              !
:      1753              ! Clear out the date text buffer before loading in date.
:      1754              !
:      1755
:      1756              incr i from 0 to 11 do
:      1757                  DATETXT [.i] = ZEROS;
:      1758
:      1759              GMANID (DATMSG, DATETXT, A, $o'177777', 8, 10, NO);      !Get date from operator
:      1760              PRINTB (SW_Q2_NOTICE);      !Print notice message
:      1761              PRINTB (SW_Q4_UNATT);      !Report running in ATTENDED mode
:      1762          end
:      1763      else
:      1764          begin
:      1765              PRINTB (SW_Q2_NOTICE);      !Print notice message
:      1766              PRINTB (SW_Q3_OPER);      !Tell operator he/she must be present
:      1767          end;
:      1768
:      1769      !
:      1770      ! Everything looks good so far, move on and get the
:      1771      ! hardware question responses and save them.
:      1772      !-
:      1773
:      1774      LUN = -1;      !Start formatting at logical unit 0
:      1775
:      1776      do
:      1777          begin
:      1778              LUN = .LUN + 1;      !Up pointer to next logical unit to GPHARD
:      1779
:      1780              if .LUN gequ .L$UNIT then DOCLN;      !End host code if no logical units
:      1781          end
:      1782      end
:      1783      until (GPHARD (.LUN, PTBL PTR)) nequ ZERO;      !Repeat GPHARD s until P Table adrs returned
:      1784
:      1785      !
:      1786      ! Get the P Table parameters for this unit and go format it.
:      1787      !
:      1788      RC25 ADDR = .PTBL PTR [wrdo];      !Load up the controllers base address

```



```

:      1789      VEC_ADDR = .PTBL_PTR [wrđ1];          !Load up the controllers vector address
:      1790      ISD_STRUCT [BLK0, WRD1, S1W_VADR] = .VEC_ADDR/4;      !V2.0
:      1791      BR_LEVEL = .PTBL_PTR [wrđ2];          !Load up the controllers bus request
:      1792      UNIT_NO = .PTBL_PTR [wrđ3];          !Load up the unit number to format
:      1793
:      1794      !.
:      1795      ! Before leaving the init code section we must first do some house keeping
:      1796      ! left behind from the ISD_STRUCT preset declaration. The adapter purge interrupt
:      1797      ! bit must be defined for the type machine this formatter is running under.
:      1798      !.
:      1799
:      1800      if #bliss (bliss16)                      !Define compiler
:      1801      then
:      1802          ISD_STRUCT [BLK1, WRD1, S2W_PI] = ZERO !No purging for PDP-11
:      1803      else
:      1804          ISD_STRUCT [BLK1, WRD1, S2W_PI] = ONE; !Purging for VAX 11
:      1805
:      1806      !..
:      1807      ! This commented code will load an external copy of the formatter from
:      1808      ! the local load media into azkel6 module space starting at AZFMTR : VECTOR (.word)
:      1809      !.
:      1810      !.
:      1811      ! Now load in the RC25 formatter DM code from the local boot device
:      1812      ! into the blank buffer allocated in azkel6. Report load error if
:      1813      ! an error code is returned.
:      1814      !.
:      1815      ! The DM code will only be read in during start! or restarts of the
:      1816      ! host code. Block zero of the the DM .sav file will thrown out.
:      1817      !.
:      1818      !.
:      1819      ! if LOAD_FILE (AZFMTR, DM_FN$EXT, DM_SIZE) then DECODE ();
:      1820      !.
:      1821      !..
:      1822      !.
:      1823      !.
:      1824      ! Now calculate the overlay sections starting address and store the result
:      1825      ! in global location 'OVSA' for future reference.
:      1826      !.
:      1827      ! This takes the DM buffer starting adrs and adds to it the number of bytes
:      1828      ! in the (initial load + remote program header size) resulting in the first
:      1829      ! adrs of the overlay section.
:      1830      !.
:      1831      OVSA = AZFMTR + (.AZFMTR [WRD0]);          !Calculate overlay start adrs
:      1832      end
:      1833      else
:      1834      begin
:      1835      !.
:      1836      ! The continue flag set will cause the current units formatting to be
:      1837      ! aborted. The next sequential logical unit will then be formatted.
:      1838      !.
:      1839      !.
:      1840      ! F READEF (EF CONTINUE)
:      1841      then
:      1842      begin
:      1843      !.
:      1844      ! End this Host code if this .lun is the the last logical unit to format.
:      1845      ! Else do another GPHARD macro to get the next logical unit to format.

```

ZRCMB3
REV B PATCH 00

ZRCMB0 RC25 DISK FORMATTER
INITIALIZE SECTION

5 Apr-1984 13:33:45
5 Apr-1984 13:31:29

VAX-11 Bliss-16 V3-555
SPIDER\USERS:[NEALE.AZTEC]ZRCMB3.B16;5 (8)

SEQ 0112
Page 13

```

:      1846      ;
:      1847      PRINTB (FMT3, .LUN, .UNIT_NO);      !Report this luns format was aborted
:      1848      WRT_RC25 (RCIP, ONES);      !Abort this current units formatting
:      1849
:      1850      if (.LUN + 1) gequ .L$UNIT
:      1851      then
:      1852      begin
:      1853      PRINTB (NO ADD UNITS);      !Report no more unit to format
:      1854      DOCLN;
:      1855      end;
:      1856
:      1857      end;
:      1858
:      1859      ;
:      1860      ! The new pass event flag set means that all logical units have
:      1861      ! been formatted. Therefore end this host code and return to DS>.
:      1862      ;
:      1863
:      1864      if READEF (EF_NEW) then DOCLN;      !New pass means all units formatted
:      1865
:      1866      ;
:      1867      ! Repeat doing GPWARDS until a non zero value is returned on all logical units
:      1868      ! have been GPWARD'ed. End this host code if all logical units have been formatted.
:      1869      ;
:      1870
:      1871      do
:      1872      begin
:      1873      LUN = .LUN + 1;      !Up pointer to next logical unit
:      1874
:      1875      if .LUN gequ .L$UNIT then DOCLN;      !End the host if all logical units GPWARD'ed
:      1876
:      1877      end
:      1878      until (GPWARD (.LUN, PTBL_PTR)) nequ ZERO;      !Repeat until a P table adrs is returned
:      1879
:      1880      ;
:      1881      ! Get the P_Table parameters for this unit and go format 't.
:      1882      ;
:      1883      RC25_ADDR = .PTBL_PTR [wrdo];      !Load up the controllers base address
:      1884      VEC_ADDR = .PTBL_PTR [wrdo1];      !Load up the controllers vector address
:      1885      ISD_STRUCT [BLKO, WRD1, SIW_VADR] = .VEC_ADDR/4;      !V2.0
:      1886      BR_LEVEL = .PTBL_PTR [wrdo2];      !Load up the controllers bus request
:      1887      UNIT_NO = .PTBL_PTR [wrdo3];      !Load up the unit number to format
:      1888      end;
:      1889
:      1890      ;
:      1891      ! Tell the operator which RC25 controller is being initialized this pass.
:      1892      ;
:      1893      PRINTB (XCRLF);      !Space between each formatting pass
:      1894      PRINTB (FMT2, .PTBL_PTR [WRD0]);
:      1895      ENDINIT;

```

000000	004167	000000G	.SBTTL	LINIT INITIALIZE SECTION		
000004	012700	000034	LINIT:	JSR	R1,\$SAVE2	;
000010	104447			MOV	#34,R0	;
000012	103023			TRAP	47	
				BHIS	1\$	

1674
1711

ZRCHB3
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
INITIALIZE SECTION

5-Apr-1984 13:33:45
5 Apr 1984 13:31:29

VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB3.B16;5 (8)

000014	012746	000000G		MOV	#PWR.MSG, -(SP)	:	1714
000020	012746	000001		MOV	#1, (SP)	:	
000024	010600			MOV	SP,RO	: SP,*	
000026	104414			TRAP	14	:	
000030	012716	000000G		MOV	#ABO.MSG,(SP)	:	1715
000034	012746	000001		MOV	#1, (SP)	:	
000040	010600			MOV	SP,RO	: SP,*	
000042	104414			TRAP	14	:	
000044	012700	177777		MOV	#-1,RO	: *,RC#M.REG	1716
000050	010077	000000G		MOV	RO,#RC25.ADDR	: RC#M.REG,*	
000054	104444			TRAP	44	:	
000056	062706	000006		ADD	#6,SP	:	1713
000062	012700	000040	1\$:	MOV	#40,RO	:	1726
000066	104447			TRAP	47	:	
000070	103404			BCS	2\$:	
000072	012700	000037		MOV	#37,RO	:	
000076	104447			TRAP	47	:	
000100	103167			BHIS	9\$:	
000102	026727	000000G 000020	2\$:	CMP	L\$UNIT,#20	:	1735
000110	101417			BLOS	3\$:	
000112	012746	000000G		MOV	#TO.MANY.UNITS, -(SP)	:	1738
000116	012746	000001		MOV	#1, -(SP)	:	
000122	010600			MOV	SP,RO	: SP,*	
000124	104414			TRAP	14	:	
000126	012716	000000G		MOV	#GOOD.NUM.UNITS,(SP)	:	1739
000132	012746	000001		MOV	#1, -(SP)	:	
000136	010600			MOV	SP,RO	: SP,*	
000140	104414			TRAP	14	:	
000142	104444			TRAP	44	:	
000144	062706	000006		ADD	#6,SP	:	1737
000150	032767	000001 000000G	3\$:	BIT	#1,SW.UNATT	:	1749
000156	001434			BEQ	5\$:	
000160	005000			CLR	RO	: I	1756
000162	105060	000000G	4\$:	CLRB	DATETXT(RO)	: *(I)	1757
000166	005200			INC	RO	: I	1756
000170	020027	000013		CMP	RO,#13	: I,*	
000174	003772			BLE	4\$:	
000176	104443			TRAP	43	:	1759
000200	000406			.WORD	406	:	
000202	000000G			.WORD	DATETXT	:	
000204	000142			.WORD	142	:	
000206	000000G			.WORD	DATMSG	:	
000210	177777			.WORD	-1	:	
000212	000010			.WORD	10	:	
000214	000012			.WORD	12	:	
000216	012746	000000G		MOV	#SW.Q2.NOTICE, (SP)	:	1760
000222	012746	000001		MOV	#1, -(SP)	:	
000226	010600			MOV	SP,RO	: SP,*	
000230	104414			TRAP	14	:	
000232	012716	000000G		MOV	#SW.Q4.UNATT,(SP)	:	1761
000236	012746	000001		MOV	#1, (SP)	:	
000242	010600			MOV	SP,RO	: SP,*	
000244	104414			TRAP	14	:	
000246	000414			BR	6\$:	1749
000250	012746	000000G	5\$:	MOV	#SW.Q2.NOTICE, (SP)	:	1765
000254	012746	000001		MOV	#1, (SP)	:	
000260	010600			MOV	SP,RO	: SP,*	

ZRCHB3
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
INITIALIZE SECTION

5-Apr-1984 13:33:45
5-Apr-1984 13:31:29

VAX-11 Bliss-16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCHB3.B16;5 (8)

000262	104414				TRAP	14			
000264	012716	000000G			MOV	#SW.Q3.OPER,(SP)	:		1766
000270	012746	000001			MOV	#1,-(SP)	:		
000274	010600				MOV	SP,R0	:	SP,*	
000276	104414				TRAP	14			
000300	112767	000377	000000G	6\$:	MOVB	#377,LUN	:		1774
000306	105267	000000G		7\$:	INCB	LUN	:		1778
000312	005000				CLR	R0	:		1780
000314	156700	000000G			BISB	LUN,R0			
000320	020067	000000G			CMP	R0,L\$UNIT			
000324	103401				BLO	8\$			
000326	104444				TRAP	44			
000330	005000			8\$:	CLR	R0	:		1783
000332	156700	000000G			BISB	LUN,R0			
000336	104442				TRAP	42			
000340	010067	000000G			MOV	R0,PTBL.PTR			
000344	001760				BEQ	7\$			
000346	010001				MOV	R0,R1	:	PTBL.PTR,*	1788
000350	011167	000000G			MOV	(R1),RC25.ADDR			
000354	016167	000002	000000G		MOV	2(R1),VEC.ADDR	:		1789
000362	016716	000000G			MOV	VEC.ADDR,(SP)	:		1790
000366	012746	000004			MOV	#4,-(SP)			
000372	004767	000000G			JSR	PC,BL\$DIV			
000376	010002				MOV	R0,R2			
000400	042702	177600			BIC	#177600,R2			
000404	142767	000177	000002G		BICB	#177,ISD.STRUCT+2			
000412	050267	000002G			BIS	R2,ISD.STRUCT+2			
000416	016167	000004	000000G		MOV	4(R1),BR.LEVEL	:		1791
000424	016167	000006	000000G		MOV	6(R1),UNIT.NO	:		1792
000432	142767	000001	000006G		BICB	#1,ISD.STRUCT+6	:		1800
000440	016767	000000G	000000G		MOV	AZFMTR,OVSA	:		1831
000446	062767	000000G	000000G		ADD	#AZFMTR,OVSA			
000454	022626				CMP	(SP)+,(SP)+	:		1728
000456	000523				BR	14\$:		1726
000460	012700	000036		9\$:	MOV	#36,R0	:		1840
000464	104447				TRAP	47			
000466	103040				BHIS	11\$			
000470	016746	000000G			MOV	UNIT.NO,-(SP)	:		1847
000474	005046				CLR	-(SP)			
000476	116716	000000G			MOVB	LUN,(SP)			
000502	012746	000000G			MOV	#FMT3,-(SP)			
000506	012746	000003			MOV	#3,-(SP)			
000512	010600				MOV	SP,R0	:	SP,*	
000514	104414				TRAP	14			
000516	012700	177777			MOV	#1,R0	:	*,RC\$M.REG	1848
000522	010077	000000G			MOV	R0,#RC25.ADDR	:	RC\$M.REG,*	
000526	005000				CLR	R0	:		1850
000530	156700	000000G			BISB	LUN,R0			
000534	005200				INC	R0			
000536	020067	000000G			CMP	R0,L\$UNIT			
000542	103410				BLO	10\$			
000544	012716	000000G			MOV	#NO.ADD.UNITS,(SP)	:		1853
000550	012746	000001			MOV	#1,-(SP)			
000554	010600				MOV	SP,R0	:	SP,*	
000556	104414				TRAP	14			
000560	104444				TRAP	44			
000562	005726				TST	(SP)+	:		1852

ZRCHB3
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
INITIALIZE SECTION

5-Apr-1984 13:33:45
5-Apr-1984 13:31:29

SEQ 0115
Page 16
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB3.816;5 (8)

000564	062706	000010	10\$:	ADD	#10,SP	:	1842
000570	012700	000035	11\$:	MOV	#35,RO	:	1864
000574	104447			TRAP	47		
000576	103001			BHIS	12\$		
000600	104444			TRAP	44		
000602	105267	000000G	12\$:	INCB	LUN	:	1873
000606	005000			CLR	RO	:	1875
000610	156700	000000G		BISB	LUN,RO		
000614	020067	000000G		CMP	RO,L\$UNIT		
000620	103401			BLO	13\$		
000622	104444			TRAP	44		
000624	005000		13\$:	CLR	RO	:	1878
000626	156700	000000G		BISB	LUN,RO		
000632	104442			TRAP	42		
000634	010067	000000G		MOV	RO,PTBL.PTR		
000640	001760			BEQ	12\$		
000642	010001			MOV	RO,R1	: PTBL.PTR,*	1883
000644	011167	000000G		MOV	(R1),RC25.ADDR		
000650	016167	000002 000000G		MOV	2(R1),VEC.ADDR	:	1884
000656	016746	000000G		MOV	VEC.ADDR,-(SP)	:	1885
000662	012746	000004		MOV	#4,-(SP)		
000666	004767	000000G		JSR	PC,BL\$DIV		
000672	010002			MOV	RO,R2		
000674	042702	177600		BIC	#177600,R2		
000700	142767	000177 000002G		BICB	#177,ISD.STRUCT+2		
000706	050267	000002G		BIS	R2,ISD.STRUCT+2		
000712	016167	000004 000000G		MOV	4(R1),BR.LEVEL	:	1886
000720	016167	000006 000000G		MOV	6(R1),UNIT.NO	:	1887
000726	012716	000000G	14\$:	MOV	#XCRLF,(SP)	:	1893
000732	012746	000001		MOV	#1,-(SP)		
000736	010600			MOV	SP,RO	: SP,*	
000740	104414			TRAP	14		
000742	017716	000000G		MOV	#PTBL.PTR,(SP)	:	1894
000746	012746	000000G		MOV	#FMT2,-(SP)		
000752	012746	000002		MOV	#2,-(SP)		
000756	010600			MOV	SP,RO	: SP,*	
000760	104414			TRAP	14		
000762	062706	000012		ADD	#12,SP	:	1674
000766	000207			RTS	PC		

; Routine Size: 252 words, Routine Base: AA\$CODE + 0156
; Maximum stack depth per invocation: 10 words

000000	004767	177004		.SBTTL	L\$INIT INITIALIZE SECTION		
000004	104411		L\$INIT::	JSR	PC,LINIT	:	1894
000006	000207			TRAP	11		
				RTS	PC		

Routine Size: 4 words, Routine Base: AA\$CODE + 1146
Maximum stack depth per invocation: 2 words

ZRCHB3 ZRCHB0 RC25 DISK FORMATTER
 REV B PATCH 00 AUTODROP SECTION

5-Apr-1984 13:33:45
 5-Apr-1984 13:31:29

SEQ 0116
 Page 17
 VAX-11 Bliss-16 V3-555
 SPIDER\$USERS:[NEALE.AZTEC]ZRCHB3.B16;5 (9)

```

:      1896 %sbttl 'AUTODROP SECTION
:      1897 !*
:      1898 ! This code is executed immediately after the initialize code if
:      1899 ! the "ADR" flag was set. The unit(s) under test are checked to
:      1900 ! see if they will respond. Those that don't are immediately
:      1901 ! dropped from testing.
:      1902 !-
:      1903 RGNAUTO;
:      1904 return;
:      1905 ENDAUTO;
    
```

```

000000 000207          LAUTO:  .SBTTL  LAUTO AUTODROP SECTION          ;          1895
                                RTS    PC
: Routine Size: 1 word,      Routine Base: AA$CODE + 1156
: Maximum stack depth per invocation: 0 words
    
```

```

000000 004767 177772  L$AUTO:  .SBTTL  L$AUTO AUTODROP SECTION          ;          1904
000004 104461          TRAP    PC,LAUTO
000006 000207          RTS     61
                                PC
: Routine Size: 4 words,      Routine Base: AA$CODE + 1160
: Maximum stack depth per invocation: 2 words
    
```

ZRCHB3
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
CLEANUP CODING SECTION

5 Apr-1984 13:33:45
5-Apr-1984 13:31:29

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB3.816;5 (10)

```

:      1906  #sbttl 'CLEANUP CODING SECTION'
:      1907  ;
:      1908  ! Cleanup coding is assembled with the diagnostic program, utilizing in
:      1909  ! itiating (BGNCLN) and ending (ENDCLN) directives. The coding can be
:      1910  ! used by either the diagnostic program or the supervisor to affect the
:      1911  ! return of a test device to a static state.
:      1912  ;
:      1913  ! The clean-up code is invoked in three different ways:
:      1914  !
:      1915  !     A. At end of every sub-pass
:      1916  !     B. Issuance of DOCLN macro
:      1917  !     C. Operator ^C
:      1918  !
:      1919  BGNCLN;
:      1920  !
:      1921  ! Before initing the RC25 controller check first to make sure that
:      1922  ! that this is not a non-existant register.
:      1923  !
:      1924  !
:      1925  if not .FLG_WRD [NON_EXIST_REG] then WRT_RC25 (RCIP, ONES);      !Abort DM code execution
:      1926  !
:      1927  return;
:      1928  ENDCLN;

```

```

000000 032767 000002 000000G          .SBTTL  LCLEAN CLEANUP CODING SECTION
000006 001004          LCLEAN: BIT    @2,FLG.WRD          ;
000010 012700 177777          BNE     1$              ;
000014 010077 000000G          MOV     @-1,R0         ; *,RC$M.REG
000020 000207          1$:  MOV     R0,@RC25.ADDR      ; RC$M.REG,*
                                RTS     PC              ;
                                                1905

```

```

: Routine Size: 9 words,      Routine Base: AA$CODE + 1170
: Maximum stack depth per invocation: 0 words

```

```

000000 004767 177752          .SBTTL  L$CLEAN CLEANUP CODING SECTION
000004 104412          L$CLEAN: JSR     PC,LCLEAN          ;
000006 000207          TRAP   12              ;
                                RTS     PC              ;
                                                1927

```

```

: Routine Size: 4 words,      Routine Base: AA$CODE + 1212
: Maximum stack depth per invocation: 2 words

```

ZRCHB3
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
DROP UNIT SECTION

5-Apr-1984 13:33:45
5-Apr-1984 13:31:29

VAX-11 B110-16 V3 555

SPIDER#USERS:[NEALE.AZTEC]ZRCHB3.B16;5 (11)

SEQ 0118

Page 19

```

:      1929  .sbt1 'DROP UNIT SECTION'
:      1930  !
:      1931  ! The drop code is invoked by a DODU macro or a drop command, and con-
:      1932  ! tains any code that needs to be executed in conjunction with the drop
:      1933  ! ping of a unit from the test cycle. No coding is required in this
:      1934  ! section.
:      1935  !
:      1936  ! The effect of a DODU is the same whether executed in the init code or
:      1937  ! in a hardware test. It invokes the drop unit coding and causes subse-
:      1938  ! quent GPHARD'S for that logical unit to be returned 'NOT COMPLETE'.
:      1939  ! This effect lasts only for the duration of the current command.
:      1940  !
:      1941  BGNDU;
:      1942  return;
:      1943  ENDDU;
    
```

```

000000 000207          LDU:      .SBTTL  LDU DROP UNIT SECTION
                                RTS      PC
:
: Routine Size: 1 word,      Routine Base: AA$CODE + 1222
: Maximum stack depth per invocation: 0 words
                                ;
                                1928
    
```

```

000000 004767 177772  L$DU::  .SBTTL  L$DU DROP UNIT SECTION
000004 104453          JSR      PC,LDU
000006 000207          TRAP    53
                                RTS      PC
:
: Routine Size: 4 words,      Routine Base: AA$CODE + 1224
: Maximum stack depth per invocation: 2 words
                                ;
                                1942
    
```


ZRCMB3
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
ADD UNIT SECTION5-Apr-1984 13:33:45
5 Apr-1984 13:31:29VAX-11 Bliss-16 V3-555
SPIDER\$USERS:(NEALE.AZTEC)ZRCMB3.B16;5 (12)SEQ 0119
Page 20

```

:      1944 .sbtll 'ADD UNIT SECTION
:      1945 :
:      1946 : The add code is invoked by the ADD command, and contains any code that
:      1947 : needs to be executed in conjunction with adding a unit back to the
:      1948 : test cycle. No coding is required in this section.
:      1949 :
:      1950 : Units may be added to the test sequence only through the use of opera
:      1951 : tor ADD command. Each unit must have a P-TABLE in memory due to an
:      1952 : earlier hardware dialogue (i.e. the unit was previously dropped).
:      1953 : The ADD code must be delimited by BGNAU, ENDAU. There is no particu-
:      1954 : lar coding required in the add code to cause the add to be effective:
:      1955 :
:      1956 : The section is just for programmer housekeeping.
:      1957 :
:      1958 BGNAU;
:      1959 return;
:      1960 ENDAU;

```

```

000000 000207          LAU:      .SBTTL  LAU ADD UNIT SECTION
:                                RTS      PC
:                                ;
:                                1943
: Routine Size: 1 word,      Routine Base: AA$CODE + 1234
: Maximum stack depth per invocation: 0 words

```

```

000000 004767 177772  L$AU:: .SBTTL  L$AU ADD UNIT SECTION
000004 104452          JSR      PC,LAU
000006 000207          TRAP     52
:                                RTS      PC
:                                ;
:                                1959
: Routine Size: 4 words,      Routine Base: AA$CODE + 1236
: Maximum stack depth per invocation: 2 words

```

```

:      1961 end
:      1962
:      1963 eludom

```

```

:
:      OTS external references
:      .GLOBL $SAVE2, BL$DIV

```

PSECT SUMMARY

```

:
:      Psect Name          Words      Attributes
:      AA$CODE             339      RO , I , LCL. REL. CON

```

```

:
:      LIBRARY STATISTICS
:

```

D10

ZRCH83 ZRCH80 RC25 DISK FORMATTER
REV B PATCH 00 ADD UNIT SECTION

5-Apr-1984 13:33:45
5-Apr-1984 13:31:29

SEQ 0120
Page 21
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCH83.B16;5 (12)

File	Total	----- Symbols Loaded	-- Percent	----- Blocks Read
SPIDER\$USERS:(NEALE.AZTEC)ZRCH80.L16;2	398	63	15	24

COMMAND QUALIFIERS

BLISS /PDP11/LIST ZRCH83.B16

: Size: 289 code + 50 data words
: Run Time: 00:18.7
: Elapsed Time: 02:34.9
: Memory Used: 222 pages
: Compilation Complete

ZRCHB4

ZRCHB0 RC25 DISK FORMATTER

5-Apr-1984 13:44:04
23-Sep-1983 14:13:18VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB4.B16;2(1)SEQ 0121
Page 1

```

: 0001 MODULE ZRCHB4 (#TITLE 'ZRCHB0 RC25 DISK FORMATTER'
: 0002 IDENT = 'REV B PATCH 00',
: 0003 ADDRESSING_MODE (RELATIVE) ,
: 0004 ENVIRONMENT (NOEIS)
: 0005 ) =
: 0006 BEGIN
: 0007 #sbttl 'MODULE DECLARATIONS'
: 0008 !
: 0009 ! Pretty Declarations
: 0010 !
: 0011 !<blf/lowercase_key>
: 0012 !
: 0013 !
: 0014 library 'ZRCHB0'; !Define RC25 formatter library
: 0015 !
: 0016 require 'BLSMAC.REQ'; !Define Bliss Macro require file
: 1505 !
: 1506 !*
: 1507 ! The psect named "code or $code$" is redefined here
: 1508 ! to be called "ab$code". This is done to organize the
: 1509 ! formatters test sections into a seperate psect.
: 1510 !-
: 1511 psect
: 1512 code = ab$code;
: 1513 !
: 1514 !*
: 1515 ! Structure declarations used within this module.
: 1516 !-
: 1517 !
: 1518 structure
: 1519 !
: 1520 !*
: 1521 ! RC25 register accessing structure. This
: 1522 ! structure allows RC25 register accessing
: 1523 ! to be transportable between the PDP-11 and
: 1524 ! VAX Diagnostic Supervisors.
: 1525 !
: 1526 ! This also defines an access algorithm for
: 1527 ! VAX to allow field reference to MBA address
: 1528 ! space without generating machine checks.
: 1529 !-
: 1530 !
: 1531 RC25 [0, P, S, E] =
: 1532 begin
: 1533 local
: 1534 RC$S_REG;
: 1535 !
: 1536 !
: 1537 RC$S_REG = .(RC25 * #upval*0)<0, #bpval, 0>;
: 1538 RC$S_REG
: 1539 end
: 1540 <P, S, E>;
: 1541 !
: 1542 !<blf/page>

```

ZRCHB4
REV B PATCH 00ZRCHB0 RC25 DISK FORMATTER
MODULE DECLARATIONS5-Apr-1984 13:44:04
23-Sep-1983 14:13:18VAX-11 Bliss 16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB4.B16;2 (2)

SEQ 0122

Page 2

```
:      1543  !*
:      1544  ! External Routines declared outside this module.
:      1545  !-
:      1546
:      1547  external routine
:      1548      ABORT,
:      1549      GET_DUST_STATUS,
:      1550      EX_SUP_PROG,
:      1551      EX_LOC_PROG,
:      1552      REC_DATA,
:      1553      SEND_DATA,
:      1554      SET_CNTRLR_CHAR,
:      1555      ON_LINE,
:      1556      DUP$I_SERVICE : INT_LNK$TYP novalue,
:      1557      INT$I_SERVICE : INT_LNK$TYP novalue,
:      1558      INIT_COM_AREA,
:      1559      BOOT_RC25,
:      1560      DECODE : novalue;
:      1561
:      1562  !<blf/page>
```

```

:      1563  !*
:      1564  ! External Declaration of datums declared outside of this module.
:      1565  !-
:      1566
:      1567  external
:      1568  !
:      1569  ! Micellaneous external data declarations
:      1570  !
:      1571  SW_UNATT,                !Unattended reformat mode flag
:      1572  DATETXT : vector [12, byte], !Stores operator date input
:      1573  LUN : word,                !Logical unit number storage
:      1574  UNIT_NO : word,            !Physical unit number storage
:      1575  FLG_WRD : bitvector [16],  !Global flag word
:      1576  ISD_STRUCT : blockvector [4, 2, word] field (ISD_FIELD), !Init sequence data
:      1577  PTBL_PTR : ref vector [4, word] volatile,
:      1578  NEX_FLAG : word,            !RC25 register existence flag
:      1579  NXT_CRN : byte,            !Next seq command ref number
:      1580  RET_EN$AD : ref block [RB_SIZE + 2, word] field (ENV_FIELD),
:      1581  FCT_BUF : block [256, word], !External FCT sector load buffer
:      1582  SND_BUF : vector [SNOB_SIZE, word], !DUP send cmd text buffer
:      1583  REC_BUF : block [RECB_SIZE, word] field (RECB_FIELD), !DUP receive cmd text buffer
:      1584  MSGADR,                    !Pointer to DM sent ascii text
:      1585  NSD_SLOT : word,            !Stores next send ring slot to load cmd into
:      1586  NRD_SLOT : word,            !Stores next receive slot to expect response in
:      1587  VEC_ADDR : word,           !Stores controllers vector address
:      1588  RET_STATUS : word,         !Stores return status of called routines
:      1589  PID_SAVE : vector [2, word], !Saves process indicator word
:      1590  RC25_ADDR : ref RC25 field (ISD_FIELD), !RC25 reference structure
:      1591  !
:      1592  ! Init sequence code error
:      1593  !
:      1594  BOOT_FAILURE,
:      1595  PROT0_VIOLATION,
:      1596  PORT_INIT_ERR,
:      1597  ACTIVE_DUP_SERVER,
:      1598  INACTIVE_DUP_SERVER,
:      1599  !
:      1600  ! External FCT request message during RESTORE formatting mode
:      1601  !
:      1602  FCT_REQ_MSG,
:      1603  !
:      1604  ! Printing format strings and ascii text strings
:      1605  !
:      1606  DATMSG,
:      1607  CRLF,
:      1608  FMT1,
:      1609  FMT5,
:      1610  FMT6,
:      1611  FMT7;
:      1612

```

```

:      1613 #sbttl 'RC25 REGISTER EXISTENCE TEST'
:      1614 BGNTST;
:      1615
:      1616 :..
:      1617 : Functional Description :
:      1618 :     This section will be the first code to be executed during a formatting
:      1619 :     session and will ensure that the object drive to format actually does
:      1620 :     exist. The format for the unit will be aborted if the drive fails to
:      1621 :     respond to this initial test.
:      1622 :
:      1623 : Implicit Inputs :
:      1624 :     none
:      1625 :
:      1626 : Implicit Outputs :
:      1627 :     none
:      1628 :
:      1629 : Completion Codes :
:      1630 :     none
:      1631 :
:      1632 : Side Effects :
:      1633 :     none
:      1634 :-
:      1635
:      1636 local
:      1637     EOF,
:      1638     DUMMY : volatile,
:      1639     TEMP : volatile,
:      1640     RETRIES;
:      1641
:      1642 FLG_WRD [NON_EXIST_REG] = ZERO;
:      1643 NEX_FLAG = ZEROS;
:      1644 SETVEC (4, INT$I_SERVICE, PRI07);
:      1645 TEMP = .PTBL_PTR [WRD0];
:      1646 TEMP = .TEMP * 2;
:      1647
:      1648 if ..TEMP
:      1649 then
:      1650     begin
:      1651     DUMMY = ZEROS;
:      1652     end;
:      1653
:      1654 CLRVEC (4);
:      1655
:      1656 if .NEX_FLAG eq1 ONES
:      1657 then
:      1658     begin
:      1659     PRINTB (FMT6, .PTBL_PTR [WRD0]);
:      1660     FLG_WRD [NON_EXIST_REG] = ONE;
:      1661     DOCLN;
:      1662     end;
:      1663

```

```

!EOF indicator
!Dummy variable for register ext test
!Store number of retries to perform
!Initially clear nonexistent register flag
!Clear out nex flag
!Set up for an nex trap
!Get this controllers IP address
!Make it into the SA register address
!Read the sa register
!This is so that if there
!Is an nex there will be
!A single operand inst.
!So that it will trap
!correctly.
!See if we got an nex
!Address not there
!Print error message
!Set flag indicating non existant reg

```

```

: 1664 *subttl 'INITIALIZE RC25 CONTROLLER'
: 1665 :
: 1666 : The next thing to be done is to boot the RC25 controller. We will allow
: 1667 : a few retries if not successful after the first boot before we considered
: 1668 : the Controller dead.
: 1669 :
: 1670 : But before we do the boot sequence the processors priority and interrupt
: 1671 : vector address must first be loaded. During the init sequence the Init
: 1672 : sequence interrupt service routine will just flag any interrupts and ignore
: 1673 : them since interrupts are disabled during init sequence. Later the DUP
: 1674 : interrupt service routine will be load and do the DUP communications protocol.
: 1675 :
: 1676 : The following priorities will be assigned:
: 1677 :     1. Processor will run at priority zero.
: 1678 :     2. The RC25 runs at priority 5 by default.
: 1679 :     3. The DUP interrupt routine will run at priority 7.
: 1680 :
: 1681 SETPRI (PRI00);           !Set the processors priority
: 1682 CLRVEC (.VEC_ADDR);      !Clear out the vector before setting
: 1683 SETVEC (.VEC_ADDR, INT$I SERVICE, PRI07); !Set the interrupt service priority
: 1684 :
: 1685 :
: 1686 : Retry the RC25 booting until the return is true or the retry limit is reached.
: 1687 :
: 1688 RETRIES = -1;           !Reset the retry counter
: 1689 :
: 1690 do
: 1691     begin
: 1692     RET_STATUS = BOOT_RC25 ();           !Boot the RC25 controller
: 1693     RETRIES = .RETRIES + 1;           !Up the retry count
: 1694     end
: 1695 until (.RET_STATUS) or (.RETRIES eqv ONE); !Repeat the Boot until done
: 1696 :
: 1697 :
: 1698 : Report booting error if the return status never came back as true.
: 1699 :
: 1700 :
: 1701 if not .RET_STATUS           !D'ed the Controller boot
: 1702 then
: 1703     begin
: 1704     PRINTB (BOOT_FAILURE);           !Report a boot error
: 1705     DOCLN;                           !Abort the program
: 1706     end;
: 1707 :
: 1708 :
: 1709 : Now that the RC25 controller is booted, check to make sure that the controller
: 1710 : has done its part of the DUP protocol by clearing out the port communications area.
: 1711 :
: 1712 : While we're there set up the communication area for the up and coming communications
: 1713 : between the remote and host program.
: 1714 :
: 1715 :
: 1716 if INIT_COM_AREA ()           !Was the com area cleared out as expected
: 1717 then
: 1718     begin
: 1719     PRINTB (PROTO_VIOLATION);
: 1720     PRINTB (PORT_INIT_ERR);

```

ZRCMB4
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
INITIALIZE RC25 CONTROLLER5-Apr-1984 13:44:04
23 Sep-1983 14:13:18

VAX-11 Bliss-16 V3 555

SPIDER\$USERS:[NEALE.AZTEC]ZRCMB4.B16;2 (5)

SEQ 0126

Page 6

```
:      1721      DOCLN;
:      1722      end;
:      1723
:      1724      !
:      1725      ! Before writing the go to the IP register load the DUP interrupt service adrs
:      1726      ! into the RC25 vector address for the up and coming communications between
:      1727      ! the host and controller.
:      1728      !
:      1729      CLRVEC (.VEC_ADDR);          !Clear out the vector before starting
:      1730      SETVEC (.VEC_ADDR, DUP$I SERVICE, PRI07);    !Set the interrupt service priority
:      1731      !
:      1732      ! Start the controllers functional micro code by writing the SA Register
:      1733      ! with step four write data with the go bit set.
:      1734      !-
:      1735      WRT_RC25 (RCSA, (.ISD STRUCT [BLK3, ISWRT, ISW_ALL] or %o'000001'));    !Let the controller go
```



```

:      1736 *sbttl 'BRING RC25 CONTROLLER ONLINE'
:      1737
:      1738 !+
:      1739 ! Functional Description :
:      1740 !       This section will set the RC25 controllers characteristics and place
:      1741 !       the unit on-line.
:      1742 !
:      1743 ! Implicit Inputs :
:      1744 !       none
:      1745 ! Implicit Outputs :
:      1746 !       none
:      1747 ! Completion Codes :
:      1748 !       none
:      1749 ! Side Effects :
:      1750 !       none
:      1751 !--
:      1752
:      1753 !+
:      1754 ! Set the controller characteristics. Default values will be taken in all cases
:      1755 ! except for the host time out value which will be changed to 'wait for ever'.
:      1756 !-
:      1757
:      1758 if SET_CNTLRL_CHAR ( ) then DECODE ( );           !Call decode if not successful
:      1759
:      1760 !+
:      1761 ! Do a unit on line cmd which will spin up the device and load the heads. The
:      1762 ! unit to be placed on line will be the last unit number received from the 'gphard'
:      1763 ! macro in the init code.
:      1764 !-
:      1765 PRINTB (FMT5, .LUN, .UNIT_NO);                       !Tell operator which units coming online
:      1766 !
:      1767
:      1768 if ON_LINE ( ) then DECODE ( );           !Call decode if not successful
:      1769

```

```

: 1770 *sbttl 'DOWN LINE LOAD FORMATTER AND MONITOR TASK'
: 1771
: 1772 !**
: 1773 ! Functional Description :
: 1774 !     This section will down line-load the RC25 formatter and monitor
: 1775 !     the formatting task.
: 1776 !
: 1777 ! Implicit Inputs :
: 1778 !     none
: 1779 ! Implicit Outputs :
: 1780 !     none
: 1781 ! Completion Codes :
: 1782 !     none
: 1783 ! Side Effects :
: 1784 !     none
: 1785 !--
: 1786
: 1787 !
: 1788 ! See if the Dup server in the RC25 Controller is in an
: 1789 ! idle state. To do this first get the dust status and
: 1790 ! then look at the flag field bit 3 for a :
: 1791 !     0 = idle
: 1792 !     1 = active
: 1793 !
: 1794
: 1795 if GET_DUST_STATUS () then DECODE ();           !Call Decode if connection error
: 1796
: 1797 !
: 1798 ! Look in the flag field bit 3 to see if the server is active.
: 1799 !
: 1800
: 1801 if .RET_EN$AD [FLG_B3]                           !If the server 's active exit and reboot
: 1802 then
: 1803     begin
: 1804     PRINTB (ACTIVE_DUP_SERVER);
: 1805     DOCLN;
: 1806     end;
: 1807
: 1808 !
: 1809 ! The server is not active so down line load the formatter
: 1810 ! and start its execution by issuing a "Execute supplied program".
: 1811 ! Call the decode routine if a connection error is detected.
: 1812 !
: 1813
: 1814 if EX_SUP_PROG () then DECODE ();           !Call decode if connection error
: 1815
: 1816 !
: 1817 ! Get the dust status to see if the server is in an active
: 1818 ! state. An active state is what we want so error if the
: 1819 ! server is in an idle state.
: 1820 !
: 1821 ! If in the active state then save the progress indicator
: 1822 ! in "Pid_save" for future reference.
: 1823 !
: 1824
: 1825 if GET_DUST_STATUS () then DECODE ();           !Call decode if connection error
: 1826

```

```

: 1827 !
: 1828 ! Look at the flag field bit 3 to see if the server is active.
: 1829 !
: 1830
: 1831 if not (.RET_EN$AD [FLG_B3])           !Reboot if server is idle
: 1832 then
: 1833     begin
: 1834     PRINTB (INACTIVE_DUP_SERVER);
: 1835     DOCLN;
: 1836     end
: 1837 else
: 1838     begin
: 1839     PID_SAVE [0] = .RET_EN$AD [PLO_IND];       !Save progress indicator lo word
: 1840     PID_SAVE [1] = .RET_EN$AD [PHI_IND];     !Save progress indicator hi word
: 1841     end;
: 1842
: 1843 !*
: 1844 ! The Dup server is in the active state running the formatter
: 1845 ! program. This DO LOOP will loop on the DUP sub-protocol
: 1846 ! doing the "send and receive" data commands. These commands
: 1847 ! establish the communications between this host program and
: 1848 ! the remote formatter program running in the RC25 controller.
: 1849 !-
: 1850 RETRIES = ZERO;           !Clear out the retry flag
: 1851
: 1852 while TRUE do
: 1853     begin
: 1854
: 1855     !*
: 1856     ! Do a 'Receive_data' command which poll's the remote program
: 1857     ! for a message. The returned message can either be a:
: 1858     !
: 1859     ! 1. Question
: 1860     !     Where the ascii text is a prompt for information.
: 1861     !
: 1862     ! 2. Default question
: 1863     !     Where the default question message is identical
: 1864     !     to the question message except that a null (zero
: 1865     !     length) send data is taken to be a default answer
: 1866     !     to the question.
: 1867     !
: 1868     ! 3. Information
: 1869     !     Where the ascii text is an informative message.
: 1870     !
: 1871     ! 4. Termination
: 1872     !     Where the ascii text is an normal termination message.
: 1873     !
: 1874     ! 5. Fatal Error
: 1875     !     Where the ascii text is a fatal error message.
: 1876     !
: 1877     ! 6. Special
: 1878     !     This type is used when only a host program could
: 1879     !     respond.
: 1880     !-
: 1881
: 1882     if REC_DATA () then DECODE ();           !Call decode if connection error
: 1883

```

N10

ZRCH84
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

5-Apr-1984 13:44:04
23-Sep-1983 14:13:18

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH84.B16;2 (7)

SEQ 0130
Page 10

```

1884      !
1885      ! From the first word in the send/receive data buffer, look
1886      ! to see what type message the remote program has sent to
1887      ! the Host program. Use this message type number to index
1888      ! into the select expression to perform the requested action
1889      ! by the remote program.
1890      !
1891      !
1892      $CLRSBUF;                !Clear out the send buffer area
1893      !
1894      selectoneu .REC_BUF [MSG_TYP] of      !Select the appropriate action
1895      set
1896      !
1897      [1] :                    !Question message type
1898      begin
1899      !
1900      ! Look into the send/receive data buffer at the message
1901      ! number field and see what question the remote program
1902      ! is asking. Use the fields value to index into the
1903      ! select expression to perform the appropriate action.
1904      !
1905      !
1906      selectoneu .REC_BUF [MSG_NUM] of      !Select the requested question
1907      set
1908      !
1909      [0] :                    !Enter current date <MM-DD-YYYY>
1910      begin
1911      !
1912      ! If in un-attended reformat mode then give to the DM the
1913      ! date received from the init code else let the DM prompt
1914      ! the operator for the date.
1915      !
1916      !
1917      if .SW_UNATT             !Is the host running in unattended mode
1918      then
1919      begin
1920      !
1921      if .RETRIES             !Did DM not verify the date the last ime
1922      then
1923      begin
1924      !
1925      ! Clear out the date text buffer before loading in
1926      ! date and ask the operator for the date again.
1927      !
1928      !
1929      incr i from 0 to 11 do
1930      DATETXT [.i] = ZEROS;
1931      !
1932      GMANID (DATMSG, DATETXT, A, #0'17777', 8, 10, NO); !Get date from operator
1933      end;
1934      !
1935      $FILLDATE;             !Fill the send buffer with the date
1936      RETRIES = ONE;        !Set the retry flag
1937      end
1938      else
1939      begin
1940      GMANID (MSGADR, SND_BUF, A, #0'17777', 8, 10, NO);

```

```

:      1941          end;
:      1942
:      1943          PRINTB (CRLF);
:      1944
:      1945          if SEND DATA ( ) then DECODE ( );
:      1946
:      1947          end;
:      1948
:      1949          [7] :          !Enter a non zero serial number
:      1950          begin
:      1951          GMANID (MSGADR, SND BUF, A, #0'177777', 1, 64, NO);
:      1952          PRINTB (CRLF);
:      1953
:      1954          if SEND_DATA ( ) then DECODE ( );
:      1955
:      1956          end;
:      1957
:      1958          [otherwise] :          !This message number is unknown
:      1959          begin
:      1960          RET_STATUS = UMN_CODE;          !Unknown message number error code
:      1961          DECODE ( );          !Report error and die
:      1962          end;
:      1963          tes;
:      1964
:      1965          end;
:      1966
:      1967          [2] :          !Default Question
:      1968          begin
:      1969
:      1970          selectoneu .REC BUF [MSG_NUM] of
:      1971          set
:      1972
:      1973          [1] :          !Enter unit number to format <0>
:      1974          begin
:      1975          !
:      1976          ! If in un-attended reformat mode then give to the DM the
:      1977          ! unit number obtained from the hardware P Table else let
:      1978          ! the DM prompt the operator for the unit to format.
:      1979          !
:      1980
:      1981          if .SW_UNATT          !Is e host running in unattended mode
:      1982          then
:      1983          begin
:      1984          SND_BUF = .UNIT NO;
:      1985          PRINTB (FMT7, .UNIT NO);          !Report which unit is being formatted
:      1986          end
:      1987          else
:      1988          begin
:      1989          GMANID (MSGADR, SND BUF, A, #0'177777', 1, 3, YES);
:      1990          end;
:      1991
:      1992          PRINTB (CRLF);
:      1993
:      1994          if SEND DATA ( ) then DECODE ( );
:      1995
:      1996          end;
:      1997

```

C11

ZRCMB4
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK5 Apr-1984 13:44:04
23 Sep-1983 14:13:18VAX-11 Bliss-16 V3-555
SPIDER#USERS:(NEALE,AZTEC)ZRCMB4.B16;2 (7)SEQ 0132
Page 12

```

: 1998      [4] :           !Use existing bad block information <N>
: 1999      begin
: 2000      :
: 2001      : If in un-attended reformat mode then tell the DM to
: 2002      : do a reformat mode else let the DM prompt the operator
: 2003      : for the formatting mode to use for this unit.
: 2004      :
: 2005      :
: 2006      if .SW_UNATT           !Is the host running in unattended mode
: 2007      then
: 2008      begin
: 2009      SND_BUF = %c'Y';         !Default to do reformat mode
: 2010      end
: 2011      else
: 2012      begin
: 2013      GMANID (MSGADR, SND_BUF, A, %o'177777', 1, 3, YES);
: 2014      end;
: 2015      PRINTB (CRLF);
: 2016      if SEND_DATA () then DECODE ();
: 2017      :
: 2018      :
: 2019      :
: 2020      end;
: 2021      :
: 2022      [5] :           !Use down line load <N>
: 2023      begin
: 2024      GMANID (MSGADR, SND_BUF, A, %o'177777', 1, 3, YES);
: 2025      PRINTB (CRLF);
: 2026      :
: 2027      if SEND_DATA () then DECODE ();
: 2028      :
: 2029      :
: 2030      end;
: 2031      [6] :           !Continue if bad block information is inaccessible <N>
: 2032      begin
: 2033      :
: 2034      if .SW_UNATT           !Is the host running in unattended mode
: 2035      then
: 2036      begin
: 2037      SND_BUF = %c'N';         !Default not to continue
: 2038      end
: 2039      else
: 2040      begin
: 2041      GMANID (MSGADR, SND_BUF, A, %o'177777', 1, 3, YES);
: 2042      end;
: 2043      :
: 2044      PRINTB (CRLF);
: 2045      if SEND_DATA () then DECODE ();
: 2046      :
: 2047      :
: 2048      end;
: 2049      :
: 2050      [otherwise] :         !This message number is unknown
: 2051      begin
: 2052      RET_STATUS = UMN CODE;     !Unknown message number error code
: 2053      DECODE ();               !Report error and die
: 2054      end;

```

ZRCMB4
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK5 Apr-1984 13:44:04
23 Sep-1983 14:13:18VAX-11 Bliss 16 V3-555
SPIDER#USERS:(NEALE.AZTEC)ZRCMB4.B16.2(7)
SEQ 0133
Page 13

```

:      2055          tes;
:      2056
:      2057          end;
:      2058
:      2059      [3] :          !Informational message
:      2060          begin
:      2061
:      2062          selectoneu .REC_BUF [MSG_NUM] of
:      2063          set
:      2064
:      2065          [0, 1, 2, 3, 4, 5, 7, 9, 11, 14, 16] : !All possible information msg s
:      2066          begin
:      2067          PRINTB (FMT1, MSGADR);
:      2068          end;
:      2069
:      2070          [otherwise] :          !This message number is unknown
:      2071          begin
:      2072          RET_STATUS = UMN_CODE;          !Unknown message number error code
:      2073          DECODE ();          !Report error and die
:      2074          end;
:      2075          tes;
:      2076
:      2077          end;
:      2078
:      2079      [4] :          !Termination message
:      2080          begin
:      2081
:      2082          selectoneu .REC_BUF [MSG_NUM] of
:      2083          set
:      2084
:      2085          [12, 13] :          !Abort and normal termination type
:      2086          begin
:      2087          PRINTB (FMT1, MSGADR);
:      2088          WRT_RC25 (RCIP, ONES);          !Stop the remote program
:      2089          exitloop          !Return to init code
:      2090          end;
:      2091
:      2092          [otherwise] :          !This message number is unknown
:      2093          begin
:      2094          RET_STATUS = UMN_CODE;          !Unknown message number error code
:      2095          DECODE ();          !Report error and die
:      2096          end;
:      2097          tes;
:      2098
:      2099          end;
:      2100
:      2101      [5] :          !Fatal error message
:      2102          begin
:      2103
:      2104          selectoneu .REC_BUF [MSG_NUM] of
:      2105          set
:      2106
:      2107          [1 to 24] :          !Error messages are from 1 to 24
:      2108          begin
:      2109          PRINTB (FMT1, MSGADR);
:      2110          WRT_RC25 (RCIP, ONES);          !Stop the remote program
:      2111          DOCLN;          !Kill this formatter and return to init code

```

```

:      2112          end;
:      2113
:      2114          [otherwise] :          !This message number is unknown
:      2115          begin
:      2116          RET_STATUS = UMN CODE;          !Unknown message number error code
:      2117          DECODE ();          !Report error and die
:      2118          end;
:      2119          tes;
:      2120
:      2121          end;
:      2122
:      2123          [6] :          !Special message type
:      2124          begin
:      2125
:      2126          selectoneu .REC_BUF [MSG_NUM] of
:      2127          set
:      2128
:      2129          [1] :          !FCT sector request
:      2130          begin
:      2131          !
:      2132          ! If this is the first FCT sector request (sector 0) then ask
:      2133          ! the operator for the external FCT file name and open the file.
:      2134          ! The supervisor will error and return to DRS> monitor if the
:      2135          ! requested file is not found on the local load media.
:      2136          !
:      2137
:      2138          if .REC_BUF [MSG_TXT] eq1 FCT_SEC_0
:      2139          then
:      2140          begin
:      2141          GMANID (FCT_REQ_MSG, SND_BUF, A, #0'177777', 1, 10, NO);
:      2142          OPEN (SND_BUF);
:      2143          end;
:      2144
:      2145          !
:      2146          ! Load into the FCT_BUF the next sequential 256 FCT sector words
:      2147          !
:      2148
:      2149          incr FCT_WRD from FCT_BUF to FCT_BUF + #0'776' by #0'2' do
:      2150          begin
:      2151          EOF = GETWORD (.FCT_WRD);
:      2152          end;
:      2153
:      2154          !
:      2155          ! Test to see if the EOF indicator was returned from the GETWORD macro
:      2156          ! before 16 FCT sectors were read indicating an illegal FCT file format.
:      2157          !
:      2158          ! End of file is reached when .EOF = 0;
:      2159          !
:      2160
:      2161          if ((.EOF eq1 ZERO) and (.REC_BUF [MSG_TXT] lss FCT_SEC_15)) or (.REC_BUF [MSG_TXT] gtr
:      2162          FCT_SEC_15)
:      2163          then
:      2164          begin
:      2165          SND_BUF [0] = ONES;          !Indicate failure to get good FCT file
:      2166
:      2167          if SEND DATA () then DECODE (); !Send failure to DM
:      2168

```



```

:      2169          RET_STATUS = ILL_FCT;      !Host will also report the failure
:      2170          DECODE ();                !Error out and die
:      2171          end;
:      2172
:      2173          :
:      2174          : Return to the remote program a success indicator and the
:      2175          : unibus lo and hi adrs of where this FCT sector can be found
:      2176          : and send the information back to the remote program.
:      2177          :
:      2178          SND_BUF [0] = ZERO;            !Indicate success
:      2179          SND_BUF [1] = FCT_BUF;      !Lo unibus adrs of FCT sector
:      2180          SND_BUF [2] = ZERO;        !Hi unibus adrs of FCT sector
:      2181
:      2182          if SEND_DATA () then DECODE (); !Send reply back to DM
:      2183
:      2184          :
:      2185          : Close this External FCT file if this is FCT sector 15.
:      2186          :
:      2187          :
:      2188          if .REC_BUF [MSG_TXT] eq1 FCT SEC_15 then CLOSE;
:      2189
:      2190          end;
:      2191
:      2192          [otherwise] :
:      2193          begin
:      2194          RET_STATUS = UMN_CODE;
:      2195          DECODE ();
:      2196          end;
:      2197          tes;
:      2198
:      2199          end;
:      2200
:      2201          [otherwise] :                !This message number is unknown
:      2202          begin
:      2203          RET_STATUS = UMT CODE;        !Unknown message number error code
:      2204          DECODE ();                !Report error and die
:      2205          end;
:      2206          tes;
:      2207
:      2208          end;
:      2209
:      2210          ENDTST;

```

```

.TITLE ZRCMB4 ZRCMB0 RC25 DISK FORMATTER
.IDENT /REV B /

.GLOBL ABORT, GET.DUST.STATUS, EX.SUP.PROG
.GLOBL EX.LOC.PROG, REC.DATA, SEND.DATA
.GLOBL SET.CNTRL.CHAR, ON.LINE, DUP$I.SERVICE
.GLOBL INT$I.SERVICE, INIT.COM.AREA, BOOT.RC25
.GLOBL DECODE, SW.UNATT, DATETXT, LUN
.GLOBL UNIT.NO, FLG.WRD, ISD.STRUCT, PTBL.PTR
.GLOBL NEX.FLAG, NXT.CRN, RET.EN$AD, FCT.BUF
.GLOBL SND.BUF, REC.BUF, MSGADR, NSD.SLOT
.GLOBL NRD.SLOT, VEC.ADDR, RET.STATUS
.GLOBL PID.SAVE, RC25.ADDR, BOOT.FAILURE

```

.GLOBL PROTO.VIOLATION, PORT.INIT.ERR
.GLOBL ACTIVE.DUP.SERVER, INACTIVE.DUP.SERVER
.GLOBL FCT.REQ.MSG, DATMSG, CRLF, FMT1
.GLOBL FMT5, FMT6, FMT7

.SBTTL \$T1 DOWN LINE LOAD FORMATTER AND MONITOR TASK
.PSECT AB\$CODE, RO

Address	Offset	Label	Instruction	Comment	PC
000000	004167	000000G			
000004	024646		\$T1: JSR R1,\$SAVE3		1611
000006	142767	000002 000000G	CMP (SP),(SP)		
000014	005067	000000G	BICB #2,FLG.WRD		1642
000020	012746	000340	CLR NEX.FLAG		1643
000024	012746	000000G	MOV #340,-(SP)		1644
000030	012746	000004	MOV #INT\$I.SERVICE,-(SP)		
000034	012746	000003	MOV #4,-(SP)		
000040	104437		MOV #3,(SP)		
000042	017766	000000G 000010	TRAP 37		
000050	062766	000702 000010	MOV #PTBL.PTR,10(SP)	; *.TEMP	1645
000056	032776	000001 000010	ADD #2,10(SP)	; *.TEMP	1646
000064	001402		BIT #1,@10(SP)	; *.TEMP	1648
000066	005066	000012	BEQ 1\$		
000072	012700	000004	CLR 12(SP)	; DUMMY	1651
000076	104436		MOV #4,R0		1654
000100	026727	000000G 177777	TRAP 36		
000106	001015		CMP NEX.FLAG,#1		1656
000110	017716	000000G	BNE 2\$		
000114	012746	000000G	MOV #PTBL.PTR,(SP)		1659
000120	012746	000002	MOV #FMT6,(SP)		
000124	010600		MOV #2,(SP)		
000126	104414		MOV SP,R0	; SP.*	
000130	152767	000002 000000G	TRAP 14		
000136	104444		BISB #2,FLG.WRD		1660
000140	022626		TRAP 44		
000142	005000		CMP (SP)*,(SP)*		1658
000144	104441		CLR R0		1681
000146	016700	000000G	TRAP 41		
000152	104436		MOV VEC.ADDR,R0		1682
000154	012716	000340	TRAP 36		
000160	012746	000000G	MOV #340,(SP)		1683
000164	016746	000000G	MOV #INT\$I.SERVICE,(SP)		
000170	012746	000003	MOV VEC.ADDR,(SP)		
000174	104437		MOV #3,(SP)		
000176	012703	177777	TRAP 37		
000202	004767	000000G	MOV #1,R3	; *.RETRIES	1688
000206	010067	000000G	\$3: JSR PC,BOOT.RC25		1692
000212	005203		MOV R0,RET.STATUS		
000214	032700	000001	INC R3	; RETRIES	1693
000220	001003		BIT #1,R0	; *.RET.STATUS	1695
000222	020327	000001	BNE 4\$		
000226	001365		CMP R3,#1	; RETRIES.*	
000230	032767	000001 000000G	BNE 3\$		
000236	001010		\$4: BIT #1,RET.STATUS		1701
000240	012716	000000G	BNE 5\$		
000244	012746	000001	MOV #BOOT.FAILURE,(SP)		1704
000250	010600		MOV #1,(SP)		
			MOV SP,R0	; SP.*	

H I I

ZRCH84
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

5 Apr 1984 13:44:04
23 Sep 1983 14:13:18

VAX 11 B119 16 V3 555
SPIDER\$USERS:[NEALE,AZTEC]ZRCH84.B16;2 (7)

SEQ 0137

Page 17

000252	104414			TRAP	14		
000254	104444			TRAP	44		
000256	005726			TST	(SP).	:	1703
000260	004767	000000G	5:	JSR	PC,INIT.COM.AREA	:	1716
000264	006000			ROR	R0		
000266	103016			BCC	6:		
000270	012716	000000G		MOV	#PROTO.VIOLATION,(SP)	:	1719
000274	012746	000001		MOV	#1,(SP)		
000300	010600			MOV	SP,R0	: SP,.	
000302	104414			TRAP	14		
000304	012716	000000G		MOV	#PORT.INIT.ERR,(SP)	:	1720
000310	012746	000001		MOV	#1,(SP)		
000314	010600			MOV	SP,R0	: SP,.	
000316	104414			TRAP	14		
000320	104444			TRAP	44		
000322	022626			CMP	(SP).,(SP).	:	1718
000324	016700	000000G	6:	MOV	VEC.ADDR,R0	:	1729
000330	104436			TRAP	36		
000332	012716	000340		MOV	#340,(SP)	:	1730
000336	012746	000000G		MOV	#DUP\$I.SERVICE,-(SP)		
000342	016746	000000G		MOV	VEC.ADDR,-(SP)		
000346	012746	000003		MOV	#3,(SP)		
000352	104437			TRAP	37		
000354	016701	000016G		MOV	ISD.STRUCT.16,R1	: #,RC\$M.REG	1735
000360	052701	000001		BIS	#1,R1	: #,RC\$M.REG	
000364	016700	000000G		MOV	RC25.ADDR,R0		
000370	010160	000002		MOV	R1.2(R0)	: RC\$M.REG,.	
000374	004767	000000G		JSR	PC.SET.CNTRLR.CHAR	:	1758
000400	006000			ROR	R0		
000402	103002			BCC	7:		
000404	004767	000000G		JSR	PC.DECODE		
000410	016716	000000G	7:	MOV	UNIT.NO,(SP)	:	1765
000414	016746	000000G		MOV	LUN,-(SP)		
000420	012746	000000G		MOV	#FMT5,-(SP)		
000424	012746	000003		MOV	#3,(SP)		
000430	010600			MOV	SP,R0	: SP,.	
000432	104414			TRAP	14		
000434	004767	000000G		JSR	PC.ON.LINE	:	1768
000440	006000			ROR	R0		
000442	103002			BCC	8:		
000444	004767	000000G		JSR	PC.DECODE		
000450	004767	000000G	8:	JSR	PC.GET.DUST.STATUS	:	1795
000454	006000			ROR	R0		
000456	103002			BCC	9:		
000460	004767	000000G		JSR	PC.DECODE		
000464	016700	000000G	9:	MOV	RET.EN\$AD,R0	:	1801
000470	032760	004000 000022		BIT	#4000,22(R0)		
000476	001410			BEQ	10:		
000500	012716	000000G		MOV	#ACTIVE.DUP.SERVER,(SP)	:	1804
000504	012746	000001		MOV	#1,(SP)		
000510	010600			MOV	SP,R0	: SP,.	
000512	104414			TRAP	14		
000514	104444			TRAP	44		
000516	005726			TST	(SP).	:	1803
000520	004767	000000G	10:	JSR	PC.EX.SUP.PROG	:	1814
000524	006000			ROR	R0		
000526	103002			BCC	11:		

000530	004767	000000G		JSR	PC,DECODE			
000534	004767	000000G	11:	JSR	PC,GET.DUST.STATUS	:	1825	
000540	006000			ROR	R0			
000542	103002			BCC	12:			
000544	004767	000000G		JSR	PC,DECODE			
000550	016701	000000G	12:	MOV	RET.EN\$AD,R1	:	1831	
000554	032761	004000	000022	BIT	#4000,22(R1)			
000562	001011			BNE	13:			
000564	012716	000000G		MOV	#INACTIVE.DUP.SERVER,(SP)	:	1834	
000570	012746	000001		MOV	#1,-(SP)			
000574	010600			MOV	SP,R0	: SP,*		
000576	104414			TRAP	14			
000600	104444			TRAP	44			
000602	005726			TST	(SP),	:	1833	
000604	000406			BR	14:		1831	
000606	016167	000024	000000G	MOV	24(R1),PID.SAVE	:	1839	
000614	016167	000026	000002G	MOV	26(R1),PID.SAVE+2	:	1840	
000622	005003			CLR	R3	: RETRIES	1850	
000624	004767	000000G		15:	JSR	PC,REC.DATA	:	1882
000630	006000			ROR	R0			
000632	103002			BCC	16:			
000634	004767	000000G		JSR	PC,DECODE			
000640	005000			CLR	R0	: I		
000642	005060	000000G		17:	CLR	SND.BUF(R0)	: *(I)	
000646	062700	000002		ADD	#2,R0	: *,I		
000652	020027	000110		CMP	R0,#110	: I,*		
000656	101771			BLOS	17:			
000660	016701	000000G		MOV	REC.BUF,R1	:	1894	
000664	006201			ASR	R1			
000666	006201			ASR	R1			
000670	006201			ASR	R1			
000672	006201			ASR	R1			
000674	000301			SWAB	R1			
000676	042701	177760		BIC	#177760,R1			
000702	020127	000001		CMP	R1,#1			
000706	001107			BNE	24:			
000710	016700	000000G		MOV	REC.BL^,R0	:	1906	
000714	042700	170000		BIC	#170000,R0			
000720	001060			BNE	23:			
000722	032767	000001	000000G	BIT	#1,SW.UNATT	:	1917	
000730	001435			BEQ	21:			
000732	032703	000001		BIT	#1,R3	: *,RETRIES	1921	
000736	001417			BEQ	19:			
000740	005000			CLR	R0	: I	1929	
000742	105060	000000G	18:	CLRB	DATETXT(R0)	: *(I)	1930	
000746	005200			INC	R0	: I	1929	
000750	020027	000013		CMP	R0,#13	: I,*		
000754	003772			BLE	18:			
000756	104443			TRAP	43	:	1932	
000760	000406			.WORD	406			
000762	000000G			.WORD	DATETXT			
000764	000142			.WORD	142			
000766	000000G			.WORD	DATMSG			
000770	177777			.WORD	-1			
000772	000010			.WORD	10			
000774	000012			.WORD	12			
000776	005000		19:	CLR	R0	: I	1933	

ZRCH84
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

5-Apr-1984 13:44:04
23 Sep-1983 14:13:18

SEQ 0139
Page 19
VAX-11 Bliss 16 V3-555
SPIDER#USERS:[NEALE.AZTEC]ZRCH84.B16;2(7)

001000	116060	000000G	000000G	20\$:	MOVB	DATEXT(RO),SND.BUF(RO)	:	*(I),*(I)	
001006	005200				INC	RO	:	I	
001010	020027	000013			CMP	RO,#13	:	I,*	
001014	101771				BLOS	20\$:		
001016	012703	000001			MOV	#1,R3	:	*,RETRIES	1936
001022	000410				BR	22\$:		1917
001024	104443			21\$:	TRAP	43	:		1940
001026	000406				.WORD	406			
001030	000000G				.WORD	SND.BUF			
001032	000142				.WORD	142			
001034	000000G				.WORD	MSGADR			
001036	177777				.WORD	1			
001040	000010				.WORD	10			
001042	000012				.WORD	12			
001044	012716	000000G		22\$:	MOV	#CRLF,(SP)	:		1943
001050	012746	000001			MOV	#1,-(SP)	:		
001054	010600				MOV	SP,RO	:	SP,*	
001056	104414				TRAP	14	:		
001060	000547				BR	31\$:		1945
001062	020027	000007		23\$:	CMP	RO,#7	:		1906
001066	001153				BNE	33\$:		
001070	104443				TRAP	43	:		1951
001072	000406				.WORD	406			
001074	000000G				.WORD	SND.BUF			
001076	000142				.WORD	142			
001100	000000G				.WORD	MSGADR			
001102	177777				.WORD	-1			
001104	000001				.WORD	1			
001106	000100				.WORD	100			
001110	012716	000000G			MOV	#CRLF,(SP)	:		1952
001114	012746	000001			MOV	#1,-(SP)	:		
001120	010600				MOV	SP,RO	:	SP,*	
001122	104414				TRAP	14	:		
001124	000525				BR	31\$:		1954
001126	020127	000002		24\$:	CMP	R1,#2	:		1894
001132	001175				BNE	39\$:		
001134	016700	000000G			MOV	REC.BUF,RO	:		1970
001140	042700	170000			BIC	#170000,RO	:		
001144	020027	000001			CMP	RO,#1	:		
001150	001040				BNE	27\$:		
001152	032767	000001	000000G		BIT	#1,SW.UNATT	:		1981
001160	001415				BEQ	25\$:		
001162	016767	000000G	000000G		MOV	UNIT.NO,SND.BUF	:		1984
001170	016716	000000G			MOV	UNIT.NO,(SP)	:		1985
001174	012746	000000G			MOV	#FMT7,(SP)	:		
001200	012746	000002			MOV	#2,(SP)	:		
001204	010600				MOV	SP,RO	:	SP,*	
001206	104414				TRAP	14	:		
001210	022626				CMP	(SP)*,(SP)*	:		1983
001212	000410				BR	26\$:		1981
001214	104443			25\$:	TRAP	43	:		1989
001216	000406				.WORD	406			
001220	000000G				.WORD	SND.BUF			
001222	000152				.WORD	152			
001224	000000G				.WORD	MSGADR			
001226	177777				.WORD	1			
001230	000001				.WORD	1			

ZRCH84
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

5-Apr-1984 13:44:04
23-Sep-1983 14:13:18

SEQ 0140
Page 20
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH84.B16;2 (7)

001232	000003			.WORD	3			
001234	012716	000000G	26\$:	MOV	0CRLF,(SP)	:		1992
001240	012746	000001		MOV	01,-(SP)	:		
001244	010600			MOV	SP,RO	:	SP,*	
001246	104414			TRAP	14	:		
001250	000453			BR	31\$:		1994
001252	020027	000004	27\$:	CMP	RO,04	:		1970
001256	001027			BNE	30\$:		
001260	032767	000001	00C000G	BIT	01,SW.UNATT	:		2006
001266	001404			BEQ	28\$:		
001270	012767	000131	000000G	MOV	0131,SND.BUF	:		2009
001276	000410			BR	29\$:		2006
001300	104443		28\$:	TRAP	43	:		2013
001302	000406			.WORD	406	:		
001304	000000G			.WORD	SND.BUF	:		
001306	000152			.WORD	152	:		
001310	000000G			.WORD	MSGADR	:		
001312	177777			.WORD	-1	:		
001314	000001			.WORD	1	:		
001316	000003			.WORD	3	:		
001320	012716	000000G	29\$:	MOV	0CRLF,(SP)	:		2016
001324	012746	000001		MOV	01,-(SP)	:		
001330	010600			MOV	SP,RO	:	SP,*	
001332	104414			TRAP	14	:		
001334	000421			BR	31\$:		2018
001336	020027	000005	30\$:	CMP	RO,05	:		1970
001342	001023			BNE	32\$:		
001344	104443			TRAP	43	:		2024
001346	000406			.WORD	406	:		
001350	000000G			.WORD	SND.BUF	:		
001352	000152			.WORD	152	:		
001354	000000G			.WORD	MSGADR	:		
001356	177777			.WORD	-1	:		
001360	000001			.WORD	1	:		
001362	000003			.WORD	3	:		
001364	012716	000000G		MOV	0CRLF,(SP)	:		2025
001370	012746	000001		MOV	01,-(SP)	:		
001374	010600			MOV	SP,RO	:	SP,*	
001376	104414			TRAP	14	:		
001400	004767	000000G	31\$:	JSR	PC.SEND.DATA	:		2027
001404	006000			ROR	RO	:		
001406	103436			BLO	36\$:		
001410	000437			BR	37\$:		2023
001412	020027	000006	32\$:	CMP	RO,06	:		1970
001416	001036		33\$:	BNE	38\$:		
001420	032767	000001	000000G	BIT	01,SW.UNATT	:		2034
001426	001404			BEQ	34\$:		
001430	012767	000116	000000G	MOV	0116,SND.BUF	:		2037
001436	000410			BR	35\$:		2034
001440	104443		34\$:	TRAP	43	:		2041
001442	000406			.WORD	406	:		
001444	000000G			.WORD	SND.BUF	:		
001446	000152			.WORD	152	:		
001450	000000G			.WORD	MSGADR	:		
001452	177777			.WORD	-1	:		
001454	000001			.WORD	1	:		
001456	000003			.WORD	3	:		

ZRCH84
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

5-Apr-1984 13:44:04
23-Sep-1983 14:13:18

SEQ 0141
Page 21
VAX-11 Bliss 16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCH84.B16;2 (7)

001460	012716	000000G		35\$:	MOV	#CRLF,(SP)	:		2044
001464	012746	000001			MOV	#1,-(SP)			
001470	010600				MOV	SP,R0	:	SP,*	
001472	104414				TRAP	14			
001474	004767	000000G			JSR	PC,SEND.DATA	:		2046
001500	006000				ROR	R0			
001502	103002				BCC	37\$			
001504	004767	000000G		36\$:	JSR	PC,DECODE			
001510	005726			37\$:	TST	(SP)*	:		2032
001512	000533				BR	44\$:		1970
001514	012767	004001	000000G	38\$:	MOV	#4001,RET.STATUS	:		2052
001522	000167	000532			JMP	57\$:		2053
001526	020127	000003		39\$:	CMP	R1,#3	:		1894
001532	001037				BNE	41\$			
001534	016700	000000G			MOV	REC.BUF,R0	:		2062
001540	042700	170000			BIC	#170000,R0			
001544	020027	000005			CMP	R0,#5			
001550	101417				BLOS	40\$			
001552	020027	000007			CMP	R0,#7			
001556	001414				BEQ	40\$			
001560	020027	000011			CMP	R0,#11			
001564	001411				BEQ	40\$			
001566	020027	000013			CMP	R0,#13			
001572	001406				BEQ	40\$			
001574	020027	000016			CMP	R0,#16			
001600	001403				BEQ	40\$			
001602	020027	000020			CMP	R0,#20			
001606	001342				BNE	38\$			
001610	012716	000000G		40\$:	MOV	#MSGADR,(SP)	:		2067
001614	012746	000000G			MOV	#FMT1,-(SP)			
001620	012746	000002			MOV	#2,-(SP)			
001624	010600				MOV	SP,R0	:	SP,*	
001626	104414				TRAP	14			
001630	000463				BR	43\$:		2066
001632	020127	000004		41\$:	CMP	R1,#4	:		1894
001636	001030				BNE	42\$			
001640	016700	000000G			MOV	REC.BUF,R0	:		2082
001644	042700	170000			BIC	#170000,R0			
001650	020027	000014			CMP	R0,#14			
001654	103717				BLO	38\$			
001656	020027	000015			CMP	R0,#15			
001662	101314				BHI	38\$			
001664	012716	000000G			MOV	#MSGADR,(SP)	:		2087
001670	012746	000000G			MOV	#FMT1,(SP)			
001674	012746	000002			MOV	#2,-(SP)			
001700	010600				MOV	SP,R0	:	SP,*	
001702	104414				TRAP	14			
001704	022626				CMP	(SP)*,(SP)*			
001706	012700	177777			MOV	#-1,R0	:	*,RC\$M.REG	2088
001712	010077	000000G			MOV	R0,#RC25.ADDR	:	RC\$M.REG,*	
001716	000564				BR	59\$:		2086
001720	020127	000005		42\$:	CMP	R1,#5	:		1894
001724	001027				BNE	45\$			
001726	016700	000000G			MOV	REC.BUF,R0	:		2104
001732	042700	170000			BIC	#170000,R0			
001736	001666				BEQ	38\$			
001740	020027	000030			CMP	R0,#30			

001744	101263		BHI	38\$		
001746	012716	000000G	MOV	#MSGADR,(SP)	:	2109
001752	012746	000000G	MOV	#FMT1,(SP)		
001756	012746	000002	MOV	#2,-(SP)		
001762	010600		MOV	SP,R0	: SP,*	
001764	104414		TRAP	14		
001766	012700	177777	MOV	#-1,R0	: *,RC#M.REG	2110
001772	010077	000000G	MOV	R0,#RC25.ADDR	: RC#M.REG,*	
001776	104444		TRAP	44		
002000	022626		43\$: CMP	(SP)+,(SP)+	:	2108
002002	000530		44\$: BR	58\$:	2104
002004	020127	000006	45\$: CMP	R1,#6	:	1894
002010	001120		BNE	56\$		
002012	016700	000000G	MOV	REC.BUF,R0	:	2126
002016	042700	170000	BIC	#170000,R0		
002022	020027	000001	CMP	R0,#1		
002026	001232		BNE	38\$		
002030	005767	000002G	TST	REC.BUF+2	:	2138
002034	001013		BNE	46\$		
002036	104443		TRAP	43	:	2141
002040	000406		.WORD	406		
002042	000000G		.WORD	SND.BUF		
002044	000142		.WORD	142		
002046	000000G		.WORD	FCT.REQ.MSG		
002050	177777		.WORD	-1		
002052	000001		.WORD	1		
002054	000012		.WORD	12		
002056	012700	000000G	MOV	#SND.BUF,R0	:	2142
002062	104434		TRAP	34		
002064	012701	000000G	46\$: MOV	#FCT.BUF,R1	: *,FCT.WRD	2149
002070	000412		BR	50\$		
002072	010100		47\$: MOV	R1,R0	: FCT.WRD,*	2151
002074	104427		TRAP	27		
002076	103004		BHIS	48\$		
002100	010011		MOV	R0,(R1)	: R0,FCT.WRD	
002102	012702	000001	MOV	#1,R2	: *,EOF	
002106	000401		BR	49\$		
002110	005002		48\$: CLR	R2	: EOF	
002112	062701	000002	49\$: ADD	#2,R1	: *,FCT.WRD	2149
002116	020127	000776G	50\$: CMP	R1,#FCT.BUF+776	: FCT.WRD,*	
002122	003763		BLE	47\$		
002124	005702		TST	R2	: EOF	2161
002126	001004		BNE	51\$		
002130	026727	000002G 000017	CMP	REC.BUF+2,#17		
002136	002404		BLT	52\$		
002140	026727	000002G 000017	51\$: CMP	REC.BUF+2,#17		
002146	003416		BLE	54\$		
002150	012767	177777 000000G	52\$: MOV	#-1,SND.BUF	:	2165
002156	004767	000000G	JSR	PC,SEND.DATA	:	2167
002162	006000		ROR	R0		
002164	103002		BCC	53\$		
002166	004767	000000G	JSR	PC,DECODE		
002172	012767	006001 000000G	53\$: MOV	#6001,RET.STATUS	:	2169
002200	004767	000000G	JSR	PC,DECODE	:	2170
002204	005067	000000G	54\$: CLR	SND.BUF	:	2178
002210	012767	000000G 000002G	MOV	#FCT.BUF,SND.BUF+2	:	2179
002216	005067	000004G	CLR	SND.BUF+4	:	2180

N11

ZRCH84 ZRCH80 RC25 DISK FORMATTER
 REV B PATCH 00 DOWN LINE LOAD FORMATTER AND MONITOR TASK

5-Apr-1984 13:44:04
 23-Sep 1983 14:13:18

SEQ 0143
 Page 23
 VAX-11 Bliss-16 V3 555
 SPIDER\$USERS:[NEALE.AZTEC]ZRCH80.L16;2 (7)

002222	004767	000000G		JSR	PC,SEND.DATA	:	2182
002226	006000			ROR	R0	:	
002230	103002			BCC	55\$:	
002232	004767	000000G		JSR	PC,DECODE	:	
002236	026727	000002G 000017	55\$:	CMP	REC.BUF+2,017	:	2188
002244	001007			BNE	58\$:	
002246	104435			TRAP	35	:	
002250	000405			BR	58\$:	2126
002252	012767	001001 000000G	56\$:	MOV	01001,RET.STATUS	:	2203
002260	004767	000000G	57\$:	JSR	PC,DECODE	:	2204
002264	000167	176334	58\$:	JMP	15\$:	1852
002270	062706	000036	59\$:	ADD	036,SP	:	1611
002274	000207			RTS	PC	:	

: Routine Size: 607 words, Routine Base: AB\$CODE + 0000
 : Maximum stack depth per invocation: 23 words

000000	004767	175476		.SBTTL	T1 DOWN LINE LOAD FORMATTER AND MONITOR TASK		
000000			T1::				
000004	104466		1\$:	JSR	PC,\$T1	:	2208
000006	006000			TRAP	66	:	
000010	103773			ROR	R0	:	
000012	000207			BLO	1\$:	
				RTS	PC	:	

: Routine Size: 6 words, Routine Base: AB\$CODE + 2276
 : Maximum stack depth per invocation: 2 words

: 2211 end
 : 2212
 : 2213 eludom

:
 : OTS external references
 : .GLOBL \$SAVE3

:
 : PSECT SUMMARY
 :
 : Psect Name Words Attributes
 : AB\$CODE 613 R0 , I , LCL, REL, CON

:
 : LIBRARY STATISTICS
 :
 : File Total Symbols Loaded Percent Blocks Read

: SPIDER\$USERS:[NEALE.AZTEC]ZRCH80.L16;2	398	168	42	38
---	-----	-----	----	----

B12

ZRCH84 ZRCH80 RC25 DISK FORMATTER
REV B PATCH 00 DOWN LINE LOAD FORMATTER AND MONITOR TASK

5 Apr-1984 13:44:04
23 Sep 1983 14:13:18

SEQ 0144
Page 24
VAX-11 Bliss 16 V3 555
SPIDER#USERS:(NEALE.AZTEC)ZRCH84.B16;2 (7)

COMMAND QUALIFIERS

:
: BLISS /PDP11/LIST ZRCH84.916
: Size: 613 code * 0 data words
: Run Time: 00:24.7
: Elapsed Time: 02:14.8
: Memory Used: 386 pages
: Compilation Complete

```

: 0001 MODULE ZRCHB5 (*TITLE 'ZRCHB0 RC25 DISK FORMATTER'
: 0002 IDENT = 'REV B PATCH 00',
: 0003 ADDRESSING_MODE (RELATIVE) ,
: 0004 ENVIRONMENT (NOEIS)
: 0005 ) *
: 0006 BEGIN
: 0007 *$bttl 'MODULE DECLARATIONS'
: 0008 !
: 0009 ! Pretty Declarations
: 0010 !
: 0011 !<blf/lowercase_key>
: 0012 !
: 0013 !
: 0014 library 'ZRCHB0'; !Define RC25 Formatter library
: 0015 !
: 0016 require 'BLSMAC.REQ'; !Define Bliss Macro require file
: 1505 !
: 1506 !*
: 1507 ! A forward-routine-declaration declares a name to be a routine-name whose
: 1508 ! definition is given later in the same block, and associates with that name the
: 1509 ! set of attributes needed for generation of call to the named routine.
: 1510 !
: 1511 ! The following is a list of all routines declared within this module and can
: 1512 ! be noted that this module is only place where global routines are declared.
: 1513 !-
: 1514 !
: 1515 forward routine
: 1516 LOAD_FILE, !Load file from local load media
: 1517 GET_NSD, !Get next send descriptor slot index
: 1518 GET_NRD, !Get next receive descriptor slot index
: 1519 LOAD_OUT$STD_BUF, !Load out standing command buffer
: 1520 GET_CMD$REF, !Get unique command reference number
: 1521 DECODE : novalue, !Decode return status error code
: 1522 DUP$I_SERVICE : INT_LNK$TYP novalue, !Dup/UQ port interrupt service routine
: 1523 CTO_WAIT, !Command time out wait
: 1524 ABORT, !Abort Dup command
: 1525 GET_DUST_STATUS, !Get Dust Status command
: 1526 EX_SUP_PROG, !Execute Supplied Program command
: 1527 EX_LOC_PROG, !Execute Local Program command
: 1528 SEND_DATA, !Send Data command
: 1529 REC_DATA, !Receive Data command
: 1530 SET_CNTRLR_CHAR, !Set Controller Characteristics command
: 1531 ON_LINE, !On Line command
: 1532 INT$I_SERVICE : INT_LNK$TYP novalue, !Initialization sequence interrupt service
: 1533 IS_TIMER, !Initialization sequence time out wait
: 1534 BOOT_RC25, !Initialize sequence for RC25 controller
: 1535 INIT_COM_AREA, !Initialize UQ Port communication area
: 1536 !
: 1537 !<BLF/PAGE>

```

```

:      1538  !.
:      1539  ! The psect named "code or $code$" is redefined here to be called "ac$code".
:      1540  ! This is done to organize formatter routine code into a seperate psect.
:      1541  !-
:      1542  psect
:      1543      code = ac$code;
:      1544
:      1545  !.
:      1546  ! Structure declarations used within this module.
:      1547  !
:      1548
:      1549  structure
:      1550
:      1551      !.
:      1552      ! RC25 register accessing structure. This structure allows RC25 register
:      1553      ! accessing to be transportable between the PDP-11 and VAX Diagnostic Supervisors.
:      1554      !
:      1555      ! This also defines an access algorithm for VAX to allow field reference
:      1556      ! to MBA address space without generating machine checks.
:      1557      !-
:      1558
:      1559      RC25 [0, P, S, E] =
:      1560          begin
:      1561              local
:      1562                  RC$S_REG;
:      1563
:      1564                  RC$S_REG = .(RC25 * $upval+0)<0, $bpval, 0>;
:      1565                  RC$S_REG
:      1566                  end
:      1567                  <P, S, E>;
:      1568
:      1569
:      1570  !<blf/page>

```

```

:      1571  !*
:      1572  ! External Declaration of  datums declared outside of this module.
:      1573  !-
:      1574
:      1575  external
:      1576  !
:      1577  ! Communications area declarations
:      1578  !
:      1579  COM_AREA : blockvector [REC_ALLOCATE + SND_ALLOCATE + HDR_SIZ, 2, word],
:      1580  HEAD_AREA : ref block [4, word] field (HDR_FIELD),
:      1581  RECEIVE_RING : ref blockvector [REC_ALLOCATE, 2, word] field (DSC_FIELD),
:      1582  SEND_RING : ref blockvector [SND_ALLOCATE, 2, word] field (DSC_FIELD),
:      1583  REC_ENVELOPE : blockvector [REC_ALLOCATE, RB_SIZE + 2, word] field (ENV_FIELD),
:      1584  SND_ENVELOPE : blockvector [SND_ALLOCATE, SB_SIZE + 2, word] field (ENV_FIELD),
:      1585  RET_EN$AD : ref block [RB_SIZE + 2, word] field (ENV_FIELD),
:      1586  REC_BUF : block [RECB_SIZE, word] field (RECB_FIELD),
:      1587  SND_BUF : vector [SNDB_SIZE, word],
:      1588  OUT$STD_BUF : blockvector [REC_ALLOCATE, 2, word] field (OUT$FIELD),
:      1589  !
:      1590  ! Miscellaneous external data declarations
:      1591  !
:      1592  NEX_FLAG : word,                !Non-existent RC25 register flag
:      1593  NRD_SLOT : word,             !Next receive descriptor slot
:      1594  NSD_SLOT : word,             !Next send descriptor slot
:      1595  RC25_ADDR : ref RC25 field (ISD_FIELD), !Controller reg access struct
:      1596  RET_STATUS : word,           !Global return status location
:      1597  PID_SAVE : vector [2, word], !Saves program indicator field
:      1598  VEC_ADDR : word,              !RC25 interrupt vector address
:      1599  RSVD_STRUCT : vector [4, word], !Stores init seq reserved fields
:      1600  ISD_STRUCT : blockvector [4, 2, word] field (ISD_FIELD), !Init seq data
:      1601  UNIT_NO : word,               !P_table unit number to format
:      1602  NXT_CRN : byte,              !Stores next cmd ref number
:      1603  !
:      1604  ! Error Messages Structures
:      1605  !
:      1606  PFE_STRUCT : vector [23],     !Port fatal error msg struct
:      1607  EMSG_STRUCT : vector [23],   !Error message structure
:      1608  RC_STRUCTURE : vector [39],  !RC25 SA register fatal error
:      1609  SDUP_STRUCT : vector [7],    !DUP return status code messages
:      1610  SMSCP_STRUCT : vector [13],  !MSCP return status code messages
:      1611  !<blf/page>

```

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr 1984 13:46:24
11-Jan-1984 13:23:12

VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (4)

SEQ 0148
Page 4

```

:      1612      :
:      1613      : Formatted ascii strings
:      1614      :
:      1615      : PID FMT,                !Progress indicator printing string
:      1616      : FMT4,                  !Micro code version printing format
:      1617      :
:      1618      : DM code parameter declarations
:      1619      :
:      1620      : The following declarations are declared in module azkel6 'DM code
:      1621      : module' and point to the starting address of specific DM code buffer areas.
:      1622      :
:      1623      : (OVSA is declared in azkel2 and is a memory location however)
:      1624      :
:      1625      : It is a hack but the azfmtr storage allocation must be edited in manually
:      1626      : any time the DM code buffer changes. Get this value from the output of
:      1627      : DMCONV program. (Bliss wants this allocation value to be a compile time constant).
:      1628      :
:      1629      : AZFMTR : vector [8989, word],      !RC25 formatter DM code buffer
:      1630      : MDSA,                          !Host DM code header address
:      1631      : DMSA,                          !Host DM code initial load address
:      1632      : OVSA;                          !Host First overlay address
:      1633      :

```

```

:      1634 global routine LOAD_FILE (BUF LOC, FN EXT, BUF SIZ) = !Load file from local load media
:      1635
:      1636 !..
:      1637 ! Functional Description :
:      1638 !   This multi-purpose routine is used to read in files from the boot
:      1639 !   device into host memory space at the specified buffer address.
:      1640
:      1641 ! Formal Parameters :
:      1642 !   BUF_LOC           This contains the host memory address where
:      1643 !   words read in from the boot device are to
:      1644 !   be loaded.
:      1645
:      1646 !   FN_EXT           This contains a pointer to an asciz string
:      1647 !   of the file name to be read in.
:      1648
:      1649 !   BUF_SIZ         This contains the number of words allocated
:      1650 !   for the buffer where the file is to be read
:      1651 !   in. This is used for protect against buffer
:      1652 !   over runs.
:      1653 ! Implicit Inputs :
:      1654 !   none
:      1655
:      1656 ! Implicit Outputs :
:      1657 !   none
:      1658
:      1659 ! Completion Codes :
:      1660 !   FRE_CODE        Is returned if the end-of-file indicator is
:      1661 !   not returned before the buffer is full.
:      1662
:      1663 !   PAS_CODE        Is returned if end-of-file indicator is returned
:      1664 !   before the buffer is full.
:      1665
:      1666 ! Side Effects :
:      1667 !   Data previously loaded in the specified load buffer is
:      1668 !   destroyed by the incoming file from disk.
:      1669
:      1670
:      1671 begin
:      1672
:      1673 local
:      1674     NEOF,           !Return status from getword macro
:      1675     EOB;           !End of buffer address storage
:      1676
:      1677 OPEN (.FN_EXT);   !Open the file for input
:      1678 EOB = (.BUF SIZ*2) + .BUF_LOC; !Calculate the last buffer address
:      1679
:      1680 !..
:      1681 ! Read the first block from the input file and through
:      1682 ! it away since it only contains RI 11 information.
:      1683 !
:      1684
:      1685 incru i from 0 to 255 do
:      1686     begin
:      1687     GETWORD (.BUF LOC);
:      1688     end;
:      1689
:      1690 !..

```

```

:      1691      ! Loop on getting a word from the load file. After
:      1692      ! increment to the next word by adding %0'2' to the
:      1693      ! buffer load addrss. Continue getting words until
:      1694      ! the not complete indicator is returned (0) or the
:      1695      ! buffer is over run. Return an error code if the
:      1696      ! buffer is over run.
:      1697      !
:      1698      !
:      1699      do
:      1700          begin
:      1701              NEOF = GETWORD (.BUF LOC);           !Read next word into .mem_loc
:      1702              BUF_LOC = .BUF_LOC + %0'2';       !Increment to next word
:      1703          end
:      1704      until ( not .NEOF) or (.BUF_LOC eqv .EOB); !Repeat until eof or buf overrun
:      1705      !
:      1706      !
:      1707      ! Return an error code if not eof (neof = 1) else close the file and return a pass code.
:      1708      !
:      1709      !
:      1710      if .NEOF                                     !Was the eof indicator returned
:      1711      then
:      1712          return RET_STATUS = FRE_CODE           !Return error code to caller
:      1713      else
:      1714          begin
:      1715              CLOSE;                               !Eof was returned so close file
:      1716          end;
:      1717      !
:      1718      return RET_STATUS = PAS CODE;             !Return a pass code to the caller
:      1719      end;

```

```

.TITLE ZRCH85 ZRCH80 RC25 DISK FORMATTER
.IDENT /REV B /

.GLOBL COM.AREA, HEAD.AREA, RECEIVE.RING
.GLOBL SEND.RING, REC.ENVELOPE, SND.ENVELOPE
.GLOBL RET.EN$AD, REC.BUF, SND.BUF, OUT$STD.BUF
.GLOBL NEX.FLAG, MRD.SLOT, NSD.SLOT, RC25.ADDR
.GLOBL RET.STATUS, PID.SAVE, VEC.ADDR
.GLOBL RSVD.STRUCT, ISD.STRUCT, UNIT.NO
.GLOBL NXT.CRN, PFE.STRUCT, EMSG.STRUCT
.GLOBL RC.STRUCTURE, SDUP.STRUCT, SMSCP.STRUCT
.GLOBL PID.FMT, FMT4, AZFMTR, MDSA, DMSA
.GLOBL OVSA

```

000000

```

.SBTTL LOAD.FILE MODULE DECLARATIONS
.PSECT AC$CODE, RO

```

000000 004167 000000G

LOAD.FILE::

000004	016600	000014	JSR	R1, \$SAVE3	:	1634
000010	104434		MOV	14(SP),R0	: FN.EXT,*	1677
000012	016600	000012	TRAP	34		
000016	006300		MOV	12(SP),R0	: BUF.SIZ,*	1678
000020	016602	000016	ASL	R0		
000024	060200		MOV	16(SP),R2	: BUF.LOC,*	
			ADD	R2,R0		

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0151
Page 7
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:{NEALE.AZTEC}ZRCH85.B16;4 (5)

000026	010003			MOV	R0,R3	:	*.EOB	
000030	005001			CLR	R1	:	I	1685
000032	010200		1\$:	MOV	R2,R0	:		1687
000034	104427			TRAP	27	:		
000036	103001			BHIS	2\$:		
000040	010012			MOV	R0,(R2)	:	R0,*	
000042	005201		2\$:	INC	R1	:	I	1685
000044	020127	000377		CMP	R1,#377	:	I,*	
000050	101770			BLOS	1\$:		
000052	016600	000016		MOV	16(SP),R0	:	BUF.LOC,*	1701
000056	104427			TRAP	27	:		
000060	103005			BHIS	4\$:		
000062	010076	000016		MOV	R0,@16(SP)	:	R0,BUF.LOC	
000066	012702	000001		MOV	#1,R2	:	*,NEOF	
000072	000401			BR	5\$:		
000074	005002		4\$:	CLR	R2	:	NEOF	
000076	062766	000002 000016		ADD	#2,16(SP)	:	*,BUF.LOC	1702
000104	032702	000001		BIT	#1,R2	:	*,NEOF	1704
000110	001403			BEQ	6\$:		
000112	026603	000016		CMP	16(SP),R3	:	BUF.LOC,EOB	
000116	001355			BNE	3\$:		
000120	006002		6\$:	ROR	R2	:	NEOF	1710
000122	103005			BCC	7\$:		
000124	012700	005001		MOV	#5001,R0	:		1712
000130	010067	000000G		MOV	R0,RET.STATUS	:		
000134	000207			RTS	PC	:		
000136	104435		7\$:	TRAP	35	:		1714
000140	005067	000000G		CLR	RET.STATUS	:		1718
000144	005000			CLR	R0	:		1671
000146	000207			RTS	PC	:		1634

: Routine Size: 52 words, Routine Base: AC\$CODE + 0000
: Maximum stack depth per invocation: 6 words

: 1720

```

:      1721 global routine GET_NSD =                !Chooses the next send slot
:      1722
:      1723 !**
:      1724 ! Functional Description :
:      1725 !     This routine will determine which send ring descriptor the port/controller
:      1726 !     is polling and returns that dsc slot number to the calling routine.
:      1727 !     This host program will call this routine each time it wishes to deposit
:      1728 !     another command into the command (send) ring.
:      1729 ! Formal Parameters :
:      1730 !     none
:      1731 ! Implicit Inputs :
:      1732 !     NSD_SLOT :      Global storage for the next send descriptor slot.
:      1733 !                   Stores where the host should place this command for
:      1734 !                   processing by the port/controller.
:      1735 ! Implicit Outputs :
:      1736 !     The global storage "Nsd_slot" is updated to the
:      1737 !     present send slot where the port/controller is polling.
:      1738 ! Completion Codes :
:      1739 !     Returns the contents of "Nsd_slot" to the calling routine.
:      1740 ! Side Effects :
:      1741 !     none
:      1742 !--
:      1743
:      1744     begin
:      1745     !
:      1746     ! Increment the next send descriptor_slot by one
:      1747     !
:      1748     NSD_SLOT = .NSD_SLOT + 1;
:      1749     !
:      1750     ! Set the slot pointer back to zero if it wraps around to the top of the ring.
:      1751     !
:      1752     !
:      1753     if .NSD_SLOT gtru SND_ALLOCATE 1 then NSD_SLOT = ZERO;
:      1754     !
:      1755     !
:      1756     ! Return the next send descriptor_slot to the caller
:      1757     !
:      1758     return .NSD_SLOT;
:      1759     end;

```

000000	005267	000000G	.SBTTL	GET.NSD MODULE DECLARATIONS		
			GET.NSD:			
000004	026727	000000G 000003	INC	NSD.SLOT	:	1748
000012	101402		CMP	NSD.SLOT,#3	:	1753
000014	005067	000000G	BLOS	1\$		
000020	016700	000000G	CLR	NSD.SLOT		
000024	000207		1\$:	MOV	NSD.SLOT,R0	1744
				RTS	PC	1721

```

: Routine Size: 11 words,      Routine Base: AC$CODE + 0150
: Maximum stack depth per invocation: 0 words

```

```

:      1760

```

```

:      1761 global routine GET_NRD =                !Chooses the next receive slot
:      1762
:      1763 !..
:      1764 ! Functional Description :
:      1765 !     This routine will determine which receive ring descriptor the port/controller
:      1766 !     is polling and returns that dsc slot number to the calling routine. This
:      1767 !     host program will call this routine each time it wishes to process another
:      1768 !     receive ring descriptor.
:      1769 ! Formal Parameters :
:      1770 !     none
:      1771 ! Implicit Inputs :
:      1772 !     NRD_SLOT :      Global storage for the next receive descriptor slot.
:      1773 !                   Stores where the port should return this commands
:      1774 !                   response indicator.
:      1775 ! Implicit Outputs :
:      1776 !     The global storage "Nrd_slot" is updated to the present receive slot
:      1777 !     where the port/controller is polling.
:      1778 ! Completion Codes :
:      1779 !     Returns the contents of "Nrd_slot" to the calling routine.
:      1780 ! Side Effects :
:      1781 !     none
:      1782 !--
:      1783
:      1784     begin
:      1785     !
:      1786     ! Increment the next receive descriptor_slot by one
:      1787     !
:      1788     NRD_SLOT = .NRD_SLOT + 1;
:      1789     !
:      1790     ! Set the slot pointer back to zero if it wraps around to the top of the ring.
:      1791     !
:      1792     !
:      1793     if .NRD_SLOT gtru REC_ALLOCATE - 1 then NRD_SLOT = ZERO;
:      1794     !
:      1795     !
:      1796     ! Return the next receive descriptor_slot to the caller
:      1797     !
:      1798     return .NRD_SLOT;
:      1799     end;

```

000000	005267	000000G	GET.NRD: :	.SBTTL GET.NRD MODULE DECLARATIONS		
000004	026727	000000G 000003	INC	NRD.SLOT	:	1788
000012	101402		CMP	NRD.SLOT,#3	:	1793
000014	005067	000000G	BLOS	1\$		
000020	016700	000000G	CLR	NRD.SLOT		
000024	000207		1\$: MOV	NRD.SLOT,R0	:	1784
			RTS	PC	:	1761

```

: Routine Size: 11 words, Routine Base: AC$CODE + 0176
: Maximum stack depth per invocation: 0 words

```

```

:      1800

```

```

:      1801 global routine LOAD_OUT$STD_BUF (REF_NUM) =      !Load out$std_buffer with this command
:      1802
:      1803 !**
:      1804 ! Functional Description :
:      1805 !   The outstanding command buffer "out$std_buf" is used by
:      1806 !   this host program to determine if an outstanding command
:      1807 !   issued to the port has been processed yet. This is done
:      1808 !   by examining the receive flag 'Rec_flg' in a buffer slot
:      1809 !   for a '1' which is set by the interrupt service routine
:      1810 !   during response ring interrupts.
:      1811
:      1812 !   This buffer can be looked at as a window between the port
:      1813 !   driver receiving & processing the response envelopes and the
:      1814 !   host class driver issuing commands to the port.
:      1815
:      1816 !   This routine loads into an empty out$std_buf slot the
:      1817 !   following values:
:      1818
:      1819 !   1. This commands reference number.
:      1820 !   2. Clears 'rec_flg' indicating this command is outstanding.
:      1821 !   3. Clears out the second word in slot where the returned
:      1822 !   envelope address will go.
:      1823
:      1824 ! IMPORTANT NOTE:
:      1825 ! -----
:      1826 ! To quarentee a command loaded into the out$std_buffer will
:      1827 ! never be lost (i.e. having this routine return a buffer
:      1828 ! slot not yet received by the interrupt service routine),
:      1829 ! only the cto_wait (controller time out wait) routine is
:      1830 ! permitted to return a out$std_buf slot to the unused pool
:      1831 ! (i.e. by loading a slots first word with $o'100000').
:      1832 ! This routine is therefore guaranteed to return an unused
:      1833 ! out$std_buffer slot when this unique value of $o'100000'
:      1834 ! is found. To further quarentee this, unique command ref
:      1835 ! numbers will never use zero as a reference number.
:      1836
:      1837 ! Formal Parameters :
:      1838 !   REF_NUM      This is the unique reference number of this command set to the port.
:      1839
:      1840 ! Implicit Inputs :
:      1841 !   none
:      1842
:      1843 ! Implicit Outputs :
:      1844 !   none
:      1845
:      1846 ! Completion Codes :
:      1847 !   The out$standing buffer slot index where this command was put
:      1848 !   is routines value and is returned to the caller.
:      1849
:      1850 ! Side Effects :
:      1851 !   none
:      1852
:      1853 ! --
:      1854
:      1855     begin
:      1856
:      1857     !.

```

```

:      1858      ! Search through the out standing command buffer and look for the first
:      1859      ! open slot. Return this first slot index to the caller if one is found.
:      1860      ! If no open slots are found then return an error code.
:      1861      !
:      1862
:      1863      incru i from 0 to REC_ALLOCATE - 1 do          !Find the first open slot
:      1864      !
:      1865      ! An open slot is by definition the value %0'100000' in the slots first word.
:      1866      !
:      1867
:      1868      if .OUT$STD_BUF [.i, CMD_WRD] eqlu %0'100000'  !Is this slot open
:      1869      then
:      1870      begin
:      1871      OUT$STD_BUF [.i, REC_FLG] = FALSE;  !Clear the received flag
:      1872      OUT$STD_BUF [.i, CMD_REF] = .REF_NUM;  !Load this cmd's ref num
:      1873      OUT$STD_BUF [.i, ENV_ADR] = ZERO;  !Clear the previous env adr
:      1874      return .i;  !Return buffer index to caller
:      1875      end;
:      1876
:      1877      !
:      1878      ! The buffer is full if the code reaches here. This should never happen so
:      1879      ! report an error for debug purposes.
:      1880      !
:      1881      return RET_STATUS = OBF_CODE;  !Report an 'out$std buffer full" error
:      1882      end;

```

```

000000 004167 000000G      .SBTTL LOAD.OUT$STD.BUF MODULE DECLARATIONS
                                LOAD.OUT$STD.BUF::
000004 005001                JSR      R1,$SAVE2                ;
000006 010100                CLR      R1                ; I
                                1$: MOV      R1,R0                ; I.*
000010 006300                ASL      R0
000012 006300                ASL      R0
000014 012702 000000G      MOV      #OUT$STD.BUF,R2
000020 060002                ADD      R0,R2
000022 021227 100000        CMP      (R2),#-100000
000026 001010                BNE      2$
000030 042712 100000        BIC      #100000,(R2)                ;
000034 116612 000010        MOV      10(SP),(R2)                ; REF.NUM.*
000040 005060 000002G      CLR      OUT$STD.BUF+2(R0)        ;
000044 010100                MOV      R1,R0                ; I.*
000046 000207                RTS      PC                ; I.*
                                2$: INC      R1                ; I
000050 005201                INC      R1                ; I.*
000052 020127 000003        CMP      R1,#3                ; I.*
000056 101753                BLOS    1$
000060 012700 002001        MOV      #2001,R0                ;
000064 010067 000000G      MOV      R0,RET.STATUS            ;
000070 000207                RTS      PC                ;

```

```

; Routine Size: 29 words,      Routine Base: AC$CODE + 0224
; Maximum stack depth per invocation: 4 words

```

; 1883

```

:      1884 global routine GET_CMD$REF =                !Gets next unique cmd ref number
:      1885
:      1886 !..
:      1887 ! Functional Description :
:      1888 !   A 32 bit unique non-zero number used to identify host commands.
:      1889 !   Class drivers should supply a unique reference number in each
:      1890 !   command that the send to a DUP server. A class driver may supply
:      1891 !   a zero reference number if it does not need to associate a command
:      1892 !   with its end message.
:      1893
:      1894 !   Command reference numbers must be unique across all commands that
:      1895 !   are outstanding on the same connection i.e., they must be unique
:      1896 !   across all outstanding commands issued by a single class driver
:      1897 !   (Host) to a single DUP server. The class driver may re-use a
:      1898 !   commands reference number when the command is no longer
:      1899 !   outstanding -- i.e., after receiving the commands end message or
:      1900 !   after re-synchronizing with the DUP server. Command reference
:      1901 !   numbers need not be unique for commands issued by different class
:      1902 !   drivers --- i.e. commands issued by different host or commands for
:      1903 !   diffeent DUP servers from the same host. Therefore controllers
:      1904 !   must internally use the combination of a command reference number
:      1905 !   and the connection on which the command was received as the unique
:      1906 !   identifier of an outstanding command.
:      1907
:      1908 !   This routine will generate a unique command reference number and
:      1909 !   will search the out$standing command buffer to see if already used.
:      1910 !   The first unused unique command reference found will be returned
:      1911 !   to the calling routine.
:      1912
:      1913 ! Formal Parameters :
:      1914 !   none
:      1915
:      1916 ! Implicit Inputs :
:      1917 !   NXT_CRN          This global location stores the next unique cmd
:      1918 !                   reference number to be used.
:      1919
:      1920 ! Implicit Outputs :
:      1921 !   NXT_CRN          This global location is loaded with the next
:      1922 !                   unique command reference number.
:      1923
:      1924 ! Completion Codes :
:      1925 !   The contents of .NXT_CRN is returned to the calling routine.
:      1926
:      1927 ! Side Effects :
:      1928 !   none
:      1929 !--
:      1930
:      1931     begin
:      1932
:      1933     local
:      1934         DONE;
:      1935
:      1936     !
:      1937     ! Increment the global unique command reference number before anything is done.
:      1938     !
:      1939     NXT_CRN = .NXT_CRN + 1;
:      1940

```

```

: 1941      !.
: 1942      ! Repeat generating and searching the outstanding command buffer until a
: 1943      ! unique command reference number is found.
: 1944      !-
: 1945
: 1946      do
: 1947      begin
: 1948      BREAK;                                !Flag any operator control C's
: 1949      !
: 1950      ! Wrap this next command reference number around
: 1951      ! back to one if it is greater than 255 (decimal).
: 1952      !
: 1953
: 1954      if .NXT_CRN gtr 255 then NXT_CRN = 1;
: 1955
: 1956      DONE = TRUE;                          !Clear the all done indicator flag
: 1957
: 1958      !.
: 1959      ! Now search the outstd_buffer for this command
: 1960      ! reference number. If not there then done stays
: 1961      ! true and the loop will end else increment to the
: 1962      ! next unique command ref number and make done false
: 1963      ! to continue the loop.
: 1964      !-
: 1965
: 1966      incu i from 0 to REC_ALLOCATE - 1 do  !Search buffer for this cmd ref num
: 1967
: 1968      if .OUT$STD_BUF [.i, CMD_REF] eqv .NXT_CRN !Does it already exist
: 1969      then
: 1970      begin                                  !It already exists
: 1971      NXT_CRN = .NXT_CRN + 1;                !Try the next sequential cmd ref num
: 1972      DONE = FALSE;                          !Make code loop again
: 1973      exitloop;                              !Exit this incr loop
: 1974      end;
: 1975
: 1976      end
: 1977      until .DONE;                          !Repeat loop until done
: 1978
: 1979      !
: 1980      ! Return the unique command reference number to the caller.
: 1981      !
: 1982      return .NXT_CRN;
: 1983      end;

```

000000	010146		.SBTTL	GET.CMD\$REF	MODULE DECLARATIONS		
			GET.CMD\$REF::				
000002	105267	000000G		MOV	R1, (SP)		1884
000006	104422		1\$:	INCB	NXT_CRN		1939
000010	012701	000001		TRAP	22		1947
000014	005000			MOV	01,R1		1956
000016	126067	000000G 000000G	2\$:	CLR	R0		1966
000024	001004			CMPB	OUT\$STD.BUF(R0),NXT_CRN		1968
000026	105267	000000G		BNE	3\$		
000032	005001			INCB	NXT_CRN		1971
000034	000405			CLR	R1		1972
				BR	4\$		1970

ZRCHB5
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
MODULE DECLARATIONS

5 Apr 1984 13:46:24
11 Jan 1984 13:23:12

SEQ 0158
Page 14
VAX-11 B1199-16 V3 555
SPIDER#USERS:(NEALE.AZTEC)ZRCHB5.B16;4 (9)

000036	062700	000004	3:	ADD	#4,RO	:	*,I	1966
000042	020027	000014		CMP	RO,#14	:	I,*	
000046	101763			BLOS	2:			
000050	006001		4:	ROR	R1	:	DONE	1977
000052	103355			BCC	1:			
000054	005000			CLR	RO	:		1931
000056	156700	000000G		BISB	NXT.CRN,RO			
000062	012601			MOV	(SP),R1	:		1884
000064	000207			RTS	PC			

: Routine Size: 27 words, Routine Base: AC#CODE + 0316
: Maximum stack depth per invocation: 3 words

: 1984


```

:      1985 global routine DECODE : novalue *           !Decodes failing SA reg data
:      1986
:      1987 :..
:      1988 : Functional Description :
:      1989 : Due to the implimentation of the DUP and UQ Port protocol there
:      1990 : are two levels at which an issued command to a port/controller
:      1991 : can fail and they are:
:      1992 :
:      1993 :     1. The issued command can time out.
:      1994 :
:      1995 :     2. An error can be posted in SA register bit 15 by the port to
:      1996 : report an error.
:      1997 :
:      1998 :     3. The issued command to the port/controller can be executed
:      1999 : correctly without any errors but the response packet status
:      2000 : field could have an error or status other than success posted.
:      2001 :
:      2002 : These errors or status's returned are all returned to the host
:      2003 : routine which queued the DUP command via the global storage
:      2004 : "RET_STATUS". The host to port/controller communications
:      2005 : connection having the highest priority (ie. if the SA Reg error
:      2006 : bit is set the DUP interrupt routine returns a PFE_CODE and this
:      2007 : code is passed up to the calling host routine with out being
:      2008 : intercepted by any routine on the way up).
:      2009 :
:      2010 : This routine will then be called when the return from a queued
:      2011 : command comes back with an error code or non successfull status
:      2012 : code. This is by definition when bit 0 in the returned status
:      2013 : is equal to 1.
:      2014 :
:      2015 : An appropriate recovery action will be done for each individual
:      2016 : error.
:      2017 :
:      2018 : Formal Parameters :
:      2019 :     none
:      2020 :
:      2021 : Implicit Inputs :
:      2022 :     RET_STATUS:   Stored in this global storage is the returned error
:      2023 :                   code or non-successful status code from a queued
:      2024 :                   command.
:      2025 :
:      2026 : Implicit Outputs :
:      2027 :     none
:      2028 :
:      2029 : Completion Codes :
:      2030 :     none
:      2031 :
:      2032 : Side Effects :
:      2033 :     All formatter errors are fatal, therefor after execution
:      2034 :     of this routine the RC25 controller is initialized
:      2035 :     aborting any DM code running in the controller.
:      2036 :
:      2037 :
:      2038 : begin
:      2039 :
:      2040 : local
:      2041 :     SA VALUE;           !Save SA register fatal error code

```

```

: 2042
: 2043
: 2044 : Use the contents of "RET_STATUS" to select what type error or non successful
: 2045 : status code is to be processed.
: 2046
: 2047
: 2048 selectoneu .RET_STATUS of
: 2049 set
: 2050
: 2051 : "Communication area initialize" error code
: 2052
: 2053 : This error code indicates that the port did
: 2054 : not init the com area in the host memory after
: 2055 : step 2 of the initialization sequence.
: 2056
: 2057
: 2058 [CIE_CODE] : !Code equals %o'01'
: 2059 begin
: 2060 PRINTB (.EMSG_STRUCT [MSG4]);
: 2061 end;
: 2062
: 2063 : "Port/Controller time out" error code
: 2064
: 2065 : Port/Controller timed out after the specified time out interval.
: 2066
: 2067
: 2068 [CTO_CODE] : !Code equals %o'11'
: 2069 begin
: 2070 PRINTB (.EMSG_STRUCT [MSG21]);
: 2071 end;
: 2072
: 2073 : "Port fatal error" code
: 2074
: 2075 : The error bit in the SA Register was set when examined in the DUP$I SERVICE
: 2076 : routine. This error indicates a Port fatal error code.
: 2077
: 2078
: 2079 [PFE_CODE] : !Code equals %o'21'
: 2080 begin
: 2081 SA_VALUE = .RC25_ADDR [RCSA, ERR CODE]; !Get error code from SA register
: 2082
: 2083 : Is the error code within the RC25 error code range, or is it
: 2084 : an generic, all controllers, error code?
: 2085
: 2086
: 2087 if .SA_VALUE gequ 200
: 2088 then
: 2089 begin
: 2090 PRINTB (.RC STRUCTURE [.SA VALUE - 200]); !RC25 error code range
: 2091 end
: 2092 else
: 2093 begin
: 2094 PRINTB (.PFE STRUCT [.SA VALUE]); !Generic error code
: 2095 end;
: 2096
: 2097 end;
: 2098

```

```

:      2099      : "Return status error" code
:      2100      :
:      2101      : This indicates that a non successful return status code was returned from an issued command.
:      2102      :
:      2103      :
:      2104      : [RSE_CODE] :                               !Code equals #0'31'
:      2105      :     begin
:      2106      :     PRINTB (.EMSG_STRUCT [MSG0]);
:      2107      :     :
:      2108      :     : Look at the UQPORT connection ID field to determine the type of response
:      2109      :     :
:      2110      :
:      2111      :     if .RET_EN$AD [CONN_ID] eq lu DUP
:      2112      :     then
:      2113      :     begin
:      2114      :     PRINTB (.SDUP_STRUCT [.RET_EN$AD [STATUS]]);
:      2115      :     end
:      2116      :     else
:      2117      :     begin
:      2118      :     PRINTB (.SMSCP_STRUCT [.RET_EN$AD [STA CODE]]);
:      2119      :     end;
:      2120      :
:      2121      :     end;
:      2122      :
:      2123      : "Port Portocol violation" error code
:      2124      :
:      2125      : A protocol violation error was detected during host processing of an issued command.
:      2126      :
:      2127      :
:      2128      : [PVE_CODE] :                               !Code equals #0 41'
:      2129      :     begin
:      2130      :     PRINTB (.EMSG_STRUCT [MSG1]);
:      2131      :     end;
:      2132      :
:      2133      : "Remote program died" error code
:      2134      :
:      2135      : This indicates that the remote program running
:      2136      : in the DM machine did not responded within the
:      2137      : designated time out interval and that the progress
:      2138      : indicator was not increase after subsiquent time out
:      2139      : delays. It is assumed that the remote program is dead
:      2140      : and is treated as a fatal error.
:      2141      :
:      2142      :
:      2143      : [RPD_CODE] :                               !Code equals #0'51
:      2144      :     begin
:      2145      :     PRINTB (.EMSG_STRUCT [MSG2]);
:      2146      :     end;
:      2147      :
:      2148      : "Port to host synchronious error" code
:      2149      :
:      2150      :
:      2151      : [PSE_CODE] :                               !Code equals #0 61
:      2152      :     begin
:      2153      :     PRINTB (.EMSG_STRUCT [MSG5]);
:      2154      :     end;
:      2155      :

```

ZRCMB5
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
MODULE DECLARATIONS5 Apr-1984 13:46:24
11 Jan-1984 13:23:12VAX-11 Bliss-16 V3 555
SPIDER\USERS:[NEALE.AZTEC]ZRCMB5.B16;4 (10)
SEO 0162
Page 18

```

:      2156      : 'Message length error' code
:      2157      :
:      2158      :
:      2159      : [MLE_CODE] :           !Code equals %o'71'
:      2160      :     begin
:      2161      :     PRINTB (.EMSG_STRUCT [MSG6]);
:      2162      :     end;
:      2163      :
:      2164      : "Unknown end code' error code
:      2165      :
:      2166      :
:      2167      : [UEC_CODE] :           !Code equals %o'101'
:      2168      :     begin
:      2169      :     PRINTB (.EMSG_STRUCT [MSG7]);
:      2170      :     end;
:      2171      :
:      2172      : "Adaptor purge request' error code
:      2173      :
:      2174      :
:      2175      : [APR_CODE] :           !Code equals %o 201
:      2176      :     begin
:      2177      :     PRINTB (.EMSG_STRUCT [MSG8]);
:      2178      :     end;
:      2179      :
:      2180      : "Unknown interrupt" error code
:      2181      :
:      2182      :
:      2183      : [UIN_CODE] :           !Code equals %o 301'
:      2184      :     begin
:      2185      :     PRINTB (.EMSG_STRUCT [MSG9]);
:      2186      :     end;
:      2187      :
:      2188      : "ATTENTION MSG ENDCODE" error code
:      2189      :
:      2190      :
:      2191      : [ATN_CODE] :           !Code equals %o 401
:      2192      :     begin
:      2193      :     PRINTB (.EMSG_STRUCT [MSG12]);
:      2194      :     end;
:      2195      :
:      2196      : "COMMAND MSG ENDCODE' error code
:      2197      :
:      2198      :
:      2199      : [CMD_CODE] :           !Code equals %o 501'
:      2200      :     begin
:      2201      :     PRINTB (.EMSG_STRUCT [MSG13]);
:      2202      :     end;
:      2203      :
:      2204      : "SERIOUS EXCEPTION' error code
:      2205      :
:      2206      :
:      2207      : [SEX_CODE] :           !Code equals %o 601'
:      2208      :     begin
:      2209      :     PRINTB (.EMSG_STRUCT [MSG14]);
:      2210      :     end;
:      2211      :
:      2212      : INVALID COMMAND' error code

```

```

:      2213      :
:      2214      :
:      2215      [IVC_CODE] :          !Code equals %o'701'
:      2216          begin
:      2217          PRINTB (.EMSG_STRUCT [MSG15]);
:      2218          end;
:      2219      :
:      2220      : "UNKNOWN MESSAGE TYPE" error code
:      2221      :
:      2222      :
:      2223      [UMT_CODE] :          !Code equals %o'1001'
:      2224          begin
:      2225          PRINTB (.EMSG_STRUCT [MSG16]);
:      2226          end;
:      2227      :
:      2228      : "UNKNOWN MESSAGE NUMBER" error code
:      2229      :
:      2230      :
:      2231      [UMN_CODE] :          !Code equals %o'4001'
:      2232          begin
:      2233          PRINTB (.EMSG_STRUCT [MSG19]);
:      2234          end;
:      2235      :
:      2236      : Out standing buffer slots are all filled up
:      2237      :
:      2238      :
:      2239      [OBF_CODE] :          !Code equals %o'2001'
:      2240          begin
:      2241          PRINTB (.EMSG_STRUCT [MSG17]);
:      2242          end;
:      2243      :
:      2244      : Out standing command buffer out of sync error
:      2245      :
:      2246      :
:      2247      [OSE_CODE] :          !Code equals %o'3001'
:      2248          begin
:      2249          PRINTB (.EMSG_STRUCT [MSG18]);
:      2250          end;
:      2251      :
:      2252      : File read error from local load media
:      2253      :
:      2254      :
:      2255      [FRE_CODE] :          !Code equals %o'5001'
:      2256          begin
:      2257          PRINTB (.EMSG_STRUCT [MSG20]);
:      2258          end;
:      2259      :
:      2260      : Illegal FCT file length
:      2261      :
:      2262      :
:      2263      [ILL_FCT] :          !Code equals %o'6001'
:      2264          begin
:      2265          PRINTB (.EMSG_STRUCT [MSG22]);
:      2266          end;
:      2267      :
:      2268      : This is here to trap any unknow return status codes sent to this routine.
:      2269      :

```

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11 Jan-1984 13:23:12

VAX-11 Bliss 16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (10)

SEQ 0164
Page 20

```

:      2270
:      2271      [otherwise] :      !Code equals non of the above
:      2272      begin
:      2273      PRINTB (.EMSG STRUCT [MSG3]);
:      2274      end;
:      2275      tes;
:      2276
:      2277      !
:      2278      ! All errors are fatal so init the RC25 and jump to the clean up
:      2279      ! code section to abort this units format.
:      2280      !
:      2281      RC25_ADDR [RCIP, RC_ALL] = ONES;      !Init the controller
:      2282      WRT_RC25 (RCIP, ONES);      !Init the controller
:      2283      DOCLN;      !Jump to the clean-up code section
:      2284      end;
    
```

```

000000 004167 000000G      .SBTTL  DECODE MODULE DECLARATIONS
000004 024646      DECODE::JSR  R1,$SAVE2      ;      1985
000006 016701 000000G      CMP    -(SP),-(SP)      ;
000012 020127 000001      MOV    RET.STATUS,R1      ;      2048
000016 001007      CMP    R1,#1      ;
000020 016746 000010G      BNE   1$      ;
000024 012746 000001      MOV    EMSG.STRUCT+10,-(SP)      ;      2060
000030 010600      MOV    #1,-(SP)      ;
000032 104414      MOV    SP,R0      ; SP,*
000034 000567      TRAP  14      ;
000036 020127 000011      BR    11$      ;      2048
000042 001007      1$:  CMP    R1,#11      ;
000044 016746 000052G      BNE   2$      ;
000050 012746 000001      MOV    EMSG.STRUCT+52,-(SP)      ;      2070
000054 010600      MOV    #1,-(SP)      ;
000056 104414      MOV    SP,R0      ; SP,*
000060 000567      TRAP  14      ;
000062 020127 000021      BR    13$      ;      2048
000066 001034      2$:  CMP    R1,#21      ;
000070 016700 000000G      BNE   4$      ;
000074 016066 000002 000002      MOV    RC25.ADDR,R0      ;      2081
000102 016600 000002      MOV    2(R0),2(SP)      ; *,RC$S.REG
000106 042700 174000      MOV    2(SP),R0      ; RC$S.REG,SA.VALUE
000112 020027 000310      BIC   #174000,R0      ; *,SA.VALUE
000116 103410      CMP    R0,#310      ; SA.VALUE,*      2087
000120 006300      BLO   3$      ;
000122 016046 177160G      ASL   R0      ;      2090
000126 012746 000001      MOV    RC.STRUCTURE-620(R0),(SP)      ;
000132 010600      MOV    #1,(SP)      ;
000134 104414      MOV    SP,R0      ; SP,*
000136 000576      TRAP  14      ;
000140 006300      BR    17$      ;      2087
000142 016046 000000G      3$:  ASL   R0      ;      2094
000146 012746 000001      MOV    PFE.STRUCT(R0),(SP)      ;
000152 010600      MOV    #1,-(SP)      ;
000154 104414      MOV    SP,R0      ; SP,*
000156 000566      TRAP  14      ;
000160 020127 000031      BR    17$      ;      2048
000164 001044      4$:  CMP    R1,#31      ;
      BNE  7$
    
```

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0165
Page 21
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (10)

000166	016746	000000G		MOV	EMSG.STRUCT,-(SP)	:		2106
000172	012746	000001		MOV	#1,-(SP)	:		
000176	010600			MOV	SP,R0	:	SP,*	
000200	104414			TRAP	14	:		
000202	016702	000000G		MOV	RET.EN\$AD,R2	:		2114
000206	010200			MOV	R2,R0	:	RET.EN\$AD,*	2111
000210	126027	000003	000002	CMPB	3(R0),#2	:		
000216	001012			BNE	5\$:		
000220	016200	000016		MOV	16(R2),R0	:		2114
000224	006300			ASL	R0	:		
000226	016016	000000G		MOV	SDUP.STRUCT(R0),(SP)	:		
000232	012746	000001		MOV	#1,-(SP)	:		
000236	010600			MOV	SP,R0	:	SP,*	
000240	104414			TRAP	14	:		
000242	000413			BR	6\$:		2111
000244	116202	000016	5\$:	MOVB	16(R2),R2	:		2118
000250	042702	177740		BIC	#177740,R2	:		
000254	006302			ASL	R2	:		
000256	016216	000000G		MOV	SMSCP.STRUCT(R2),(SP)	:		
000262	012746	000001		MOV	#1,-(SP)	:		
000266	010600			MOV	SP,R0	:	SP,*	
000270	104414			TRAP	14	:		
000272	005726		6\$:	TST	(SP),*	:		2105
000274	000567			BR	22\$:		2048
000276	020127	000041	7\$:	CMP	R1,#41	:		
000302	001007			BNE	8\$:		
000304	016746	000002G		MOV	EMSG.STRUCT+2,-(SP)	:		2130
000310	012746	000001		MOV	#1,-(SP)	:		
000314	010600			MOV	SP,R0	:	SP,*	
000316	104414			TRAP	14	:		
000320	000567			BR	24\$:		2048
000322	020127	000051	8\$:	CMP	R1,#51	:		
000326	001007			BNE	9\$:		
000330	016746	000004G		MOV	EMSG.STRUCT+4,-(SP)	:		2145
000334	012746	000001		MOV	#1,-(SP)	:		
000340	010600			MOV	SP,R0	:	SP,*	
000342	104414			TRAP	14	:		
000344	000567			BR	26\$:		2048
000346	020127	000061	9\$:	CMP	R1,#61	:		
000352	001007			BNE	10\$:		
000354	016746	000012G		MOV	EMSG.STRUCT+12,(SP)	:		2153
000360	012746	000001		MOV	#1,(SP)	:		
000364	010600			MOV	SP,R0	:	SP,*	
000366	104414			TRAP	14	:		
000370	000567			BR	28\$:		2048
000372	020127	000071	10\$:	CMP	R1,#71	:		
000376	001007			BNE	12\$:		
000400	016746	000014G		MOV	EMSG.STRUCT+14,(SP)	:		2161
000404	012746	000001		MOV	#1,-(SP)	:		
000410	010600			MOV	SP,R0	:	SP,*	
000412	104414			TRAP	14	:		
000414	000567		11\$:	BR	30\$:		2048
000416	020127	000101	12\$:	CMP	R1,#101	:		
000422	001007			BNE	14\$:		
000424	016746	000016G		MOV	EMSG.STRUCT+16,(SP)	:		2169
000430	012746	000001		MOV	#1,(SP)	:		
000434	010600			MOV	SP,R0	:	SP,*	

ZRCMB5
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
MODULE DECLARATIONS5-Apr-1984 13:46:24
11-Jan-1984 13:23:12VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCMB5.B16;4 (10)
SEQ 0166
Page 22

000436	104414		TRAP	14		
000440	000576	13\$:	BR	33\$:	2048
000442	020127	000201	14\$:	CMP	R1,#201	
000446	001007		BNE	15\$		
000450	016746	000020G	MOV	MSG.STRUCT+20, -(SP)	:	2177
000454	012746	000001	MOV	#1, -(SP)		
000460	010600		MOV	SP,R0	: SP,▲	
000462	104414		TRAP	14		
000464	000564		BR	33\$:	2048
000466	020127	000301	15\$:	CMP	R1,#301	
000472	001007		BNE	16\$		
000474	016746	000022G	MOV	MSG.STRUCT+22, -(SP)	:	2185
000500	012746	000001	MOV	#1, -(SP)		
000504	010600		MOV	SP,R0	: SP,▲	
000506	104414		TRAP	14		
000510	000552		BR	33\$:	2048
000512	020127	000401	16\$:	CMP	R1,#401	
000516	001007		BNE	18\$		
000520	016746	000030G	MOV	MSG.STRUCT+30, -(SP)	:	2193
000524	012746	000001	MOV	#1, -(SP)		
000530	010600		MOV	SP,R0	: SP,▲	
000532	104414		TRAP	14		
000534	000540		BR	33\$:	2048
000536	020127	000501	17\$:	CMP	R1,#501	
000542	001007		BNE	19\$		
000544	016746	000032G	MOV	MSG.STRUCT+32, -(SP)	:	2201
000550	012746	000001	MOV	#1, -(SP)		
000554	010600		MOV	SP,R0	: SP,▲	
000556	104414		TRAP	14		
000560	000526		BR	33\$:	2048
000562	020127	000601	19\$:	CMP	R1,#601	
000566	001007		BNE	20\$		
000570	016746	000034G	MOV	MSG.STRUCT+34, -(SP)	:	2209
000574	012746	000001	MOV	#1, -(SP)		
000600	010600		MOV	SP,R0	: SP,▲	
000602	104414		TRAP	14		
000604	000514		BR	33\$:	2048
000606	020127	000701	20\$:	CMP	R1,#701	
000612	001007		BNE	21\$		
000614	016746	000036G	MOV	MSG.STRUCT+36, -(SP)	:	2217
000620	012746	000001	MOV	#1, -(SP)		
000624	010600		MOV	SP,R0	: SP,▲	
000626	104414		TRAP	14		
000630	000502		BR	33\$:	2048
000632	020127	001001	21\$:	CMP	R1,#1001	
000636	001007		BNE	23\$		
000640	016746	000040G	MOV	MSG.STRUCT+40, -(SP)	:	2225
000644	012746	000001	MOV	#1, (SP)		
000650	010600		MOV	SP,R0	: SP,▲	
000652	104414		TRAP	14		
000654	000470		BR	33\$:	2048
000656	020127	004001	22\$:	CMP	R1,#4001	
000662	001007		BNE	25\$		
000664	016746	000046G	MOV	MSG.STRUCT+46, -(SP)	:	2233
000670	012746	000001	MOV	#1, -(SP)		
000674	010600		MOV	SP,R0	: SP,▲	
000676	104414		TRAP	14		

000700	000456		24\$:	BR	33\$:		2048
000702	020127	002001	25\$:	CMP	R1,#2001	:		
000706	001007			BNE	27\$:		
000710	016746	000042G		MOV	EMSG.STRUCT+42,-(SP)	:		2241
000714	012746	000001		MOV	#1,-(SP)	:		
000720	010600			MOV	SP,R0	:	SP,*	
000722	104414			TRAP	14	:		
000724	000444		26\$:	BR	33\$:		2048
000726	020127	003001	27\$:	CMP	R1,#3001	:		
000732	001007			BNE	29\$:		
000734	016746	000044G		MOV	EMSG.STRUCT+44,-(SP)	:		2249
000740	012746	000001		MOV	#1,-(SP)	:		
000744	010600			MOV	SP,R0	:	SP,*	
000746	104414			TRAP	14	:		
000750	000432		28\$:	BR	33\$:		2048
000752	020127	005001	29\$:	CMP	R1,#5001	:		
000756	001007			BNE	31\$:		
000760	016746	000050G		MOV	EMSG.STRUCT+50,-(SP)	:		2257
000764	012746	000001		MOV	#1,-(SP)	:		
000770	010600			MOV	SP,R0	:	SP,*	
000772	104414			TRAP	14	:		
000774	000420		30\$:	BR	33\$:		2048
000776	020127	006001	31\$:	CMP	R1,#6001	:		
001002	001007			BNE	32\$:		
001004	016746	000054G		MOV	EMSG.STRUCT+54,-(SP)	:		2265
001010	012746	000001		MOV	#1,-(SP)	:		
001014	010600			MOV	SP,R0	:	SP,*	
001016	104414			TRAP	14	:		
001020	000406			BR	33\$:		2048
001022	016746	000006G	32\$:	MOV	EMSG.STRUCT+6,-(SP)	:		2273
001026	012746	000001		MOV	#1,-(SP)	:		
001032	010600			MOV	SP,R0	:	SP,*	
001034	104414			TRAP	14	:		
001036	012766	177777 000004	33\$:	MOV	#-1,4(SP)	:	*.RC\$S.REG	2281
001044	012700	177777		MOV	#-1,R0	:	*.RC\$M.REG	2282
001050	010077	000000G		MOV	R0,RC25.ADDR	:	RC\$M.REG,*	
001054	104444			TRAP	44	:		
001056	062706	000010		ADD	#10,SP	:		1985
001062	000207			RTS	PC	:		

: Routine Size: 282 words, Routine Base: AC\$CODE + 0404
 : Maximum stack depth per invocation: 10 words

: 2285

```

: 2286 global routine DUP$I_SERVICE : INT_LNK$TYP novalue = !Signals receive queue entry
: 2287
: 2288 !**
: 2289 ! Functional Description :
: 2290 ! The transmission of a message will result in a host interrupt if and
: 2291 ! only if interrupts were armed suitably during initialization and one
: 2292 ! of the following conditions has been met:
: 2293
: 2294 ! 1. The message was a command with F=1 and the port's fetching it
: 2295 ! caused the command ring to transition from full to not-full.
: 2296 ! (This interrupt means that the host may place another command
: 2297 ! in the ring.)
: 2298
: 2299 ! 2. The message was a response with F=1 and the port's depositing
: 2300 ! it caused the response ring to transition from empty to not
: 2301 ! empty. (This interrupt means that there is a response for
: 2302 ! the host to process.)
: 2303
: 2304 ! 3. The port is interfaced to the host via a bus abapter and a
: 2305 ! command requires the port/controller to re-access a given
: 2306 ! location during data transfer. (This interrupt means that
: 2307 ! the port/controller is requesting the host to purge the
: 2308 ! indicated chanel of the bus adapter.)
: 2309
: 2310 ! This interrupt service routine is entered when any of the above
: 2311 ! conditions occure. When entered it will be determined what type
: 2312 ! interrupt was executed and take the necessary action.
: 2313
: 2314 ! Formal Parameters :
: 2315 ! none
: 2316
: 2317 ! Implicit Inputs :
: 2318 ! Nrd_slot: A global flag which points to the next receive descriptor
: 2319 ! slot where the port/controller should be polling on and
: 2320 ! where to expect the first response packet to process.
: 2321
: 2322 ! Implicit Outputs :
: 2323 ! Ret_status: This global flag is the mechinism by which these DUP
: 2324 ! and UQ Port protocol routines pass status code back to
: 2325 ! to the host routine's requesting communications over
: 2326 ! the established connections. The status returned is
: 2327 ! decoded by the caller to determine if an error or bad
: 2328 ! response packet status was discovered.
: 2329
: 2330 ! Out$std_buf: This buffer is used to save all commands issued to the
: 2331 ! port and are considered outstanding when in this buffer.
: 2332 ! This interrupt service routine will indicate this command
: 2333 ! is nolonger outstanding by setting the rec_flg in the
: 2334 ! slot matching this response envelope command ref number.
: 2335
: 2336 ! Completion Codes :
: 2337 ! none
: 2338
: 2339 ! Side Effects :
: 2340 ! none
: 2341 ! --
: 2342

```

```

:      2343      begin
:      2344
:      2345      local
:      2346          TEMP,                !Holds nrd_slot + 1 reference
:      2347          FOUND_CMD,          !Found command flag
:      2348          REF_NUM;            !Stores response packets cmd ref number
:      2349
:      2350      !
:      2351      ! Before this interrupt service routine does anything
:      2352      ! look at the SA register for any port fatal errors
:      2353      ! posted. If there are errors posted then report the
:      2354      ! error and kick the bucket.
:      2355      !
:      2356
:      2357      if .RC25_ADDR [RCSA, ERR_BIT]          !Are there any errors posted
:      2358      then
:      2359          begin
:      2360          RET_STATUS = PFE_CODE;                !Indicate the error type
:      2361          DECODE ();                          !Decode and print the error
:      2362          return;                              !Just for show. Decode will kill 't
:      2363          end;
:      2364
:      2365      !
:      2366      ! See what kind of interrupt got us here.
:      2367
:      2368      ! We could have a:
:      2369
:      2370      ! 1. Response ring transition interrupt.
:      2371      ! 2. Send ring transition interrupt.
:      2372      ! 3. A adaptor purge request interrupt (which is
:      2373      ! illegal running under the PDP-11 Diagnostic
:      2374      ! supervisor and is flagged as a fatal controller
:      2375      ! error.)
:      2376      ! 4. Or an unknown interrupt not known by this program
:      2377      ! which also results in a fatal controller error.
:      2378      !
:      2379      !
:      2380
:      2381      !
:      2382
:      2383      ! Check to see if we get here because of a response ring
:      2384      ! transition interrupt. This is more likely to be the
:      2385      ! most frequent interrupt so check it first.
:      2386      !
:      2387
:      2388      if .HEAD_AREA [RSP_INT]
:      2389      then
:      2390          begin
:      2391          GET_NRD ();                            !Get the resp slot location to process
:      2392          !
:      2393          ! Check the host protocol for being out of sequence
:      2394          ! with the controller by making sure that this slot
:      2395          ! is owned by the host (meaning that there is a resp
:      2396          ! envelope at this ring slot to process.
:      2397          !
:      2398
:      2399      if .RECEIVE_RING [.NRD SLOT, OWN_BIT] nequ HOST_OWNED    !Is this owned by host

```

```

:      2400      then
:      2401      begin
:      2402      RET_STATUS = PSE CODE;
:      2403      DECODE ();
:      2404      return;
:      2405      end;
:
:      2406
:      2407
:      2408      ! Per DUP protocol once interrupted due to a response ring
:      2409      ! interrupt, the host code should process all response packets
:      2410      ! found in the response ring. This while loop will continue
:      2411      ! to process the response packets in the response ring until
:      2412      ! none remain.
:      2413
:      2414
:      2415      while TRUE do
:      2416      begin
:      2417      BREAK;
:      2418
:      2419      ! Load the Reference structure "Ret_en$ad" with the address
:      2420      ! of this response envelope to process (The minus $o'4' is
:      2421      ! done to address the first word in the envelope packet
:      2422      ! and is equal to location "text-4").
:      2423
:      2424      RET_EN$AD = (.RECEIVE_RING [.NRD_SLOT, LO EN$AD]) - $o'4';
:      2425
:      2426
:      2427      ! Test the end packet for its possible three end types.
:      2428
:      2429      ! End message opcodes (also called endcodes) are formed by adding the end
:      2430      ! message flag to the command opcode. For example, a READ commands end
:      2431      ! message contains the value OP.RED + OP.END in its opcode field. The Invalid
:      2432      ! command end message contains just the end message flag (i.e., OP.END) in
:      2433      ! its opcode field. The serious exception opcode shown above (i.e. OP.SEX +
:      2434      ! OP.END) in its opcode field.
:      2435
:      2436      ! Commands opcode bits 6 and 7 indicate the type of message (command, end or
:      2437      ! attention message. Command opcodes bits 5 through 5 indicate the command
:      2438      ! category (immediate, sequential or no sequential) and whether or not the
:      2439      ! command includes a buffer descriptor.
:      2440
:      2441      ! See MSCP document appendix "A 1 NOTE:" for more information on this topic.
:      2442
:      2443
:      2444      selectoneu .RET EN$AD [TYP$MSG] of !Select the endpacket size
:      2445      set
:      2446
:      2447      [END$MSG] :
:      2448      begin
:      2449
:      2450
:      2451      ! Select off of the endcode to make sure the communications
:      2452      ! mechanism transferred the correct number of byte for this
:      2453      ! end packet. If this number of bytes transferred is not
:      2454      ! correct for the commands end packet then load the error
:      2455      ! code into return status, call decode to report the
:      2456      ! error and kick the bucket and die. This endcode is formed

```

```

:      2457      ! by adding the end message flag to '200' to the commands
:      2458      ! opcode.
:      2459      !
:      2460
:      2461      select neu .RET EN$AD [ENDCODE] of
:      2462      set
:      2463      !
:      2464      ! "RECEIVE DATA" command end packet
:      2465      !
:      2466
:      2467      [EOP RED] :
:      2468      begin
:      2469
:      2470      if .RET_EN$AD [MSG_LENGTH] nequ ESZ_RED      !Is the byte count correct
:      2471      then
:      2472      begin
:      2473      RET_STATUS = MLE_CODE; !Return a "message length error code"
:      2474      DECODE ();      !Report the error and kick the bucket
:      2475      return;      !Just for show. Decode kills it
:      2476      end;
:      2477
:      2478      end;
:      2479      !
:      2480      ! "SEND DATA" command end packet
:      2481      !
:      2482
:      2483      [EOP SED] :
:      2484      begin
:      2485
:      2486      if .RET_EN$AD [MSG_LENGTH] nequ ESZ_SED      !Is the byte count correct
:      2487      then
:      2488      begin
:      2489      RET_STATUS = MLE_CODE; !Return a "message length error code"
:      2490      DECODE ();      !Report the error and kick the bucket
:      2491      return;      !Just for show. Decode kills it
:      2492      end;
:      2493
:      2494      end;
:      2495      !
:      2496      ! "GET DUST STATUS" command end packet
:      2497      !
:      2498
:      2499      [EOP GDS] :
:      2500      begin
:      2501
:      2502      if .RET_EN$AD [MSG_LENGTH] nequ ESZ_GDS      !Is the byte count c   ect
:      2503      then
:      2504      begin
:      2505      RET_STATUS = MLE_CODE; !Return a "message length error code"
:      2506      DECODE ();      !Report the error and kick the bucket
:      2507      return;      !Just for show. Decode kills it
:      2508      end;
:      2509
:      2510      end;
:      2511      !
:      2512      ! "EXECUTE SUPPLIED PROGRAM" command end packet
:      2513      !

```

```

:      2514
:      2515      [EOP ESP] :
:      2516      begin
:      2517
:      2518      if .RET_EN$AD [MSG_LENGTH] nequ ESZ_ESP      !Is the byte count correct
:      2519      then
:      2520      begin
:      2521      RET_STATUS = MLE_CODE; !Return a "message length error code"
:      2522      DECODE ();      !Report the error and kick the bucket
:      2523      return;      !Just for show. Decode kills it
:      2524      end;
:      2525
:      2526      end;
:      2527      :
:      2528      : "EXECUTE LOCAL PROGRAM" command end packet
:      2529      :
:      2530
:      2531      [EOP ELP] :
:      2532      begin
:      2533
:      2534      if .RET_EN$AD [MSG_LENGTH] nequ ESZ_ELP      !Is the byte count correct
:      2535      then
:      2536      begin
:      2537      RET_STATUS = MLE_CODE; !Return a "message length error code"
:      2538      DECODE ();      !Report the error and kick the bucket
:      2539      return;      !Just for show. Decode kills it
:      2540      end;
:      2541
:      2542      end;
:      2543      :
:      2544      : "ABORT PROGRAM" command end packet
:      2545      :
:      2546
:      2547      [EOP ABT] :
:      2548      begin
:      2549
:      2550      if .RET_EN$AD [MSG_LENGTH] nequ ESZ_ABT      !Is the byte count correct
:      2551      then
:      2552      begin
:      2553      RET_STATUS = MLE_CODE; !Return a "message length error code"
:      2554      DECODE ();      !Report the error and kick the bucket
:      2555      return;      !Just for show. Decode kills it
:      2556      end;
:      2557
:      2558      end;
:      2559      :
:      2560      : 'SET CONTROLLER CHAR' command end packet
:      2561      :
:      2562
:      2563      [EOP_SCC] :
:      2564      begin
:      2565
:      2566      if .RET_EN$AD [MSG_LENGTH] nequ ESZ_SCC      !Is the byte count correct
:      2567      then
:      2568      begin
:      2569      RET_STATUS = MLE_CODE; !Return a message length error code
:      2570      DECODE ();      !Report the error and kick the bucket

```

```

:      2571          return;          !Just for show. Decode kills it
:      2572          end;
:      2573
:      2574          end;
:      2575          :
:      2576          : "ON LINE" command end packet
:      2577          :
:      2578
:      2579          [EOP_ONL] :
:      2580          begin
:      2581
:      2582          if .RET_EN$AD [MSG_LENGTH] nequ ESZ_ONL      !Is the byte count correct
:      2583          then
:      2584          begin
:      2585          RET_STATUS = MLE_CODE; !Return a "message length error code"
:      2586          DECODE ();      !Report the error and kick the bucket
:      2587          return;          !Just for show. Decode kills it
:      2588          end;
:      2589
:      2590          end;
:      2591          :
:      2592          : The "OP_END" end message flag all by its self tells
:      2593          : us that the controller is flagging us of an illegal
:      2594          : command sent over the connection. Error and kick the
:      2595          : bucket.
:      2596          :
:      2597
:      2598          [OP_END] :
:      2599          begin
:      2600          RET_STATUS = IVC_CODE;
:      2601          DECODE ();
:      2602          return;
:      2603          end;
:      2604          :
:      2605          : The controller is telling us that a serious exception
:      2606          : has occured. Error and kick the bucket.
:      2607          :
:      2608
:      2609          [EOP_SEX] :
:      2610          begin
:      2611          RET_STATUS = SEX_CODE;
:      2612          DECODE ();
:      2613          return;
:      2614          end;
:      2615          :
:      2616          : Unknown end packet endcode type
:      2617          :
:      2618
:      2619          [otherwise] :
:      2620          begin
:      2621          RET_STATUS = UEC_CODE;      !Return an "unknown end code"
:      2622          DECODE ();      !Report the error and kick the bucket
:      2623          return;          !Just for show. Decode kills it
:      2624          end;
:      2625          tes;
:      2626
:      2627          !

```

```

: 2628      ! The port/controller sent the endpacket over the connection
: 2629      ! with out any problems. Now find this commands owner in the
: 2630      ! outstanding buffer and indicate to them that the command
: 2631      ! has been received.
: 2632      !
: 2633      REF_NUM = .RET_EN$AD [CMD_LREF];      !Get this rec packets cmd ref number
: 2634      !
: 2635      ! Search the outstanding command buffer for this commands
: 2636      ! reference number.
: 2637      !
: 2638      ! If found, load the buffer location with the ret en$ad
: 2639      ! and set the received flag to signify that this command
: 2640      ! has been received by this interrupt service routine.
: 2641      !
: 2642      FOUND_CMD = FALSE;      !Clear the found cmd flag
: 2643      !
: 2644      incru i from 0 to REC_ALLOCATE 1 do      !Search the buffer
: 2645      !
: 2646      if .OUT$STD_BUF [.i, CMD_REF] eqv .REF_NUM      !Is this the cmd ref
: 2647      then
: 2648      begin
: 2649      OUT$STD_BUF [.i, REC_FLG] = TRUE;      !Indicate command is received
: 2650      OUT$STD_BUF [.i, ENV_ADR] = .RET EN$AD;      !Return envelope adrs
: 2651      FOUND_CMD = TRUE;      !Indicate it was found
: 2652      exitloop;      !Exit the loop
: 2653      end;
: 2654      !
: 2655      !
: 2656      ! If the search through the command ref
: 2657      ! buffer failed to find this commands cmd
: 2658      ! reference number then die.
: 2659      !
: 2660      !
: 2661      if not .FOUND_CMD
: 2662      then
: 2663      begin
: 2664      RET_STATUS = PSE_CODE;
: 2665      DECODE ();
: 2666      return;
: 2667      end;
: 2668      !
: 2669      end;      !End of END$MSG processing
: 2670      !
: 2671      ! The set controller characteristics command
: 2672      ! disabled the reporting of attentions messages
: 2673      ! so treat this as a fatal error and die.
: 2674      !
: 2675      !
: 2676      [ATT$MSG] :      !Attention end message type
: 2677      begin
: 2678      RET_STATUS = ATN_CODE;
: 2679      DECODE ();
: 2680      return;
: 2681      end;
: 2682      !
: 2683      ! It doesn't make any since for this end message
: 2684      ! packet not to have the end message flag added

```



```

: 2685      : to this command opcode as treat it as a fatal
: 2686      : error and die.
: 2687      :
: 2688
: 2689      [CMD$MSG] :
: 2690      begin
: 2691      RET_STATUS = CMD_CODE;
: 2692      DECODE ();
: 2693      return;
: 2694      end;
: 2695
: 2696      : This end code type is of unknown origin so
: 2697      : treat is as a fatal error and die.
: 2698      :
: 2699
: 2700      [otherwise] :
: 2701      begin
: 2702      RET_STATUS = UEC_CODE;      !Unknown message type code received
: 2703      DECODE ();
: 2704      return;
: 2705      end;
: 2706      tes;
: 2707
: 2708
: 2709      : Before we leave put this receive envelope message length
: 2710      : field back to the envelope size, in bytes (Per UQ Spec).
: 2711      : This size does not include the 2 UQ's words preceding the
: 2712      : command text area.
: 2713
: 2714      RET_EN$AD [MSG_LENGTH] = RB_SIZE*2;
: 2715
: 2716      : Return this receive slot descriptor back to
: 2717      : the port to fulfill my part of the protocol.
: 2718
: 2719      RECEIVE_RING [.NRD_SLOT, OWN_BIT] = PORT_OWNED;
: 2720
: 2721      : Look at the next response ring descriptor. If its
: 2722      : host owned then continue this process else exit the
: 2723      : loop. First see if the ring reference has wrapped
: 2724      : around to the top of the ring.
: 2725
: 2726
: 2727      if .NRD_SLOT + 1 gtru REC ALLOCATE 1      !Has the ring ref wrapped around
: 2728      then
: 2729      TEMP = ZERO      !Wrap it back to zero dsc slot
: 2730      else
: 2731      TEMP = .NRD_SLOT + 1;      !Look at the next seq dsc slot
: 2732
: 2733
: 2734      : Now see if the next receive descriptor slot is host owned.
: 2735
: 2736
: 2737      if .RECEIVE_RING [.TEMP, OWN_BIT] ealu MOST OWNED      !Are we done yet
: 2738      then
: 2739      GET NRD ()      !Get the next resp desc to process
: 2740      else
: 2741      ex'tloop;      !No more to do so exit

```

```

: 2742
: 2743         end;                                !End of WHILE LOOP
: 2744
: 2745         !
: 2746         ! All response ring descriptors have been processed with out any
: 2747         ! detected errors so return to the main host code with an pass
: 2748         ! return code. But before we go clear out the interrupt indicators.
: 2749         !
: 2750         HEAD_AREA [RSP_INT] = ZEROS;           !Clear the interrupt indicator location
: 2751         HEAD_AREA [CMD_INT] = ZEROS;           !Clear out the indicator
: 2752         return RET_STATUS = PAS_CODE;         !Return a "pass code"
: 2753         end;
: 2754
: 2755         !*****END OF RESPONSE RING INTERRUPT PROCESSING*****
: 2756
: 2757         !
: 2758         ! A send ring transition interrupt could happen if at
: 2759         ! some date only one descriptor slot is allocated for
: 2760         ! commands.
: 2761         !
: 2762         ! Clear the interrupt indicator if this is true and do
: 2763         ! a return with no errors.
: 2764         !-
: 2765
: 2766         if .HEAD_AREA [CMD_INT]                 !Is this a com ring transition interrupt
: 2767         then
: 2768             begin
: 2769                 HEAD_AREA [CMD_INT] = ZERO;     !Clear out the indicator
: 2770                 return;                          !Continue on with the host code
: 2771             end;
: 2772
: 2773         !*****END OF COMMAND RING INTERRUPT PROCESSING*****
: 2774
: 2775         !
: 2776         ! Check to see if an adaptor purge is being requested
: 2777         ! by the port/controller in order to complete excution
: 2778         ! of a issued command. Remember that this is illegal
: 2779         ! during PDP-11 formatting and is concidered to be a
: 2780         ! fatal error.
: 2781         !-
: 2782
: 2783         if .HEAD AREA [ADP_CH] nequ ZERO         !Is the an adaptor purge request?
: 2784         then
: 2785             begin
: 2786                 RET_STATUS = APR CODE;           !Indicate the error code
: 2787                 DECODE ();                       !Report the error and kick the bucket
: 2788                 return;                          !Just for show. Decode kills it
: 2789             end;
: 2790
: 2791         !*****END OF ADAPTOR PURGE INTERRUPT PROCESSING*****
: 2792
: 2793         !
: 2794         ! The host program has been interrupted by an unknown interrupt
: 2795         ! source if the routine program flow reaches here.
: 2796         !
: 2797         ! Load the error code into return status and call decode to take
: 2798         ! appropriate action.

```

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5 Apr-1984 13:46:24
11 Jan-1984 13:23:12

SEQ 0177
Page 33
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (11)

```

:      2799      !
:      2800
:      2801      RET_STATUS = UIN CODE;
:      2802      DECODE ();
:      2803      return;
:      2804      end;
    
```

```

000000 010046      .SBTTL  DUP$I.SERVICE MODULE DECLARATIONS
                                DUP$I.SERVICE::
000002 010146      MOV      R0, (SP) ; 2286
000004 010246      MOV      R1, -(SP)
000006 010346      MOV      R2, -(SP)
000010 010446      MOV      R3, -(SP)
000012 010546      MOV      R4, -(SP)
000014 016700 000000G  MOV      R5, -(SP)
000020 016046 000002  MOV      RC25.ADDR, R0 ;
000024 100004      BPL      1$ ; *,RC$S.REG 2357
000026 012767 000021 000000G  MOV      #21,RET.STATUS ;
000034 000546      BR      9$ ; 2360
000036 016700 000000G 1$: MOV      HEAD.AREA, R0 ; 2361
000042 032760 000001 000006  BIT      #1,6(R0) ; 2388
000050 001002      BNE      2$
000052 000167 000550      JMP      21$
000056 004767 176424 2$: JSR      PC.GET.NRD ;
000062 016700 000000G  MOV      NRD.SLOT, R0 ; 2391
000066 006300      ASL      R0 ; 2399
000070 006300      ASL      R0
000072 066700 000000G  ADD      RECEIVE.RING, R0
000076 032760 100000 000002  BIT      #100000,2(R0)
000104 001404      BEQ      4$
000106 012767 000061 000000G 3$: MOV      #61,RET.STATUS ; 2402
000114 000570      BR      16$ ; 2403
000116 104422 4$: TRAP      22 ; 2416
000120 016700 000000G  MOV      NRD.SLOT, R0 ; 2424
000124 006300      ASL      R0
000126 006300      ASL      R0
000130 066700 000000G  ADD      RECEIVE.RING, R0
000134 011067 000000G  MOV      (R0),RET.EN$AD
000140 162767 000004 000000G  SUB      #4,RET.EN$AD
000146 016701 000000G  MOV      RET.EN$AD, R1 ;
000152 116100 000014      MOV      14(R1), R0 ; 2444
000156 006200      ASR      R0
000160 006200      ASR      R0
000162 006200      ASR      R0
000164 006200      ASR      R0
000166 006200      ASR      R0
000170 006200      ASR      R0
000172 042700 177774      BIC      #177774, R0
000176 020027 000002      CMP      R0, #2
000202 001121      BNE      14$
000204 005002      CLR      R2 ;
000206 156102 000014  BISB      14(R1), R2 ; 2461
000212 020227 000205      CMP      R2, #205
000216 001007      BNE      6$
000220 021127 000060 5$: CMP      (R1), #60 ; 2470
    
```

000224	001453			BEQ	10\$			
000226	012767	000071	000000G	MOV	#71,RET.STATUS	:		2473
000234	000520			BR	16\$:		2474
000236	020227	000204		6\$: CMP	R2,#204	:		2461
000242	001766			BEQ	5\$:		2486
000244	020227	000201		CMP	R2,#201	:		2461
000250	001763			BEQ	5\$:		2502
000252	020227	000202		CMP	R2,#202	:		2461
000256	001760			BEQ	5\$:		2518
000260	020227	000203		CMP	R2,#203	:		2461
000264	001755			BEQ	5\$:		2534
000266	020227	000206		CMP	R2,#206	:		2461
000272	001752			BEQ	5\$:		2550
000274	020227	000204		CMP	R2,#204	:		2461
000300	001747			BEQ	5\$:		2566
000302	020227	000211		CMP	R2,#211	:		2461
000306	001744			BEQ	5\$:		2582
000310	020227	000200		CMP	R2,#200	:		2461
000314	001004			BNE	7\$:		
000316	012767	000701	000000G	MOV	#701,RET.STATUS	:		2600
000324	000565			BR	24\$:		2601
000326	020227	000207		7\$: CMP	R2,#207	:		2461
000332	001004			BNE	8\$:		
000334	012767	000601	000000G	MOV	#601,RET.STATUS	:		2611
000342	000556			BR	24\$:		2612
000344	012767	000101	000000G	8\$: MOV	#101,RET.STATUS	:		2621
000352	000552			9\$: BR	24\$:		2622
000354	016700	000000G		10\$: MOV	RET.EN\$AD,R0	:		2633
000360	016005	000004		MOV	4(R0),R5	:	*.REF.NUM	
000364	005004			CLR	R4	:	FOUND.CMD	2642
000366	005000			CLR	R0	:	I	2644
000370	005001			11\$: CLR	R1	:		2646
000372	156001	000000G		BISB	OUT\$STD.BUF(R0),R1	:	*(I),*	
000376	020105			CMP	R1,R5	:	*.REF.NUM	
000400	001011			BNE	12\$:		
000402	052760	100000	000000G	BIS	#100000,OUT\$STD.BUF(R0)	:	*.*(I)	2649
000410	016760	000000G	000002G	MOV	RET.EN\$AD,OUT\$STD.BUF+2(R0)	:	*.*(I)	2650
000416	012704	000001		MOV	#1,R4	:	*.FOUND.CMD	2651
000422	000405			BR	13\$:		2648
000424	062700	000004		12\$: ADD	#4,R0	:	*.I	2644
000430	020027	000014		CMP	R0,#14	:	I,*	
000434	101755			BLOS	11\$:		
000436	032704	000001		13\$: BIT	#1,R4	:	*.FOUND.CMD	2661
000442	001016			BNE	17\$:		
000444	000620			BR	3\$:		2664
000446	020027	000001		14\$: CMP	R0,#1	:		2444
000452	001004			BNE	15\$:		
000454	012767	000401	000000G	MOV	#401,RET.STATUS	:		2678
000462	000506			BR	24\$:		2679
000464	005700			15\$: TST	R0	:		2444
000466	001326			BNE	8\$:		
000470	012767	000501	000000G	MOV	#501,RET.STATUS	:		2691
000476	000500			16\$: BR	24\$:		2692
000500	012777	000074	000000G	17\$: MOV	#74,RET.EN\$AD	:		2714
000506	016700	000000G		MOV	NRD.SLOT,R0	:		2719
000512	006300			ASL	R0	:		
000514	006300			ASL	R0	:		

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0179
Page 35
VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (11)

000516	066700	000000G		ADD	RECEIVE.RING,RO		
000522	052760	100000	000002	BIS	#100000,2(RO)		
000530	016700	000000G		MOV	NRD.SLOT,RO	:	2727
000534	005200			INC	RO		
000536	020027	000003		CMP	RO,#3		
000542	101402			BLOS	18\$		
000544	005003			CLR	R3	: TEMP	2729
000546	000401			BR	19\$:	2727
000550	010003		18\$:	MOV	RO,R3	: *.TEMP	2731
000552	010300		19\$:	MOV	R3,RO	: TEMP,*	2737
000554	006300			ASL	RO		
000556	006300			ASL	RO		
000560	066700	000000G		ADD	RECEIVE.RING,RO		
000564	032760	100000	000002	BIT	#100000,2(RO)		
000572	001004			BNE	20\$		
000574	004767	175706		JSR	PC,GET.NRD	:	2739
000600	000167	177312		JMP	4\$:	2737
000604	016700	000000G	20\$:	MOV	HEAD.AREA,RO	:	2750
000610	005060	000006		CLR	6(RO)		
000614	005060	000004		CLR	4(RO)	:	2751
000620	005067	000000G		CLR	RET.STATUS	:	2752
000624	000427			BR	25\$:	2390
000626	016700	000000G	21\$:	MOV	HEAD.AREA,RO	:	2766
000632	032760	000001	000004	BIT	#1,4(RO)		
000640	001403			BEQ	22\$		
000642	005060	000004		CLR	4(RO)	:	2769
000645	000416			BR	25\$:	2768
000650	016700	000000G	22\$:	MOV	HEAD.AREA,RO	:	2783
000654	105760	000003		TSTB	3(RO)		
000660	001404			BEQ	23\$		
000662	012767	000201	000000G	MOV	#201,RET.STATUS	:	2786
000670	000403			BR	24\$:	2787
000672	012767	000301	000000G	MOV	#301,RET.STATUS	:	2801
000700	004767	176010	23\$:	JSR	PC,DECODE	:	2802
000704	005726		24\$:	TST	(SP)+	:	2286
000706	012605		25\$:	MOV	(SP)+,R5		
000710	012604			MOV	(SP)+,R4		
000712	012603			MOV	(SP)+,R3		
000714	012602			MOV	(SP)+,R2		
000716	012601			MOV	(SP)+,R1		
000720	012600			MOV	(SP)+,RO		
000722	000002			RTI			

: Routine Size: 234 words, Routine Base: AC\$CODE - 14/0
: Maximum stack depth per invocation: 13 words

: 2805

```

:      2806 global routine CTO_WAIT (TO_VALUE, REF_NUM, BUF$LOC) = !Controller time out wait
:      2807
:      2808 !**
:      2809 ! Functional Description :
:      2810 !   This routine is called to wait for the port/controller to either
:      2811 !   complete the queued command or time out the command.
:      2812 ! Formal Parameters :
:      2813 !   TO_VALUE      Indicate the time-out interval for this command.
:      2814 !   REF_NUM       This argument contains the unique reference number
:      2815 !                 assigned to this command being timed out by this
:      2816 !                 routine.
:      2817 !   BUF$LOC       This argument points to the out$std_buf location
:      2818 !                 where this command is saved. At this locat on the
:      2819 !                 received flag "rec_flg" bit is examined within the
:      2820 !                 the timeout loop and when it equals true will
:      2821 !                 signal that this command has been received by the
:      2822 !                 interrupt service routine.
:      2823 ! Implicit Inputs :
:      2824 !   none
:      2825 ! Implicit Outputs :
:      2826 !   The command word in the out$std_buffer 'word zero of a command
:      2827 !   slot' is cleared out with the value #0'100000' to indicate this
:      2828 !   is an unused out$std_buffer slot and that it can be reused.
:      2829 ! Completion Codes :
:      2830 !   There are two levels of return status returned by this routine.
:      2831 !   1. The DUP interrupt service returns to this routine a status code
:      2832 !   to indicate the success of the connection/communications mechanism
:      2833 !   to complete the queued command. If the port/controller does not
:      2834 !   time out then this return status is returned as this routines
:      2835 !   return status code.
:      2836 !   2. If the port/controller times out then the SA Register error bit
:      2837 !   is examined for the error bit set. If set then an port fatal
:      2838 !   error code is returned to the calling routine else a controller
:      2839 !   time out error code is returned.
:      2840 !   In all cases, if an error code is returned (bit 0 = 1) then the
:      2841 !   routine decode is called to decode the error code and does the
:      2842 !   necessary recovery actions.
:      2843 !   At the next higher level of return from this routine is another level
:      2844 !   of return status returned. This level test the success of the
:      2845 !   connection and also test t ; status field in the returned response
:      2846 !   envelope for the success of the controller to successfully complete
:      2847 !   the requested command.
:      2848 ! Side Effects :
:      2849 !   none
:      2850 !--
:      2851
:      2852 begin
:      2853
:      2854 !*
:      2855 ! Before doing the timeout wait make sure that this buffer location
:      2856 ! that we're suppose to time out actually contains the command ref
:      2857 ! number that was sent to us via the formal argument. Error and
:      2858 ! kick the bucket if not the same.
:      2859 !-
:      2860
:      2861 if .OUT$STD BUF [.BUF$LOC, CMD_REF] nequ .REF_NUM !Is this the same ref_num
:      2862 then

```

```

:      2863      begin
:      2864      RET_STATUS = OSE_CODE;          !Indicate the error code
:      2865      DECODE ();                    !Call decode to report the error
:      2866      end;
:      2867
:      2868      !+
:      2869      ! Loop on a one micro second delay for the number of times
:      2870      ! requested by the caller. After each delay see if the flag
:      2871      ! "rec_flg" has been set yet. Return "Ret_status" and clear the
:      2872      ! command word to %o'100000' to indicate this command has been
:      2873      ! received if this flag gets set before the timer expires.
:      2874      ! Return a error code if the timer expires before the flag gets set.
:      2875      !-
:      2876
:      2877      incru i from 0 to .TO_VALUE do      !Loop for time-out_value
:      2878      begin
:      2879      DELAY (C_US);                      !Do the one micro second delay
:      2880      !
:      2881      ! Exit routine with the DUP interrupt service routines "ret_status" if
:      2882      ! "rec_flg" got set before the timer expires.
:      2883      !
:      2884
:      2885      if .OUT$STD_BUF [.BUF$LOC, REC_FLG] eq 1 TRUE !Is this command received yet
:      2886      then
:      2887      begin
:      2888      OUT$STD_BUF [.BUF$LOC, CMD_WRD] = %o'100000'; !Return the slot to the unused state
:      2889      return .RET_STATUS;                    !Return the interrupt service status
:      2890      end;
:      2891
:      2892      !
:      2893      ! Check the SA register for posted port errors just in case the
:      2894      ! port does not interrupt the host when a port error occurs.
:      2895      !
:      2896
:      2897      if .RC25_ADDR [RCSA, ERR_BIT] then return RET_STATUS = PFE_CODE;
:      2898
:      2899      BREAK;                                !Service any control C's
:      2900      end;
:      2901
:      2902      !
:      2903      ! The port/controller timed out if the code reached here. Return an error
:      2904      ! code to the caller and exit the routine.
:      2905      !
:      2906
:      2907      if .RC25_ADDR [RCSA, ERR_BIT]          !Is the SA error bit set
:      2908      then
:      2909      return RET_STATUS = PFE_CODE          !Port timed out with fatal error
:      2910      else
:      2911      return RET_STATUS = CTO_CODE;        !Port just timed out
:      2912
:      2913      end;

```

.GLOBL L\$DLY

.SBTTL CTO.WAIT MODULE DECLARATIONS

N14

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0182
Page 38
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (12)

```

000000 004167 000000G      CTO.WAIT::
000004 162706 000006      JSR    R1,$SAVE3          ;
000010 016600 000020      SUB    #6,SP              ;
000014 006300          MOV    20(SP),R0          ; BUF$LOC,*
000016 006300          ASL    R0
000020 012703 000000G      ASL    R0
000024 060003          MOV    #OUT$STD.BUF,R3
000026 005000          ADD    R0,R3
000030 151300          CLR    R0
000032 020066 000022      BISB  (R3),R0
000036 001405          CMP    R0,22(SP)
000040 012767 003001 000000G    BEQ    1$
000046 004767 175716          MOV    #3001,RET.STATUS
000052 005002          JSR    PC,DECODE
000054 000436          CLR    R2
000056 012701 000001      BR     10$
000062 001411          2$:  MOV    #1,R1
000064 016700 000000G      3$:  BEQ    6$
000070 001404          MOV    L$DLY,R0
000072 005066 000004      BEQ    5$
000076 005300          4$:  CLR    4(SP)
000100 001374          DEC    R0
000102 005301          BNE   4$
000104 000766          5$:  DEC    R1
000106 005713          BR     3$
000110 100005          6$:  TST   (R3)
000112 012713 100000      BPL   7$
000116 016700 000000G    MOV   #-100000,(R3)
000122 000427          MOV   RET.STATUS,R0
000124 016700 000000G      BR     12$
000130 016066 000002 000002  7$:  MOV   RC25.ADDR,R0
000136 100003          MOV   2(R0),2(SP)
000140 012700 000021      BPL   9$
000144 000414          8$:  MOV   #21,R0
000146 104422          BR     11$
000150 005202          9$:  TRAP  22
000152 020266 000024      INC   R2
000156 101737          10$: CMP   R2,24(SP)
000160 016700 000000G      BLOS  2$
000164 016016 000002      MOV   RC25.ADDR,R0
000170 100763          MOV   2(R0),(SP)
000172 012700 000011      BMI   8$
000176 010067 000000G      11$: MOV   #11,R0
000202 062706 000006      12$: MOV   R0,RET.STATUS
000206 000207          ADD   #6,SP
          RTS   PC

```

; Routine Size: 68 words, Routine Base: AC\$CODE + 2414
; Maximum stack depth per invocation: 9 words

; 2914


```

: 2915 global routine ABORT * !Aborts remote program
: 2916
: 2917 **
: 2918 : Functional Description :
: 2919 : The abort program command is used to terminate the execution of a
: 2920 : remote program in an orderly fashion. When a successful response
: 2921 : is received to this command the remote program has stopped
: 2922 : executing and the server is in idle state. Note that the sending
: 2923 : of this command does not preclude further send data or receive data
: 2924 : exchanges: On the contrary, the remote program may be designed to
: 2925 : send out termination status and possibly even ask questions during
: 2926 : its forced exit sequence. The time out for this command is a fixed
: 2927 : 10 seconds and if a response is not received by then the
: 2928 : connection to the dust should be terminated. This command is only
: 2929 : legal if the dust is in active state.
: 2930
: 2931 : Formal Parameters :
: 2932 : none
: 2933
: 2934 : Implicit Inputs :
: 2935 : NSD_SLOT This global storage gets loaded by the routine
: 2936 : 'Get_nsd' and in it is stored the next send ring
: 2937 : descriptor slot where the port/controller should
: 2938 : be polling on and the place to put this commands
: 2939 : command packet.
: 2940
: 2941 : Implicit Outputs :
: 2942 : none
: 2943
: 2944 : Completion Codes :
: 2945 : RET_STATUS: Return status passes back to the calling routine
: 2946 : the status of the just issued command.
: 2947
: 2948 : Side Effects :
: 2949 : Any remote program running in the controllers DM machine will
: 2950 : be aborted.
: 2951 :--
: 2952
: 2953 begin
: 2954
: 2955 local
: 2956 REF_NUM, !Stores unique cmd ref number
: 2957 ABO_BUF$LOC, !Stores outstanding cmd buffer location
: 2958 TEMP; !A place to put the read IP register data
: 2959
: 2960
: 2961 : Before we load up the command packet up with all this good information get
: 2962 : thenext send descriptor slot and a unique command reference number.
: 2963
: 2964 GET_NSD (); !Get the next send desc slot
: 2965 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
: 2966
: 2967 : UQ Port command envelope Header field definition
: 2968
: 2969 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_ABT; !Load the length of envelope
: 2970 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE; !Load credits
: 2971 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0; !Load message type

```

```

: 2972 SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP; !Load connection ID
: 2973 :
: 2974 ! DUP command envelope field definition
: 2975 :
: 2976 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM; !Define reference number
: 2977 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO; !Hi order ref number
: 2978 SND_ENVELOPE [.NSD_SLOT, UN_LUSED] = ZERO; !Unused low order
: 2979 SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO; !Unused hi order
: 2980 SND_ENVELOPE [.NSD_SLOT, OPCODE] = OP_ABT; !Load opcod
: 2981 SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO; !Reserved field
: 2982 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;
: 2983 :
: 2984 ! Call the load outstanding command buffer routine
: 2985 ! and load this command into the buffer. The return
: 2986 ! from this routine will point us to the buffer location
: 2987 ! where this command is stored. Later we can look at
: 2988 ! this location to see if the interrupt service routine
: 2989 ! has received and process it.
: 2990 :
: 2991 ABO_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
: 2992 :
: 2993 if .ABO_BUF$LOC eqv OBF_CODE then DECODE (); !Error if buffer is full
: 2994 :
: 2995 :
: 2996 ! Set the ownership bit to 1 giving this slot to the port/controller
: 2997 :
: 2998 SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
: 2999 :
: 3000 ! Read the IP register to stimulate port polling
: 3001 :
: 3002 TEMP = .RC25_ADDR [RCIP, RC_ALL];
: 3003 :
: 3004 ! Time out the port/controller processing the command.
: 3005 :
: 3006 ! The first test tests the connections ability to
: 3007 ! respond to this command without any errors in the SA
: 3008 ! register and for the command not timing out.
: 3009 :
: 3010 ! The second tests the DUP server for good status. If
: 3011 ! bad status is sent back then an error code is returned
: 3012 ! to the calling routine where the routine "decode" will
: 3013 ! decode and take the appropriate recovery. The time
: 3014 ! out routine will loop on delaying and checking the hi
: 3015 ! bit of the first word in the outstd buf for a true.
: 3016 ! When true signals us that the interrupt service routine
: 3017 ! has received the endpacket and no connection errors
: 3018 ! were detected.
: 3019 :
: 3020 :
: 3021 if CTO_WAIT (3000, .REF_NUM, .ABO_BUF$LOC) then DECODE (); !Is return an error
: 3022 :
: 3023 :
: 3024 ! Get the return envelope address from the outstd buf
: 3025 ! at this commands buffer location and check the packet
: 3026 ! for good status error and die if bad status was returned
: 3027 :
: 3028 RET_ENV$ADR = .OUT$STD_BUF [.ABO_BUF$LOC, ENV_ADR]; !Get the ret env adr

```

```

:      3029      :
:      3030      : Now test for good status
:      3031      :
:      3032      :
:      3033      : if .RET EN$AD [STATUS] nequ ZERO
:      3034      : then
:      3035      :     return RET STATUS + RSE CODE
:      3036      : else
:      3037      :     return .RET STATUS;
:      3038      :
:      3039      : end;

```

```

!Test the status
!Return a "Response status err" code
!This ret status is good or bad

```

000000	004167	000000G	ABORT::	.SBTTL	ABORT MODULE DECLARATIONS		
000004	005746			JSR	R1,\$SAVE2	:	2915
000006	004767	175312		TST	(SP)	:	
000012	004767	175454		JSR	PC,GET.NSD	:	2964
000016	010002			JSR	PC,GET.CMD\$REF	:	2965
000020	016746	000000G		MOV	R0,R2	:	*.REF.NUM
000024	012746	000054		MOV	NSD.SLOT,-(SP)	:	2969
000030	004767	000000G		MOV	#54,-(SP)	:	
000034	012760	000014 000000G		JSR	PC,BL\$MUL	:	
000042	012701	000002G		MOV	#14,SND.ENVELOPE(R0)	:	
000046	060001			MOV	#SND.ENVELOPE+2,R1	:	2970
000050	112711	000001		ADD	R0,R1	:	
000054	112761	000002 000001		MOVB	#1,(R1)	:	2971
000062	010260	000004G		MOVB	#2,1(R1)	:	2972
000066	005060	000006G		MOV	R2,SND.ENVELOPE+4(R0)	:	REF.NUM,*
000072	005060	000010G		CLR	SND.ENVELOPE+6(R0)	:	2976
000076	005060	000012G		CLR	SND.ENVELOPE+10(R0)	:	2977
000102	112760	000006 000014G		CLR	SND.ENVELOPE+12(R0)	:	2978
000110	105060	000015G		MOVB	#6,SND.ENVELOPE+14(R0)	:	2979
000114	005060	000016G		CLR	SND.ENVELOPE+15(R0)	:	2980
000120	010216			CLR	SND.ENVELOPE+16(R0)	:	2981
000122	004767	175252		MOV	R2,(SP)	:	REF.NUM,*
000126	010001			JSR	PC,LOAD.OUT\$STD.BUF	:	2991
000130	020127	002001		MOV	R0,R1	:	*.ABO.BUF\$LOC
000134	001002			CMP	R1,#2001	:	ABO.BUF\$LOC,*
000136	004767	175416		BNE	1\$:	2993
000142	016700	000000G	1\$:	JSR	PC,DECODE	:	
000146	006300			MOV	NSD.SLOT,R0	:	2998
000150	006300			ASL	R0	:	
000152	0667 0	000000G		ASL	R0	:	
000156	052760	100000 000002		ADD	SEND.RING,R0	:	
000164	017766	000000G 000004		BIS	#100000,2(R0)	:	
000172	016600	000004		MOV	#RC25.ADDR,4(SP)	:	*.RC\$S.REG
000176	012716	005670		MOV	4(SP),R0	:	RC\$S.REG,TEMP
000202	010246			MOV	#5670,(SP)	:	3002
000204	010146			MOV	R2,(SP)	:	REF.NUM,*
000206	004767	177356		MOV	R1,(SP)	:	ABO.BUF\$LOC,*
000212	022626			JSR	PC,CTO.WAIT	:	
000214	006000			CMP	(SP)*,(SP)*	:	
000216	103002			ROR	R0	:	
000220	004767	175334		BCC	2\$:	
000224	010100		2\$:	JSR	PC,DECODE	:	
000226	006300			MOV	R1,R0	:	ABO.BUF\$LOC,*
				ASL	R0	:	3028

E15

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0186
Page 42
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE,AZTEC]ZRCH85.B16;4 (13)

000230	006300			ASL	R0		
000232	016067	000002G	0J0000G	MOV	OUT\$STD.BUF+2(R0),RET.EN\$AD		
000240	016000	000002G		MOV	OUT\$STD.BUF+2(R0),R0	:	3033
000244	005760	000016		TST	16(R0)		
000250	001405			BEQ	3\$		
000252	012700	000031		MOV	#31,R0	:	3035
000256	010067	000000G		MOV	R0,RET.STATUS		
000262	000402			BR	4\$:	2953
000264	016700	000000G	3\$:	MOV	RET.STATUS,R0		
000270	062706	000006	4\$:	ADD	#6,SP	:	2915
000274	000207			RTS	PC		

: Routine Size: 95 words, Routine Base: AC\$CODE + 2624
: Maximum stack depth per invocation: 9 words

: 3040

```

: 3041 global routine GET_DUST STATUS = !Gets DUP server status
: 3042
: 3043 !..
: 3044 ! Functional Description :
: 3045 ! This command allows the host program to interrogate the DUP server
: 3046 ! to determine its characteristics, its state and the state of the
: 3047 ! program currently running (if any). It is legal in either idle or
: 3048 ! active state and does not affect the state of server. It has a
: 3049 ! fixed timeout interval of 3 seconds. If the response times out, the
: 3050 ! host should break the connection.
: 3051
: 3052 ! Formal Parameters :
: 3053 ! none
: 3054
: 3055 ! Implicit Inputs :
: 3056 ! NSD_SLOT This global storage gets loaded by the routine
: 3057 ! 'Get_nsd' and in it is stored the next send ring
: 3058 ! descriptor slot where the port/controller should
: 3059 ! be polling on and the place to put this commands
: 3060 ! command packet.
: 3061
: 3062 ! Implicit Outputs :
: 3063 ! none
: 3064
: 3065 ! Completion Codes :
: 3066 ! RET_STATUS: Return status passes back to the calling routine
: 3067 ! the status of the just issued command.
: 3068
: 3069 ! Side Effects :
: 3070
: 3071
: 3072 begin
: 3073
: 3074 local
: 3075 REF_NUM, !Stores unique cmd ref number
: 3076 GDS_BUF$LOC, !Stores outstanding cmd buffer location
: 3077 TEMP; !A place to put the IP read data
: 3078
: 3079
: 3080 ! Before we load up the command packet up with all this good information get
: 3081 ! the next send descriptor slot and a unique command reference number.
: 3082
: 3083 GET_NSD (); !Get the next send desc slot
: 3084 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
: 3085
: 3086 ! UQ Port command envelope Header field definition
: 3087
: 3088 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ GDS; !Load the envelope size
: 3089 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE; !Load the credit size
: 3090 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0; !Load the message type (Sequential)
: 3091 SND_ENVELOPE [.NSD_SLOT, CONN ID] = DUP; !Load the connection ID
: 3092
: 3093 ! DUP generic command envelope field definition
: 3094
: 3095 SND ENVELOPE [.NSD SLOT, CMD_LREF] = .REF_NUM; !Load command reference number
: 3096 SND ENVELOPE [.NSD SLOT, CMD_HREF] = ZERO; !Command reference low order
: 3097 SND ENVELOPE [.NSD SLOT, UN LUSED] = ZERO; !Low order unused

```

```

:      3098     SND_ENVELOPE [.NSD_SLOT, UN MUSED] = ZERO;  !Hi order unused
:      3099     SND_ENVELOPE [.NSD_SLOT, OPCODE] = OP_GDS;  !Load opcode
:      3100     SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;      !Reserved field
:      3101     SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;  !Load modifier field
:      3102     :
:      3103     ! Call the load outstanding command buffer routine
:      3104     ! and load this command into the buffer. The return
:      3105     ! from this routine will point us to the buffer location
:      3106     ! where this command is stored. Later we can look at
:      3107     ! this location to see if the interrupt service routine
:      3108     ! has received and process it.
:      3109     :
:      3110     GDS_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM);  !Load the command
:      3111     :
:      3112     if .GDS_BUF$LOC eqv OBF_CODE then DECODE ();  !Error if buffer is full
:      3113     :
:      3114     :
:      3115     ! Set the ownership bit to 1 giving this slot to the port/controller
:      3116     :
:      3117     SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
:      3118     :
:      3119     ! Read the IP register to stimulate port polling
:      3120     :
:      3121     TEMP = .RC25_ADDR [RCIP, RC_ALL];
:      3122     :
:      3123     ! Time out the port/controller processing the command.
:      3124     :
:      3125     ! The first test tests the connections ability to
:      3126     ! respond to this command without any errors in the SA
:      3127     ! register and for the command not timing out.
:      3128     :
:      3129     ! The second tests the DUP server for good status. If
:      3130     ! bad status is sent back then an error code is returned
:      3131     ! to the calling routine where the routine "decode" will
:      3132     ! decode and take the appropriate recovery. The time
:      3133     ! out routine will loop on delaying and checking the hi
:      3134     ! bit of the first word in the out$std_buf for a true.
:      3135     ! When true signals us that the interrupt service routine
:      3136     ! has received the endpacket and no connection errors
:      3137     ! were detected.
:      3138     :
:      3139     :
:      3140     if CTO_WAIT (3500, .REF_NUM, .GDS_BUF$LOC) then DECODE ();  !Is return an error
:      3141     :
:      3142     :
:      3143     ! Get the return envelope address from the out$std_buf
:      3144     ! at this commands buffer location and check the packet
:      3145     ! for good status error and die if bad status was returned
:      3146     :
:      3147     RET_EN$AD = .OUT$STD_BUF [.GDS_BUF$LOC, ENV ADR];  !Get the ret env adr
:      3148     :
:      3149     ! Now test for good status
:      3150     :
:      3151     :
:      3152     if .RET_EN$AD [STATUS] neqv ZERO  !Test the status
:      3153     then
:      3154         return RET STATUS = RSE CODE  !Return a 'Response status err' code

```

```

:      3155      else
:      3156      return .RET_STATUS;
:      3157
:      3158      end;

```

!This ret_status is good or bad

```

000000 004167 000000G      .SBTTL GET.DUST.STATUS MODULE DECLARATIONS
                                GET.DUST.STATUS:
000004 005746      JSR      R1,$SAVE2      ; 3041
000006 004767 175014      TST      -(SP)
000012 004767 175156      JSR      PC,GET.NSD      ; 3083
000016 C10002      JSR      PC,GET.CMD$REF  ; 3084
000020 016746 000000G      MOV      R0,R2      ; *,REF.NUM
000024 012746 000054      MOV      NSD.SLOT,-(SP) ; 3088
000030 004767 000000G      JSR      PC,BL$MUL
000034 012760 000014 000000G  MOV      #14,SND.ENVELOPE(R0)
000042 012701 000002G      MOV      #SND.ENVELOPE+2,R1 ; 3089
000046 060001      ADD      R0,R1
000050 112711 000001      MOV      #1,(R1) ; 3090
000054 112761 000002 000001      MOV      #2,1(R1) ; 3091
000062 010260 000004G      MOV      R2,SND.ENVELOPE+4(R0) ; REF.NUM,* 3095
000066 005060 000006G      CLR      SND.ENVELOPE+6(R0) ; 3096
000072 005060 000010G      CLR      SND.ENVELOPE+10(R0) ; 3097
000076 005060 000012G      CLR      SND.ENVELOPE+12(R0) ; 3098
000102 112760 000001 000014G  MOV      #1,SND.ENVELOPE+14(R0) ; 3099
000110 105060 000015G      CLR      SND.ENVELOPE+15(R0) ; 3100
000114 005060 000016G      CLR      SND.ENVELOPE+16(R0) ; 3101
000120 010216      MOV      R2,(SP) ; REF.NUM,* 3110
000122 004767 174754      JSR      PC,LOAD.OUT$STD.BUF
000126 010001      MOV      R0,R1 ; *,GDS.BUF$LOC
000130 020127 002001      CMP      R1,#2001 ; GDS.BUF$LOC,* 3112
000134 001002      BNE      1$
000136 004767 175120      JSR      PC,DECODE
000142 016700 000000G      1$: MOV      NSD.SLOT,R0 ; 3117
000146 006300      ASL      R0
000150 006300      ASL      R0
000152 066700 000000G      ADD      SEND.RING,R0
000156 052760 100000 000002      BIS      #100000,2(R0)
000164 017766 000000G 000004      MOV      #RC25.ADDR,4(SP) ; *,RC$S.REG 3121
000172 016600 000004      MOV      4(SP),R0 ; RC$S.REG,TEMP
000176 012716 006654      MOV      #6654,(SP) ; 3140
000202 010246      MOV      R2,-(SP) ; REF.NUM,*
000204 010146      MOV      R1,-(SP) ; GDS.BUF$LOC,*
000206 004767 177060      JSR      PC,CTO.WAIT
000212 022626      CMP      (SP)*,(SP)*
000214 006000      ROR      R0
000216 103002      BCC      2$
000220 004767 175036      JSR      PC,DECODE
000224 010100 2$: MOV      R1,R0 ; GDS.BUF$LOC,* 3147
000226 006300      ASL      R0
000230 006300      ASL      R0
000232 016067 000002G 000000G  MOV      OUT$STD.BUF+2(R0),RET.EN$AD
000240 016000 000002G      MOV      OUT$STD.BUF+2(R0),R0 ; 3152
000244 005760 000016      TST      16(R0)
000250 001405      BEQ      3$
000252 012700 000031      MOV      #31,R0 ; 3154

```

ZRCH85 ZRCH80 RC25 DISK FORMATTER
REV B PATCH 00 MODULE DECLARATIONS

5-Apr 1984 13:46:24
11 Jan 1984 13:23:12

SEQ 0190
Page 46
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (14)

000256	010067	000000G		MOV	RO,RET.STATUS		
000262	000402			BR	4\$:	3072
000264	016700	000000G	3\$:	MOV	RET.STATUS,RO		
000270	062706	0000006	4\$:	ADD	06,SP	:	3041
000274	000207			RTS	PC		

: Routine Size: 95 words, Routine Base: AC\$CODE + 3122
: Maximum stack depth per invocation: 9 words

: 3159


```

:      3160 global routine EX SUP_PROG =                !Executes supplied program
:      3161
:      3162 !..
:      3163 ! Functional Description :
:      3164 !   This command causes the server to transfer the program from host
:      3165 !   memory to an area in the controller and start its execution. The
:      3166 !   host supplies the address and length (in bytes) of a buffer
:      3167 !   containing the program header and initial load; the starting
:      3168 !   of the program, its memory requirements and any relocation information
:      3169 !   needed to run under the server are in the program header in a format
:      3170 !   which is none of the host business. This command is only legal when
:      3171 !   the server is in the idle state and return of a successful end packet
:      3172 !   puts the server into to active state.
:      3173
:      3174 !   The time out interval for this command is 30 seconds.
:      3175
:      3176 ! Formal Parameters :
:      3177 !   none
:      3178
:      3179 ! Implicit Inputs :
:      3180 !   NSD_SLOT      This global storage gets loaded by the routine
:      3181 !                   'Get_nsd' and in it is stored the next send ring
:      3182 !                   descriptor slot where the port/controller should
:      3183 !                   be polling on and the place to put this commands
:      3184 !                   command packet.
:      3185
:      3186
:      3187 ! Implicit Outputs :
:      3188 !   AZFMTR:        Azfmtr is the vector produced by DMCONV program and
:      3189 !                   is decalared in module AZKEL6.
:      3190 !   DMSA:          These three bound addresses point to specific area
:      3191 !   OVSA:          in the DM code buffer 'azfmtr' and are used to
:      3192 !   MDSA:          define the buffer descriptors within this command.
:      3193
:      3194 ! Completion Codes :
:      3195 !   RET_STATUS:   Return status passes back to the calling routine
:      3196 !                   the status of the just issued command.
:      3197
:      3198 ! Side Effects :
:      3199 !   The DM machine in the controller goes from the idle state to the
:      3200 !   active state on return of a successful return packet.
:      3201 !..
:      3202
:      3203 begin
:      3204
:      3205 local
:      3206     REF_NUM,                !Stores unique cmd ref number
:      3207     ESP_BUF$LOC,          !Stores outstanding cmd buffer location
:      3208     TEMP;                 !A place to put the read IP register data
:      3209
:      3210
:      3211 ! Before we load up the command packet up with all this good information get
:      3212 ! the next send descriptor slot and a unique command reference number.
:      3213
:      3214 GET_NSD ();                !Get the next send desc slot
:      3215 REF_NUM = GET_CMD$REF (); !Get a un que command ref num
:      3216

```

```

: 3217      ! UQ Port command envelope Header field definition
: 3218      !
: 3219      SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_ESP;      !Load the message length
: 3220      SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE;      !Load the credits field
: 3221      SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0;      !Define the msg type 'Sequential'
: 3222      SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP;      !Define the conection ID
: 3223      !
: 3224      ! DUP generic command envelope field definition
: 3225      !
: 3226      SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;      !Load command ref number
: 3227      SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;      !Zero Hi order word of cmd ref
: 3228      SND_ENVELOPE [.NSD_SLOT, UN_LUSED] = ZERO;      !Not used in DUP implimentation
: 3229      SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO;      !Not used in DUP implimentation
: 3230      SND_ENVELOPE [.NSD_SLOT, OPCODE] = OP_ESP;      !Load the command op-code
: 3231      SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;      !Not used
: 3232      SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;
: 3233      !
: 3234      ! Command specific command envelope field definition
: 3235      !
: 3236      ! Byte count of initial transfer (from bytes 0-3 of the program header).
: 3237      !
: 3238      SND_ENVELOPE [.NSD_SLOT, BLO_CNT] = .AZFMTR [WRD0]; !Byte count low word
: 3239      SND_ENVELOPE [.NSD_SLOT, BHI_CNT] = .AZFMTR [WRD1]; !Byte count high word
: 3240      !
: 3241      ! Buffer descriptor definition for initial load. First
: 3242      ! byte of this buffer is byte 0 of program header.
: 3243      !
: 3244      SND_ENVELOPE [.NSD_SLOT, BPA_LO] = MDSA;      !Low unibus adrs <0-15>
: 3245      SND_ENVELOPE [.NSD_SLOT, BPA_HI] = ZERO;      !Unibus adrs bits <16-17>
: 3246      SND_ENVELOPE [.NSD_SLOT, QBUS_EXT] = ZERO;      !Q_bus extention adrs
: 3247      SND_ENVELOPE [.NSD_SLOT, RSV] = ZERO;      !Reserved field
: 3248      SND_ENVELOPE [.NSD_SLOT, UBA_CHAN] = ZERO;      !Unibus adaptor channel number
: 3249      SND_ENVELOPE [.NSD_SLOT, RSV0] = ZERO;      !These next four words are not
: 3250      SND_ENVELOPE [.NSD_SLOT, RSV1] = ZERO;      !used in the DUP implementation
: 3251      SND_ENVELOPE [.NSD_SLOT, RSV2] = ZERO;      !
: 3252      SND_ENVELOPE [.NSD_SLOT, RSV3] = ZERO;      !
: 3253      !
: 3254      ! These next field definitions are the same as above except they are for the
: 3255      ! overlay buffer descriptors. To make life easy for me I'll use the same names
: 3256      ! and just prefix them with a $ for uniqueness.
: 3257      !
: 3258      ! The overlay area immediately follows the initial load image in the program image.
: 3259      !
: 3260      SND_ENVELOPE [.NSD_SLOT, $BPA_LO] = .OVSA;      !Low unibus adrs <0-15>
: 3261      SND_ENVELOPE [.NSD_SLOT, $BPA_HI] = ZERO;      !Unibus adrs bits <16-17>
: 3262      SND_ENVELOPE [.NSD_SLOT, $QBUS_EXT] = ZERO;      !Q_bus extention adrs
: 3263      SND_ENVELOPE [.NSD_SLOT, $RSV] = ZERO;      !Reserved field
: 3264      SND_ENVELOPE [.NSD_SLOT, $UBA_CHAN] = ZERO;      !Unibus adaptor channel number
: 3265      SND_ENVELOPE [.NSD_SLOT, $RSV0] = ZERO;      !These next four words are not
: 3266      SND_ENVELOPE [.NSD_SLOT, $RSV1] = ZERO;      !used in the DUP implementation
: 3267      SND_ENVELOPE [.NSD_SLOT, $RSV2] = ZERO;      !
: 3268      SND_ENVELOPE [.NSD_SLOT, $RSV3] = ZERO;      !
: 3269      !
: 3270      ! Call the load outstanding command buffer routine
: 3271      ! and load this command into the buffer. The return
: 3272      ! from this routine will point us to the buffer location
: 3273      ! where this command is stored. Later we can look at

```

```

:      3274      ! this location to see if the interrupt service routine
:      3275      ! has received and process it.
:      3276      !
:      3277      ESP_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
:      3278      !
:      3279      if .ESP_BUF$LOC eqv OBF_CODE then DECODE ();      !Error if buffer is full
:      3280      !
:      3281      !
:      3282      ! Set the ownership bit to 1 giving this slot to the port/controller
:      3283      !
:      3284      SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
:      3285      !
:      3286      ! Read the IP register to stimulate port polling
:      3287      !
:      3288      TEMP = .RC25_ADDR [RCIP, RC_ALL];
:      3289      !
:      3290      ! Time out the port/controller processing the command.
:      3291      !
:      3292      ! The first test tests the connections ability to
:      3293      ! respond to this command without any errors in the SA
:      3294      ! register and for the command not timing out.
:      3295      !
:      3296      ! The second tests the DUP server for good status. If
:      3297      ! bad status is sent back then an error code is returned
:      3298      ! to the calling routine where the routine "decode" will
:      3299      ! decode and take the appropriate recovery. The time
:      3300      ! out routine will loop on delaying and checking the hi
:      3301      ! bit of the first word in the out$std_buf for a true.
:      3302      ! When true signals us that the interrupt service routine
:      3303      ! has received the endpacket and no connection errors
:      3304      ! were detected.
:      3305      !
:      3306      !
:      3307      if CTO_WAIT (3500, .REF_NUM, .ESP_BUF$LOC) then DECODE (); !Is return an error
:      3308      !
:      3309      !
:      3310      ! Get the return envelope address from the out$std_buf
:      3311      ! at this commands buffer location and check the packet
:      3312      ! for good status error and die if bad status was returned
:      3313      !
:      3314      RET_EN$AD = .OUT$STD_BUF [.ESP_BUF$LOC, ENV_ADR]; !Get the ret env adr
:      3315      !
:      3316      ! Now test for good status
:      3317      !
:      3318      !
:      3319      if .RET_EN$AD [STATUS] neqv ZERO      !Test the status
:      3320      then
:      3321          return RET_STATUS = RSE_CODE      !Return a "Response status err" code
:      3322      else
:      3323          return .RET_STATUS;      !This ret_status 's good or bad
:      3324      !
:      3325      end;

```

M15

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

VAX-11 B11:16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (15)

SEQ 0194
Page 50

000004	005746		TST	-(SP)		
000006	004767	174516	JSR	PC,GET.NSD	:	3214
000012	004767	174660	JSR	PC,GET.CMD\$REF	:	3215
000016	010002		MOV	R0,R2	: *,REF.NUM	
000020	016746	000000G	MOV	NSD.SLOT,-(SP)	:	3219
000024	012746	000054	MOV	#54,-(SP)		
000030	004767	000000G	JSR	PC,BL\$MUL		
000034	012760	000050 000000G	MOV	#50,SND.ENVELOPE(R0)		
000042	012701	000002G	MOV	#SND.ENVELOPE+2,R1	:	3220
000046	060001		ADD	R0,R1		
000050	112711	000001	MOV	#1,(R1)	:	3221
000054	112761	000002 000001	MOV	#2,1(R1)	:	3222
000062	010260	000004G	MOV	R2,SND.ENVELOPE+4(R0)	: REF.NUM,*	3226
000066	005060	000006G	CLR	SND.ENVELOPE+6(R0)	:	3227
000072	005060	000010G	CLR	SND.ENVELOPE+10(R0)	:	3228
000076	005060	000012G	CLR	SND.ENVELOPE+12(R0)	:	3229
000102	112760	000002 000014G	MOV	#2,SND.ENVELOPE+14(R0)	:	3230
000110	105060	000015G	CLRB	SND.ENVELOPE+15(R0)	:	3231
000114	005060	000016G	CLR	SND.ENVELOPE+16(R0)	:	3232
000120	016760	000000G 000020G	MOV	AZFMTR,SND.ENVELOPE+20(R0)	:	3238
000126	016760	000002G 000022G	MOV	AZFMTR+2,SND.ENVELOPE+22(R0)	:	3239
000134	012760	000000G 000024G	MOV	#HDSA,SND.ENVELOPE+24(R0)	:	3244
000142	012701	000026G	MOV	#SND.ENVELOPE+26,R1	:	3245
000146	060001		ADD	R0,R1		
000150	105011		CLRB	(R1)	:	3247
000152	105061	000001	CLRB	1(R1)	:	3248
000156	005060	000030G	CLR	SND.ENVELOPE+30(R0)	:	3249
000162	005060	000032G	CLR	SND.ENVELOPE+32(R0)	:	3250
000166	005060	000034G	CLR	SND.ENVELOPE+34(R0)	:	3251
000172	005060	000036G	CLR	SND.ENVELOPE+36(R0)	:	3252
000176	016760	000000G 000040G	MOV	OVSA,SND.ENVELOPE+40(R0)	:	3260
000204	012701	000042G	MOV	#SND.ENVELOPE+42,R1	:	3261
000210	060001		ADD	R0,R1		
000212	105011		CLRB	(R1)	:	3263
000214	105061	000001	CLRB	1(R1)	:	3264
000220	005060	000044G	CLR	SND.ENVELOPE+44(R0)	:	3265
000224	005060	000046G	CLR	SND.ENVELOPE+46(R0)	:	3266
000230	005060	000050G	CLR	SND.ENVELOPE+50(R0)	:	3267
000234	005060	000052G	CLR	SND.ENVELOPE+52(R0)	:	3268
000240	010216		MOV	R2,(SP)	: REF.NUM,*	3277
000242	004767	174336	JSR	PC,LOAD.OUT\$STD.BUF		
000246	010001		MOV	R0,R1	: *,ESP.BUF\$LOC	
000250	020127	002001	CMP	R1,#2001	: ESP.BUF\$LOC,*	3279
000254	001002		BNE	1\$		
000256	004767	174502	JSR	PC,DECODE		
000262	016700	000000G	MOV	NSD.SLOT,R0	:	3284
000266	006300		ASL	R0		
000270	006300		ASL	R0		
000272	066700	000000G	ADD	SEND.RING,R0		
000276	052760	100000 000002	BIS	#100000,2(R0)		
000304	017766	000000G 000004	MOV	#RC25.ADDR,4(SP)	: *,RC\$S.REG,	3288
000312	016600	000004	MOV	4(SP),R0	: RC\$S.REG,TEMP	
000316	012716	006654	MOV	#6654,(SP)	:	3307
000322	010246		MOV	R2,-(SP)	: REF.NUM,*	
000324	010146		MOV	R1,-(SP)	: ESP.BUF\$LOC,*	
000326	004767	176442	JSR	PC,CTO.WAIT		
000332	022626		CMP	(SP),*(SP)		

N15

ZRCH85 ZRCH80 RC25 DISK FORMATTER
REV B PATCH 00 MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0195
Page 51
VAX-11 Bliss-16 v3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (15)

000334	006000		ROR	R0		
000336	103002		BCC	2\$		
000340	004767	174420	JSR	PC,DECODE		
000344	010100		MOV	R1,R0	; ESP.BUF\$LOC,*	3314
000346	006300		ASL	R0		
000350	006300		ASL	R0		
000352	016067	000002G 000000G	MOV	OUT\$STD.BUF+2(R0),RET.EN\$AD		
000360	016000	000002G	MOV	OUT\$STD.BUF+2(R0),R0	;	3319
000364	005760	000016	TST	16(R0)		
000370	001405		BEQ	3\$		
000372	012700	000031	MOV	#31,R0	;	3321
000376	010067	000000G	MOV	R0,RET.STATUS	;	3203
000402	000402		BR	4\$;	3160
000404	016700	000000G	MOV	RET.STATUS,R0	;	
000410	062706	000006	ADD	#6,SP	;	
000414	000207		RTS	PC		

; Routine Size: 135 words, Routine Base: AC\$CODE + 3420
; Maximum stack depth per invocation: 9 words

; 3326

```

:      3327 global routine EX LOC PROG .                !Executes local program
:      3328
:      3329 :..
:      3330 : Functional Description :
:      3331 :   Receipt of this command causes the controller to search its local
:      3332 :   media for the named program, load and execute it. Receipt of a
:      3333 :   successful response by the host means that the program is executing
:      3334 :   and the server is in the active state. The time out value for this
:      3335 :   command is specified in the get dust response
:      3336 :
:      3337 : Formal Parameters :
:      3338 :   none
:      3339 :
:      3340 : Implicit Inputs :
:      3341 :   NSD_SLOT      This global storage gets loaded by the routine
:      3342 :                  'Get_nsd' and in it is stored the next send ring
:      3343 :                  descriptor slot where the port/controller should
:      3344 :                  be polling on and the place to put this commands
:      3345 :                  command packet.
:      3346 :
:      3347 : Implicit Outputs :
:      3348 :   none
:      3349 :
:      3350 : Completion Codes :
:      3351 :   RET_STATUS:   Return status passes back to the calling routine
:      3352 :                  the status of the just issued command.
:      3353 :
:      3354 : Side Effects :
:      3355 :   The DM machine in the controller goes from the idle state to the
:      3356 :   active state on return of a successful return packet.
:      3357 : ..
:      3358
:      3359 begin
:      3360
:      3361 local
:      3362     REF_NUM,                !Stores unique cmd ref number
:      3363     ELP_BUF$LOC,          !Stores outstanding cmd buffer location
:      3364     TEMP;                !A place to store the read IP register data
:      3365
:      3366 :
:      3367 : Before we load up the command packet up with all this good information get
:      3368 : the next send descriptor slot and a unique command reference number.
:      3369 :
:      3370 GET_NSD ();                !Get the next send desc slot
:      3371 REF_NUM = GET_CMD$REF ();    !Get a unique command ref num
:      3372 :
:      3373 : UQ Port command envelope Header field definition
:      3374 :
:      3375 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ ELP;    !Load the message size
:      3376 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE;        !Load the credit size
:      3377 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0;         !Define the msg typ 'Sequential'
:      3378 SND_ENVELOPE [.NSD_SLOT, CONN ID] = DUP;        !Define the connectio ID
:      3379 :
:      3380 : DUP generic command envelope field definition
:      3381 :
:      3382 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;    !load the command ref number
:      3383 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;        !Zero the Hi order cmd ref num

```

ZRCMB5
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
MODULE DECLARATIONS5 Apr 1984 13:46:24
11-Jan-1984 13:23:12VAX-11 Bligg 16 V3 555
SPIDER@USERS:(NEALE.AZTEC)ZRCMB5.016;4 (16)SEQ 0197
Page 33

```

: 3384 SND_ENVELOPE [.NSD_SLOT, UN_LUSED] = ZERO; !Not used in DUP implimentation
: 3385 SND_ENVELOPE [.NSD_SLOT, UN_MUSED] = ZERO; !Not used in DUP implimentation
: 3386 SND_ENVELOPE [.NSD_SLOT, OPCODE] = OP_ELP; !Load this commands op code
: 3387 SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO; !Not used field
: 3388 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO; !Define modifiers for this cmd
: 3389
: 3390 ! Command specific command envelope field definition
: 3391
: 3392 SND_ENVELOPE [.NSD_SLOT, PN_0] = ZERO; !Program name word 0
: 3393 SND_ENVELOPE [.NSD_SLOT, PN_1] = ZERO; !Program name word 1
: 3394 SND_ENVELOPE [.NSD_SLOT, PN_2] = ZERO; !Program name word 2
: 3395
: 3396 ! Call the load outstanding command buffer routine
: 3397 ! and load this command into the buffer. The return
: 3398 ! from this routine will point us to the buffer location
: 3399 ! where this command is stored. Later we can look at
: 3400 ! this location to see if the interrupt service routine
: 3401 ! has received and process it.
: 3402
: 3403 ELP_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
: 3404
: 3405 if .ELP_BUF$LOC eqv 0BF_CODE then DECODE (); !Error if buffer is full
: 3406
: 3407
: 3408 ! Set the ownership bit to 1 giving this slot to the port/controller
: 3409
: 3410 SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
: 3411
: 3412 ! Read the IP register to stimulate port polling
: 3413
: 3414 TEMP = .RC25_ADDR [RCIP, RC_ALL];
: 3415
: 3416 ! Time out the port/controller processing the command.
: 3417
: 3418 ! The first test tests the connections ability to
: 3419 ! respond to this command without any errors in the SA
: 3420 ! register and for the command not timing out.
: 3421
: 3422 ! The second tests the DUP server for good status. If
: 3423 ! bad status is sent back then an error code is returned
: 3424 ! to the calling routine where the routine "decode" will
: 3425 ! decode and take the appropriate recovery. The time
: 3426 ! out routine will loop on delaying and checking the hi
: 3427 ! bit of the first word in the outstd_buf for a true.
: 3428 ! When true signals us that the interrupt service routine
: 3429 ! has received the endpacket and no connection errors
: 3430 ! were detected.
: 3431
: 3432
: 3433 if CTO_WAIT (3000, .REF_NUM, .ELP_BUF$LOC) then DECODE (); !Is return an error
: 3434
: 3435
: 3436 ! Get the return envelope address from the outstd_buf
: 3437 ! at this commands buffer location and check the packet
: 3438 ! for good status error and die if bad status was returned
: 3439
: 3440 RET EN$AD = .OUT$STD_BUF [.ELP_BUF$LOC, ENV_ADR]; !Get the ret env adr

```

```

:      3441      :
:      3442      : Now test for good status
:      3443      :
:      3444      :
:      3445      : if .RET EN$AD [STATUS] nequ ZERO      !Test the status
:      3446      : then
:      3447      :     return RET_STATUS = RSE_CODE      !Return a "Response status err" code
:      3448      : else
:      3449      :     return .RET_STATUS;              !This ret status is good or bad
:      3450      :
:      3451      : end;

```

			.SBTTL	EX.LOC.PROG MODULE DECLARATIONS	
000000	004167	000000G	EX.LOC.PROG::		
000004	005746		JSR	R1,\$SAVE2	3327
000006	004767	174100	TST	(SP)	
000012	004767	174242	JSR	PC,GET.NSD	3370
000016	010002		JSR	PC,GET.CMD\$REF	3371
000020	016746	000000G	MOV	R0,R2	; *,REF.NUM
000024	012746	000054	MOV	NSD.SLOT,-(SP)	3375
000030	004767	000000G	MOV	#54,-(SP)	
000034	012760	000060 000000G	JSR	PC,BL\$MUL	
000042	012701	000002G	MOV	#60,SND.ENVELOPE(R0)	
000046	060001		MOV	#SND.ENVELOPE*2,R1	3376
000050	112711	000001	ADD	R0,R1	
000054	112761	000002 000001	MOVB	#1,(R1)	3377
000062	010260	000004G	MOVB	#2,1(R1)	3378
000066	005060	000006G	MOV	R2,SND.ENVELOPE*4(R0)	; REF.NUM,*
000072	005060	000010G	CLR	SND.ENVELOPE*6(R0)	3382
000076	005060	000012G	CLR	SND.ENVELOPE*10(R0)	3383
000102	112760	000003 000014G	CLR	SND.ENVELOPE*12(R0)	3384
000110	105060	000015G	MOV	#3,SND.ENVELOPE*14(R0)	3385
000114	005060	000016G	CLRB	SND.ENVELOPE*15(R0)	3386
000120	005060	000020G	CLR	SND.ENVELOPE*16(R0)	3387
000124	005060	000022G	CLR	SND.ENVELOPE*20(R0)	3388
000130	005060	000024G	CLR	SND.ENVELOPE*22(R0)	3392
000134	010216		CLR	SND.ENVELOPE*24(R0)	3393
000136	004767	174024	MOV	R2,(SP)	; REF.NUM,*
000142	010001		JSR	PC,LOAD.OUT\$STD.BUF	3403
000144	020127	002001	MOV	R0,R1	; *,ELP.BUF\$LOC
000150	001002		CMP	R1,#2001	; ELP.BUF\$LOC,*
000152	004767	174170	BNE	1\$	3405
000156	016700	000000G	JSR	PC,DECODE	
000162	006300		MOV	NSD.SLOT,R0	3410
000164	006300		ASL	R0	
000166	066700	000000G	ASL	R0	
000172	052760	100000 000002	ADD	SEND.RING,R0	
000200	017766	000000G 000004	BIS	#100000,2(R0)	
000206	016600	000004	MOV	\$RC25.ADDR,4(SP)	; *,RC\$S.REG
000212	012716	005670	MOV	4(SP),R0	; RC\$S.REG,TEMP
000216	010246		MOV	#5670,(SP)	3422
000220	010146		MOV	R2,(SP)	; REF.NUM,*
000222	004767	176130	MOV	R1,-(SP)	; ELP.BUF\$LOC,*
000226	022626		JSR	PC,CTO.WAIT	
000230	006000		CMP	(SP)*,(SP)*	
			ROR	R0	

ZRCH85 ZRCH80 RC25 DISK FORMATTER
REV B PATCH 00 MODULE DECLARATIONS

5 Apr-1984 13:46:24
11-Jan-1984 13:23:12

VAX 11 B119 16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (16)
Page 55

000232	103002		BCC	2\$		
000234	004767	174106	JSR	PC,DECODE		
000240	010100		MOV	R1,R0	; ELP.BUF\$LOC,*	3440
000242	006300		ASL	R0		
000244	006300		ASL	R0		
000246	016067	000002G 000000G	MOV	OUT\$STD.BUF+2(R0),RET.EN\$AD		
000254	016000	000002G	MOV	OUT\$STD.BUF+2(R0),R0	;	3445
000260	005760	000016	TST	16(R0)		
000264	001405		BEG	3\$		
000266	012700	000031	MOV	#31,R0	;	3447
000272	010067	000000G	MOV	R0,RET.STATUS		
000276	000402		BR	4\$;	3329
000300	016700	000000G	MOV	RET.STATUS,R0		
000304	062706	000006	ADD	#6,SP	;	3327
000310	000207		RTS	PC		

; Routine Size: 101 words, Routine Base: AC\$CODE + 4036
; Maximum stack depth per invocation: 9 words

; 3452

```

3453 global routine SEND_DATA -           !Performs host-->port communications
3454
3455 !..
3456 ! Functional Description :
3457 ! These commands are used to communicate between the initiating host
3458 ! program and the remote program. Both send and receive commands
3459 ! specify a host buffer descriptor and a byte count. In the case of
3460 ! send data, the information in the buffer is read by the remote program
3461 ! and a send data response sent back to the host to acknowledge receipt.
3462 ! In the case of receive data the remote program writes data into the
3463 ! buffer up to the amount specified by the byte count and then sends a
3464 ! receive data response to the host to notify it of the transmission.
3465
3466 ! The send data and receive data commands are only legal when the
3467 ! server is in the active state. If the remote program terminates
3468 ! abnormally, putting the server back in the idle state, outstanding
3469 ! send data and receive data commands may be lost. In the event that
3470 ! the specified timeout interval is exceeded, the host program should
3471 ! issue a get dust status command to see if the remote program is
3472 ! still running (is the dup server is active); if it is, the
3473 ! progress indicator should be remembered and the timeout interval
3474 ! should be re-installed. If the second timeout expires without a
3475 ! response and a second get dust status shows the remote program
3476 ! having made no progress in the interim then the program should be
3477 ! considered broken and should be aborted.
3478
3479 ! Formal Parameters :
3480 ! none
3481
3482 ! Implicit Inputs :
3483 ! NSD_SLOT           This global storage gets loaded by the routine
3484 !                   'Get_nsd' and in it is stored the next send ring
3485 !                   descriptor slot where the port/controller should
3486 !                   be polling on and the place to put this commands
3487 !                   command packet.
3488
3489 ! Implicit Outputs :
3490 ! none
3491
3492 ! Completion Codes :
3493 ! RET_STATUS:       Return status passes back to the calling routine
3494 !                   the status of the just issued command.
3495
3496 ! Side Effects :
3497 ! none
3498
3499
3500 begin
3501
3502 local
3503     REF_NUM,           !Stores unique cmd ref number
3504     SND_BUF$LOC,      !Stores outstanding cmd buffer location
3505     TEMP;             !A place to put read IP register data
3506
3507
3508 ! Before we load up the command packet up with all this good information
3509 ! get the next send descriptor slot and a unique command reference number.

```

```

: 3510      !
: 3511      GET_NSD ();                !Get the next send desc slot
: 3512      REF_NUM = GET_CMD$REF (); !Get a unique command ref num
: 3513      !
: 3514      ! UQ Port command envelope Header field definition
: 3515      !
: 3516      SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_SED;      !Load the message size
: 3517      SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE;           !Load the credit size
: 3518      SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0;            !Define the message typ 'Sequential
: 3519      SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP;          !Define the connection ID
: 3520      !
: 3521      ! DUP generic command envelope field definition
: 3522      !
: 3523      SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;      !Load command reference number
: 3524      SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;         !Zero Hi order cmd ref number
: 3525      SND_ENVELOPE [.NSD_SLOT, UN_USED] = ZERO;          !Not used in DUP implimentation
: 3526      SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO;        !Not used in DUP implimentation
: 3527      SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_SED;       !Load this commands op-code
: 3528      SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;            !Not used field
: 3529      SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;        !Define the commands modifiers
: 3530      !
: 3531      ! Command specific command envelope field definition
: 3532      !
: 3533      ! Byte count of transfer
: 3534      !
: 3535      SND_ENVELOPE [.NSD_SLOT, BLO_CNT] = SNDB_SIZE;     !Byte count low word
: 3536      SND_ENVELOPE [.NSD_SLOT, BHI_CNT] = ZERO;         !Byte count high word
: 3537      !
: 3538      ! Buffer descriptor definition
: 3539      !
: 3540      SND_ENVELOPE [.NSD_SLOT, BPA_LO] = SND_BUF;       !Buffer physical adrs <0 15>
: 3541      SND_ENVELOPE [.NSD_SLOT, BPA_HI] = ZERO;         !Buffer physical adrs bits <16 17>
: 3542      SND_ENVELOPE [.NSD_SLOT, QBUS_EXT] = ZERO;        !Q_bus extention adrs
: 3543      SND_ENVELOPE [.NSD_SLOT, RSV] = ZERO;            !Reserved field
: 3544      SND_ENVELOPE [.NSD_SLOT, UBA_CHAN] = ZERO;       !Unibus adaptor channel number
: 3545      SND_ENVELOPE [.NSD_SLOT, RSV0] = ZERO;           !These next four words are not
: 3546      SND_ENVELOPE [.NSD_SLOT, RSV1] = ZERO;           !used in the UQ Port implementation
: 3547      SND_ENVELOPE [.NSD_SLOT, RSV2] = ZERO;          !
: 3548      SND_ENVELOPE [.NSD_SLOT, RSV3] = ZERO;          !
: 3549      !
: 3550      ! Call the load outstanding command buffer routine and load this command
: 3551      ! into the buffer. The return from this routine will point us to the buffer locat'on
: 3552      ! where this command is stored. Later we can look at this location to
: 3553      ! see if the interrupt service routine has received and process 't.
: 3554      !
: 3555      SND_BUF$LOC = LOAD_OUT$STD BUF (.REF_NUM); !Load the command
: 3556      !
: 3557      if .SND_BUF$LOC eqlu OBF CODE then DECODE ();      !Error if buffer is full
: 3558      !
: 3559      !
: 3560      ! Set the ownership bit to 1 giving this slot to the port/controller
: 3561      !
: 3562      SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
: 3563      !
: 3564      ! Read the IP register to stimulate port polling
: 3565      !
: 3566      TEMP = .RC25_ADDR [RCIP, RC_ALL];

```

```

: 3567
: 3568
: 3569 : Time out the port/controller for the response from this command.
: 3570
: 3571 : If the controller time out then:
: 3572 : 1. See what kind of error was returned. If the error
: 3573 : is a type other than a CTO_CODE (controller time out)
: 3574 : then call routine Decode which does the appropriate
: 3575 : action based on the error.
: 3576
: 3577 : 2. If the returned error is an CTO_CODE then do a get
: 3578 : dust status and check the progress indicator to look
: 3579 : for an increase, indicating that the remote program is
: 3580 : still running and is not dead.
: 3581
: 3582 : If the indicator hasn't changed then assume that the
: 3583 : remote program is dead and return an error code of
: 3584 : RPD_CODE (remote program dead code) and exit.
: 3585
: 3586 : If the indicator has changed then assume that the
: 3587 : remote program is still running, save a copy of its
: 3588 : value and reinstate the controller time out delay and
: 3589 : repeat the loop.
: 3590
: 3591 : As long as the progress indicator in the remote program
: 3592 : is still increasing this loop will be repeated for ever.
: 3593
: 3594 : If the controller doesn't time then return with the return
: 3595 : code returned from routine CTO_WAIT () which could be either
: 3596 : a success or error code by definition of this host code.
: 3597
: 3598
: 3599 while TRUE do !Repeat for ever
: 3600 begin
: 3601 BR^AK; !Flag control C s
: 3602
: 3603 : Do a controller time out and determine if the controller
: 3604 : has processed the command or if a fatal error has occurred.
: 3605
: 3606
: 3607 if CTO_WAIT (4000, .REF_NUM, .SND_BUF$LOC) !Is return an error
: 3608 then
: 3609 begin
: 3610
: 3611 : If the return status code equals a CTO_CODE then see if the remote
: 3612 : program is still running. If it is then save the progress indicator
: 3613 : and repeat the loop else call routine Decode ().
: 3614
: 3615
: 3616 if .RET_STATUS eqlu CTO_CODE !Is this a controller time out
: 3617 then
: 3618 begin
: 3619
: 3620 if GET_DUST_STATUS () then DECODE (); !Get the dust status
: 3621
: 3622 if .RET_EN$AD [PLO_IND] gtru .PID SAVE [0] !Any progress been made
: 3623 then

```

ZRCMB5
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
MODULE DECLARATIONS5-Apr-1984 13:46:24
11 Jan-1984 13:23:12VAX-11 Bliss-16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCMB5.816;4 (17)SEQ 0203
Page 59

```

:      3624      begin
:      3625      PID_SAVE [0] = .RET_EN$AD [PLO_IND];      !Still running save Pid
:      3626      PID_SAVE [1] = .RET_EN$AD [PHI_IND];
:      3627      PRINTB (PID_FMT, .PID_SAVE [1], .PID_SAVE [0]);
:      3628      end
:      3629      else
:      3630
:      3631      if .RET_EN$AD [PHI_IND] gtru .PID_SAVE [1]
:      3632      then
:      3633      begin
:      3634      PID_SAVE [0] = .RET_EN$AD [PLO_IND];      !Still running save P'd
:      3635      PID_SAVE [1] = .RET_EN$AD [PHI_IND];
:      3636      PRINTB (PID_FMT, .PID_SAVE [1], .PID_SAVE [0]);
:      3637      end
:      3638      else
:      3639      return RET_STATUS = RPD_CODE;      !No progress so flag error
:      3640
:      3641      end
:      3642      else
:      3643      !
:      3644      ! The return status code was not a controller time out code so something
:      3645      ! else is wrong. Call the routine Decode () to find out what went wrong.
:      3646      !
:      3647      DECODE ()
:      3648
:      3649      end
:      3650      else
:      3651      begin
:      3652      !
:      3653      ! The command has been received by the interrupt service.
:      3654      !
:      3655      ! Get this commands return envelope address out of the
:      3656      ! out$std_buf and check for good return status error and
:      3657      ! die if bad status.
:      3658      !
:      3659      RET_EN$AD = .OUT$STD_BUF [.SND_BUF$LOC, ENV ADR];      !Get the ret env adr
:      3660      !
:      3661      ! Test for good status
:      3662      !
:      3663      !
:      3664      if .RET_EN$AD [STATUS] nequ ZERO      !Test the status
:      3665      then
:      3666      return RET_STATUS = RSE_CODE      !Return a 'Response status err' code
:      3667      else
:      3668      return .RET_STATUS;      !This ret_status is good or bad
:      3669      !
:      3670      end;
:      3671
:      3672      end;
:      3673
:      3674      return .RET STATUS;      !It won't compile without this here
:      3675      end;

```

000000 004167 000000G

```

          .SBTTL SEND.DATA MODULE DECLARATIONS
SEND.DATA::
          JSR      R1,$SAVE3
          ;

```

3453

ZRCHB
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCHB5.B16;4 (17)
Page 60

000004	005746		TST	-(SP)		
000006	004767	173566	JSR	PC.GET.NSD	:	3511
000012	004767	173730	JSR	PC.GET.CMD\$REF	:	3512
000016	010003		MOV	R0,R3	: *,REF.NUM	
000020	016746	000000G	MOV	NSD.SLOT,-(SP)	:	3516
000024	012746	000054	MOV	#54,-(SP)		
000030	004767	000000G	JSR	PC.BL\$MUL		
000034	012760	000034 000000G	MOV	#34,SND.ENVELOPE(R0)		
000042	012701	000002G	MOV	#SND.ENVELOPE+2,R1	:	3517
000046	060001		ADD	R0,R1		
000050	112711	000001	MOV	#1,(R1)	:	3518
000054	112761	000002 000001	MOV	#2.1(R1)	:	3519
000062	010360	000004G	MOV	R3,SND.ENVELOPE+4(R0)	: REF.NUM,*	3523
000066	005060	000006G	CLR	SND.ENVELOPE+6(R0)	:	3524
000072	005060	000010G	CLR	SND.ENVELOPE+10(R0)	:	3525
000076	005060	000012G	CLR	SND.ENVELOPE+12(R0)	:	3526
000102	112760	000004 000014G	MOV	#4,SND.ENVELOPE+14(R0)	:	3527
000110	105060	000015G	CLR	SND.ENVELOPE+15(R0)	:	3528
000114	005060	000016G	CLR	SND.ENVELOPE+16(R0)	:	3529
000120	012760	000045 000020G	MOV	#45,SND.ENVELOPE+20(R0)	:	3535
000126	005060	000022G	CLR	SND.ENVELOPE+22(R0)	:	3536
000132	012760	000000G 000024G	MOV	#SND.BUF,SND.ENVELOPE+24(R0)	:	3540
000140	012701	000026G	MOV	#SND.ENVELOPE+26,R1	:	3541
000144	060001		ADD	R0,R1		
000146	105011		CLRB	(R1)	:	3543
000150	105061	000001	CLRB	1(R1)	:	3544
000154	005060	000030G	CLR	SND.ENVELOPE+30(R0)	:	3545
000160	005060	000032G	CLR	SND.ENVELOPE+32(R0)	:	3546
000164	005060	000034G	CLR	SND.ENVELOPE+34(R0)	:	3547
000170	005060	000036G	CLR	SND.ENVELOPE+36(R0)	:	3548
000174	010316		MOV	R3,(SP)	: REF.NUM,*	3555
000176	004767	173452	JSR	PC.LOAD.OUT\$STD.BUF		
000202	010002		MOV	R0,R2	: *,SND.BUF\$LOC	
000204	020227	002001	CMR	R2,#2001	: SND.BUF\$LOC,*	3557
000210	001002		BNE	1\$		
000212	004767	173616	JSR	PC.DECODE		
000216	016700	000000G	MOV	NSD.SLOT,R0	:	3562
000222	006300		ASL	R0		
000224	006300		ASL	R0		
000226	066700	000000G	ADD	SEND.RING,R0		
000232	052760	100000 000002	BIS	#100000,2(R0)		
000240	017766	000000G 000004	MOV	#RC25.ADDR,4(SP)	: *,RC\$S.REG	3566
000246	016600	000004	MOV	4(SP),R0	: RC\$S.REG,TEMP	
000252	104422		TRAP	22	:	3600
000254	012716	007640	MOV	#7640,(SP)	:	3607
000260	010346		MOV	R3,(SP)	: REF.NUM,*	
000262	010246		MOV	R2,-(SP)	: SND.BUF\$LOC,*	
000264	004767	175554	JSR	PC.CTO.WAIT		
000270	022626		CMR	(SP)+,(SP)+		
000272	006000		ROR	R0		
000274	103074		BCC	8\$		
000276	026727	000000G 000011	CMR	RET.STATUS,#11	:	3616
000304	001065		BNE	7\$		
000306	004767	176240	JSR	PC.GET.DUST.STATUS	:	3620
000312	006000		ROR	R0		
000314	103002		BCC	3\$		
000316	004767	173512	JSR	PC.DECODE		

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5 Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0205
Page 61
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (17)

000322	016701	000000G		3\$:	MOV	RET.EN\$AD,R1	:	3626
000326	010100				MOV	R1,R0	; RET.EN\$AD,*	3622
000330	016000	000024			MOV	24(R0),R0		
000334	020067	000000G			CMP	R0,PID.SAVE		
000340	101417				BLOS	4\$		
000342	010067	000000G			MOV	R0,PID.SAVE	:	3625
000346	016167	000026	000002G		MOV	26(R1),PID.SAVE+2	:	3626
000354	010016				MOV	R0,(SP)	; PID.SAVE,*	3627
000356	016746	000002G			MOV	PID.SAVE+2,-(SP)		
000362	012746	000000G			MOV	#PID.FMT,-(SP)		
000366	012746	000003			MOV	#3,-(SP)		
000372	010600				MOV	SP,R0	; SP,*	
000374	104414				TRAP	14		
000376	000422				BR	5\$:	3624
000400	026167	000026	000002G	4\$:	CMP	26(R1),PID.SAVE+2	:	3631
000406	101421				BLOS	6\$		
000410	010067	000000G			MOV	R0,PID.SAVE	:	3634
000414	016167	000026	000002G		MOV	26(R1),PID.SAVE+2	:	3635
000422	010016				MOV	R0,(SP)	; PID.SAVE,*	3636
000424	016746	000002G			MOV	PID.SAVE+2,-(SP)		
000430	012746	000000G			MOV	#PID.FMT,-(SP)		
000434	012746	000003			MOV	#3,-(SP)		
000440	010600				MOV	SP,R0	; SP,*	
000442	104414				TRAP	14		
000444	062706	000006		5\$:	ADD	#6,SP	:	3633
000450	000700				BR	2\$:	3631
000452	012700	000051		6\$:	MOV	#51,R0	:	3639
000456	000420				BR	9\$		
000460	004767	173350		7\$:	JSR	PC,DECODE	:	3647
000464	000672				BR	2\$:	3607
000466	010200			8\$:	MOV	R2,R0	; SND.BUF\$LOC,*	3659
000470	006300				ASL	R0		
000472	006300				ASL	R0		
000474	016067	000002G	000000G		MOV	OUT\$STD.BUF+2(R0),RET.EN\$AD		
000502	016000	000002G			MOV	OUT\$STD.BUF+2(R0),R0	:	3664
000506	005760	000016			TST	16(R0)		
000512	001405				BEQ	10\$		
000514	012700	000031			MOV	#31,R0	:	3666
000520	010067	000000G		9\$:	MOV	R0,RET.STATUS		
000524	000402				BR	11\$:	3651
000526	016700	000000G		10\$:	MOV	RET.STATUS,R0		
000532	062706	000006		11\$:	ADD	#6,SP	:	3453
000536	000207				RTS	PC		

; Routine Size: 176 words, Routine Base: AC\$CODE + 4350
; Maximum stack depth per invocation: 12 words

; 3676

```

: 3677 global routine REC_DATA : !Performs host-->port communications
: 3678
: 3679 !..
: 3680 ! Functional Description :
: 3681 ! These commands are used to communicate between the initiating host
: 3682 ! program and the remote program. Both send and receive commands
: 3683 ! specify a host buffer descriptor and a byte count. In the case of
: 3684 ! send data, the information in the buffer is read by the remote program
: 3685 ! and a send data response sent back to the host to acknowledge receipt.
: 3686 ! In the case of receive data the remote program writes data into the
: 3687 ! buffer up to the amount specified by the byte count and then sends a
: 3688 ! receive data response to the host to notify it of the transmission.
: 3689
: 3690 ! The send data and receive data commands are only legal when the
: 3691 ! server is in the active state. If the remote program terminates
: 3692 ! abnormally, putting the server back in the idle state, outstanding
: 3693 ! send data and receive data commands may be lost. In the event that
: 3694 ! the specified timeout interval is exceeded, the host program should
: 3695 ! issue a get dust status command to see if the remote program is
: 3696 ! still running (is. the dup server is active); if it is, the
: 3697 ! progress indicator should be remembered and the timeout interval
: 3698 ! should be re-installed. If the second timeout expires without a
: 3699 ! response and a second get dust status shows the remote program
: 3700 ! having made no progress in the interim then the program should be
: 3701 ! considered broken and should be aborted.
: 3702
: 3703 ! Formal Parameters :
: 3704 ! none
: 3705
: 3706 ! Implicit Inputs :
: 3707 ! NSD_SLOT This global storage gets loaded by the routine
: 3708 ! 'Get_nsd' and in it is stored the next send ring
: 3709 ! descriptor slot where the port/controller should
: 3710 ! be polling on and the place to put this commands
: 3711 ! command packet.
: 3712
: 3713 ! Implicit Outputs :
: 3714 ! none
: 3715
: 3716 ! Completion Codes :
: 3717 ! RET_STATUS: Return status passes back to the calling routine
: 3718 ! the status of the just issued command.
: 3719
: 3720 ! Side Effects :
: 3721 ! none
: 3722 !..
: 3723
: 3724 begin
: 3725
: 3726 local
: 3727 REF_NUM, !Stores unique cmd ref number
: 3728 REC_BUF$LOC, !Stores outstanding cmd buffer locat'on
: 3729 TEMP; !A place to put read IP register data
: 3730
: 3731 !
: 3732 ! Before we load up the command packet up with all this good information
: 3733 ! get the next send descriptor slot and a unique command reference number.

```



```

: 3734      !
: 3735      GET_NSD ();                !Get the next send desc slot
: 3736      REF_NUM = GET_CMD$REF (); !Get a unique command ref num
: 3737      !
: 3738      ! UQ Port command envelope Header field definition
: 3739      !
: 3740      SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_RED;      !Load message length
: 3741      SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE;           !Load credit size
: 3742      SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0;           !Define message type 'Sequential'
: 3743      SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP;          !Define connection ID
: 3744      !
: 3745      ! DUP generic command envelope field definition
: 3746      !
: 3747      SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;      !Load command reference number
: 3748      SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;         !Zero Hi order cmd ref num
: 3749      SND_ENVELOPE [.NSD_SLOT, UN_LUSED] = ZERO;        !Not used in DUP implimentation
: 3750      SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO;        !Not used in DUP implimentation
: 3751      SND_ENVELOPE [.NSD_SLOT, OPCODE] = OP_RED;       !Load this commands op-code
: 3752      SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;           !Not used field
: 3753      SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;        !Define this commands modifie s
: 3754      !
: 3755      ! Command specfic command envelope field definition
: 3756      !
: 3757      ! Byte count of transfer
: 3758      !
: 3759      SND_ENVELOPE [.NSD_SLOT, BLO_CNT] = RECB_SIZE;     !Byte count low word
: 3760      SND_ENVELOPE [.NSD_SLOT, BHI_CNT] = ZERO;         !Byte count high word
: 3761      !
: 3762      ! Buffer descriptor definition
: 3763      !
: 3764      SND_ENVELOPE [.NSD_SLOT, BPA_LO] = REC_BUF;        !Low unibus adrs <0 15>
: 3765      SND_ENVELOPE [.NSD_SLOT, BPA_HI] = ZERO;          !Unibus adrs bits <16-17>
: 3766      SND_ENVELOPE [.NSD_SLOT, QBUS_EXT] = ZERO;        !Q_bus extention adrs
: 3767      SND_ENVELOPE [.NSD_SLOT, RSV] = ZERO;             !Reserved field
: 3768      SND_ENVELOPE [.NSD_SLOT, UBA_CHAN] = ZERO;        !Unibus adaptor channel number
: 3769      SND_ENVELOPE [.NSD_SLOT, RSV0] = ZERO;           !These next four words are no'
: 3770      SND_ENVELOPE [.NSD_SLOT, RSV1] = ZERO;           !used in the UQ Port implementation
: 3771      SND_ENVELOPE [.NSD_SLOT, RSV2] = ZERO;           !
: 3772      SND_ENVELOPE [.NSD_SLOT, RSV3] = ZERO;           !
: 3773      !
: 3774      ! Call the load out$standing command buffer routine and load this command
: 3775      ! into the buffer. The return from this routine will point us to the
: 3776      ! buffer location where this command is stored. Later we can look at this
: 3777      ! location to see if the interrupt service routine has received and process
: 3778      ! it.
: 3779      !
: 3780      REC_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
: 3781      !
: 3782      if .REC_BUF$LOC eqv OBF CODE then DECODE ();      !Error if buffer is full
: 3783      !
: 3784      !
: 3785      ! Set the ownership bit to 1 giving this slot to the port/controller.
: 3786      !
: 3787      SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
: 3788      !
: 3789      ! Read the IP register to stimulate port polling
: 3790      !

```

```

:      3791      TEMP = .RC25_ADDR [RCIP, RC ALL];
:      3792
:      3793      :
:      3794      : Time out the port/controller for the response from this command.
:      3795
:      3796      : If the controller times out then:
:      3797      :   1. See what kind of error was returned. If the error
:      3798      :      is a type other than a CTO_CODE (controller time out)
:      3799      :      then call routine Decode which does the appropriate
:      3800      :      action based on the error.
:      3801
:      3802      :   2. If the returned error is an CTO_CODE then do a get
:      3803      :      dust status and check the progress indicator to look
:      3804      :      for an increase, indicating that the remote program is
:      3805      :      still running and is not dead.
:      3806
:      3807      :      If the indicator hasn't changed then assume that the
:      3808      :      remote program is dead and return an error code of
:      3809      :      RPD_CODE (remote program dead code) and exit.
:      3810
:      3811      :      If the indicator has changed then assume that the
:      3812      :      remote program is still running, save a copy of its
:      3813      :      value and reinstate the controller time out delay and
:      3814      :      repeat the loop.
:      3815
:      3816      :      As long as the progress indicator in the remote program
:      3817      :      is still increasing this loop will be repeated for ever.
:      3818
:      3819      : If the controller doesn't time then return with the return code
:      3820      : returned from routine CTO_WAIT () which could be either a success
:      3821      : or error code by definition of this host code.
:      3822
:      3823
:      3824      while TRUE do                                !Repeat for ever
:      3825      begin
:      3826      BREAK;                                       !Flag control C's
:      3827
:      3828      : Do a controller time out and determine if the controller
:      3829      : has processed the command or if a fatal error has occurred.
:      3830
:      3831
:      3832      if CTO_WAIT (8000, .REF NUM, .REC BUF$(OC))    !Is return an error
:      3833      then
:      3834      begin
:      3835
:      3836      : If the return status code eq's a CTO_CODE then see if the remote
:      3837      : program is still running. If it is then save the progress indicator
:      3838      : and repeat the loop else call routine Decode ().
:      3839
:      3840
:      3841      if .RET STATUS eq'u CTO_CODE                    !Is this a controller time out
:      3842      then
:      3843      begin
:      3844
:      3845      if GET DUST STATUS () then DECODE ();          !Get the dust status
:      3846
:      3847      if .RET EN$AD [PLO IND] gtru .PID SAVE [0]    !Any progress been made

```

```

:      3848           then
:      3849           begin
:      3850           PID_SAVE [0] = .RET_EN$AD [PLO_IND];           !Still running save Pid
:      3851           PID_SAVE [1] = .RET_EN$AD [PHI_IND];
:      3852           PRINTB (PID_FMT, .PID_SAVE [1], .PID_SAVE [0]);
:      3853           end
:      3854           else
:      3855
:      3856           if .RET_EN$AD [PHI_IND] gtru .PID_SAVE [1]
:      3857           then
:      3858           begin
:      3859           PID_SAVE [0] = .RET_EN$AD [PLO_IND];           !Still running save Pid
:      3860           PID_SAVE [1] = .RET_EN$AD [PHI_IND];
:      3861           PRINTB (PID_FMT, .PID_SAVE [1], .PID_SAVE [0]);
:      3862           end
:      3863           else
:      3864           return RET_STATUS = RPD CODE;           !No progress so flag error
:      3865
:      3866           end
:      3867           else
:      3868           !
:      3869           ! The return status code was not a controller time out code so something
:      3870           ! else 's wrong. Call the routine Decode () to find out what went wrong.
:      3871           !
:      3872           DECODE ()
:      3873
:      3874           end
:      3875           else
:      3876           begin
:      3877           !
:      3878           ! The command has been received by the interrupt service.
:      3879           !
:      3880           ! Get this commands return envelope address out of the
:      3881           ! out$std buf and check for good return status error and
:      3882           ! die if bad status.
:      3883           !
:      3884           RET_EN$AD = .OUT$STD_BUF [.REC_BUF$LOC, ENV_ADR];           !Get the ret env adr
:      3885           !
:      3886           ! Test for good status
:      3887           !
:      3888           !
:      3889           if .RET_EN$AD [STATUS] nequ ZERO           !Test the status
:      3890           then
:      3891           return RET_STATUS = RSE CODE           !Return a "Response status err" code
:      3892           else
:      3893           return .RET_STATUS;           !This ret status is good or bad
:      3894
:      3895           end;
:      3896
:      3897           end;
:      3898
:      3899           return .RET_STATUS;           !It won t compile without this here
:      3900           end;

```

000004	005746			JSR	R1,\$SAVE3			3677
000006	004767	173026		TST	.(SP)			
000012	004767	173170		JSR	PC,GET.NSD			3735
000016	01003			JSR	PC,GET.CMD\$REF			3736
000020	016746	000000G		MOV	R0,R3		*,REF.NUM	
000024	012746	000054		MOV	NSD.SLOT,.(SP)			3740
000030	004767	000000G		MOV	#54,.(SP)			
000034	012760	000034	000000G	JSR	PC,BL\$MUL			
000042	012701	000002G		MOV	#34,SND.ENVELOPE(R0)			
000046	060001			MOV	#SND.ENVELOPE+2,R1			3741
000050	112711	000001		ADD	R0,R1			
000054	112761	000002	000001	MOVB	#1,(R1)			3742
000062	010360	000004G		MCVB	#2,1(R1)			3743
000066	005060	000006G		MOV	R3,SND.ENVELOPE+4(R0)		REF.NUM,*	3747
000072	005060	000010G		CLR	SND.ENVELOPE+6(R0)			3748
000076	005060	000012G		CLR	SND.ENVELOPE+10(R0)			3749
000102	112760	000005	000014G	CLR	SND.ENVELOPE+12(R0)			3750
000110	105060	000015G		MOVB	#5,SND.ENVELOPE+14(R0)			3751
000114	005060	000016G		CLRB	SND.ENVELOPE+15(R0)			3752
000120	012760	000170	000020G	CLR	SND.ENVELOPE+16(R0)			3753
000126	005060	000022G		MOV	#170,SND.ENVELOPE+20(R0)			3759
000132	012760	000000G	000024G	CLR	SND.ENVELOPE+22(R0)			3760
000140	012701	000026G		MOV	#REC.BUF,SND.ENVELOPE+24(R0)			3764
000144	060001			MOV	#SND.ENVELOPE+26,R1			3765
000146	105011			ADD	R0,R1			
000150	105061	000001		CLRB	(R1)			3767
000154	005060	000030G		CLRB	1(R1)			3768
000160	005060	000032G		CLR	SND.ENVELOPE+30(R0)			3769
000164	005060	000034G		CLR	SND.ENVELOPE+32(R0)			3770
000170	005060	000036G		CLR	SND.ENVELOPE+34(R0)			3771
000174	010316			CLR	SND.ENVELOPE+36(R0)			3772
000176	004767	172712		MOV	R3,(SP)		REF.NUM,*	3780
000202	010002			JSR	PC,LOAD.OUT\$STD.BUF			
000204	020227	002001		MOV	R0,R2		*,REC.BUF\$LOC	
000210	001002			CMF	R2,#2001		REC.BUF\$LOC,*	3782
000212	004767	173056		BNE	1\$			
000216	016700	000000G	1\$:	JSR	PC,DECODE			
000222	006300			MOV	NSD.SLOT,R0			3787
000224	006300			ASL	R0			
000226	066700	000000G		ASL	R0			
000232	052760	100000	000002	ADD	SEND.RING,R0			
000240	017766	000000G	000004	BIS	#100000,2(R0)			
000246	016600	000004		MOV	#RC25.ADDR,4(SP)		*,RC\$S.REG	3791
000252	104422			MOV	4(SP),R0		RC\$S.REG,TEMP	
000254	012716	017500	2\$:	TRAP	2\$			3825
000260	010346			MOV	#17500,(SP)			3832
000262	010246			MOV	R3,(SP)		REF.NUM,*	
000264	004767	175014		MOV	R2,(SP)		REC.BUF\$LOC,*	
000270	022626			JSR	PC,CTO.WAIT			
000272	006000			CMF	(SP),.(SP),.			
000274	103074			ROR	R0			
000276	026727	000000G	000011	BCC	8\$			
000304	001065			CMF	RET.STATUS,#11			3841
000306	004767	175500		BNE	7\$			
000312	006000			JSR	PC,GET.DUST.STATUS			3845
000314	103002			ROR	R0			
				BCC	3\$			

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5 Apr-1984 13:46:24
11 Jan 1984 13:23:12

VAX-11 Bliss 16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCH85.816;4 (18)

000316	004767	172752			JSR	PC,DECODE		
000322	016701	000000G	3\$:		MOV	RET.EN\$AD,R1	:	3851
000326	010100				MOV	R1,R0	; RET.EN\$AD,*	3847
000330	016000	000024			MOV	24(R0),R0		
000334	020067	000000G			CMP	R0,PID.SAVE		
000340	101417				BLOS	4\$		
000342	010067	000000G			MOV	R0,PID.SAVE	:	3850
000346	016167	000026	000002G		MOV	26(R1),PID.SAVE+2	:	3851
000354	010016				MOV	R0,(SP)	; PID.SAVE,*	3852
000356	016746	000002G			MOV	PID.SAVE+2,-(SP)		
000362	012746	000000G			MOV	#PID.FMT,(SP)		
000366	012746	000003			MOV	#3,(SP)		
000372	010600				MOV	SP,R0	; SP,*	
000374	104414				TRAP	14		
000376	000422				BR	5\$:	3849
000400	026167	000026	000002G	4\$:	CMP	26(R1),PID.SAVE+2	:	3856
000406	101421				BLOS	6\$		
000410	010067	000000G			MOV	R0,PID.SAVE	:	3859
000414	016167	000026	000002G		MOV	26(R1),PID.SAVE+2	:	3860
000422	010016				MOV	R0,(SP)	; PID.SAVE,*	3861
000424	016746	000002G			MOV	PID.SAVE+2,(SP)		
000430	012746	000000G			MOV	#PID.FMT,-(SP)		
000434	012746	000003			MOV	#3,-(SP)		
000440	010600				MOV	SP,R0	; SP,*	
000442	104414				TRAP	14		
000444	062706	000006		5\$:	ADD	#6,SP	:	3858
000450	000700				BR	2\$:	3856
000452	012700	000051		6\$:	MOV	#51,R0	:	3864
000456	000420				BR	9\$		
000460	004767	172610		7\$:	JSR	PC,DECODE	:	3872
000464	000672				BR	2\$:	3832
000466	010200			8\$:	MOV	R2,R0	; REC.BUF\$LOC,*	3884
000470	006300				ASL	R0		
000472	006300				ASL	R0		
000474	016067	000002G	000000G		MOV	OUT\$STD.BUF+2(R0),RET.EN\$AD		
000502	016000	000002G			MOV	OUT\$STD.BUF+2(R0),R0	:	3889
000506	005760	000016			TST	16(R0)		
000512	001405				BEQ	10\$		
000514	012700	000031			MOV	#31,R0	:	3891
000520	010067	000000G		9\$:	MOV	R0,RET.STATUS	:	3876
000524	000402				BR	11\$:	3876
000526	016700	000000G		10\$:	MOV	RET.STATUS,R0		
000532	062706	000006		11\$:	ADD	#6,SP	:	3677
000536	000207				RTS	PC		

; Routine Size: 176 words, Routine Base: AC\$CODE + 5110
; Maximum stack depth per invocation: 12 words

; 3901

```

3902 global routine SET_CNTRLR CHAR -           !Sets control characteristics
3903
3904 !..
3905 ! Functional Description :
3906 !   The SET CONTROLLER CHARACTERISTICS command is used to set host
3907 !   settable unit characteristics and obtain those unit
3908 !   characteristics that are essential for proper class driver
3909 !   operation. This command never alters the unit's state
3910 !   ("unit online", "unit-available", "unit-offline"). It is
3911 !   meaningless to set host settable characteristics for a unit
3912 !   that is "unit available" or "unit offline".
3913
3914 ! Formal Parameters :
3915 !   none
3916
3917 ! Implicit Inputs :
3918 !   NSD_SLOT           This global storage gets loaded by the routine
3919 !                       'Get_nsd' and in it is stored the next send ring
3920 !                       descriptor slot where the port/controller should
3921 !                       be polling on and the place to put this commands
3922 !                       command packet.
3923
3924 ! Implicit Outputs :
3925 !   none
3926
3927 ! Completion Codes :
3928 !   RET_STATUS:       Return status passes back to the calling routine
3929 !                       the status of the just issued command.
3930
3931 ! Side Effects :
3932 !   Any previously defined controller characteristics will possibly
3933 !   be altered after execution of this command.
3934 !..
3935
3936 begin
3937
3938 local
3939   REF_NUM,           !Stores unique cmd ref number
3940   SCC_BUF$LOC,      !Stores outstanding cmd buffer location
3941   TEMP;             !A place to put read IP register data
3942
3943
3944 ! Before we load up the command packet up with all this good information
3945 ! get the next send descriptor slot and a unique command reference number.
3946
3947 GET_NSD ();          !Get the next send desc slot
3948 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
3949
3950 ! UQ Port command envelope Header field definition
3951
3952 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ SCC;      !Load message length
3953 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE;           !Load credit size
3954 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0;           !Define message type 'Sequential'
3955 SND_ENVELOPE [.NSD_SLOT, CONN ID] = MSCP;         !Define connection ID
3956
3957 ! MSCP generic command envelope field definition
3958

```

```

3959      !
3960      SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;      !Load command reference number
3961      SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;      !Zero Hi order cmd ref num
3962      SND_ENVELOPE [.NSD_SLOT, UN_LUSED] = ZERO;      !Not used in DUP implimentation
3963      SND_ENVELOPE [.NSD_SLOT, UN_MUSED] = ZERO;      !Not used in DUP implimentation
3964      SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_SCC;      !Load this commands op code
3965      SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;      !Not used field
3966      SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;      !Define this commands modifiers
3967      !
3968      ! Command specific command envelope field definition
3969      !
3970      SND_ENVELOPE [.NSD_SLOT, MSCP_VER] = ZERO;      !MSCP version
3971      SND_ENVELOPE [.NSD_SLOT, CTL_FLAGS] = ZERO;      !Controller flags
3972      SND_ENVELOPE [.NSD_SLOT, HOST_TOV] = ZERO;      !Host time out value
3973      SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;      !Reserved
3974      SND_ENVELOPE [.NSD_SLOT, T&D_0] = ZERO;      !Time and Date word 0
3975      SND_ENVELOPE [.NSD_SLOT, T&D_1] = ZERO;      !Time and Date word 1
3976      SND_ENVELOPE [.NSD_SLOT, T&D_2] = ZERO;      !Time and Date word 2
3977      SND_ENVELOPE [.NSD_SLOT, T&D_3] = ZERO;      !Time and Date word 3
3978      SND_ENVELOPE [.NSD_SLOT, CDP_LO] = ZERO;      !Cntlr dep parameter lo word
3979      SND_ENVELOPE [.NSD_SLOT, CDP_HI] = ZERO;      !Cntlr dep parameter hi wrd
3980      !
3981      ! Call the load out$standing command buffer routine and load this command
3982      ! into the buffer. The return from this routine will point us to the
3983      ! buffer location where this command is stored. Later we can look at this
3984      ! location to see if the interrupt service routine has received and process it.
3985      !
3986      SCC_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM);      !Load the command
3987      !
3988      if .SCC_BUF$LOC equl OBF_CODE then DECODE ();      !Error if buffer is full
3989      !
3990      !
3991      ! Set the ownership bit to 1 giving this slot to the port/controller
3992      !
3993      SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
3994      !
3995      ! Read the IP register to stimulate port polling
3996      !
3997      TEMP = .RC25_ADDR [RCIP, RC_ALL];
3998      !
3999      ! Time out the port/controller processing the command.
4000      !
4001      ! The first test tests the connections ability to
4002      ! respond to this command without any errors in the SA
4003      ! register and for the command not timing out.
4004      !
4005      ! The second tests the DUP server for good status. If
4006      ! bad status is sent back then an error code is returned
4007      ! to the calling routine where the routine "decode" will
4008      ! decode and take the appropriate recovery. The time
4009      ! out routine will loop on delaying and checking the hi
4010      ! bit of the first word in the out$std buf for a true.
4011      ! When true signals us that the interrupt service routine
4012      ! has received the endpacket and no connection errors
4013      ! were detected.
4014      !
4015      !

```

```

:      4016      if CTD WAIT (3000, .REF_NUM, .SCC_BUF$LOC) then DECODE (); !Is return an error
:      4017
:      4018      :
:      4019      : Get the return envelope address from the out$std buf
:      4020      : at this commands buffer location and check the packet
:      4021      : for good status error and die if bad status was returned
:      4022      :
:      4023      RET_EN$AD = .OUT$STD BUF [.SCC_BUF$LOC, ENV ADR]; !Get the ret env adr
:      4024      :
:      4025      : Now test for good status
:      4026      :
:      4027      :
:      4028      if .RET_EN$AD [STATUS] nequ ZERO          !Test the status
:      4029      then
:      4030          return RET_STATUS = RSE CODE          !Return a "Response status err code
:      4031      else
:      4032          return .RET STATUS;                    !This ret status is good or bad
:      4033
:      4034      end;

```

```

000000 004167 000000G      .SBTTL SET.CNTRLR.CHAR MODULE DECLARATIONS
                                SET.CNTRLR.CHAR::
000004 005746      JSR      R1,$SAVE2          ;          3902
000006 004767 172266      TST      -(SP)          ;
000012 004767 172430      JSR      PC,GET.NSD      ;          3948
000016 010002      JSR      PC,GET.CMD$REF  ;          3949
000020 016746 000000G      MOV      R0,R2          ; +,REF.NUM
000024 012746 000054      MOV      NSD.SLOT, -(SP) ;          3953
000030 004767 000000G      MOV      #54, -(SP)
000034 012760 000040 000000G JSR      PC,BL$MUL
000042 012701 000002G      MOV      #40,SND.ENVELOPE(R0)
000046 060001      MOV      #SND.ENVELOPE+2,R1 ;          3954
000050 112711 000001      ADD      R0,R1
000054 105061 000001      MOV      #1,(R1)        ;          3955
000060 010260 000004G      CLRB     1(R1)          ;          3956
000064 005060 000006G      MOV      R2,SND.ENVELOPE+4(R0) ; REF.NUM,+ 3960
000070 005060 000010G      CLR      SND.ENVELOPE+6(R0) ;          3961
000074 005060 000012G      CLR      SND.ENVELOPE+10(R0) ;          3962
000100 112760 000004 000014G CLR      SND.ENVELOPE+12(R0) ;          3963
000106 105060 000015G      MOV      #4,SND.ENVELOPE+14(R0) ;          3964
000112 005060 000016G      CLRB     SND.ENVELOPE+15(R0) ;          3965
000116 005060 000020G      CLR      SND.ENVELOPE+16(R0) ;          3966
000122 005060 000022G      CLR      SND.ENVELOPE+20(R0) ;          3970
000126 005060 000024G      CLR      SND.ENVELOPE+22(R0) ;          3971
000132 005060 000026G      CLR      SND.ENVELOPE+24(R0) ;          3972
000136 005060 000030G      CLR      SND.ENVELOPE+26(R0) ;          3973
000142 005060 000032G      CLR      SND.ENVELOPE+30(R0) ;          3974
000146 005060 000034G      CLR      SND.ENVELOPE+32(R0) ;          3975
000152 005060 000036G      CLR      SND.ENVELOPE+34(R0) ;          3976
000156 005060 000040G      CLR      SND.ENVELOPE+36(R0) ;          3977
000162 005060 000042G      CLR      SND.ENVELOPE+40(R0) ;          3978
000166 010216      CLR      SND.ENVELOPE+42(R0) ;          3979
000170 004767 172160      MOV      R2,(SP)        ; REF.NUM,+ 3986
000174 010001      JSR      PC,LOAD.OUT$STD.BUF
000176 020127 002001      MOV      R0,R1          ; +,SCC.BUF$LOC
                                CMP      R1,#2001      ; SCC.BUF$LOC,+ 3988

```


ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0215
Page 71
VAX-11 Bliss 16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4(19)

000202	001002			BNE	1\$		
000204	004767	172324		JSR	PC,DECODE		
000210	016700	000000G	1\$:	MOV	NSD.SLOT,RO	:	3993
000214	006300			ASL	RO		
000216	006300			ASL	RO		
000220	066700	000000G		ADD	SEND.RING,RO		
000224	052760	100000	000002	BIS	#100000,2(RO)		
000232	017766	000000G	000004	MOV	RC25.ADDR,4(SP)	: *,RC\$S.REG	3997
000240	016600	000004		MOV	4(SP),RO	: RC\$S.REG,TEMP	
000244	012716	005670		MOV	#5670,(SP)	:	4016
000250	010246			MOV	R2,-(SP)	: REF.NUM,*	
000252	010146			MOV	R1,-(SP)	: SCC.BUF\$LOC,*	
000254	004767	174264		JSR	PC,CTD.WAIT		
000260	022626			CMP	(SP)*,(SP)*		
000262	006000			ROR	RO		
000264	103002			BCC	2\$		
000266	004767	172242		JSR	PC,DECODE		
000272	010100		2\$:	MOV	R1,RO	: SCC.BUF\$LOC,*	4023
000274	006300			ASL	RO		
000276	006300			ASL	RO		
000300	016067	000002G	000000G	MOV	OUT\$STD.BUF+2(RO),RET.EN\$AD		
000306	016000	000002G		MOV	OUT\$STD.BUF+2(RO),RO	:	4028
000312	005760	000016		TST	16(RO)		
000316	001405			BEQ	3\$		
000320	012700	000031		MOV	#31,RO	:	4030
000324	010067	000000G		MOV	RO,RET.STATUS	:	
000330	000402			BR	4\$:	3937
000332	016700	000000G	3\$:	MOV	RET.STATUS,RO	:	
000336	062706	000006	4\$:	ADD	#6,SP	:	3902
000342	000207			RTS	PC		

: Routine Size: 114 words, Routine Base: AC\$CODE + 5650
: Maximum stack depth per invocation: 9 words

: 4035

```

:      4036 global routine ON LINE =                !Makes a unit come online to a host
:      4037
:      4038 : **
:      4039 : Functional Description :
:      4040 :     The online command is used to bring a unit "unit-online, set
:      4041 :     host settable unit characteristics and obtain those unit
:      4042 :     characteristics that are essential for proper class driver
:      4043 :     operation. The unit is spun-up, if necessary, and is heads
:      4044 :     are loaded prior to returning the online command's end
:      4045 :     message. Host settable characteristics are set exactly as if
:      4046 :     a set unit characteristics command were issued. Host settable
:      4047 :     characteristics are set after the unit has been successfully spun up
:      4048 :     and any other validity checks have succeeded. Note that the unit's
:      4049 :     host settable characteristics are not altered if the unit is already
:      4050 :     "unit-online".
:      4051 :
:      4052 : Formal Parameters :
:      4053 :     none
:      4054 :
:      4055 : Implicit Inputs :
:      4056 :     NSD_SLOT      This global storage gets loaded by the routine
:      4057 :                   'Get_nsd' and in it is stored the next send ring
:      4058 :                   descriptor slot where the port/controller should
:      4059 :                   be polling on and the place to put this commands
:      4060 :                   command packet.
:      4061 :
:      4062 : Implicit Outputs :
:      4063 :     none
:      4064 :
:      4065 : Completion Codes :
:      4066 :     RET_STATUS:   Return status passes back to the calling routine
:      4067 :                   the status of the just issued command.
:      4068 :
:      4069 : Side Effects :
:      4070 :     Any previously defined controller characteristics will possibly
:      4071 :     be altered after execution of this command.
:      4072 :
:      4073 :
:      4074 : begin
:      4075 :
:      4076 : local
:      4077 :     REF_NUM,                !Stores unique cmd ref number
:      4078 :     ONL_BUF$LOC,           !Stores outstanding cmd buffer locat'on
:      4079 :     TEMP;                  !A place to put read IP register data
:      4080 :
:      4081 :
:      4082 : ! Before we load up the command packet up with all this good information
:      4083 : ! get the next send descriptor slot and a unique command reference number.
:      4084 :
:      4085 : GET_NSD ();                !Get the next send desc slot
:      4086 : REF_NUM = GET_CMD$REF ();  !Get a unique command ref num
:      4087 :
:      4088 : ! UQ Port command envelope Header field definition
:      4089 :
:      4090 : SND ENVELOPE [.NSD_SLOT, MSG LENGTH] = SZ ONL;    !Load message length
:      4091 : SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE;         !Load cred't size
:      4092 : SND ENVELOPE [.NSD_SLOT, MSG TYPE] = 0;          !Define message type Sequential

```

```

:      4093      SNO_ENVELOPE [.NSD_SLOT, CONN_ID] = MSCP;      !Define connection ID
:      4094      :
:      4095      : MSCP generic command envelope field definition
:      4096      :
:      4097      SNO_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;      !Load command reference number
:      4098      SNO_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;      !Zero Hi order cmd ref num
:      4099      SNO_ENVELOPE [.NSD_SLOT, UNIT_NUM] = .UNIT_NO;      !Select unit to bring online
:      4100      SNO_ENVELOPE [.NSD_SLOT, UN_MUSED] = ZERO;      !Not used in DUP implimentation
:      4101      SNO_ENVELOPE [.NSD_SLOT, OPCODE] = OP_ONL;      !Load this commands op code
:      4102      SNO_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;      !Not used field
:      4103      SNO_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;      !Define this commands modifiers
:      4104      :
:      4105      : Command specific command envelope field definition
:      4106      :
:      4107      SNO_ENVELOPE [.NSD_SLOT, RSV$D] = ZERO;      !Reserved
:      4108      SNO_ENVELOPE [.NSD_SLOT, UNT_FLAGS] = ZERO;      !Unit flag field
:      4109      SNO_ENVELOPE [.NSD_SLOT, RSV$0] = ZERO;      !Reserved field
:      4110      SNO_ENVELOPE [.NSD_SLOT, RSV$1] = ZERO;      !Reserved field
:      4111      SNO_ENVELOPE [.NSD_SLOT, RSV$2] = ZERO;      !Reserved field
:      4112      SNO_ENVELOPE [.NSD_SLOT, RSV$3] = ZERO;      !Reserved field
:      4113      SNO_ENVELOPE [.NSD_SLOT, RSV$4] = ZERO;      !Reserved field
:      4114      SNO_ENVELOPE [.NSD_SLOT, RSV$5] = ZERO;      !Reserved field
:      4115      SNO_ENVELOPE [.NSD_SLOT, DDP_LO] = ZERO;      !Device dependent parameter
:      4116      SNO_ENVELOPE [.NSD_SLOT, DDP_HI] = ZERO;      !Device dependent parameter
:      4117      SNO_ENVELOPE [.NSD_SLOT, SHADOW_UNIT] = ZERO;      !Shadow unit
:      4118      SNO_ENVELOPE [.NSD_SLOT, COPY_SPEED] = ZERO;      !Copy speed
:      4119      :
:      4120      : Call the load outstanding command buffer routine
:      4121      : and load this command into the buffer. The return
:      4122      : from this routine will point us to the buffer location
:      4123      : where this command is stored. Later we can look at
:      4124      : this location to see if the interrupt service routine
:      4125      : has received and process it.
:      4126      :
:      4127      ONL_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM);      !Load the command
:      4128      :
:      4129      if .ONL_BUF$LOC eqlu OBF_CODE then DECODE ();      !Error if buffer is full
:      4130      :
:      4131      :
:      4132      : Set the ownership bit to 1 giving this slot to the port/controller
:      4133      :
:      4134      SEND_RING [.NSD_SLOT, OWN_BIT] = PORT OWNED;
:      4135      :
:      4136      : Read the IP register to stimulate port polling
:      4137      :
:      4138      TEMP = .RC25_ADDR [RCIP, RC ALL];
:      4139      :
:      4140      : Time out the port/controller processing the command.
:      4141      :
:      4142      : The first test tests the connections ability to
:      4143      : respond to this command without any errors in the SA
:      4144      : register and for the command not timing out.
:      4145      :
:      4146      : The second tests the DUP server for good status. If
:      4147      : bad status is sent back then an error code is returned
:      4148      : to the calling routine where the routine 'decode' will
:      4149      : decode and take the appropriate recovery. The time

```

ZRCHB5
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11 Jan-1984 13:23:12

SEQ 0218
Page 74
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCHB5.B16;4 (20)

```

:      4150      ! out routine will loop on delaying and checking the hi
:      4151      ! bit of the first word in the out$std_buf for a true.
:      4152      ! When true signals us that the interrupt service routine
:      4153      ! has received the endpacket and no connection errors
:      4154      ! were detected.
:      4155      !
:      4156      !
:      4157      if CTO WAIT (ONE_MINUTE, .REF_NUM, .ONL_BUF$LOC) then DECODE ();      !Is return an error
:      4158      !
:      4159      !
:      4160      ! Get the return envelope address from the out$std_buf
:      4161      ! at this commands buffer location and check the packet
:      4162      ! for good status error and die if bad status was returned
:      4163      !
:      4164      RET_EN$AD = .OUT$STD_BUF [.ONL_BUF$LOC, ENV_ADR];      !Get the ret env adr
:      4165      !
:      4166      ! Now test for good status
:      4167      !
:      4168      !
:      4169      if .RET_EN$AD [STATUS] nequ ZERO      !Test the status
:      4170      then
:      4171      return RET_STATUS = RSE CODE      !Return a "Response status err" code
:      4172      else
:      4173      return .RET_STATUS;      !This ret_status is good or bad
:      4174      !
:      4175      end;

```

			.SBTTL	ON.LINE MODULE DECLARATIONS	
000000	004167	000000G	ON.LINE::		
			JSR	R1,\$SAVE2	4036
000004	005746		TST	-(SP)	
000006	004767	171722	JSR	PC,GET.NSD	4085
000012	004767	172064	JSR	PC,GET.CMD\$REF	4086
000016	010002		MOV	R0,R2	*,REF.NUM
000020	016746	000000G	MOV	NSD.SLOT,-(SP)	4090
000024	012746	000054	MOV	#54,-(SP)	
000030	004767	000000G	JSR	PC,BL\$MUL	
000034	012760	000044 000000G	MOV	#44,SND.ENVELOPE(R0)	
000042	012701	000002G	MOV	#SND.ENVELOPE+2,R1	4091
000046	060001		ADD	R0,R1	
000050	112711	000001	MOVB	#1,(R1)	4092
000054	105061	000001	CLRB	1(R1)	4093
000060	010260	000004G	MOV	R2,SND.ENVELOPE+4(R0)	REF.NUM,*
000064	005060	000006G	CLR	SND.ENVELOPE+6(R0)	4098
000070	016760	000000G 000010G	MOV	UNIT.NO,SND.ENVELOPE+10(R0)	4099
000076	005060	000012G	CLR	SND.ENVELOPE+12(R0)	4100
000102	112760	000011 000014G	MOVB	#11,SND.ENVELOPE+14(R0)	4101
000110	105060	000015G	CLRB	SND.ENVELOPE+15(R0)	4102
000114	005060	000016G	CLR	SND.ENVELOPE+16(R0)	4103
000120	005060	000020G	CLR	SND.ENVELOPE+20(R0)	4107
000124	005060	000022G	CLR	SND.ENVELOPE+22(R0)	4108
000130	005060	000024G	CLR	SND.ENVELOPE+24(R0)	4109
000134	005060	000026G	CLR	SND.ENVELOPE+26(R0)	4110
000140	005060	000030G	CLR	SND.ENVELOPE+30(R0)	4111
000144	005060	000032G	CLR	SND.ENVELOPE+32(R0)	4112
000150	005060	000034G	CLR	SND.ENVELOPE+34(R0)	4113

M1

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0219
Page 75
VAX-11 Bliss-16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (20)

000154	005060	000036G		CLR	SND.ENVELOPE+36(R0)	:	4114
000160	005060	000040G		CLR	SND.ENVELOPE+40(R0)	:	4115
000164	005060	000042G		CLR	SND.ENVELOPE+42(R0)	:	4116
000170	005060	000044G		CLR	SND.ENVELOPE+44(R0)	:	4117
000174	005060	000046G		CLR	SND.ENVELOPE+46(R0)	:	4118
000200	010216			MOV	R2,(SP)	: REF.NUM,*	4127
000202	004767	171602		JSR	PC,LOAD.OUT\$STD.BUF		
000206	010001			MOV	R0,R1	: *,ONL.BUF\$LOC	
000210	020127	002001		CMP	R1,#2001	: ONL.BUF\$LOC,*	4129
000214	001002			BNE	1\$		
000216	004767	171746		JSR	PC,DECODE		
000222	016700	000000G	1\$:	MOV	NSD.SLOT,R0	:	4134
000226	006300			ASL	R0		
000230	006300			ASL	R0		
000232	066700	000000G		ADD	SEND.RING,R0		
000236	052760	100000	000002	BIS	#100000,2(R0)		
000244	017766	000000G	000004	MOV	@RC25.ADDR,4(SP)	: *,RC\$S.REG	4138
000252	016600	000004		MOV	4(SP),R0	: RC\$S.REG,TEMP	
000256	012716	165140		MOV	#-12640,(SP)	:	4157
000262	010246			MOV	R2,-(SP)	: REF.NUM,*	
000264	010146			MOV	R1,-(SP)	: ONL.BUF\$LOC,*	
000266	004767	173706		JSR	PC,CTO.WAIT		
000272	022626			CMP	(SP)+,(SP)+		
000274	006000			ROR	R0		
000276	103002			BCC	2\$		
000300	004767	171664		JSR	PC,DECODE		
000304	010100		2\$:	MOV	R1,R0	: ONL.BUF\$LOC,*	4154
000306	006300			ASL	R0		
000310	006300			ASL	R0		
000312	016067	000002G	000000G	MOV	OUT\$STD.BUF+2(R0),RET.EN\$AD		
000320	016000	000002G		MOV	OUT\$STD.BUF+2(R0),R0	:	4169
000324	005760	000016		TST	16(R0)		
000330	001405			BEQ	3\$		
000332	012700	000031		MOV	#31,R0	:	4171
000336	010067	000000G		MOV	R0,RET.STATUS	:	
000342	000402			BR	\$:	4074
000344	016700	000000G	3\$:	MOV	RET.STATUS,R0	:	
000350	062706	000006	4\$:	ADD	#6,SP	:	4036
000354	000207			RTS	PC		

: Routine Size: 119 words, Routine Base: AC\$CODE + 6214
: Maximum stack depth per invocation: 9 words

: 4176

```

: 4177 global routine INT$I_SERVICE : INT_LNK$TYP novalue = !Init sequence interrupt catcher
: 4178
: 4179 !..
: 4180 ! Functional Description :
: 4181 ! During the initialization sequence the IE bit is defined to be
: 4182 ! a zero. This means that the host is not requesting interrupts at
: 4183 ! the completion of steps 1 3.
: 4184 !
: 4185 ! Note that no interrupt will be generated at the completion of
: 4186 ! step 4 since this step requires only a small number of time.
: 4187 !
: 4188 ! This interrupt service routine serves to catch any interrupts that the
: 4189 ! controller might issue during the initialization sequence. The
: 4190 ! interrupt is ignored and control is returned.
: 4191 !
: 4192 ! This interrupt service routine is also used during the RC25 register
: 4193 ! existence test in determining whether P Table RC25 registers exist.
: 4194 !
: 4195 ! Formal Parameters :
: 4196 ! none
: 4197 !
: 4198 ! Implicit Inputs :
: 4199 ! NEX_FLAG A flag which is loaded with zeros during the RC25 register
: 4200 ! existence test and set to all ones by this routine in the
: 4201 ! event of attempts to read a non-existent RC25 controller.
: 4202 !
: 4203 ! Implicit Outputs :
: 4204 ! NEX_FLAG Is returned with all ones in the event of RC25 non-existent
: 4205 ! register access attempts.
: 4206 !
: 4207 ! Completion Codes :
: 4208 ! none
: 4209 !
: 4210 ! Side Effects :
: 4211 ! none
: 4212 !
: 4213 !
: 4214 ! begin
: 4215 ! NEX_FLAG = ONES; !Indicate this interrupt occurred
: 4216 ! return;
: 4217 ! end;

```

```

000000 012767 177777 000000G .SBT'L INT$I_SERVICE MODULE DECLARATIONS
000006 000002 INT$I_SERVICE:: MOV #1,NEX_FLAG ;
RTI ;

```

```

: Routine Size: 4 words, Routine Base: AC$CODE + 657
: Maximum stack depth per invocation: 0 words

```

```

: 4218

```

```

:      4219 global routine IS_TIMER (SEQ_NO) *          !Init sequence time out
:      4220
:      4221 !**
:      4222 ! Functional Description :
:      4223 !     Steps 1-3 of the init sequence, each are required to complete within
:      4224 !     10 seconds.  If any of these steps fails to complete within that period,
:      4225 !     this is to be treated as a host detected fatal error.
:      4226 !
:      4227 !     This routine will do one us delays for a total of 10 seconds.  After
:      4228 !     each delay the step field is examined to see if th s init sequence has completed.
:      4229 !
:      4230 ! Formal Parameters :
:      4231 !     SEQ_NO: Indicated which init step is presently being performed within the RC25 init sequence.
:      4232 !
:      4233 ! Implicit Inputs :
:      4234 !     none
:      4235 !
:      4236 ! Implicit Outputs :
:      4237 !     none
:      4238 !
:      4239 ! Completion Codes :
:      4240 !     TRUE:  Indicates to the calling routine that the indicated init sequence step has timed out.
:      4241 !
:      4242 !     FALSE: Indicates to the calling routine that the indicated init sequence
:      4243 !     step has not timed out.
:      4244 !
:      4245 ! Side Effects :
:      4246 !     If the init sequence step times out and an error is posted in the sa register
:      4247 !     then the routine decode will be call.
:      4248 !
:      4249 !
:      4250 begin
:      4251
:      4252 local
:      4253     TO_VALUE : word,          !Step time out value
:      4254     STEP_VAL : word;        !Temp storage of step value
:      4255
:      4256 STEP_VAL = ZERO;          !Make sure the loc is zeroed out
:      4257 !
:      4258 ! Select the step value expected from this init sequence step.
:      4259 !
:      4260
:      4261 selectoneu .SEQ_NO of      !Select the binary step value
:      4262     set
:      4263
:      4264     [0] :
:      4265         begin
:      4266             STEP_VAL = %b 0001 ;          !Step 1 binary value
:      4267             TO_VALUE = 20000;          !Timeout step one for 60 seconds
:      4268             end;
:      4269
:      4270     [1] :
:      4271         begin
:      4272             STEP_VAL = %b 0010 ;          !Step 2 binary value
:      4273             TO_VALUE = 5000;          !Timeout step two for 10 seconds
:      4274             end;
:      4275

```

C?

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5 Apr 1984 13:46:24
11 Jan 1984 13:23:12

VAX 11 Bliss 16 V3 555
SPIDER#USERS:(MCALE,AZTEC)ZRCH85.816;4 (22)

SEQ 0222
Page 78

```

:      4276      [2] :
:      4277      begin
:      4278      STEP_VAL = #b 0100 ;           !Step 3 binary value
:      4279      TO_VALUE = 5000;           !Timeout step 3 for 10 seconds
:      4280      end;
:      4281
:      4282      [3] :
:      4283      begin
:      4284      STEP_VAL = #b'1000';         !Step 4 binary value
:      4285      TO_VALUE = 5000;         !Timeout step 4 for 10 seconds
:      4286      end;
:      4287      tes;
:      4288
:      4289      !
:      4290      ! Loop on the 100 micro second delay until either the expected step field
:      4291      ! is read in the SA register or the step times out.
:      4292      !
:      4293
:      4294      incru TIM_OUT from 0 to .TO_VALUE do      !Loop on C US delay s
:      4295      begin
:      4296      DELAY (C_US);                          !Do the delay
:      4297      !
:      4298      ! Check the step bit to see if it is set yet. If it is set then return
:      4299      ! a false indicating the completion else continue delaying.
:      4300      !
:      4301
:      4302      if .RC25_ADDR [RCSA, STP_FIELD] eqv .STEP_VAL then return FALSE;
:      4303
:      4304      BREAK;                                    !Service any control C s
:      4305      end;
:      4306
:      4307      !
:      4308      ! This step has not completed within the specified time interval. Test
:      4309      ! the sa register for any errors posted and report errors if any. Return
:      4310      ! a true to the caller indicating the error.
:      4311      !
:      4312
:      4313      if .RC25_ADDR [RCSA, ERR_BIT]
:      4314      then
:      4315      begin
:      4316      RET_STATUS = PFE_CODE;                !Indicate the port/fatal error code
:      4317      DECODE ();                          !Report the error
:      4318      end;
:      4319
:      4320      return TRUE;                            !Return a failure to the caller
:      4321      end;

```

000000	004167	000000G	.SBTTL	IS.TIMER	IS.TIMER MODULE DECLARATIONS	
			IS.TIMER::			
			JSR	R1,\$SAVE4		4219
000004	162706	000006	SUB	#6,SP		
000010	005002		CLR	R2	: STEP.VAL	4256
000012	016600	000022	MOV	22(SP),R0	: SEQ.NO.*	4261
000016	001005		BNE	1\$		
000020	012702	000001	MOV	#1,R2	: *.STEP.VAL	4266
000024	012703	047040	MOV	#47040,R3	: *.TO.VALUE	4267

ZRCHB5
REV B PATCH 00

ZRCHB0 RC25 DISK FORMATTER
MODULE DECLARATIONS

5 Apr 1984 13:46:24
11 Jan-1984 13:23:12

VAX-11 Bliss 16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCHB5.B16;4 (22)

SEQ 0223
Page 79

000030	000423			BR	5:				4261
000032	020027	000001	1:	CMP		R0,#1			
000036	001003			BNE		2:			
000040	012702	000002		MOV		#2,R2		; *,STEP.VAL	4272
000044	000413			BR		4:			4273
000046	020027	000002	2:	CMP		R0,#2			4261
000052	001003			BNE		3:			
000054	012702	000004		MOV		#4,R2		; *,STEP.VAL	4278
000060	000405			BR		4:			4279
000062	020027	000003	3:	CMP		R0,#3			4261
000066	001004			BNE		5:			
000070	012702	000010		MOV		#10,R2		; *,STEP.VAL	4284
000074	012703	011610	4:	MOV		#11610,R3		; *,TO.VALUE	4285
000100	005004		5:	CLR		R4		; TIM.OUT	4294
000102	000436			BR		11:			
000104	012701	000001	6:	MOV		#1,R1		; *,\$\$TMP2	4296
000110	001411		7:	BEQ		10:			
000112	016700	000000G		MOV		L\$DLY,R0		; *,\$\$TMP1	
000116	001404			BEQ		9:			
000120	005066	000004	8:	CLR		4(SP)		; \$\$TMP	
000124	005300			DEC		R0		; \$\$TMP1	
000126	001374			BNE		8:			
000130	005301		9:	DEC		R1		; \$\$TMP2	
000132	000766			BR		7:			
000134	016700	000000G	10:	MOV		RC25.ADDR,R0			4302
000140	016066	000002 000002		MOV		2(R0),2(SP)		; *,RC\$S.REG	
000146	010201			MOV		R2,R1		; STEP.VAL,*	
000150	016600	000002		MOV		2(SP),R0		; RC\$S.REG,*	
000154	006200			ASR		R0			
000156	006200			ASR		R0			
000160	006200			ASR		R0			
000162	000300			SWAB		R0			
000164	042700	177760		BIC		#177760,R0			
000170	020001			CMP		R0,R1			
000172	001421			BEQ		13:			
000174	104422			TRAP		22			
000176	005204			INC		R4		; TIM.OUT	4294
000200	020403		11:	CMP		R4,R3		; TIM.OUT,TO.VALUE	
000202	101740			BLOS		6:			
000204	016700	000000G		MOV		RC25.ADDR,R0			4313
000210	016016	000002		MOV		2(R0),(SP)		; *,RC\$S.REG	
000214	100005			BPL		12:			
000216	012767	000021 000000G		MOV		#21,RET.STATUS			4316
000224	004767	171352		JSR		PC,DECODE			4317
000230	012700	000001	12:	MOV		#1,R0			4250
000234	000401			BR		14:			
000236	005000		13:	CLR		R0			4219
000240	062706	000006	14:	ADD		#6,SP			
000244	000207			RTS		PC			

; Routine Size: 83 words, Routine Base: AC\$CODE + 6602
; Maximum stack depth per invocation: 10 words

; 4322

```

:      4323 global routine BOOT_RC25 =                               !Performs RC25 init sequence
:      4324
:      4325 !..
:      4326 ! Functional Description :
:      4327 ! This routine performs the initialization sequence of the RC25
:      4328 ! RC25 controller.
:      4329
:      4330 ! The initialization procedure serves to:
:      4331
:      4332 ! 1. Identify the parameters of the host-resident communications
:      4333 ! region to the port.
:      4334
:      4335 ! 2. Provide a confidence check of port/controller integrity.
:      4336
:      4337 ! 3. Bring the port/controller online to the host (note that the
:      4338 ! devices attached to the controller are not thereby brought
:      4339 ! online to the class driver.)
:      4340
:      4341 ! Formal Parameters :
:      4342 ! none
:      4343
:      4344 ! Implicit Inputs :
:      4345 ! ISD_STRUCT Stores the init sequence read and write data defined
:      4346 ! for this program and controller.
:      4347
:      4348 ! Implicit Outputs :
:      4349 ! none
:      4350
:      4351 ! Completion Codes :
:      4352 ! Success: Is returned to the calling routine if this initialization
:      4353 ! sequence was executed successfully.
:      4354
:      4355 ! Failure: Is returned to the calling routine if this initialization
:      4356 ! sequence was not executed successfully.
:      4357
:      4358 ! Side Effects :
:      4359 ! Any DM code that might have been running in the DM machine will be
:      4360 ! aborted.
:      4361
:      4362 ! Any outstanding commands or response pertaining to a process using
:      4363 ! the controller will be lost.
:      4364 !
:      4365
:      4366 begin
:      4367
:      4368 local
:      4369 TEMP : word;                               !Temporary storage location
:      4370
:      4371 !..
:      4372 ! The host begins the initialization sequence
:      4373 ! either by issuing a bus init or by writing
:      4374 ! any value into the IP register; the port must
:      4375 ! guarantee that the host will read zeros in SA
:      4376 ! on the next bus cycle. Initialization then
:      4377 ! sequences through steps 1 4 as per UQSSP.DOC
:      4378 ! Version 1.5.
:      4379 !

```

```

:      4380      ! Write to the IP register and start the init
:      4381      ! sequence going.
:      4382      !-
:      4383
:      4384      WRT RC25 (RCIP, ONES);                !Begin init sequence
:      4385
:      4386      !-
:      4387      ! This incr loop performs all four steps of the
:      4388      ! initialization sequence described above. The
:      4389      ! SA write and read data is preset into the
:      4390      ! structure   ISD_STRUCT   and stands for
:      4391      ! "Initialization Sequence Data_STRUCT".
:      4392      !-
:      4393
:      4394      ! If a step time out error occurs the test
:      4395      ! invoking this routine will take the necessary
:      4396      ! retry procedure. A return code of failure is
:      4397      ! returned.
:      4398
:      4399      ! If any SA register compare error is detected after
:      4400      ! a step completion the routine Decode will decode the
:      4401      ! error and load statistical tables up pertaining data.
:      4402      !-
:      4403      !-
:      4404
:      4405      incru SEQ_NO from STEP1 to STEP4 do      !Do the four init seq steps
:      4406      begin
:      4407      !
:      4408      ! Wait for the controller to load the SA reg up with the step data.
:      4409      !
:      4410
:      4411      if IS TIMER (.SEQ_NO)                        !Did the Controller time out
:      4412      then
:      4413      begin
:      4414      !
:      4415      ! DO SOME STAT TABLE UP DATA TO SHOW THE TIME OUT
:      4416      !
:      4417      PRINTB (.EMSG_STRUCT [MSG10]);
:      4418      return FAILURE;                            !Notify DRS> init of the failure
:      4419      end;
:      4420
:      4421      !
:      4422      ! The controller did not time out so read the SA register
:      4423      ! for the expected step data and compare it to the good
:      4424      ! data stored in ISD_STRUCT.
:      4425      !
:      4426      ! If the read SA data is not what we expect then return a
:      4427      ! failure code.
:      4428      !
:      4429      ! Note that the reserved fields read in the SA register are
:      4430      ! or'ed with all ones to mask out the field before compared
:      4431      ! to the expected data stored in the structure "ISD_STRUCT".
:      4432      !
:      4433      TEMP = ((.RC25_ADDR [RCSA, RC_ALL]) or (.RSVD_STRUCT [.SEQ_NO]));
:      4434      !
:      4435      if .TEMP nequ .ISD_STRUCT [.SEQ_NO, ISRD, ISR ALL]      !Compare read to expected
:      4436      then

```

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0226
Page 82
VAX-11 Bliss 16 V3 555
SPIDER\$USERS:[NFALE.AZTEC]ZRCH85.816;4(23)

```

:      4437      begin
:      4438      |
:      4439      | Load some satistical table up with some data to indicate that the
:      4440      | init sequence had some trouble.
:      4441      |
:      4442      PRINTB (.EMSG_STRUCT [MSG11]);
:      4443      return FAILURE;          !Return a failure code
:      4444      end;
:      4445
:      4446      |
:      4447      | If this is step four then print this u-code version number.
:      4448      |
:      4449      |
:      4450      if .SEQ_NO equl STEP4 then PRINTB (FMT4, .RC25_ADDR [RCSA, S4R VER]);
:      4451
:      4452      |
:      4453      | This step read data is what we expected so write the SA register
:      4454      | with th's steps write data stored in ISD_STRUCT.
:      4455      |
:      4456      WRT_RC25 (RCSA, .ISD_STRUCT [.SEQ_NO, ISWRT, ISW_ALL]);
:      4457      end;
:      4458
:      4459      |
:      4460      | The controller initialization sequence was done successfully so return a success code.
:      4461      |
:      4462      return SUCCESS;
:      4463      end;

```

```

000000 004167 000000G      .SBITL  BOOT.RC25 MODULE DECLARATIONS
                                BOOT.RC25::
000004 024646      JSR      R1,$SAVE3          ;          4323
000006 012700 177777      CMP      -(SP), (SP)
000012 010077 000000G      MOV      @-1,R0          ; *,RC$M.REG  4384
000016 005002      MOV      R0,@RC25.ADDR ; RC$M.REG,*
000020 010246      CLR      R2          ; SEQ.NO  4405
000022 004767 177504 1$:    MOV      R2,-(SP)      ; SEQ.NO,*  4411
000026 005726      JSR      PC,IS.TIMER
000030 006000      TST      (SP)*
000032 103007      ROR      R0
000034 016746 000024G      BCC      2$
000040 012746 000001      MOV      EMSG_STRUCT+24,-(SP) ;          4417
000044 010600      MOV      @1,-(SP)
000046 104414      MOV      SP,R0          ; SP,*
000050 000427      TRAP    14
000052 016700 000000G      BR      3$          ;          4411
000056 016066 000002 000002 2$:  MOV      RC25.ADDR,R0 ;          4433
000064 010200      MOV      2(R0),2(SP)
000066 006300      MOV      R2,R0          ; *,RC$S.REG
000070 016603 000002      ASL      R0          ; SEQ.NO,*
000074 056003 000000G      MOV      2(SP),R3      ; RC$S.REG,TEMP
000100 010201      BIS      RSVD_STRJCT(R0),R3 ; *,TEMP
000102 006301      MOV      R2,R1          ; SEQ.NO,*  4435
000104 006301      ASL      R1
000106 020361 000000G      ASL      R1
000112 001410      CMP      R3,ISD_STRUCT(R1) ; TEMP,*
                                BEQ      4$

```

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11-Jan-1984 13:23:12

SEQ 0227
Page 83
VAX-11 B1199 16 V3 555
SPIDER#USERS:(NEALE.AZTEC)ZRCH85.B16;4 (23)

000114	016746	000026G		MOV	EMSG.STRUCT+26,-(SP)	:		4442
000120	012746	000001		MOV	#1,-(SP)	:		
000124	010600			MOV	SP,R0	:	SP,*	
000126	104414			TRAP	14	:		
000130	022626		3\$:	CMP	(SP)+,(SP)+	:		4435
000132	000437			BR	6\$:		4437
000134	020227	000003	4\$:	CMP	R2,#3	:	SEQ.NO,*	4450
000140	001017			BNE	5\$:		
000142	016700	000000G		MOV	RC25.ADDR,R0	:		
000146	016016	000002		MOV	2(R0),(SP)	:	*,RC\$S.REG	
000152	011646			MOV	(SP),-(SP)	:	RC\$S.REG,*	
000154	042716	177760		BIC	#177760,(SP)	:		
000160	012746	000000G		MOV	#FMT4,(SP)	:		
000164	012746	000002		MOV	#2,-(SP)	:		
000170	010600			MOV	SP,R0	:	SP,*	
000172	104414			TRAP	14	:		
000174	062706	000006		ADD	#6,SP	:		
000200	016101	000002G	5\$:	MOV	ISD.STRUCT+2(R1),R1	:	*,RC\$M.REG	4456
000204	016700	000000G		MOV	RC25.ADDR,R0	:		
000210	010160	000002		MOV	R1,2(R0)	:	RC\$M.REG,*	
000214	005202			INC	R2	:	SEQ.NO	4405
000216	020227	000003		CMP	R2,#3	:	SEQ.NO,*	
000222	101676			BLOS	1\$:		
000224	012700	000001		MOV	#1,R0	:		4366
000230	000401			BR	7\$:		
000232	005000		6\$:	CLR	R0	:		4323
000234	022626		7\$:	CMP	(SP)+,(SP)+	:		
000236	000207			RTS	PC	:		

: Routine Size: 80 words, Routine Base: AC\$CODE + 7050
: Maximum stack depth per invocation: 11 words

: 4464

```

: 4465 global routine INIT COM AREA = !Inits DUP Protocol communication area
: 4466
: 4467
: 4468 : Functional Description :
: 4469 : After initialization step 3 the port controller clears out
: 4470 : the communication area's ring buffers.
: 4471
: 4472 : This routine first makes sure that this protocol is accom
: 4473 : plished by the port before proceeding.
: 4474
: 4475 : If the port did its part of the protocol then the communic
: 4476 : ations area is initialized as follows:
: 4477
: 4478 : 1. Defines from the contiguous data storage structure
: 4479 : "COM_AREA" the header area address, receive ring
: 4480 : address and the send ring address (these structures
: 4481 : are initially declared as reference structures and
: 4482 : require an address to be defined as its value per
: 4483 : BLISS language conventions).
: 4484
: 4485 : 2. Clears the interrupt indicators and adaptor purge
: 4486 : (ring base -1, -2, -3, -4) defined as "HEAD_AREA".
: 4487
: 4488 : 3. Loads the receive and send descriptors with the values:
: 4489 : a. Envelope low, high and Q_bus address
: 4490 : b. Reserved field
: 4491 : c. Flag bit
: 4492 : d. Ownership bit
: 4493
: 4494 : 4. Load the receive envelope message length field with the
: 4495 : buffer size in bytes.
: 4496
: 4497 : 5. Initialize the Outstanding command buffer to reflect
: 4498 : that all slots are unused.
: 4499
: 4500 : Formal Parameters :
: 4501 : none
: 4502
: 4503 : Implicit Inputs :
: 4504 : HEAD_AREA, RECEIVE_RING, SEND_RING, COM_AREA
: 4505
: 4506 : Implicit Outputs :
: 4507 : The communication area as a result of this routine will be initialized
: 4508 : for host program to remote program communications per DUP and
: 4509 : UQSSP specifications.
: 4510
: 4511 : Completion Codes :
: 4512 : TRUE: Error code to indicate the port controller has
: 4513 : not fulfilled its part of the DUP protocol.
: 4514
: 4515 : FALSE: An error code to indicate the port controller
: 4516 : has fulfilled its part of the DUP protocol.
: 4517
: 4518 : Side Effects :
: 4519 : none
: 4520
: 4521

```

ZRCMB5
REV B PATCH 00ZRCMB0 RC25 DISK FORMATTER
MODULE DECLARATIONS5-Apr-1984 13:46:24
11 Jan-1984 13:23:12VAX 11 Bliss 16 V3 555
SPIDER\$USERS:(NEALE.AZTEC)ZRCMB5.B16;4 (24)SEQ 0229
Page 85

```

:      4522      begin
:      4523
:      4524      !*
:      4525      ! Make sure that the controller has done its part of
:      4526      ! the DUP protocol by clearing out the ring buffers.
:      4527      ! If the rings are not cleared out then return with
:      4528      ! an error code of true.
:      4529      !
:      4530
:      4531      incru i from 2 to RING_SIZE - 1 do          !Test all blocks for zeros
:      4532
:      4533          incru j from WRD0 to WRD1 do          !Test all words for zeros
:      4534              !
:      4535              ! Test this word for zeros.  If not zeros then exit
:      4536              ! this routine with an "communication area init"
:      4537              ! error code to indicate the Protocol violation.
:      4538              !
:      4539
:      4540              if .COM_AREA [.i, .j, WORD_REF] nequ ZERO then return CIE CODE;
:      4541
:      4542      !*
:      4543      ! The port did its part of the protocol so now
:      4544      ! define the address locations of the HEAD AREA,
:      4545      ! RECEIVE_RING and SEND_RING from the contiguous
:      4546      ! storage declared by COM_AREA.
:      4547      !-
:      4548
:      4549      HEAD_AREA = COM_AREA;                          !Define the Header area
:      4550      RECEIVE_RING = COM_AREA [REC_BASE];        !Define the receive ring area
:      4551      SEND_RING = COM AREA [SND_BASE];            !Define the send ring area
:      4552
:      4553      !*
:      4554      ! Not quite sure if the port has to clear out
:      4555      ! the header area of the communications area
:      4556      ! so I'll clear it out here just in case.
:      4557      !-
:      4558
:      4559      incru i from WRD0 to WRD3 do
:      4560          HEAD_AREA [.i, WORD_REF] = ZEROS;
:      4561
:      4562      !*
:      4563      ! Load up the Send Ring descriptors with an envelope address.
:      4564      ! define the "Flag bit" to = 1 (interrupt requested), define
:      4565      ! the "Ownership bit" to = 0 (owned by host) and load the
:      4566      ! Reserved field with zeros (per DUP spec).
:      4567      !
:      4568
:      4569      incru i from 0 to SND_ALLOCATE - 1 do
:      4570          begin
:      4571              SEND_RING [.i, LO_EN$AD] = SND_ENVELOPE [.i, CMD_LREF]; !Low-order envelope adress for all sys
:      4572              SEND_RING [.i, HI_EN$AD] = ZERO;          !High order portion of an 18 bit U/Q bus adrs
:      4573              SEND_RING [.i, QB_EXT] = ZERO;          !Q_bus extention
:      4574              SEND_RING [.i, D$RSVD] = ZERO;          !Reserved field
:      4575              SEND_RING [.i, FLAG_BIT] = SET_FLG;      !Flag bit whose meaning varies depending on dsc state
:      4576              SEND_RING [.i, OWN_BIT] = HOST OWNED;  !Indicates whether dsc is host or port owned
:      4577          end;
:      4578

```

```

: 4579 :
: 4580 : Load up the Receive Ring descriptors with an envelope
: 4581 : address, define the "Ownership bit" = 1 (owned by port),
: 4582 : define the "Flag bit" to = 1 (Interrupts requested) and
: 4583 : the reserved field set to zeros (per DUP spec).
: 4584 :
: 4585 :
: 4586 :   incru i from 0 to REC_ALLOCATE - 1 do
: 4587 :     begin
: 4588 :       RECEIVE_RING [., LO_EN$AD] = REC_ENVELOPE [., CMD_LREF];
: 4589 :       RECEIVE_RING [., HI_EN$AD] = ZEROS;
: 4590 :       RECEIVE_RING [., QB_EXT] = ZEROS;
: 4591 :       RECEIVE_RING [., D$RSVD] = ZEROS;
: 4592 :       RECEIVE_RING [., FLAG_BIT] = SET_LG;
: 4593 :       RECEIVE_RING [., OWN_BIT] = PORT_OWNE;
: 4594 :     end;
: 4595 :
: 4596 :
: 4597 :   ! Reset the communications area pointer to their initial state.
: 4598 :
: 4599 :   NRD_SLOT = -1;           !Start ring pointer at zero
: 4600 :   NSD_SLOT = -1;         !Start ring pointer at zero
: 4601 :   NXT_CRN = ZERO;       !Start unique cmd ref num at one
: 4602 :
: 4603 :
: 4604 :   ! Set the response envelope message length size equal
: 4605 :   ! to the buffer size in bytes starting at text + 0.
: 4606 :
: 4607 :
: 4608 :   incru i from 0 to REC_ALLOCATE - 1 do
: 4609 :     REC_ENVELOPE [., MSG_LENGTH] = RB_SIZE*2;   !Convert to bytes before loading
: 4610 :
: 4611 :
: 4612 :   ! Init the outstanding command buffer as follows:
: 4613 :   ! 1. Indicate that all slots are unused by loading
: 4614 :   !    the unique value #0'100000'.
: 4615 :   ! 2. Clear the envelope adrs words to zero.
: 4616 :   !-
: 4617 :
: 4618 :   incru i from 0 to REC_ALLOCATE - 1 do
: 4619 :     begin
: 4620 :       OUT$STD_BUF [., CMD_WRD] = #0'100000'; !Define the slot as unused
: 4621 :       OUT$STD_BUF [., ENV_ADR] = ZERO;      !Clear out the envelope adrs field
: 4622 :     end;
: 4623 :
: 4624 :
: 4625 :   ! No errors detected by this routine so return with an error code of false.
: 4626 :
: 4627 :   return PAS_CODE;
: 4628 : end;

```

000000	004167	000000G	.SBTIL	INIT.COM.AREA MODULE DECLARATIONS		
			INIT.COM.AREA::			
000004	012701	000004	JSR	R1,\$SAVE3	:	4465
000010	005002		MOV	#4,R1	: ,I	4531
			1\$: CLR	R2	: J	4533

L2

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5 Apr 1984 13:46:24
11 Jan 1984 13:23:12

SEQ 0231
Page 87
VAX-11 Bliss 16 V3 555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4(24)

000012	010100		2\$:	MOV	R1,R0	:	I,*	4540
000014	060200			ADD	R2,R0	:	J,*	
000016	006300			ASL	R0			
000020	005760	000000G		TST	COM.AREA(R0)			
000024	001403			BEQ	3\$			
000026	012700	000001		MOV	#1,R0			
000032	000207			RTS	PC			
000034	005202		3\$:	INC	R2	:	J	4533
000036	020227	000001		CMP	R2,#1	:	J,*	
000042	101763			BLOS	2\$			
000044	062701	000002		ADD	#2,R1	:	*,I	4531
000050	020127	000022		CMP	R1,#22	:	I,*	
000054	101755			BLOS	1\$			
000056	012767	000000G 000000G		MOV	#COM.AREA,HEAD.AREA	:		4549
000064	012767	000010G 000000G		MOV	#COM.AREA+10,RECEIVE.RING	:		4550
000072	012767	000030G 000000G		MOV	#COM.AREA+30,SEND.RING	:		4551
000100	005000			CLR	R0	:	I	4559
000102	010001		4\$:	MOV	R0,R1	:	I,*	4560
000104	066701	000000G		ADD	HEAD.AREA,R1			
000110	005011			CLR	(R1)			
000112	062700	000002		ADD	#2,R0	:	*,I	4559
000116	020027	000006		CMP	R0,#6	:	I,*	
000122	101767			BLOS	4\$			
000124	005003			CLR	R3	:	I	4569
000126	010301		5\$:	MOV	R3,R1	:	I,*	4571
000130	006301			ASL	R1			
000132	006301			ASL	R1			
000134	010102			MOV	R1,R2			
000136	066702	000000G		ADD	SEND.RING,R2			
000142	010346			MOV	R3,-(SP)	:	I,*	
000144	012746	000054		MOV	#54,-(SP)			
000150	004767	000000G		JSR	PC,BL\$MUL			
000154	062700	000004G		ADD	#SND.ENVELOPE+4,R0			
000160	010012			MOV	R0,(R2)			
000162	010100			MOV	R1,R0	:		4572
000164	066700	000000G		ADD	SEND.RING,R0			
000170	062700	000002		ADD	#2,R0			
000174	012710	040000		MOV	#40000,(R0)			4576
000200	022626			CMP	(SP)*,(SP)*	:		4570
000202	005203			INC	R3	:	I	4569
000204	020327	000003		CMP	R3,#3	:	I,*	
000210	101746			BLOS	5\$			
000212	005002			CLR	R2	:	I	4586
000214	010201		6\$:	MOV	R2,R1	:	I,*	4588
000216	006301			ASL	R1			
000220	006301			ASL	R1			
000222	010103			MOV	R1,R3			
000224	066703	000000G		ADD	RECEIVE.RING,R3			
000230	010200			MOV	R2,R0	:	I,*	
000232	000300			SWAB	R0			
000234	106000			RORB	R0			
000236	006000			ROR	R0			
000240	006000			ROR	R0			
000242	142700	000077		BICB	#77,R0			
000246	062700	000004G		ADD	#REC.ENVELOPE+4,R0			
000252	010013			MOV	R0,(R3)			
000254	010100			MOV	R1,R0	:		4584

ZRCH85
REV B PATCH 00

ZRCH80 RC25 DISK FORMATTER
MODULE DECLARATIONS

5-Apr-1984 13:46:24
11 Jan-1984 13:23:12

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]ZRCH85.B16;4 (24)

000256	066700	000000G		ADD	RECEIVE.RING,R0		
000262	062700	000002		ADD	#2,R0		
000266	012710	140000		MOV	#140000,(R0)		
000272	005202			INC	R2		4593
000274	020227	000003		CMP	R2,#3		4586
000300	101745			BLOS	6\$		
000302	012767	177777	000000G	MOV	#-1,NRD.SLOT		4599
000310	012767	177777	000000G	MOV	#1,NSD.SLOT		4600
000316	105067	000000G		CLRB	NXT.CRN		4601
000322	005000			CLR	R0		4608
000324	012760	000074	000000G	7\$: MOV	#74,REC.ENVELOPE(R0)		4609
000332	062700	000100		ADD	#100,R0		4608
000336	020027	000300		CMP	R0,#300		
000342	101770			BLOS	7\$		
000344	005000			CLR	R0		4618
000346	012760	100000	000000G	8\$: MOV	#-100000,OUT\$STD.BUF(R0)		4620
000354	005060	000002G		CLR	OUT\$STD.BUF+2(R0)		4621
000360	062700	000004		ADD	#4,R0		4618
000364	020027	000014		CMP	R0,#14		
000370	101766			BLOS	8\$		
000372	005000			CLR	R0		4522
000374	000207			RTS	PC		4465

: Routine Size: 127 words, Routine Base: AC\$CODE + 7310
: Maximum stack depth per invocation: 7 words

: 4629
: 4630 end
: 4631
: 4632 eludom

: OTS external references
.GLOBL \$SAVE4, \$SAVE3, \$SAVE2, BL\$MUL

: PSECT SUMMARY

: Psect Name	Words	Attributes
: AC\$CODE	2019	R0, I, LCL, REL, CON

: LIBRARY STATISTICS

: File	Total	Symbols Loaded	Percent	Blocks Read
: SPIDER\$USERS:[NEALE.AZTEC]ZRCH80.L16;2	398	278	69	55

ZRCMB7

ZRCMB0 RC25 DISK FORMATTER

5 Apr-1984 13:51:39
23 Sep-1983 14:16:03VAX-11 Bliss 16 V3 555
SPIDER#USERS:(NEALE,AZTEC)ZRCMB7.B16;2 (1)

SEQ 0234

Page 1

```

: 0001 MODULE ZRCMB7  (#TITLE  ZRCMB0 RC25 DISK FORMATTER'
: 0002 IDENT = 'REV B PATCH 00',
: 0003 ADDRESSING_MODE (RELATIVE),
: 0004 ENVIRONMENT (NOEIS)) *
: 0005 BEGIN
: 0006
: 0007 #SBTTL 'LAST ADDRESS AND SETUP SECTION'
: 0008 !
: 0009 ! Pretty Declarations
: 0010 !
: 0011 ! <blf/lowercase_key>
: 0012 !
: 0013
: 0014 LIBRARY  ZRCMB0';           !Define RC25 Formatter library
: 0015
: 0016 REQUIRE 'BLSMAC.REQ';     !Define Bliss macro require file
: 1505
: 1506 !*
: 1507 ! The LASTAD macro must be the final statement (except .end) in a pro
: 1508 ! gram. The call generates an even address reflecting the first word of
: 1509 ! memory unused by the program.
: 1510 ! -
: 1511
: 1512 LASTAD
: 1513
: 1514 !*
: 1515 ! Hardcoded P TABLES
: 1516 !
: 1517 ! These optional hardware P TABLES are located (when present) between
: 1518 ! the 'LASTAD' macro and the ".END" statement. These hardware P-TABLES
: 1519 ! are above and beyond the default hardware P-TABLE located in the main
: 1520 ! body of the program. These P-TABLES wind up appended to the BIN file
: 1521 ! of the diagnostic, just as though the supervisor or the 'SETUP' utility
: 1522 ! had built them there. Thus the diagnostic can be "pre-parameterized"
: 1523 ! by the programmer.
: 1524 !
: 1525 ! If this hardcoded P_TABLE section is not wanted then define "number" in
: 1526 ! the BGNSETUP macro to zero and omitt BGNTAB and ENDTAB macros.
: 1527 !
: 1528 ! Coding sample is as follows:
: 1529 !
: 1530 ! LASTAD
: 1531 !
: 1532 ! BGNSETUP (Number)           !Number of P TABLES
: 1533 !
: 1534 ! BGNPTAB
: 1535 ! (DATA)
: 1536 ! (DATA)
: 1537 ! (DATA)
: 1538 ! ENOPTAB
: 1539 !
: 1540 ! BGNPTAB
: 1541 ! (DATA)
: 1542 ! (DATA)
: 1543 ! (DATA)
: 1544 ! ENOPTAB
: 1545 !

```

ZRCHB7 ZRCHB0 RC25 DISK FORMATTER
REV B PATCH 00 LAST ADDRESS AND SETUP SECTION

5 Apr 1984 13:51:39
23 Sep 1983 14:16:03

SEQ 0235
Page 2
VAX-11 B11es 16 V3-555
SPIDER\$USERS:(NEALE.AZTEC)ZRCHB7.L16;2 (1)

: 1546 : ENDSCTUP
: 1547 : .END
: 1548 :
: 1549 :
: 1550 BGNSETUP (0);
: 1551 :
: 1552 : No optional P Tables are defined
: 1553 : within this program.
: 1554 :
: 1555 ENDSCTUP

.TITLE ZRCHB7 ZRCHB0 RC25 DISK FORMATTER
.IDENT /REV B /

000000 .PSECT \$XYZ\$ RO
000000 000004 BL\$LAS:::WORD T\$FREE
000002 000000C .WORD <<T\$FREE <BL\$LAS+4>>/2>
000004 000000 T\$FREE:::WORD 0

000004' L\$LAST== BL\$LAS+4
000000 T\$PTHV== 0

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

1504

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

: 1556 END
: 1557 ELUDOM

PSECT SUMMARY

: Psect Name Words Attributes
: \$XYZ\$ 4 RO , I , LCL, REL, CON

LIBRARY STATISTICS

: File Total Symbols Loaded Percent Blocks Read
: SPIDER\$USERS:(NEALE.AZTEC)ZRCHB0.L16;2 398 2 0 13

D3

ZRCHB' ZRCHB0 RC25 DISK FORMATTER
REV B PATCH 00 LAST ADDRESS AND SETUP SECTION

5 Apr 1984 13:51:39
23 Sep 1983 14:16:03

VAX-11 Bliss 16 V3 555
SPIDER\USERS:(NFALE.AZTEC)ZRCHB7.B16;2 (1)

SEQ 0236
Page 3

: COMMAND QUALIFIERS

: BLISS /PDP11/LIST ZRCHB7.B16

: Size: 1 code + 3 data words
: Run Time: 00:06.5
: Elapsed Time: 00:38.1
: Memory Used: 99 pages
: Compilation Complete

Partition name : DUMMY
 Identification : REV B
 Task UIC : [300,12]
 Task attributes: -MD
 Total address windows: 1.
 Task image size : 15680. words
 Task address limits: 002000 077177
 R W disk blk limits: 000002 000077 000076 00062.

*** Root segment: ZRCH82

R/w mem limits: 002000 077177 075200 31360.
 Disk blk limits: 000002 000077 000076 00062.

Memory allocation synopsis:

Section	Title	Ident	File
. BLK.:(RW,I,LCL,REL,CON)	002000	000000	00000.
AA\$COD:(RO,I,LCL,REL,CON)	002000	001434	00796.
	002000	000166	00118.
	002166	001246	00678.
AB\$COD:(RO,I,LCL,REL,CON)	003434	002312	01226.
	003434	002312	01226.
AC\$COD:(RO,I,LCL,REL,CON)	005746	007706	04038.
	005746	007706	04038.
AD\$COD:(RO,I,LCL,REL,CON)	015654	000002	00002.
	015654	000002	00002.
BL\$COD:(RO,I,LCL,REL,CON)	015656	000424	00276.
	015656	000316	00206.
	016174	000106	00070.
\$DMCOD:(RO,D,GBL,REL,CON)	016302	043072	17978.
	016302	043072	17978.
\$GLOB\$:(RO,D,GBL,REL,CON)	061374	002554	01388.
	061374	002554	01388.
\$PLIT\$:(RO,D,GBL,REL,CON)	064150	013016	05646.
	064150	013016	05646.
\$XYZ\$:(RO,I,LCL,REL,CON)	077166	000010	00008.
	077166	000010	00008.
			ZRCH82 REV B ZRCH82.OBJ;1
			ZRCH83 REV B ZRCH83.OBJ;1
			ZRCH84 REV B ZRCH84.OBJ;1
			ZRCH85 REV B ZRCH85.OBJ;1
			ZRCH86 REV B ZRCH86.OBJ;1
			B16MUL V3.0 NEISLB.OLB;4
			B16SAV V3.0 NEISLB.OLB;4
			ZRCH86 REV B ZRCH86.OBJ;1
			ZRCH82 REV B ZRCH82.OBJ;1
			ZRCH82 REV B ZRCH82.OBJ;1
			ZRCH87 REV B ZRCH87.OBJ;1

Global symbols:

ABORT	010572 R	BOOT.F	076034-R	DFPTBL	002140-R	ERRTYP	002126 R	FMT4	064354-R	GP\$1	002252-R	HW.Q1.	076476 R
ABO.MS	075604-R	BOOT.R	015016-R	DMSA	016342-R	EX.LOC	012004 R	FMT5	064440-R	GP\$2	002262-R	HW.Q2.	076534-R
ACTIVE	076240 R	BR.LEV	064112-R	DUMMY	015654-R	EX.SUP	011366-R	FMT6	064550-R	GP\$3	002272 R	HW.Q3.	076572 R
AZFMTR	016302 R	COM.AR	061374-R	DUP\$I.	007436-R	FCT.BU	062332-R	FMT7	064646 R	GP\$4	002304 R	HW.Q4.	076630-R
BL\$DIV	016102-R	CRLF	065016-R	D\$PCNT	002122-R	FCT.RE	076416-R	GET.CM	006264 R	GP\$5	002322 R	HW.UNI	002146 R
BL\$LAS	077166-R	CTO.WA	010362-R	EMSG.S	071030-R	FLG.WR	064062-R	GET.DU	011070 R	MUSA	016302 R	HW.VEC	002142-R
BL\$MOD	016114-R	DATETX	064046 R	ERRBLK	002134-R	FMT1	064150-R	GET.NR	006144 R	HEAD.A	061444-R	INACTI	076322-R
BL\$MUL	015656 R	DATMSG	075412-R	ERRMSG	002132-R	FMT2	064156-R	GET.NS	006116 R	HW.BR.	002144 R	INIT.C	015256-R
BL\$SHF	016126-R	DECODE	006352-R	ERRNBR	002130-R	FMT3	064246-R	GOOD.N	075742-R	HW.IP.	002140 R	INT\$I.	014540 R

ISD.ST 064130-R	L\$DISP 002124-R	L\$HPTP 002022-R	L\$PROT 002160-R	NEX.FL 064064-R	RECEIV 061446-R	SW.Q2. 076732-R
IS.TIM 014550-R	L\$DLY 002116-R	L\$HRDL 002250-R	L\$PRT 002112-R	NO.ADD 075444-R	REC.BU 063332-R	SW.Q3. 077022-R
LOAD.F 005746-R	L\$DTP 002040-R	L\$HW 002140-R	L\$REPP 002062-R	NRD.SL 064104-R	REC.DA 013056-R	SW.Q4. 077106-R
LOAD.O 006172-R	L\$DTYP 002034-R	L\$HWLE 002136-R	L\$REV 002010-R	NSD.SL 064102-R	REC.EN 061452-R	SW.UNA 002154-R
LUN 064074-R	L\$DU 003412-R	L\$ICP 002104-R	L\$RPT 002334-R	NXT.CR 064070-R	RET.EN 064044-R	TO.MAN 075664-R
L\$ACP 002110-R	L\$DUT 002072-R	L\$INIT 003334-R	L\$SFTL 002320-R	ON.LIN 014162-R	RET.ST 064072-R	T\$FREE 077172-R
L\$APT 002036-R	L\$DVTY 002214-R	L\$LADP 002026-R	L\$SOFT 002322-R	OUT\$ST 064024-R	RINGBA 061404-R	T\$PTHV 000000
L\$AU 003424-R	L\$EF 002052-R	L\$LAST 077172-R	L\$SPC 002056-R	OVSA 064066-R	RSVD.S 064120-R	T1 005732-R
L\$AUT 002070-R	L\$ENVI 002044-R	L\$LOAD 002100-R	L\$SPCP 002020-R	PFE.ST 067030-R	SDUP.S 074716-R	UNIT.N 064114-R
L\$AUTO 003346-R	L\$ERRT 002126-R	L\$LUN 002074-R	L\$SPTP 002024-R	PID.FM 064714-R	SEND.D 012316-R	VEC.AD 064110-R
L\$CCP 002106-R	L\$ETP 002102-R	L\$MRE 002050-R	L\$STA 002030-R	PID.SA 064076-R	SEND.R 061450-R	XCRLF 065022-R
L\$CLEA 003400-R	L\$EXP1 002046-R	L\$NAME 002000-R	L\$SW 002154-R	PORT.I 076164-R	SET.CN 013616-R	\$END.L 077174-R
L\$CO 002032-R	L\$EXP4 002064-R	L\$NDHR 002316-R	L\$SWLE 002152-R	PROTO. 076116-R	SFPTBL 002154-R	\$SAVE2 016174-R
L\$DEPO 002011-R	L\$EXP5 002066-R	L\$NDHW 002150-R	L\$TEST 002114-R	PTBL.P 064116-R	SMSCP. 075360-R	\$SAVE3 016210-R
L\$DESC 002166-R	L\$HARD 002252-R	L\$NDSF 002330-R	L\$TIML 002014-R	PWR.MS 075522-R	SND.BU 063712-R	\$SAVE4 016226-R
L\$DESP 002076-R	L\$HIME 002120-R	L\$NDSW 002156-R	L\$UNIT 002012-R	RC.STR 074364-R	SND.EN 062052-R	\$SAVE5 016246-R
L\$DEVP 002060-R	L\$HPCP 002016-R	L\$PRIO 002042-R	MSGADR 063334-R	RC25.A 064106-R	SW.Q1. 076666-R	

*** Task builder statistics:

Total work file references: 15303.
 Work file reads: 0.
 Work file writes: 0.
 Size of core pool: 3216. words (12. pages)
 Size of work file: 2304. words (9. pages)

Elapsed time:00:00:33

ZRCMB CREATED BY TKB ON 5 APR 84 AT 13:53

PAGE 1

SEQ 0239

GLOBAL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES...
ABORT	010572 R	ZRCMB4 * ZRCMB5
ABO.MS	075604 R	* ZRCMB2 ZRCMB3
ACTIVE	076240-R	* ZRCMB2 ZRCMB4
AZFMTR	016302-R	ZRCMB3 ZRCMB5 * ZRCMB6
BL\$DIV	016102-R	* B16MUL ZRCMB3
BL\$LAS	077166-R	* ZRCMB7
BL\$MOD	016114 R	* B16MUL
BL\$MUL	015656-R	* B16MUL ZRCMB5
BL\$SHF	016126 R	* B16MUL
BOOT.F	076034 R	* ZRCMB2 ZRCMB4
BOOT.R	015016-R	ZRCMB4 * ZRCMB5
BR.LEV	064112-R	* ZRCMB2 ZRCMB3
COM.AR	061374 R	* ZRCMB2 ZRCMB5
CRLF	065016-R	* ZRCMB2 ZRCMB3 ZRCMB4
CTO.WA	010362-R	* ZRCMB5
DATETX	064046-R	* ZRCMB2 ZRCMB3 ZRCMB4
DATMSG	075412-R	* ZRCMB2 ZRCMB3 ZRCMB4
DECODE	006352-R	ZRCMB3 ZRCMB4 * ZRCMB5
DFPTBL	002140-R	* ZRCMB2
DMSA	016342-R	ZRCMB5 * ZRCMB6
DUMMY	015654-R	* ZRCMB6
DUP\$I.	007436-R	ZRCMB4 * ZRCMB5
D\$PCNT	002122-R	* ZRCMB2
EMSG.S	071030 R	* ZRCMB2 ZRCMB5
ERRBLK	002134-R	* ZRCMB2
ERRMSG	002132-R	* ZRCMB2
ERRNBR	002130 R	* ZRCMB2
ERRTYP	002126 R	* ZRCMB2
EX.LOC	012004-R	ZRCMB4 * ZRCMB5
EX.SUP	011366-R	ZRCMB4 * ZRCMB5
FCT.BU	062332-R	* ZRCMB2 ZRCMB4
FCT.RE	076416-R	* ZRCMB2 ZRCMB4
FLG.WR	064062-R	* ZRCMB2 ZRCMB3 ZRCMB4
FMT1	064150-R	* ZRCMB2 ZRCMB4
FMT2	064156 R	* ZRCMB2 ZRCMB3
FMT3	064246-R	* ZRCMB2 ZRCMB3
FMT4	064354-R	* ZRCMB2 ZRCMB5
FMT5	064440 R	* ZRCMB2 ZRCMB4
FMT6	064550-R	* ZRCMB2 ZRCMB4
FMT7	064646 R	* ZRCMB2 ZRCMB4
GET.CM	006264-R	* ZRCMB5
GET.DU	011070 R	ZRCMB4 * ZRCMB5
GET.NR	006144 R	* ZRCMB5
GET.NS	006116-R	* ZRCMB5
GOOD.N	075742 R	* ZRCMB2 ZRCMB3
GP\$1	002252 R	* ZRCMB3
GP\$2	002262-R	* ZRCMB3
GP\$3	002272-R	* ZRCMB3
GP\$4	002304-R	* ZRCMB3
GP\$5	002322-R	* ZRCMB3
HDSA	016302-R	ZRCMB5 * ZRCMB6
HEAD.A	061444-R	* ZRCMB2 ZRCMB5

ZRCMB CREATED BY TKB ON 5 APR 84 AT 13:53 PAGE 2

SEQ 0240

GLOBAL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES...
HW.BR.	002144 R	♦ ZRCMB2
HW.IP.	002140-R	♦ ZRCMB2
HW.Q1.	076476 R	♦ ZRCMB2 ZRCMB3
HW.Q2.	076534-R	♦ ZRCMB2 ZRCMB3
HW.Q3.	076572-R	♦ ZRCMB2 ZRCMB3
HW.Q4.	076630-R	♦ ZRCMB2 ZRCMB3
HW.UNI	002146-R	♦ ZRCMB2
HW.VEC	002142 R	♦ ZRCMB2
INACTI	076322 R	♦ ZRCMB2 ZRCMB4
INIT.C	015256-R	♦ ZRCMB4 ♦ ZRCMB5
INT\$I.	014540 R	ZRCMB4 ♦ ZRCMB5
ISD.ST	064130 R	♦ ZRCMB2 ZRCMB3 ZRCMB4 ZRCMB5
IS.TIM	014550 R	♦ ZRCMB5
LOAD.F	005746 R	ZRCMB3 ♦ ZRCMB5
LOAD.O	006172 R	♦ ZRCMB5
LUN	064074 R	♦ ZRCMB2 ZRCMB3 ZRCMB4
L\$ACP	002110 R	♦ ZRCMB2
L\$APT	002036-R	♦ ZRCMB2
L\$AU	003424-R	ZRCMB2 ♦ ZRCMB3
L\$AUT	002070-R	♦ ZRCMB2
L\$AUTO	003346-R	ZRCMB2 ♦ ZRCMB3
L\$CCP	002106-R	♦ ZRCMB2
L\$CLEA	003400 R	ZRCMB2 ♦ ZRCMB3
L\$CO	002032 R	♦ ZRCMB2
L\$DEPO	002011-R	♦ ZRCMB2
L\$DESC	002166-R	ZRCMB2 ♦ ZRCMB3
L\$DESP	002076 R	♦ ZRCMB2
L\$DEVP	002060-R	♦ ZRCMB2
L\$DISP	002124-R	♦ ZRCMB2
L\$DLY	002116 R	♦ ZRCMB2 ZRCMB5
L\$DTP	002040 R	♦ ZRCMB2
L\$DTYP	002034-R	♦ ZRCMB2
L\$DU	003412-R	ZRCMB2 ♦ ZRCMB3
L\$DUT	002072-R	♦ ZRCMB2
L\$DVTY	002214-R	ZRCMB2 ♦ ZRCMB3
L\$EF	002052-R	♦ ZRCMB2
L\$ENVI	002044-R	♦ ZRCMB2
L\$ERRT	002126-R	♦ ZRCMB2
L\$ETP	002102-R	♦ ZRCMB2
L\$EXP1	002046-R	♦ ZRCMB2
L\$EXP4	002064-R	♦ ZRCMB2
L\$EXP5	002066 R	♦ ZRCMB2
L\$HARD	002252-R	ZRCMB2 ♦ ZRCMB3
L\$HIME	002120-R	♦ ZRCMB2
L\$HPCP	002015-R	♦ ZRCMB2
L\$HPTP	002022-R	♦ ZRCMB2
L\$HRDL	002250-R	♦ ZRCMB3
L\$HW	002140-R	♦ ZRCMB2
L\$HWLE	002136-R	♦ ZRCMB2
L\$ICP	002104-R	♦ ZRCMB2
L\$INIT	003334-R	ZRCMB2 ♦ ZRCMB3
L\$LADP	002026-R	♦ ZRCMB2

GLOBAL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES...
L\$LAST	077172-R	ZRCHB2 * ZRCHB7
L\$LOAD	002100 R	* ZRCHB2
L\$LUN	002074 R	* ZRCHB2
L\$MREV	002050-R	* ZRCHB2
L\$NAME	002000-R	* ZRCHB2
L\$NDHR	002316-R	* ZRCHB3
L\$NDHW	002150 R	* ZRCHB2
L\$NDSF	002330-R	* ZRCHB3
L\$NDSW	002156-R	* ZRCHB2
L\$PRIO	002042 R	* ZRCHB2
L\$PROT	002160 R	* ZRCHB2
L\$PRT	002112 R	* ZRCHB2
L\$REPP	002062 R	* ZRCHB2
L\$REV	002010-R	* ZRCHB2
L\$RPT	002334 R	* ZRCHB2 * ZRCHB3
L\$SF TL	002320 R	* ZRCHB3
L\$SOFT	002322-R	ZRCHB2 * ZRCHB3
L\$SPC	002056-R	* ZRCHB2
L\$SPCP	002020 R	* ZRCHB2
L\$SPTP	002024 R	* ZRCHB2
L\$STA	002030-R	* ZRCHB2
L\$SW	002154-R	* ZRCHB2
L\$SWLE	002152-R	* ZRCHB2
L\$TEST	002114-R	* ZRCHB2
L\$TIML	002014-R	* ZRCHB2
L\$UNIT	002012-R	* ZRCHB2 ZRCHB3
MSGADR	063334-R	* ZRCHB2 ZRCHB4
NEX.FL	064064-R	* ZRCHB2 ZRCHB4 ZRCHB5
NO.ADD	075444-R	* ZRCHB2 ZRCHB3
NRD.SL	064104 R	* ZRCHB2 ZRCHB4 ZRCHB5
NSD.SL	064102-R	* ZRCHB2 ZRCHB4 ZRCHB5
NXT.CR	064070-R	* ZRCHB2 ZRCHB4 ZRCHB5
ON.LIN	014162-R	ZRCHB4 * ZRCHB5
OUT\$ST	064024-R	* ZRCHB2 ZRCHB5
OVSA	064066-R	* ZRCHB2 ZRCHB3 ZRCHB5
PFE.ST	067030-R	* ZRCHB2 ZRCHB5
PID.FM	064714-R	* ZRCHB2 ZRCHB5
PID.SA	064076-R	* ZRCHB2 ZRCHB4 ZRCHB5
PORT.I	076164-R	* ZRCHB2 ZRCHB4
PROTO.	076116-R	* ZRCHB2 ZRCHB4
PTBL.P	064116-R	* ZRCHB2 ZRCHB3 ZRCHB4
PWR.MS	075522-R	* ZRCHB2 ZRCHB3
RC.STR	074364-R	* ZRCHB2 ZRCHB5
RC25.A	064106-R	* ZRCHB2 ZRCHB3 ZRCHB4 ZRCHB5
RECEIV	061446-R	* ZRCHB2 ZRCHB5
REC.BU	063332-R	* ZRCHB2 ZRCHB4 ZRCHB5
REC.DA	013056-R	ZRCHB4 * ZRCHB5
REC.EN	061452-R	* ZRCHB2 ZRCHB5
RET.EN	064044-R	* ZRCHB2 ZRCHB4 ZRCHB5
RET.ST	064072-R	* ZRCHB2 ZRCHB4 ZRCHB5
RINGBA	061404 R	* ZRCHB2
RSVD.S	064120 R	* ZRCHB2 ZRCHB5

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...
SDUP.S	074716-R	♦ ZRCH82 ZRCH85
SEND.D	012316-R	ZRCH84 ♦ ZRCH85
SEND.R	061450-R	♦ ZRCH82 ZRCH85
SET.CN	013616 R	ZRCH84 ♦ ZRCH85
SFPTBL	002154-R	♦ ZRCH82
SMSCP.	075360-R	♦ ZRCH82 ZRCH85
SND.BU	063712-R	♦ ZRCH82 ZRCH84 ZRCH85
SND.EN	062052 R	♦ ZRCH82 ZRCH85
SW.Q1.	076666-R	♦ ZRCH82 ZRCH83
SW.Q2.	076732 R	♦ ZRCH82 ZRCH83
SW.Q3.	077022-R	♦ ZRCH82 ZRCH83
SW.Q4.	077106 R	♦ ZRCH82 ZRCH83
SW.UNA	002154-R	♦ ZRCH82 ZRCH83 ZRCH84
TO.MAN	075664-R	♦ ZRCH82 ZRCH83
T\$FREE	077172 R	♦ ZRCH87
T\$PTHV	000000	ZRCH82 ♦ ZRCH87
T1	005732-R	ZRCH82 ♦ ZRCH84
UNIT.N	064114-R	♦ ZRCH82 ZRCH83 ZRCH84 ZRCH85
VEC.AD	064110 R	♦ ZRCH82 ZRCH83 ZRCH84 ZRCH85
XCRLF	065022 R	♦ ZRCH82 ZRCH83
\$END.L	077174-R	♦ ZRCH87
\$SAVE2	016174-R	B16MUL ♦ B16SAV ZRCH83 ZRCH85
\$SAVE3	016210 R	♦ B16SAV ZRCH84 ZRCH85
\$SAVE4	016226-R	♦ B16SAV ZRCH85
\$SAVE5	016246-R	B16MUL ♦ B16SAV