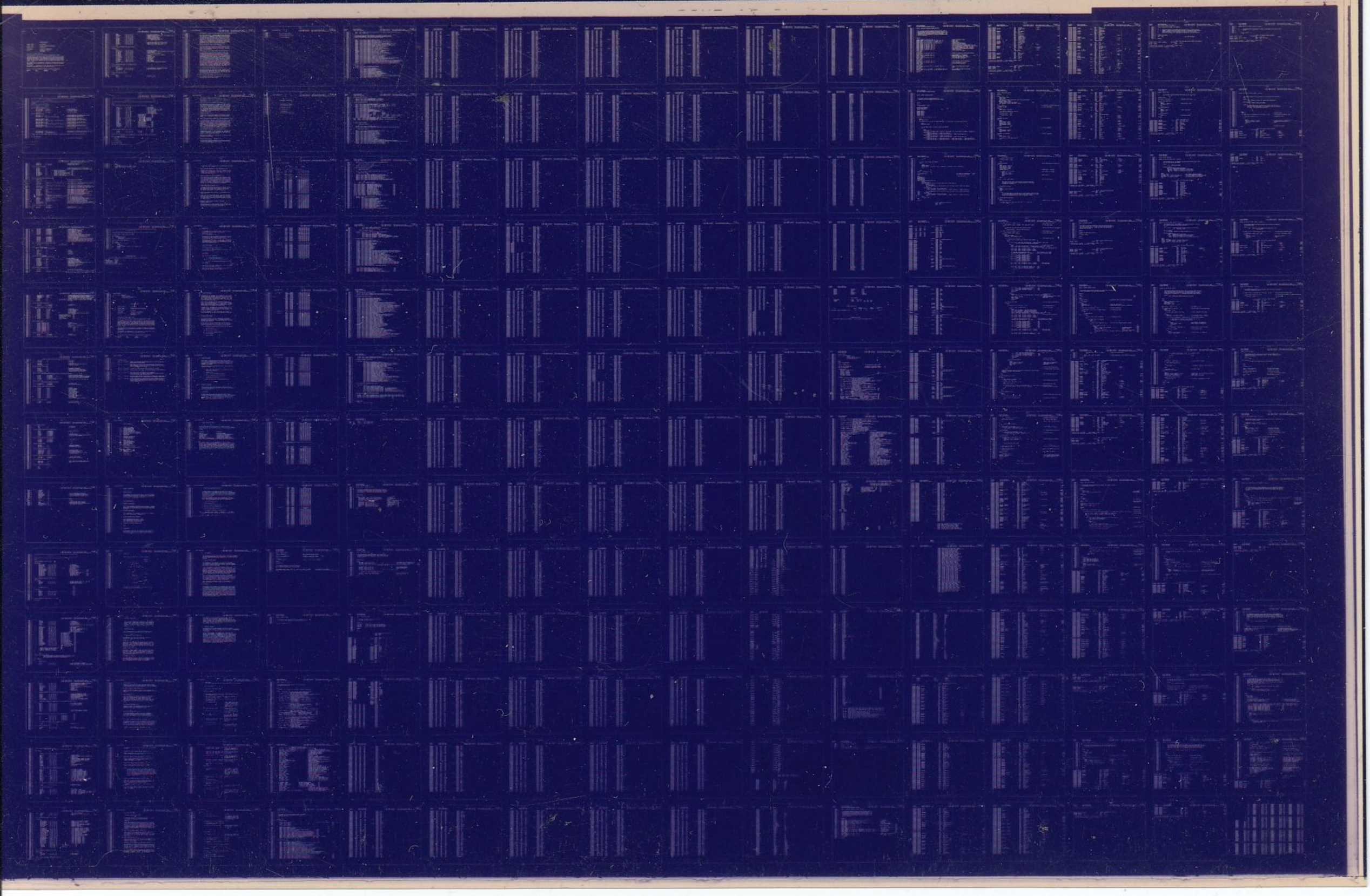
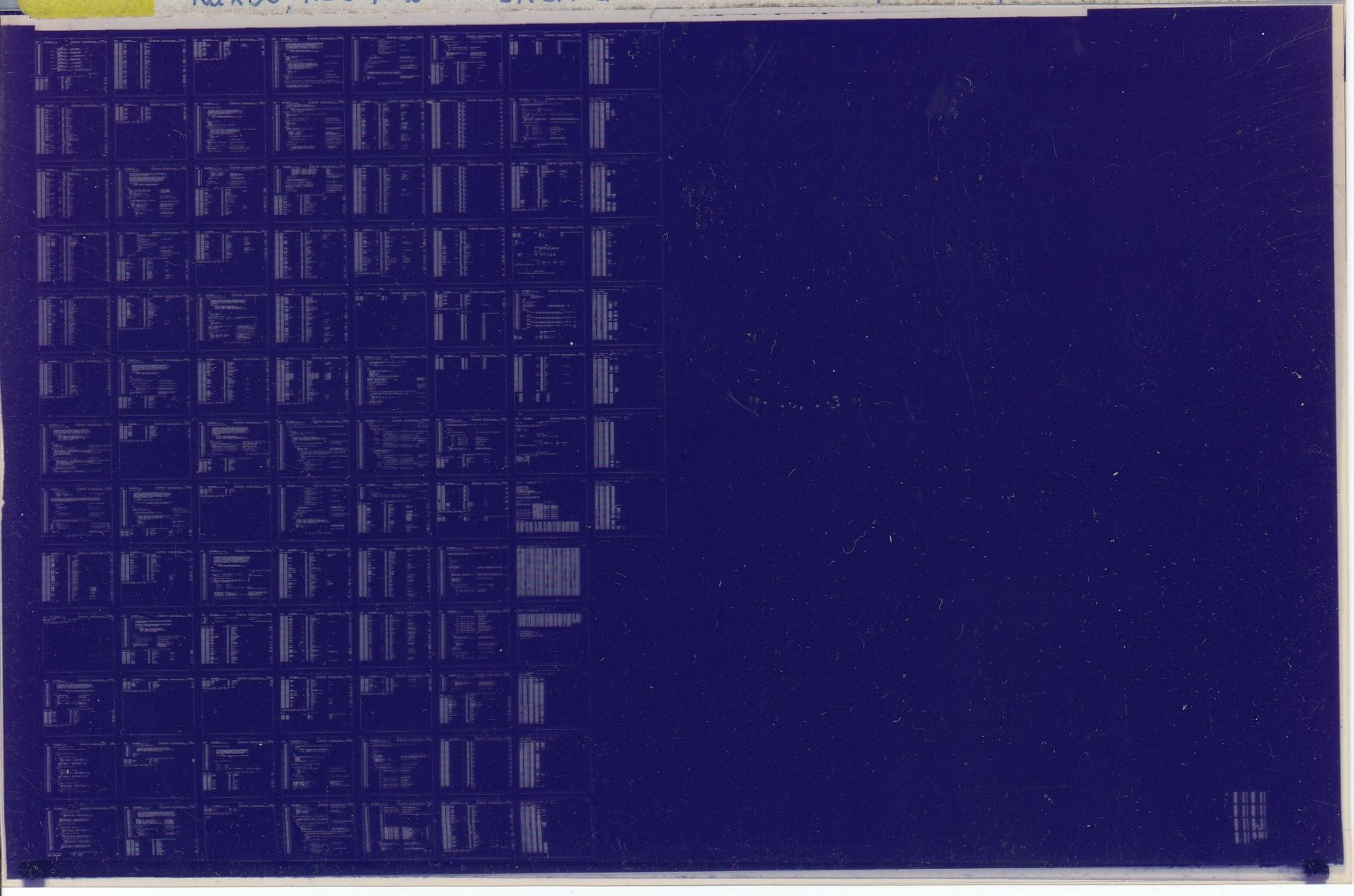
RQDX1/2/3 RX50 RQDX/RUX Exerciser AH-T399G-MC
RUX 50 RD51/52 CZRQ AGO 1 of 3 Juli 85



RQDX1/2/3 RX50 RQDX/RUX Exerciser AH-T399G-MC
RUX50 RD51/52 CZRQ AGO 2 of 3 Juli 85

RQDX 1,2,3 RX50 RQDX/RUX50 Exercise AH-T399G-MC
RUX50, RD51,52 CZRQAGO 3 of 3 July 85



IDENTIFICATION

PRODUCT CODE:	AC-T398G-MC
PRODUCT NAME:	CZRQAGO RQDX/RUX50 EXERCISER
PRODUCT DATE:	04-APR-85
MAINTAINER:	DIAGNOSTIC ENGINEERING
AUTHOR:	RAVINDER K. KARWAN BOB POWERS

Copyright (C) 1983, 1984, 1985

Digital Equipment Corporation, Maynard, Massachusetts 01754

This software is furnished under a license for use only on a single computer system and may be copied only with the inclusion of the above copyright notice. This software, or any other copies thereof, may not be provided or otherwise made available to any other person except for use on such system and to one who agrees to these license terms. Title to and ownership of the software shall at all times remain in DEC.

the information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

```
;  0001  0     !*****************************************************************************
;  0002  0     !                                                                            *
;  0003  0     !                          L I T E R A L S                                   *
;  0004  0     !                                                                            *
;  0005  0     !*****************************************************************************
;  0006  0     !
;  0007  0     LITERAL
;  0008  0     !
;  0009  0     !****** ODT TRAP VECTOR LOCATION
;  0010  0     !
;  0011  0             O_TVEC          = %o'14',
;  0012  0     !
;  0013  0     !****** HARDWARE ADDRESSES ETC.
;  0014  0     !
;  0015  0             INIT_INTR_VECT  = %o'154',                      ! VECTOR ADDRESS
;  0016  0             INIT_IP_ADDR    = %o'172150',                   ! IP REGISTER ADDRESS
;  0017  0             INIT_BR_LEVEL   = %o'4',                        ! BUS REQUEST LEVEL
;  0018  0     !
;  0019  0             LINE_CLOCK      = %o'177546',                   ! LINE-CLOCK ADDRESS
;  0020  0     !
;  0021  0     !****** HARDWARE LIMITS
;  0022  0     !
;  0023  0             MAX_CTLR        = 1,                            ! MAXIMUM NUMBER OF LCP CONTROLLERS ALLOWED
;  0024  0             UNITS_PER_CNTR  = 4,                            ! MAXIMUM UNITS PER CONTROLLER
;  0025  0             MAX_UNITS       = MAX_CTLR * UNITS_PER_CNTR,    ! MAXIMUM NUMBER OF UNITS TO TEST
;  0026  0
;  0027  0     !                                                       !HIGHEST RD51 LBN = 52137 OCT
;  0028  0             RD51_MAX_TRACK  = 1200,                         ! MAXIMUM NUMBER OF TRACKS FOR RD51
;  0029  0             RD51_SEC_PER_TRK= 18,                           ! NUMBER OF SECTORS PER TRACK FOR RD51
;  0030  0             RD51_MAX_LBN    = RD51_MAX_TRACK * RD51_SEC_PER_TRK - 1,! MAX LBN FOR RD51
;  0031  0
;  0032  0             RD52_MAX_TRACK  = 2976,                         ! MAXIMUM NUMBER OF TRACKS FOR RD52
;  0033  0             RD52_SEC_PER_TRK= 18,                           ! NUMBER OF SECTORS PER TRACK FOR RD52
;  0034  0             RD52_MAX_LBN    = RD52_MAX_TRACK * RD52_SEC_PER_TRK - 1,! MAX LBN FOR RD52
;  0035  0
;  0036  0             RX50_MAX_TRACK  = 80,                           ! MAXIMUM NUMBER OF TRACKS FOR RX50
;  0037  0             RX50_SEC_PER_TRK= 10,                           ! NUMBER OF SECTORS PER TRACK FOR RX50
;  0038  0             RX50_MAX_LBN    = RX50_MAX_TRACK * RX50_SEC_PER_TRK - 1,! MAX LBN FOR RX50
;  0039  0
;  0040  0     !                                                       !HIGHEST RD53 LBN = 416660 OCT (2/016660)
;  0041  0
;  0042  0
;  0043  0
;  0044  0             BYTES_PER_SECT  = 512,                          ! BYTES/SECTOR (AT PRESENT SAME FOR RDs AND RXs)
;  0045  0     !       MAX_XFER_SIZE   = 2 * BYTES_PER_SECT,           ! ARTBITRARY MAX SIZE OF EACH DISK I/O
;  0046  0             MAX_XFER_SIZE   = BYTES_PER_SECT * 3 / 2,
;  0047  0     !
;  0048  0     ! NOTE - BOTH OF THESE NUMBERS ARE NOW ARBITRARILY CHOSEN AS THE NUMBER OF LBNS CONTAINED PER UNIT/10 .
;  0049  0
;  0050  0     !
;  0051  0     !****** RING SIZES
;  0052  0     !
;  0053  0             CR_LOG          = 2,                            ! LOG2 LENGTH OF COMMAND RING
```

```
;    0054  0               RR_LOG          = 2,                                    ! LOG2 LENGTH OF RESPONSE RING
;    0055  0               CRING_LEN       = 1 ^ CR_LOG,                           ! COMMAND RING LENGTH
;    0056  0               RRING_LEN       = 1 ^ RR_LOG,                           ! RESPONSE RING LENGTH
;    0057  0       !
;    0058  0       !****** OFFSETS (IN WORDS)
;    0059  0       !
;    0060  0               OF_UN           = 3,                                    ! OFFSET FROM START OF CST TO FIPST UNIT
;    0061  0               OF_DATA         = 0,                                    ! OFFSET TO DISK UNIT FLAGS   WITHIN UNIT'S CST
;    0062  0               OF_BEG          = 1,                                    ! OFFSET TO BEGINNING BLK NO. WITHIN UNIT'S CST
;    0063  0               OF_BEG1         = 2,        !OFFSET TO START BK HI          ZZZ
;    0064  0               OF_END          = 3,        !OFFSET TO END BLOCK LO         ZZZ
;    0065  0               OF_END1         = 4,        !OFFSET TO END BK HI            ZZZ
;    0066  0               OF_NAME_0       = 5,        !OFFSET TO 1st 2 CHARS OF NAME  ZZZ
;    0067  0               OF_NAME_2       = 6,        !OFFSET TO 2nd 2 CHARS OF NAME  ZZZ
;    0068  0               OF_DUPFLAGS     = 8,        !OFFSET TO DUP FLAGS            ZZZ
;    0069  0               OF_COUNT        = 9,        !OFFSET TO MSCP FUNCTION COUNTER ZZZ
;    0070  0               OF_DBN          = 8,        !OFFSET TO RELATIVE DBN         ZZZ
;    0071  0       !
;    0072  0       !
;    0073  0       !****** TABLE AND OTHER STRUCTURE SIZES
;    0074  0       !
;    0075  0               LBNADR_LEN      = 2,                                    !MAX_LBN'S ARE 2 WD ADDRESSES
 ZZZ
;    0076  0               HWPT_LEN        = 8,                                    ! SIZE (WORDS) OF HW P-TABLE
 ZZZ
;    0077  0               COMM_LEN        = (RRING_LEN * 2) + (CRING_LEN * 2) + 4,! SIZE (WORDS) OF COMMUNICATION AREA PER CONTROLLER
;    0078  0               UNIT_SIZE       = 10,                                   ! SIZE (WORDS) OF CST UNIT ENTRY
 ZZZ
;    0079  0               CST_LEN         = UNITS_PER_CNTR * UNIT_SIZE + OF_UN,   ! SIZE (WORDS) OF A CONTROLLER STATUS TABLE
;    0080  0               TALLY_CLEAR     = 7,                                    ! SIZE (WORDS) OF STATISTICS TBL CLEARED EVERY PASS
;    0081  0               TALLY_TOTALS    = 20,                                   ! SIZE (WORDS) OF STATISTICS TABLE FOR TOTALS
 ZZZ
;    0082  0               TALLY_LEN       = TALLY_CLEAR + TALLY_TOTALS,           ! SIZE (WORDS) OF A STATISTICS TABLE
;    0083  0               C_ERR_LEN       = 1,                                    ! SIZE (WORDS) OF CONTROLLER ERROR TABLE
;    0084  0               RP_LEN          = 22,                                   ! SIZE (WORDS) OF A RETURN PACKET
;    0085  0               MSG_LEN         = 30,                                   ! SIZE (WORDS) OF AN MSCP MESSAGE (TEXT PORTION)
;    0086  0               PKT_LEN         = MSG_LEN + 5,                          ! SIZE (WORDS) OF AN MSCP PACKET
;    0087  0               DCT_LEN         = 9,                                    ! SIZE (WORDS) OF A DRIVER CONTROLLER TABLE
;    0088  0               RDM_LEN         = 16,                                   ! SIZE (WORDS) OF THE RANDOM NUMBER TABLE
;    0089  0               MAX_UDP_CNT     = 16,                                   ! MAX SIZE OF USER DATA PATTERN
;    0090  0               MAX_BUF_CNT     = (CRING_LEN * 2) * MAX_CTLR,           ! MAX NO. OF I/O BUFFERS (BUFF_ADDR & BUFF OWN)
;    0091  0               PKT_CNT         = ((CRING_LEN * 2) + RRING_LEN) * MAX_CTLR,
;    0092  0                                                                       ! NO. OF MSCP PACKETS IN POOL
;    0093  0               RP_CNT          = PKT_CNT - (RRING_LEN * MAX_CTLR),     ! NO. OF RETURN PACKETS IN POOL
;    0094  0               IODQ_LEN        = RP_CNT,                               ! NO. OF ENTRIES IN I/O DONE QUEUE (IODQ)
;    0095  0               OUTC_CNT        = CRING_LEN * 2,                        ! NO. OF ENTRIES/CONTROLLER'S OUTSTANDING CMD LIST
;    0096  0               DP_CNT          = 21,                                   ! NO. OF PRE-DEFINED DATA PATTERNS
;    0097  0               EP_CNT          = MAX_CTLR * RRING_LEN * 3,             ! NO. OF ERROR-LOG PACKET SAVE BUFFERS
;    0098  0               EP_LEN          = PKT_LEN - 3 + 1,                      ! LENGTH OF EACH ERROR-LOG SAVE BUFFER
;    0099  0               LAST_PKT_LEN    = 3,                                    ! BUFFER LENGTH TO SAVE INFO. ABOUT LAST RESPONSE
;    0100  0               TOO_MANY_READS  = 2,                                    !FOR READ/WRITE BALANCE WITH HOST READ COMPARES  ZZZ
;    0101  0               DESC_SIZ        = 4,                                    !NO. OF BYTES IN A PACKET DESCRIPTOR             ZZZ
;    0102  0       !
;    0103  0       !****** SW P-TABLE FLAGS (SWP_FLAGS)
;    0104  0       !
;    0105  0       !ZZZ    SWF_TRC         = %o'000001',                           ! DIAGNOSTIC TRACE
;    0106  0               SWF_APT         = %o'000001',                           !RUNNING UNDER A.P.T. MONITOR                    ZZZ
```

```
;    0107  0              SWF_RDM          = %o'000002',                    ! RANDOM SEEK MODE
;    0108  0              SWF_CRC          = %o'000004',                    ! READ-COMPARE AT CONTROLLER
;    0109  0      !       SWF_DCC          = %o'000010',                    ! DRIVE COMPLEMENT COMPLETE
;    0110  0              SWF_CWC          = %o'000020',                    ! WRITE-COMPARE AT CONTROLLER
;    0111  0              SWF_HWC          = %o'000040',                    ! WRITE-COMPARE AT HOST
;    0112  0              SWF_UDP          = %o'000100',                    ! USER-DEFINED DATA PATTERN
;    0113  0              SWF_CST          = %o'000200',                    ! CLEAR STATISTICAL TABLES
;    0114  0      !       SWF_DIA          = %o'000400',                    ! DIAGNOSTIC PACKAGE, WHEN THIS IS SELECTED
;    0115  0                                                                ! ALL INTERRUPTS ARE WAITED FOR, E.G. ONLY
;    0116  0                                                                ! ONE MSCP PACKET IS OUTSTANDING AT A TIME
;    0117  0              SWF_SEQ          = %o'001000',                    ! RANDOM OR FIXED SEQUENTIAL STEPPING
;    0118  0      !       SWF_DUP          = %o'002000',                    ! RUN DUP DIAGNOSTIC
;    0119  0              SWF_FER          = %o'004000',                    ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT DETECTED
;    0120  0              SWF_HRD          = %o'010000',                    ! HALT ON HARD ERRORS ALSO WITH 'HOE' DRS FLAG?
;    0121  0              SWF_SFT          = %o'020000',                    ! HALT ON SOFT ERRORS ALSO WITH 'HOE' DRS FLAG?
;    0122  0              SWF_BLK          = %o'040000',                    ! HALT ON BAD-BLOCK ERRORS ALSO WITH 'HOE' DRS FLAG?
;    0123  0              SWF_TRY          = %o'100000',                    ! COUNT EACH RETRY AS ANOTHER EXTRA SOFT-ERROR
;    0124  0      !
;    0125  0      !****** FLAGS FOR DUP EXERCISER (DUP_FLAGS)                                    ZZZ
;    0126  0      !                                                                             ZZZ
;    0127  0              SWP_DINT         = %o'2',                !DUP CAUSED INIT           ZZZ
;    0128  0
;    0129  0      !
;    0130  0      !****** ENTRY_REASON VALUES
;    0131  0      !               (HOW PROGRAM WAS INVOKED)
;    0132  0      !
;    0133  0              START            = 1,                             ! START
;    0134  0              RESTART          = 2,                             ! RESTART
;    0135  0              CONT             = 3,                             ! CONTINUE
;    0136  0              PWR_FAIL         = 4,                             ! POWER FAIL
;    0137  0              NEW_PASS         = 5,                             ! NEW PASS
;    0138  0      !
;    0139  0      !****** DROP UNIT REASONS
;    0140  0      !               (LOADED INTO DUR VECTOR)
;    0141  0      !
;    0142  0              DU_USER          = 0,                             ! USER COMMAND
;    0143  0              DU_CONF          = 1,                             ! CONFIGURATION ERROR
;    0144  0              DU_INIT          = 2,                             ! INITIALIZATION ERROR
;    0145  0              DU_XFER          = 3,                             ! TRANSFER LIMIT REACHED
;    0146  0              DU_HERR          = 4,                             ! HARD ERROR LIMIT REACHED
;    0147  0              DU_DFATAL        = 5,                             ! UNRECOVERABLE DEVICE ERROR
;    0148  0              DU_CFATAL        = 6,                             ! UNRECOVERABLE CONTROLLER ERROR
;    0149  0              DU_ONLINE        = 7,                             ! ONLINE FAILED
;    0150  0              DU_ACCESS        = 8,                             ! ACCESS TO LAST TRACK FAILED
;    0151  0              DU_PROTECT       = 9,                             ! WRITE PROTECT CONFLICT
;    0152  0              DU_TIME          = 10,                            ! COMMAND TIME OUT
;    0153  0      !
;    0154  0      !****** MISCELLANEOUS LITERALS
;    0155  0      !
;    0156  0              MAX_DBN          = 63,                            !HIGHEST RELATIVE DBN NUMBER                ZZ
Z
;    0157  0              INI_ATT          = 2,                             ! NO. OF HW INIT ATTEMPTS BEFORE FAILURE IS ASSUMED
;    0158  0              WR_RING          = ((%o'200') or (CR_LOG ^ 3) or (RR_LOG)),
;    0159  0                                                                ! WR-BIT-AND-RING-LENGTH (STEP 1 WRITE/STEP 2 READ)
```

```
;    0160  0              QIO_PER_CTLR    = CRING_LEN * 2,                ! MAXIMUM ! MBER OF OUTSTANDING QIOS PER CONTROLLER
;    0161  0      !       MAX_XFER        = 256,                          ! MAXIMUM SIZE (WORDS) OF AN I/O TRANSFER
;    0162  0              REMOVABLE_BIT   = %o'0',                        ! BIT IN HARDWARE TABLES MARKING A REMOVABLE DISK
;    0163  0              FIXED_BIT       = %o'20',                       ! BIT IN HARDWARE TABLES MARKING A FIXED    DISK
;    0164  0              REMOVABLE       = 0,                            ! NUMBER FOR REMOVABLE DISK WHEN SHIFTED RIGHT
;    0165  0              FIXED           = 1,                            ! NUMBER FOR FIXED     DISK WHEN SHIFTED RIGHT
;    0166  0              RX_50           = 0,                            !D_TYPE FLAG = 0 FOR RX50  (THESE FLAGS AREN'T USED. INSTEAD,)   ZZZ
;    0167  0              RD_51           = 1,                            !D_TYPE FLAG = 1 FOR RD51  (D_TYPE = 1 FOR FIXED, 0 FOR REMOV)   ZZZ
;    0168  0              RD_52           = 2,                            !                2 FOR RD52                                      ZZZ
;    0169  0                                                              !                                                                ZZZ
;    0170  0      !
;    0171  0      !****** MSCP PACKET DESCRIPTOR
;    0172  0      !
;    0173  0              ED_OWN          = %o'100000',                   ! OWNERSHIP BIT
;    0174  0              ED_FLAG         = %o'040000',                   ! FLAG BIT
;    0175  0      !
;    0176  0      !****** MSCP COMMAND PACKET OPCODES
;    0177  0      !
;    0178  0              OP_MSK          = %o'177',                      ! OPCODE MASK
;    0179  0              OP_END          = %o'200',                      ! ENDCODE DESIGNATOR
;    0180  0              OP_ACC          = %o'20',                       ! ACCESS COMMAND
;    0181  0              OP_ONL          = %o'11',                       ! ONLINE COMMAND
;    0182  0              OP_RD           = %o'41',                       ! READ COMMAND
;    0183  0              OP_SCC          = %o'4',                        ! SET CONTROLLER CHARACTERISTICS COMMAND
;    0184  0              OP_WRT          = %o'42',                       ! WRITE COMMAND
;    0185  0              OP_GDS          = %o'1',                        !get dust status            ZZZ
;    0186  0              OP_ESP          = %o'2',                        !execute supplied prog      ZZZ
;    0187  0              OP_ELP          = %o'3',                        !execute local program      ZZZ
;    0188  0              OP_SDD          = %o'4',                        !send data                  ZZZ
;    0189  0              OP_RCD          = %o'5',                        !receive data               ZZZ
;    0190  0              OP_ABT          = %o'6',                        !abort program              ZZZ
;    01    0
;    0192  0      !
;    0193  0      !****** PACKET SIZES
;    0194  0      !
;    0195  0              SZ_ACC = %decimal '32',                         ! ACCESS
;    0196  0              SZ_ONL = %decimal '36',                         ! ON LINE COMMAND
;    0197  0              SZ_RD  = %decimal '32',                         ! READ
;    0198  0              SZ_SCC = %decimal '32',                         ! SET CONTROLLER CHARACTERISTICS
;    0199  0              SZ_WRT = %decimal '32',                         ! WRITE
;    0200  0              SZ_GEN = %decimal '32',                         ! GENERAL PACKET SIZE
;    0201  0              SZ_REC = %DECIMAL '28',                 !               ZZZ
;    0202  0              SZ_SEN = %DECIMAL '28',                 !               ZZZ
;    0203  0              SZ_ELP = %DECIMAL '18',                 !               ZZZ
;    0204  0              SZ_ABT = %DECIMAL '12',                 !               ZZZ
;    0205  0              SZ_GDS = %DECIMAL '12',                 !               ZZZ
;    0206  0      !
;    0207  0      !****** MSCP COMMAND MODIFIERS
;    0208  0      !
;    0209  0              MD_CMP          = %o'040000',                   ! COMPARE
;    0210  0              MD_EXP          = %o'100000',                   ! EXPRESS REQUEST
;    0211  0      !
;    0212  0      !****** CONNECTION ID VALUES (MSCP_PKT, RETPKT)
```

```
;    0213  0      !                (SERVE AS SOURCES AND DESTINATIONS OF MSCP MESSAGES)
;    0214  0      !
;    0215  0              CID_DISK        = 0,                                    ! DISK MSCP
;    0216  0              CID_MSCP        = 0,                                    !DISK MSCP
 !ZZZ
;    0217  0              CID_TAPE        = 1,                                    ! TAPE MSCP
;    0218  0              CID_DUP         = 2,                                    ! DIAGNOSTIC AND UTILITIES PROTOCOL
;    0219  0              CID_DRIVER      = 3,                                    ! EXERCISER "DRIVER"
;    0220  0      !
;    0221  0      !****** MESSAGE TYPE VALUES
;    0222  0      !
;    0223  0              MT_SEQ          = 0,                                    ! SEQUENTIAL (FROM PORT)
;    0224  0              MT_DG           = 1,                                    ! DATAGRAM (FROM PORT)
;    0225  0              MT_CRD          = 2,                                    ! CREDIT NOTIFICATION (FROM PORT)
;    0226  0              MT_FATAL        = 3,                                    ! FATAL DEVICE ERROR (FROM "DRIVER")
;    0227  0              MT_TIMEOUT      = 4,                                    ! COMMAND TIMEOUT (FROM "DRIVER")
;    0228  0      !
;    0229  0      !****** CONTROLLER FLAGS
;    0230  0      !                (IN SET CONTROLLER CHARACTERISTICS COMMAND AND RESPONSE)
;    0231  0      !
;    0232  0              CF_ATN          = %o'000200',                           ! ENABLE ATTENTION MESSAGES
;    0233  0              CF_MSC          = %o'000100',                           ! ENABLE MISCELLANEOUS ERROR LOG MESSAGES
;    0234  0              CF_OTH          = %o'000040',                           ! ENABLE OTHER HOST'S ERROR LOG MESSAGES
;    0235  0              CF_THS          = %o'000020',                           ! ENABLE THIS HOST'S ERROR LOG MESSAGES
;    0236  0      !       CF_MASK         = CF_ATN or CF_MSC or CF_THS,
;    0237  0              CF_MASK         = CF_MSC or CF_THS,                     ! RELEVENT BITS IN CTLR FLAGS WORD
;    0238  0      !
;    0239  0      !****** UNIT FLAGS
;    0240  0      !                (IN ONLINE COMMAND AND RESPONSE)
;    0241  0      !
;    0242  0              UF_REMOVABLE    = %o'000200',                           ! REMOVABLE MEDIA
;    0243  0              UF_WPH          = %o'020000',                           ! WRITE PROTECT (HARDWARE)
;    0244  0      !
;    0245  0      !****** STATUS / EVENT CODE DEFINITIONS
;    0246  0      !
;    0247  0              ST_SUC          = %o'0',                                ! SUCCESS
;    0248  0              ST_CMD          = %o'1',                                ! INVALID COMMAND
;    0249  0              ST_ABO          = %o'2',                                ! COMMAND ABORTED
;    0250  0              ST_OFL          = %o'3',                                ! UNIT OFFLINE
;    0251  0              ST_AVL          = %o'4',                                ! DRIVE AVAILABLE
;    0252  0              ST_MFE          = %o'5',                                ! MEDIA FORMAT ERROR
;    0253  0              ST_WPT          = %o'6',                                ! WRITE PROTECTED
;    0254  0              ST_CMP          = %o'7',                                ! COMPARE ERROR
;    0255  0              ST_DAT          = %o'10',                               ! DATA ERROR
;    0256  0              ST_HST          = %o'11',                               ! HOST BUFFER ACCESS ERROR
;    0257  0              ST_CNT          = %o'12',                               ! CONTROLLER ERROR
;    0258  0              ST_DRV          = %o'13',                               ! DRIVE ERROR
;    0259  0              ST_DIA          = %o'37',                               ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    0260  0      !
;    0261  0      !****** END MESSAGE FLAGS
;    0262  0      !
;    0263  0              EF_BBR          = %o'200',                              ! BAD BLOCK REPORTED
;    0264  0              EF_BBU          = %o'100',                              ! BAD BLOCK NOT REPORTED
;    0265  0      !
```

```
;  0266  0     !****** RDRX LITERALS
;  0267  0     !
;  0268  0             RCIP            = 0,                        ! IP REGISTER
;  0269  0             RCSA            = 1,                        ! SA REGISTER
;  0270  0     !
;  0271  0     !****** COMMON SA REGISTER BIT DEFINITIONS
;  0272  0     !
;  0273  0             SA_S1           = %o'004000',               ! STEP 1 STATUS BIT
;  0274  0             SA_S2           = %o'010000',               !    :  2
;  0275  0             SA_S3           = %o'020000',               !    :  3
;  0276  0             SA_S4           = %o'040000',               !    V  4
;  0277  0             SA_ERR          = %o'100000',               ! ERROR INDICATOR
;  0278  0             SA_INT          = %o'000200',               ! INTERRUPT ENABLE DURING INITIALIZATION
;  0279  0             SA_GO           = %o'000001',               ! GO BIT TO START FIRMWARE
;  0280  0     !
;  0281  0     !****** INITIALIZATION STEP READ MASKS
;  0282  0     !
;  0283  0             S1_MASK         = %o'176000',               ! STEP 1 READ BITS
;  0284  0             S2_MASK         = %o'174377',               !    :  2
;  0285  0             S3_MASK         = %o'174377',               !    :  3
;  0286  0             S4_MASK         = %o'174000',               !    V  4
;  0287  0     !
;  0288  0     !****** COMMAND TYPES
;  0289  0     !
;  0290  0             IMM_CMD         = 0,                        ! IMMEDIATE COMMAND
;  0291  0             SEQ_CMD         = 1,                        ! SEQUENTIAL COMMAND
;  0292  0             NON_SEQ_CMD     = 2,                        ! NON-SEQUENTIAL COMMAND
;  0293  0     !
;  0294  0     !****** ERROR-LOG FORMAT TYPES
;  0295  0     !
;  0296  0             FORMAT_CNTR     = %o'0',                    ! CONTROLLER ERROR
;  0297  0             FORMAT_HOST     = %o'1',                    ! HOST MEMORY ACCESS ERROR
;  0298  0             FORMAT_XFER     = %o'2',                    ! DISK TRANSFER ERROR
;  0299  0             FORMAT_SDI      = %o'3',                    ! 'STANDARD DISK INTECONNECT' ERROR
;  0300  0             FORMAT_SDE      = %o'4',                    ! SMALL DISK ERROR
;  0301  0     !
;  0302  0     !****** ERROR-LOG BLOCK NUMBER INFORMATION
;  0303  0     !
;  0304  0             TYPE_LBN        = %o'0000',                 ! LOGICAL BLOCK NUMBER
;  0305  0             TYPE_RBN        = %o'0110',                 ! REPLACEMENT BLOCK NUMBER
;  0306  0     !
;  0307  0     !****** MSCP DISK MODEL CODES
;  0308  0     !
;  0309  0     !       MODEL_RX50      = 7,                        ! RX50  THESE ARE NO LONGER USED. THE
;  0310  0     !       MODEL_RD51      = 6,                        ! RD51  MODEL IS DETERMINED ANOTHER WAY.
;  0311  0     !       MODEL_RD52      = 8,                        ! RD52
;  0312  0     !
;  0313  0     !****** LITERALS FOR READABILITY
;  0314  0     !
;  0315  0             YES             = 1,
;  0316  0             NO              = 0,
;  0317  0             TRUE            = 1,
;  0318  0             FALSE           = 0,
```

```
;    0319  0          SUCCESS          = 1,
;    0320  0          FAILURE          = 0,
;    0321  0          FOUND            = 1,
;    0322  0          NOT_FOUND        = 0,
;    0323  0          PRESENT          = 1,                ! DISK IS PRESENT IN CONTROLLER
;    0324  0          NOT_PRESENT      = 0,                ! DISK IS NOT PRESENT IN CONTROLLER
;    0325  0          UNPROTECTED      = 1,                ! DISK HAS UNPROTECTED CUSTOMER LBN'S
;    0326  0          PROTECTED        = 0,                ! DISK HAS PROTECTED CUSTOMER LBN'S
;    0327  0          ONLINE           = 1,
;    0328  0          OFFLINE          = 0,
;    0329  0          IDLE             = 0,                !IDLE
 !ZZZ
;    0330  0          ACTIVE           = 1,                !ACTIVE
 !ZZZ
;    0331  0          FULL             = 1,                ! ERROR-LOG SAVE PACKET FILLED
;    0332  0          EMPTY            = 0,                ! ERROR-LOG SAVE PACKET PRINTED
;    0333  0          HRD_OCCURED      = 1,                ! HARD ERROR DETECTED IN RESPONSE PACKET
;    0334  0          HRD_NOT_OCCURED  = 0,                ! HARD ERROR NOT DETECTED
;    0335  0          ALL_ONES         = %o'177777';
```

```
;  0336  0     !*****************************************************************************
;  0337  0     !                                                                            *
;  0338  0     !                          F I E L D S                                       *
;  0339  0     !                                                                            *
;  0340  0     !*****************************************************************************
;  0341  0     !
;  0342  0     FIELD
;  0343  0     !
;  0344  0     !****** HARDWARE P-TABLE FIELDS
;  0345  0     !
;  0346  0     HWP_FIELDS =
;  0347  0             set
;  0348  0             HWP_IP_ADDR     = [0, 0, 16, 0],        ! IP ADDRESS
;  0349  0             HWP_VECTOR      = [1, 0, 16, 0],        ! VECTOR ADDRESS
;  0350  0             HWP_BR_LEVEL    = [2, 0, 16, 0],        ! BUS REQUEST LEVEL
;  0351  0             HWP_DISK        = [3, 0, 16, 0],        ! DISK (ALL FIELDS)
;  0352  0             HWP_DISK_NUM    = [3, 0, 4, 0],         ! DISK NUMBER
;  0353  0             HWP_DISK_TYPE   = [3, 4, 1, 0],         ! DISK TYPE
;  0354  0             HWP_DISK_DUPEX  = [3, 5, 1, 0],         !RUN DUP EXERCISER              !ZZZ
;  0355  0             HWP_DISK_DUPWT  = [3, 6, 1, 0],         !DUP WRITE FLAG                 !ZZZ
;  0356  0             HWP_ENTIRE      = [3, 7, 1, 0],         !TEST ENTIRE DISK               !ZZZ
;  0357  0             HWP_DISK_CP     = [3, 15, 1, 0],        ! PROTECT CUSTOMER DATA BIT
;  0358  0             HWP_BEG_TRK     = [4, 0, 16, 0],        ! BEGINNING TRACK LO            !ZZZ
;  0359  0             HWP_BEG_TRK1    = [5, 0, 16, 0],        ! BEGINNING TRACK HI            !ZZZ
;  0360  0             HWP_END_TRK     = [6, 0, 16, 0],        ! ENDING TRACK LO               !ZZZ
;  0361  0             HWP_END_TRK1    = [7, 0, 16, 0]         ! ENDING TRACK HI               !ZZZ
;  0362  0             tes.
;  0363  0     !
;  0364  0     !****** COMMUNICATION AREA HEADER FIELDS
;  0365  0     !
;  0366  0     COM_FIELDS =
;  0367  0             set
;  0368  0             ADAP_CH         = [1, 8, 8, 0],         ! ADAPTER CHANNEL NUMBER FOR PURGES
;  0369  0             CMD_INT         = [2, 0, 16, 0],        ! COMMAND RING INTERRUPT
;  0370  0             RSP_INT         = [3, 0, 16, 0]         ! RESPONSE RING INTERRUPT
;  0371  0             tes,
;  0372  0
;  0373  0     !                                                                                ZZZ
;  0374  0     !       DUP BUFFER FIELD                                                         ZZZ
;  0375  0     !                                                                                ZZZ
;  0376  0     DP_FIELDS =                                                                      !ZZZ
;  0377  0             SET                                                                      !ZZZ
;  0378  0             DUPBF0          = [0, 0, 16, 0],                                         !ZZZ
;  0379  0             DUPBF1          = [1, 0, 16, 0],                                         !ZZZ
;  0380  0             DUPBF2          = [2, 0, 16, 0],                                         !ZZZ
;  0381  0             DUPTYPE         = [0, 12, 4, 0],                                         !ZZZ
;  0382  0             DUPMESG         = [0, 0, 12, 0]                                          !ZZZ
;  0383  0             TES,                                                                     !ZZZ
;  0384  0
;  0385  0
;  0386  0     !****** CONTROLLER STATUS TABLE (CST) FIELDS
;  0387  0     !
;  0388  0     CST_FIELDS =
```

```
;  0389  0          set
;  0390  0          IP_ADDR         = [0, 0, 16, 0],                ! IP ADDRESS
;  0391  0          VEC_ADDR        = [1, 0, 9, 0],                 ! VECTOR ADDRESS
;  0392  0          STATE           = [1, 15, 1, 0],                ! CONTROLLER STATUS
;  0393  0          BR_LEV          = [2, 0, 8, 0],                 ! BUS REQUEST LEVEL
;  0394  0          U_CNT           = [2, 8, 8, 0]                  ! NUMBER OF UNITS (DISKS) FOR THIS CONTROLLER
;  0395  0
;  0396  0          DO_ALL          = [3, 0, 16, 0],                ! DISK 0 (ALL FIELDS)
;  0397  0          DO_DISK_NUM     = [3, 0, 4, 0],                 ! DISK NUMBER
;  0398  0          DO_TYPE         = [3, 4, 1, 0],                 ! DISK TYPE
;  0399  0          DO_UNIT         = [3, 8, 4, 0],                 ! DISK 0 UNIT NUMBER (DRS UNIT)
;  0400  0          DO_FATAL        = [3, 12, 1, 0],                ! DISK 0 FATAL ERROR BIT
;  0401  0          DO_STAT         = [3, 13, 1, 0],                ! DISK 0 STATUS BIT
;  0402  0          DO_PRES         = [3, 14, 1, 0],                ! DISK 0 PRESENT BIT
;  0403  0          DO_PROT         = [3, 15, 1, 0],                ! DK 0 PROTECT CUSTOMER DATA
;  0404  0          DO_BEG0         = [4, 0, 16, 0],    !DK 0 BEGIN TK LO          ZZZ
;  0405  0          DO_BEG1         = [5, 0, 16, 0],    !DK 0 BEGIN TK HI          ZZZ
;  0406  0          DO_END0         = [6, 0, 16, 0],    !DK 0 END TK LO            ZZZ
;  0407  0          DO_END1         = [7, 0, 16, 0],    !DK 0 END TK HI            ZZZ
;  0408  0          DO_NAME0        = [8, 0, 8, 0],     !DK 0 NAME BYTE 0          ZZZ
;  0409  0          DO_NAME1        = [8, 8, 8, 0],     !DK 0 NAME BYTE 1          ZZZ
;  0410  0          DO_NAME2        = [9, 0, 8, 0],     !DK 0 NAME BYTE 2          ZZZ
;  0411  0          DO_NAME3        = [9, 8, 8, 0],     !DK 0 NAME BYTE 3          ZZZ
;  0412  0          DO_NUL          = [10, 0, 16, 0]    !NUL AFTER NAME            ZZZ
;  0413  0          DO_DBN          = [11, 0, 8, 0],    !DK 0 RELATIVE DBN         ZZZ
;  0414  0          DO_WRITE        = [11, 12, 1, 0],   !DK 0 DUP WRITE FLAG       ZZZ
;  0415  0          DO_ACTIVE       = [11, 13, 1, 0],   !DK 0 ACTIVE FLAG          ZZZ
;  0416  0          DO_DUPERROR     = [11, 14, 1, 0],   !DK 0 DUP ERROR FLAG       ZZZ
;  0417  0          DONODUPMED      = [11, 15, 1, 0],   !DK 0 NO DUP MEDIA FLAG    ZZZ
;  0418  0          DO_COUNT        = [12, 0, 16, 0],   !DK 0 RELATIVE MSCP FUN-   ZZZ
;  0419  0                                                    CTION COUNTER        ZZZ
;  0420  0                                                                         ZZZ
;  0421  0          ! REPEAT WORDS 3 THROUGH 12 ABOVE AS: !                        ZZZ
;  0422  0          ! WORDS 13 THROUGH 21 FOR DRIVE 1     !                        ZZZ
;  0423  0          ! WORDS 22 THROUGH 30 FOR DRIVE 2     !                        ZZZ
;  0424  0          ! WORDS 31 THROUGH 39 FOR DRIVE 3     !                        ZZZ
;  0425  0          !                                     !                        ZZZ
;  0426  0                                                                         ZZZ
;  0427  0          tes,
;  0428  0
;  0429  0     !****** MSCP PACKET FIELDS
;  0430  0     !               (NOTE: BASE ADDRESS OF PACKET REFERENCES THE PACKET'S OWN
;  0431  0     !               BUFFER DESCRIPTOR, RATHER THAN THE MESSAGE BODY (TEXT + 0).
;  0432  0     !               SEE DOCUMENTATION FOR LAYOUT OF MSCP PACKETS.)
;  0433  0     !
;  0434  0     PKT_FIELDS =
;  0435  0          set
;  0436  0     !
;  0437  0     !    HEADER FIELDS
;  0438  0     !
;  0439  0          PKT_LO          = [0, 0, 16, 0],                ! PACKET DESCRIPTOR (LO ORDER)
;  0440  0          PKT_HI          = [1, 0, 16, 0],                ! PACKET DESCRIPTOR (HI ORDER - ALL FIELDS)
;  0441  0          PKT_U           = [1, 0, 2, 0],                 ! PACKET DESCRIPTOR (HI ORDER UNIBUS BITS)
```

```
;  0442  0          PKT_Q           = [1, 2, 4, 0],                 ! PACKET DESCRIPTOR (HI ORDER Q-BUS BITS)
;  0443  0          PKT_F           = [1, 14, 1, 0],                ! PACKET DESCRIPTOR FLAG BIT
;  0444  0          PKT_O           = [1, 15, 1, 0],                ! PACKET DESCRIPTOR OWNERSHIP BIT
;  0445  0          CMD_TYPE        = [2, 0, 8, 0],                 ! COMMAND TYPE
;  0446  0          RSP_RECEIVED    = [2, 8, 8, 0],                 ! FLAG SET IF RESPONSE TO COMMAND RECEIVED
;  0447  0          MSGLEN          = [3, 0, 16, 0],                ! MESSAGE LENGTH
;  0448  0          CREDITS         = [4, 0, 4, 0],                 ! CREDITS
;  0449  0          MSGTYP          = [4, 4, 4, 0],                 ! MESSAGE TYPE
;  0450  0          CONNID          = [4, 8, 8, 0],                 ! CONNECTION ID
;  0451  0  !
;  0452  0  !       GENERIC COMMAND PACKET AND END PACKET HEADER FILEDS
;  0453  0  !
;  0454  0          CRN_LO          = [5, 0, 16, 0],                ! COMMAND REF NUMBER (LO ORDER)
;  0455  0          CRN_HI          = [6, 0, 16, 0],                ! COMMAND REF NUMBER (HI ORDER)
;  0456  0          DK_NUM          = [7, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0457  0          OPCODE          = [9, 0, 8, 0],                 ! OPCODE AND ENDCODE
;  0458  0          MODIFY          = [10, 0, 16, 0],               ! COMMAND MODIFIERS
;  0459  0          STATUS_CODE     = [10, 0, 5, 0],                ! STATUS (PART OF RESPONSE PACKET)
.  0460  0          STATUS_SUBCODE  = [10, 5, 11, 0],               ! SUBCODE (PART OF RESPONSE PACKET)
;  0461  0  !
;  0462  0  !       READ, WRITE, AND ACCESS COMMAND FIELDS (FOR COMMAND AND END PACKETS)
;  0463  0  !
;  0464  0          BC_LO           = [11, 0, 16, 0],               ! BYTE COUNT (LO ORDER)
;  0465  0          BC_HI           = [12, 0, 16, 0],               ! BYTE COUNT (HI ORDER)
;  0466  0          BUF_0           = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR
;  0467  0          BUF_1           = [14, 0, 16, 0],               !       :
;  0468  0          BUF_2           = [15, 0, 16, 0],               !       :
;  0469  0          BUF_3           = [16, 0, 16, 0],               !       :
;  0470  0          BUF_4           = [17, 0, 16, 0],               !       :
;  0471  0          BUF_5           = [18, 0, 16, 0],               !       V
;  0472  0          LBN_L           = [19, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
;  0473  0          LBN_H           = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
;  0474  0
;  0475  0  !
;  0476  0  ! DUP PROGRAM LETTER FIELDS (FOR EXECUTE LOCAL PROGRAM CMD)          ZZZ
;  0477  0  !                                                                    ZZZ
;  0478  0          L1              = [11, 0, 8, 0],        !LETTER NO 1        ZZZ
;  0479  0          L2              = [11, 8, 8, 0],        !LETTER NO 2        ZZZ
;  0480  0          L3              = [12, 0, 8, 0],        !LETTER NO 3        ZZZ
;  0481  0          L4              = [12, 8, 8, 0],        !LETTER NO 4        ZZZ
;  0482  0          L5              = [13, 0, 8, 0],        !LETTER NO 5        ZZZ
;  0483  0          L6              = [13, 8, 8, 0],        !LETTER NO 6        ZZZ
;  0484  0  !
;  0485  0  !       SET CONTROLLER CHARACTERISTICS COMMAND FIELDS
;  0486  0  !
;  0487  0          C_FLAGS         = [12, 0, 16, 0],               ! CONTROLLER FLAGS
;  0488  0  !
;  0489  0  !       ONLINE COMMAND FIELDS
;  0490  0  !
;  0491  0          U_FLAGS         = [12, 0, 16, 0],               ! UNIT FLAGS
;  0492  0          DDPAR           = [19, 0, 16, 0]                ! DEVICE-DEPENDENT PARAMETERS
;  0493  0          tes.
;  0494  0  !
```

```
;  0495  0     !****** RETURN PACKET (RETPKT) FIELDS
;  0496  0     !               (SIMILAR, BUT NOT IDENTICAL, TO MSCP PACKET FIELDS)
;  0497  0     !
;  0498  0     RP_FIELDS =
;  0499  0             set
;  0500  0     !
;  0501  0     !       COMMON TO ALL RETURN PACKETS FROM DISK MSCP
;  0502  0     !
;  0503  0             MESLEN          = [0, 0, 16, 0],                ! MESSAGE LENGTH
;  0504  0             CTLR            = [1, 0, 4, 0],                 ! CONTROLLER NUMBER (CREDITS OVERWRITTEN)
;  0505  0             MESTYP          = [1, 4, 4, 0],                 ! MESSAGE TYPE
;  0506  0             CONID           = [1, 8, 8, 0],                 ! CONNECTION ID
;  0507  0             CRF_LO          = [2, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (LO ORDER)
;  0508  0             CRF_HI          = [3, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (HI ORDER)
;  0509  0             DISK            = [4, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0510  0             CMDMOD          = [5, 0, 16, 0],                ! COMMAND MODIFIERS
;  0511  0             ENDCOD          = [6, 0, 8, 0],                 ! END CODE
;  0512  0             FLAGS           = [6, 8, 8, 0],                 ! FLAGS
;  0513  0             STATUS          = [7, 0, 16, 0],                ! STATUS AND SUB-CODE
;  0514  0             STSCOD          = [7, 0, 5, 0],                 ! STATUS CODE
;  0515  0             SUBCOD          = [7, 5, 11, 0],                ! SUB-CODE
;  0516  0     !
;  0517  0     !       READ, WRITE, AND ACCESS COMMAND RETURN PACKETS
;  0518  0     !
;  0519  0             BCNT_LO         = [8, 0, 16, 0],                ! BYTE COUNT (LO ORDER)
;  0520  0             BCNT_HI         = [9, 0, 16, 0],                ! BYTE COUNT (HI ORDER)
;  0521  0             BUFF_0          = [10, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 0)
;  0522  0             BUFF_1          = [11, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 1)
;  0523  0             BUFF_2          = [12, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 2)
;  0524  0             BUFF_3          = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 3)
;  0525  0             BUFF_4          = [14, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 4)
;  0526  0             BUFF_5          = [15, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 5)
;  0527  0             BBLK_LO         = [16, 0, 16, 0],               ! FIRST BAD BLOCK (LO ORDER)
;  0528  0             BBLK_HI         = [17, 0, 16, 0],               ! FIRST BAD BLOCK (HI ORDER)
;  0529  0             CBCNT_LO        = [18, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (LO ORDER)
;  0530  0             CBCNT_HI        = [19, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (HI ORDER)
;  0531  0             LBN_LO          = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
;  0532  0             LBN_HI          = [21, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
;  0533  0     !
;  0534  0     !       SET CONTROLLER CHARACTERISTICS RETURN PACKET
;  0535  0     !
;  0536  0             C_FLGS          = [9, 0, 16, 0],                ! CONTROLLER FLAGS
;  0537  0             C_TIME          = [10, 0, 16, 0],               ! CONTROLLER TIMEOUT
;  0538  0     !
;  0539  0     !       UNIT ONLINE RETURN PACKET
;  0540  0     !
;  0541  0             U_FLGS          = [9, 0, 16, 0],                ! UNIT FLAGS
;  0542  0             R_MODEL         = [13, 0, 8, 0],                ! 2 DIGIT MODEL NUMBER                          ZZZ
;  0543  0             NAME_NUM        = [14, 0, 6, 0],                ! MODEL NAME - 2 DIGIT NUMBER
;  0544  0             NAME_1_LO       = [14, 12, 4, 0],               ! MODEL NAME - 2ND CHARACTER (LOW  ORDER 4 BITS)
;  0545  0             NAME_1_HI       = [15, 0, 1, 0],                ! MODEL NAME - 2ND CHARACTER (HIGH ORDER 1 BIT)
;  0546  0             NAME_0          = [15, 1, 5, 0],                ! MODEL NAME - 1ST CHARACTER
;  0547  0     !ZZZ    USIZ_LO         = [18, 0, 16, 0],               ! UNIT SIZE (LO ORDER)
```

```
;  0548  0     !ZZZ    USIZ_HI         = [19, 0, 16, 0],          ! UNIT SIZE (HI ORDER)
;  0549  0             SIZE0           = [18, 0, 16, 0],          !LOWER WD OF MAX LBNS OR UNIT SIZE        ZZZ
;  0550  0             SIZE1           = [19, 0, 16, 0]           !UPPER WD     "          "       "        ZZZ
;  0551  0             tes.
;  0552  0     !
;  0553  0     !****** STATISTICS TABLE (TALLY) FIELDS
;  0554  0     !
;  0555  0     T_FIELDS =
;  0556  0             set
;  0557  0             BYTES_READ_LO   = [0, 0, 16, 0],           ! NUMBER OF BYTES READ (LO ORDER)
;  0558  0             BYTES_READ_HI   = [1, 0, 16, 0],           ! NUMBER OF BYTES READ (HI ORDER)
;  0559  0             MBYTES_READ     = [2, 0, 16, 0],           ! MEGABYTES READ
;  0560  0             BYTES_WRIT_LO   = [3, 0, 16, 0],           ! NUMBER OF BYTES WRITTEN (LO ORDER)
;  0561  0             BYTES_WRIT_HI   = [4, 0, 16, 0],           ! NUMBER OF BYTES WRITTEN (HI ORDER)
;  0562  0             MBYTES_WRIT     = [5, 0, 16, 0],           ! MEGABYTES WRITTEN
;  0563  0             ERR_HARD        = [6, 0, 16, 0],           ! NUMBER OF HARD ERRORS
;  0564  0     !
;  0565  0             TOT_READS_LO    = [7, 0, 16, 0],           ! TOTAL NUMBER OF READS (LO ORDER)
;  0566  0             TOT_READS_HI    = [8, 0, 16, 0],           ! TOTAL NUMBER OF READS (HI ORDER)
;  0567  0             TOT_WRITES_LO   = [10, 0, 16, 0],          ! TOTAL NUMBER OF WRITES (LO ORDER)
;  0568  0             TOT_WRITES_HI   = [11, 0, 16, 0],          ! TOTAL NUMBER OF WRITES (HI ORDER)
;  0569  0             TOT_BYT_RED_LO  = [13, 0, 16, 0],          ! TOTAL BYTES READ (LO ORDER)
;  0570  0             TOT_BYT_RED_HI  = [14, 0, 16, 0],          ! TOTAL BYTES READ (HI ORDER)
;  0571  0             MTOT_BYT_RED    = [15, 0, 16, 0],          ! TOTAL MEGABYTES READ
;  0572  0             TOT_BYT_WRT_LO  = [16, 0, 16, 0],          ! TOTAL BYTES WRITTEN (LO ORDER)
;  0573  0             TOT_BYT_WRT_HI  = [17, 0, 16, 0],          ! TOTAL BYTES WRITTEN (HI ORDER)
;  0574  0             MTOT_BYT_WRT    = [18, 0, 16, 0],          ! TOTAL MEGABYTES WRITTEN
;  0575  0             ERR_HRD_SEK     = [19, 0, 8, 0],           ! TOTAL HARD ERRORS - SEEK
;  0576  0             ERR_HRD_DAT     = [19, 8, 8, 0],           ! TOTAL HARD ERRORS - DATA
;  0577  0             ERR_HRD_DRV     = [20, 0, 8, 0],           ! TOTAL HARD ERRORS - DRIVE
;  0578  0             ERR_HRD_HST     = [20, 8, 8, 0],           ! TOTAL HARD ERRORS - HOST
;  0579  0             ERR_SFT_SEK     = [21, 0, 8, 0],           ! TOTAL SOFT ERRORS - SEEK
;  0580  0             ERR_SFT_DAT     = [21, 8, 8, 0],           ! TOTAL SOFT ERRORS - DATA
;  0581  0             ERR_SFT_DRV     = [22, 0, 8, 0],           ! TOTAL SOFT ERRORS - DRIVE
;  0582  0             ERR_SFT_HST     = [22, 8, 8, 0],           ! TOTAL SOFT ERRORS - HOST
;  0583  0             T_BLK_WT        = [23, 0,16, 0],    !                          ZZZ
;  0584  0             T_DBN_WT        = [24, 0,16, 0],    !DBNS WRITTEN              ZZZ
;  0585  0             T_BLK_RD        = [25, 0,16, 0],    !                          ZZZ
;  0586  0             T_DBN_RD        = [26, 0,16, 0]     !DBNS READ                 ZZZ
;  0587  0
;  0588  0             tes.
;  0589  0     !
;  0590  0     !****** CONTROLLER ERROR TALLY FIELDS
;  0591  0     !
;  0592  0     C_ERR_FIELDS =
;  0593  0             set
;  0594  0             C_ERR_HRD       = [0, 0, 8, 0],            ! HARD ERRORRS
;  0595  0             C_ERR_SFT       = [0, 8, 8, 0]             ! SOFT ERRORS
;  0596  0             tes.
;  0597  0     !
;  0598  0     !****** DRIVER CONTROLLER TABLE (DCT) FIELDS
;  0599  0     !
;  0600  0     DCT_FIELDS =
```

```
;    0601  0          set
;    0602  0          WORD0          = [0, 0, 16, 0],        ! ALL FIELDS IN WORD 0
;    0603  0          CRING_CNT      = [0, 0, 8, 0],         ! NUMBER OF SLOTS IN CRING NOT YET RETURNED TO HOST
;    0604  0          IG_INT         = [0, 14, 1, 0],        ! IGNORE INTERRUPT BIT
;    0605  0          STAT           = [0, 15, 1, 0],        ! ONLINE / OFFLINE STATUS
;    0606  0          SA_SAVE        = [1, 0, 16, 0],        ! SA REGISTER SAVE WORD
;    0607  0          RR_BEG         = [2, 0, 16, 0],        ! FIXED ADDRESSES OF START AND
;    0608  0          RR_END         = [3, 0, 16, 0],        !       END OF EACH RING
;    0609  0          CR_BEG         = [4, 0, 16, 0],        !            :
;    0610  0          CR_END         = [5, 0, 16, 0],        !            V
;    0611  0          RR_POLL        = [6, 0, 16, 0],        ! ADDR OF NEXT RRING SLOT TO BE POLLED
;    0612  0          CR_POLL        = [7, 0, 16, 0],        ! ADDR OF NEXT CRING SLOT TO BE POLLED
;    0613  0          CR_NEXT        = [8, 0, 16, 0]         ! ADDR OF NEXT AVAIL CRING SLOT
;    0614  0          tes,
;    0615  0      !
;    0616  0      !****** ERROR LOG PACKET SAVE AREA FIELDS
;    0617  0      !
;    0618  0      EP_FIELDS =
;    0619  0          set
;    0620  0          EL_CNTR        = [0, 0, 8, 0],         ! CONTROLLER NUMBER
;    0621  0          EL_CONTENTS    = [0, 8, 8, 0],         ! FLAG INDICATES IF PACKET CONTENTS ALREADY PRINTED
;    0622  0          EL_MSGLEN      = [1, 0, 16, 0],        ! PACKET LENGTH
;    0623  0          EL_CRN_LO      = [3, 0, 16, 0],        ! COMMAND REFERENCE NUMBER
;    0624  0          EL_CRN_HI      = [4, 0, 16, 0],        !
;    0625  0          EL_DK_NUM      = [5, 0, 16, 0],        ! DISK ADDRESS (RD/RX DISK NUMBER)
;    0626  0          EL_FORMAT      = [7, 0, 8, 0],         ! FORMAT
;    0627  0          EL_CONTINUE    = [7, 14, 1, 0],        ! CONTINUE FLAG
;    0628  0          EL_SUCCESS     = [7, 15, 1, 0],        ! SUCCESS FLAG
;    0629  0          EL_CODE        = [8, 0, 5, 0],         ! ERROR CODE
;    0630  0          EL_SUBCODE     = [8, 5, 11, 0],        ! SUB CODE
;    0631  0          EL_RETRY       = [20, 8, 8, 0],        ! RETRY COUNT
;    0632  0          EL_BLOCK       = [23, 0, 16, 0],       ! BLOCK NUMBER
;    0633  0          EL_BLOCK_TYPE  = [24, 12, 4, 0]        ! TYPE OF BLOCK NUMBER INFO RETURNED
;    0634  0          tes,
;    0635  0      !
;    0636  0      !****** INFORMATION ABOUT LAST RESPONSE PACKET
;    0637  0      !
;    0638  0      LAST_PKT_FIELDS =
;    0639  0          set
;    0640  0          LAST_HRD_ERR   = [0, 0, 16, 0],        ! FLAG INDICATES IF HARD ERROR OCCURED
;    0641  0          LAST_CRN_LO    = [1, 0, 16, 0],        ! COMMAND REFERENCE NUMBER
;    0642  0          LAST_CRN_HI    = [2, 0, 16, 0]         !
;    0643  0          tes,
;    0644  0      !
;    0645  0      !****** RDRX REGISTER FIELDS
;    0646  0      !
;    0647  0      RC_REG =
;    0648  0          set
;    0649  0          RC_ALL         = [0, 16, 0]            ! DEFINE ALL BITS
;    0650  0          tes;
```

```
;    0651  0     !*****************************************************************************
;    0652  0     !                                                                            *
;    0653  0     !                          M A C R O S                                       *
;    0654  0     !                                                                            *
;    0655  0     !*****************************************************************************
;    0656  0     !
;    0657  0     macro
;    0658  0     !
;    0659  0     !****** CST FIELDS. MODEL FOR WDS 3-12, 13-21, 22-30, AND 31-39.       ZZZ
;    0660  0     !
;    0661  0             D_ALL           = 0, 16, 0%,                        ! ALL FIELDS
;    0662  0             D_DISK_NUM      = 0, 4, 0%,                         ! DISK ADDRESS
;    0663  0             D_TYPE          = 4, 1, 0%,                         !DISK TYPE - 1 BIT       ZZZ
;    0664  0             D_UNIT          = 8, 4, 0%,                         ! DISK UNIT NUMBER (DRS UNIT)
;    0665  0             D_FATAL         = 12, 1, 0%,                        ! FATAL ERROR BIT
;    0666  0             D_STAT          = 13, 1, 0%,                        ! DISK STATUS BIT
;    0667  0             D_PRES          = 14, 1, 0%,                        ! DISK PRESENT BIT
;    0668  0             D_PROT          = 15, 1, 0%,                        ! DISK PROTECTION BIT
;    0669  0             D_BEGO          = 0, 16, 0%,        !BEGIN TRACK LO          ZZZ
;    0670  0             D_BEG1          = 0, 16, 0%,        !BEGIN TRACK HI          ZZZ
;    0671  0             D_ENDO          = 0, 16, 0%,        !END TRACK LO            ZZZ
;    0672  0             D_END1          = 0, 16, 0%,        !END TRACK HI            ZZZ
;    0673  0             D_NAME_0        = 0, 8, 0%,         ! NAME (FIRST  CHARACTER)
;    0674  0             D_NAME_1        = 8, 8, 0%,         ! NAME (SECOND CHARACTER)
;    0675  0             D_NAME_2        = 0, 8, 0%,         ! NAME (THIRD  CHARACTER)
;    0676  0             D_NAME_3        = 8, 8, 0%,         ! NAME (FOURTH CHARACTER)
;    0677  0     !       D_NUL           = 0, 16, 0%,        !NUL AFTER NAME          ZZZ
;    0678  0             D_DBN           = 0, 8, 0%,         !RELATIVE DBN            ZZZ
;    0679  0             DUPWRITE        = 12, 1, 0%,        !DUP WRITE FLAG          ZZZ
;    0680  0             D_ACTIVE        = 13, 1, 0%,        !ACTIVE STATE            ZZZ
;    0681  0             DUPERROR        = 14, 1, 0%,        !DUP ERROR FLAG          ZZZ
;    0682  0             NODUPMEDIA      = 15, 1, 0%,        !NO DUP MEDIA            ZZZ
;    0683  0             D_COUNT         = 0, 16, 0%,        !MSCP FUNCTION COUNTER   ZZZ
;    0684  0
;    0685  0
;    0686  0     !************ BST FIELDS *************                                   ZZZ
;    0687  0     !                                                                      ZZZ
;    0688  0             HI_WRD          = 1, 0, 16, 0%,     !HI LBN                  ZZZ
;    0689  0             LO_WRD          = 0, 0, 16, 0%,     !LO LBN                  ZZZ
;    0690  0     !
;    0691  0     !****** BIT TEST
;    0692  0     !               (CAUTION: THE FIRST ARGUMENT IS THE ADDRESS AND NOT THE CONTENTS)
;    0693  0     !
; M  0694  0     BIT_TST (ADDR, EXPECTED) =
; M  0695  0         (if (.ADDR and EXPECTED) eql EXPECTED
; M  0696  0         then
; M  0697  0             TRUE
; M  0698  0         else
;    0699  0             FALSE )%,
;    0700  0     !
;    0701  0     !****** RDRX WRITE
;    0702  0     !
; M  0703  0     WRT_RDRX (0, FIELDNAM, IMAGE) =
```

```
; M 0704  0       begin
; M 0705  0       local
; M 0706  0           RC_REG;
; M 0707  0       RC_REG <%fieldexpand (FIELDNAM)> + IMAGE;
; M 0708  0       (.RDRX_ADDR + (%upval * 0)) + .RC_REG;
;   0709  0       end%;
```

```
;  0710  0     !**************************************************************************
;  0711  0     !                                                                         *
;  0712  0     !                        S T R U C T U R E S                              *
;  0713  0     !                                                                         *
;  0714  0     !**************************************************************************
;  0715  0     !
;  0716  0     !****** NIBBLE (4-BIT) VECTOR STRUCTURE
;  0717  0     !
;  0718  0     !structure
;  0719  0     !    NIBVECTOR [I; N] =
;  0720  0     !        [(N + 1) / 2]
;  0721  0     !        (NIBVECTOR + I / 2) <(I ^ 2) and 4, 4>;
;  0722  0     !
;  0723  0     !****** RDRX ACCESS ALGORITHM
;  0724  0     !
;  0725  0     structure
;  0726  0         RDRX [O, P, S, E] =
;  0727  1             begin
;  0728  1             local
;  0729  1                 RC_REG;
;  0730  1             RC_REG = .(RDRX + %upval + 0) <0, %bpval, 0>;
;  0731  1             RC_REG
;  0732  1             end
;  0733  0             <P, S, E>;
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAGO.REQ/LIST=ZRQAGO.LIS/LIBRARY=ZRQAGO.L16/SOURCE=PAGE:53

; Run Time:          00:06.8
; Elapsed Time:      01:13.5
; Lines/CPU Min:      6458
; Lexemes/CPU-Min:  33991
; Memory Used:   72 pages
; Library Precompilation Complete
```

F2


```
;    0001  0     module ZRQAM1 (
;    0002  0
.    0003  0     %title 'RD/RX EXERCISER'
;    0004  0                     ident = 'V02.2',
;    0005  0                     addressing_mode (absolute),
;    0006  0                     environment (noeis)
;    0007  0                     ) =
;    0008  0
;    0009  1     begin
;    0010  1
;    0011  1
; C  0012  1     %(
; C  0013  1                                     IDENTIFICATION
; C  0014  1                                     ---------- ---
; C  0015  1
; C  0016  1             PRODUCT CODE:           AC-T398G-MC
; C  0017  1
; C  0018  1             PRODUCT NAME:           CZRQAG0 RQDX/RUX50 EXERCISER
; C  0019  1
; C  0020  1             PRODUCT DATE:           04-APR-85
; C  0021  1
; C  0022  1             MAINTAINER:             DIAGNOSTIC ENGINEERING
; C  0023  1
; C  0024  1             AUTHOR:                 RAVINDER K. KARWAN
; C  0025  1                                     BOB POWERS
; C  0026  1
; C  0027  1
; C  0028  1             Copyright (C) 1983, 1984, 1985
; C  0029  1
; C  0030  1             Digital Equipment Corporation, Maynard, Massachusetts  01754
; C  0031  1
; C  0032  1             This software is furnished  under a license for use only on a single
; C  0033  1             computer  system and  may be  copied only with  the inclusion of the
; C  0034  1             above copyright notice.  This software, or any other copies thereof,
; C  0035  1             may not be provided or  otherwise made available to any other person
; C  0036  1             except for use on such system and to one who agrees to these license
; C  0037  1             terms.  Title to and  ownership of the  software  shall at all times
; C  0038  1             remain in DEC.
; C  0039  1
; C  0040  1             the information in this document is subject to change without notice
; C  0041  1             and should  not be construed  as a commitment  by Digital  Equipment
; C  0042  1             Corporation.
; C  0043  1
; C  0044  1             DEC assumes  no  responsibility  for  the use or  reliability of its
; C  0045  1             software on equipment which is not supplied by DEC.
; C  0046  1
; C  0047  1
; C  0048  1             The following are trademarks of Digital Equipment Corporation:
; C  0049  1
; C  0050  1             DIGITAL         PDP             UNIBUS          MASSBUS
; C  0051  1             DEC             DECUS           DECTAPE
```

```
;  C 0052  1
;  C 0053  1
;  C 0054  1          REVISION HISTORY:
;  C 0055  1
;  C 0056  1          REV 1.6    11-APR-84   MERGED FIELD AND MANUFACTURING VERSIONS OF THE RD/RX EXERCISER.
;  C 0057  1                                 ADDED SUPPORT FOR THE RUX50.
;  C 0058  1
;  C 0059  1          REV 1.7    01 MAY-84   ADDED CODE TO GET DEVICE TYPE FROM CONTROLLER CHARACTERISTICS;
;  C 0060  1                                 ADDED APT BREAKS IN UNIT_INIT ROUTINE; CORRECTED SOFT SEEK ERROR
;  C 0061  1                                 TOTALS; PROTECT MEDIA ON DEFAULT.
;  C 0062  1
;  C 0063  1          REV 1.8    06-JUL-84   ELIMINATE GETTING DISK TYPE FROM ID BLOCK ON A RESTART;
;  C 0064  1
;  C 0065  1          REV 1.9    19-SEP-84   ON END OF PASS, WAIT UNTIL LAST PACKET RETURNED BEFORE WRITING IP.

;  C 0066  1
;  C 0067  1          REV 2.0    09-NOV-84   DON'T OUTPUT DUP STATS HEADER IF NO WINCHESTER.
;  C 0068  1                                 FIXED 'CMD REF NO. NOT SENT BY HOST' PROBLEM BY USING OPERATOR-
;  C 0069  1                                 SPECIFIED BR LEVEL WHEN SENDING PACKETS.
;  C 0070  1
;  C 0071  1          REV 2.1    27-DEC-84   ADDED APT MODE QUESTION; ADDED RETRIES TO DUP TESTS. ADDED CODE TO
;  C 0072  1                                 INT_GEN ROUTINE TO MAKE IT COMPATIBLE WITH MICROCODE VERSION 9.1.
;  C 0073  1
;  C 0074  1          REV 2.2    04-APR-85   IN POLL_RRING AND POLL_CRING ROUTINES, ZERO INTERRUPT COUNTERS ON
;  C 0075  1                                 ENTRY TO PREVENT SLOWDOWN PROBLEM. ADDED 32-BIT LBN ADDRESSING.
;  C 0076  1                                 CHANGED HEADER MACRO AND .REQ FILE FOR COMPATIBILITY WITH XXDP V2.
;  C 0077  1                                 MADE DUP COMMAND TYPE = 0.
```

```
;  C 0078  1
;  C 0079  1                    TABLE OF CONTENTS
;  C 0080  1
;  C 0081  1
;  C 0082  1
;  C 0083  1          1.0     GENERAL INFORMATION
;  C 0084  1          1.1     PROGRAM ABSTRACT
;  C 0085  1          1.2     SYSTEM REQUIREMENTS
;  C 0086  1          1.2.1   HARDWARE REQUIREMENTS
;  C 0087  1          1.2.2   SOFTWARE REQUIREMENTS
;  C 0088  1          1.3     RELATED DOCUMENTS AND STANDARDS
;  C 0089  1          1.4     DIAGNOSTIC HIERARCHY PREREQUISITES
;  C 0090  1          1.5     ASSUMPTIONS
;  C 0091  1          1.6     MEMORY MAP
;  C 0092  1
;  C 0093  1          2.0     OPERATING INSTRUCTIONS
;  C 0094  1          2.1     HARDWARE QUESTIONS
;  C 0095  1          2.2     SOFTWARE QUESTIONS
;  C 0096  1
;  C 0097  1          3.0     ERROR TYPES
;  C 0098  1          3.1     ERROR INFORMATION
;  C 0099  1          3.2     INITIALIZATION ERRORS
;  C 0100  1          3.3     EXERCISER ERRORS
;  C 0101  1          3.4     ERROR LOG MESSAGES
;  C 0102  1          3.5     MSCP ERRORS
;  C 0103  1          3.6     SAMPLE ERROR STATEMENT
;  C 0104  1
;  C 0105  1          4.0     PERFORMANCE AND PROGRESS REPORTS
;  C 0106  1
;  C 0107  1          5.0     TEST SUMMARY
;  C 0108  1          5.1     INITIALIZATION SUBTEST
;  C 0109  1          5.2     EXERCISER
;  C 0110  1          5.3     DROP UNIT SUMMARY
;  C 0111  1
;  C 0112  1          6.0     ERROR CODES
;  C 0113  1
;  C 0114  1          7.0     DATA PATTERNS
```

```
; C 0115  1
; C 0116  1
; C 0117  1
; C 0118  1
; C 0119  1
; C 0120  1
; C 0121  1
; C 0122  1
; C 0123  1
; C 0124  1
; C 0125  1
; C 0126  1
; C 0127  1
; C 0128  1
; C 0129  1
; C 0130  1
; C 0131  1
; C 0132  1
; C 0133  1
; C 0134  1
; C 0135  1
; C 0136  1
; C 0137  1
; C 0138  1
; C 0139  1
; C 0140  1
; C 0141  1
; C 0142  1
; C 0143  1
; C 0144  1
; C 0145  1
; C 0146  1
; C 0147  1
; C 0148  1
; C 0149  1
; C 0150  1
; C 0151  1
; C 0152  1
; C 0153  1
; C 0154  1
; C 0155  1
; C 0156  1
; C 0157  1
; C 0158  1
; C 0159  1
; C 0160  1
; C 0161  1
; C 0162  1
; C 0163  1
; C 0164  1
; C 0165  1
```

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

This program will functionally verify and exercise RQDX or RUX50 Controller/Disk Drive subsystems. It is designed to verify that the subsystem is functioning correctly and operating within design specifications.

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE REQUIREMENTS

LSI - 11/23 processor with 28K or more of memory, console device (eg. VT100) and RQDX or RUX50 controller board and attached RD51 or RD52 WINCHESTER drive(s) and RX-50 FLOPPY drive(s)

1.2.2 SOFTWARE REQUIREMENTS

This diagnostic is designed to run with the Diagnostic Supervisor as described in paragraph 2.0.

1.3 RELATED DOCUMENTS AND STANDARDS

XXDP+ SUPERVISOR/USERS MANUAL CHQUS
UQSSP UNIBUS/Q-BUS STORAGE SYSTEMS PORT
MSCP MASS STORAGE SYSTEM PROTOCOL

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

NONE

1.5 ASSUMPTIONS

The hardware, other than the subsystem being tested, is assumed to work properly. False errors may be reported if the processor, memory, etc., do not function properly.

```
; C 0166  1          1.6     MEMORY MAP
; C 0167  1          - - --
; C 0168  1
; C 0169  1
; C 0170  1          Memory layout on 28k machine   XXDP environment
; C 0171  1          -- ----  ---  - ------------------ -------
; C 0172  1
; C 0173  1                                                ADDRESS
; C 0174  1
; C 0175  1                    +-------------------+       160000
; C 0176  1                    |                   |
; C 0177  1                    |    XXDP+ MONITOR  |
; C 0178  1                    |                   |
; C 0179  1                    +-- -- ------ ------+
; C 0180  1                    |                   |
; C 0181  1                    | XXDP+  TTY I/O AREA |
; C 0182  1                    |                   |
; C 0183  1                    +-- ---------------- -+
; C 0184  1                    |                   |
; C 0185  1                    |                   |
; C 0186  1                    |       DRS         |
; C 0187  1                    |                   |
; C 0188  1                    |                   |
; C 0189  1                    +-------------------+       121000
; C 0190  1                    |                   |
; C 0191  1                    |    EXERCISER      |
; C 0192  1                    |                   |
; C 0193  1                    +-------------------- -+    2000
; C 0194  1                    |                   |
; C 0195  1                    |      STACKS       |
; C 0196  1                    |                   |
; C 0197  1                    +-----------------------+   400
; C 0198  1                    |                   |
; C 0199  1                    |    VECTOR AREA    |
; C 0200  1                    |                   |
; C 0201  1                    +-------------------+       0
; C 0202  1
; C 0203  1
; C 0204  1          In a machine  with more memory,  free space will occur
; C 0205  1          between the exerciser and the DRS.
```

```
; C 0206  1
; C 0207  1
; C 0208  1
; C 0209  1
; C 0210  1
; C 0211  1
; C 0212  1
; C 0213  1
; C 0214  1
; C 0215  1
; C 0216  1
; C 0217  1
; C 0218  1
; C 0219  1
; C 0220  1
; C 0221  1
; C 0222  1
; C 0223  1
; C 0224  1
; C 0225  1
; C 0226  1
; C 0227  1
; C 0228  1
; C 0229  1
; C 0230  1
; C 0231  1
; C 0232  1
; C 0233  1
; C 0234  1
; C 0235  1
; C 0236  1
; C 0237  1
; C 0238  1
; C 0239  1
; C 0240  1
; C 0241  1
; C 0242  1
; C 0243  1
; C 0244  1
; C 0245  1
; C 0246  1
; C 0247  1
; C 0248  1
; C 0249  1
; C 0250  1
; C 0251  1
; C 0252  1
; C 0253  1
; C 0254  1
```

2.0 OPERATING INSTRUCTIONS

This is a Rev C Supervisor Diagnostic: for operating instructions, please see chapter 5 of XXDP+ operator's manual. They are no longer included in the diagnostic because it is desired that a change in those instructions not require a re-assembly of all Supervisor Diagnostics.

2.1 HARDWARE QUESTIONS

The following series of questions collect the parameters necessary to identify each disk subsystem.

Hardware Configuration Questions

The program will ask the following questions in response to a START command (non-script).

1. CHANGE HW (L) Y ?

   Answer NO to use the pre-built answers for all hardware questions. This program will be released pre-built to test three units with default answers shown below. The pre-built answers may be changed at any time with the setup utility. Answer YES if you want all the hardware questions to be asked.

2. NUMBER OF UNITS (D) ?

   No default. Answer with the number of disk drive units to be exercised or tested. This answer will determine how many times the following questions are asked. A range of 1 to 4 units may be specified. A unit number will be assigned sequentially from 0 by the Diagnostic supervisor for each unit.

3. IP ADDRESS (O) 172150 ?

   Enter the address of the IP register of one RQDX or RUX50 as addressed by the processor with memory management turned off. The program expects an even 16-bit address in the range of 160000 to 177774. 172150 is the default.

```
; C 0255  1
; C 0256  1
; C 0257  1
; C 0258  1
; C 0259  1
; C 0260  1
; C 0261  1
; C 0262  1
; C 0263  1
; C 0264  1
; C 0265  1
; C 0266  1
; C 0267  1
; C 0268  1
; C 0269  1
; C 0270  1
; C 0271  1
; C 0272  1
; C 0273  1
; C 0274  1
; C 0275  1
; C 0276  1
; C 0277  1
; C 0278  1
; C 0279  1
; C 0280  1
; C 0281  1
; C 0282  1
; C 0283  1
; C 0284  1
; C 0285  1
; C 0286  1
; C 0287  1
; C 0288  1
; C 0289  1
; C 0290  1
; C 0291  1
; C 0292  1
; C 0293  1
; C 0294  1
; C 0295  1
; C 0296  1
; C 0297  1
; C 0298  1
; C 0299  1
; C 0300  1
; C 0301  1
; C 0302  1
; C 0303  1
; C 0304  1
; C 0305  1
; C 0306  1
; C 0307  1
```

4. VECTOR ADDRESS (O) 154 ?

Answer with the interrupt vector ofthe same RQDX or RUX50 controller describea in the above question. A vector address in the range of 4 to 774 may be specified. 154 is the default.

5. BR LEVEL [USUALLY 4-RQDX 5-RUX50] (D) 4 ?

Answer with the bus request interrupt level used by the above controller. Levels 4 through 7 are acceptable. 4 is the default.

6. DRIVE NUMBER (D) 0 ?

Enter the logical unit number for one drive associated with the IP address above. Drive numbers are in the range of 0 through 15. The number entered here must match the unit plug on the front panel of the drive, and must be within the range implied by the jumper (LUN0-7) on the RQDX or RUX50 controller board. 0 is the default answer.

7. ALSO RUN DUP EXERCISER (L) N ?

ANSWER Y TO HAVE TESTS PERFORMED SPECIFICALLY WITH THE DIAGNOSTIC BLOCKS. SUCH DUP TESTING, IF SELECTED, IS INTERLEAVED WITH NORMAL EXERCISER TESTING.

8. WRITE ON DIAGNOSTIC AREA (L) N ?

IF THE DUP EXERCISER IS CHOSEN TO BE RUN, ANSWERING Y TO THIS QUESTION ADDS WRITE TESTING IN THE DIAGNOSTIC BLOCK AREA. THIS CAN BE USED TO DETERMINE WHETHER A UNIT IS WRITING PROPERLY, WITHOUT USING THE CUSTOMER AREA.

9. TEST ENTIRE CUSTOMER DATA AREA OF THIS DISK (L) Y?

This question is asked to give the opportunity of limiting the addressing range over which the testing will be performed. An affirmative answer will cause no limits to be imposed for the unit in question. A negative answer will cause limits to be imposed, as defined by the following four questions.

10. LOWER OCTAL WORD OF BEGINNING LBN ADDRESS (O) 0?

Enter ir octal the less significant 16-bit word of the lowest

```
;C 0308  1          LBN address in the test range. The value may be from 000000
;C 0309  1          to 177777.
;C 0310  1
;C 0311  1
;C 0312  1        11.  HIGHER OCTAL WORD OF BEGINNING LBN ADDRESS (0) 0?
;C 0313  1
;C 0314  1          Enter in octal the more significant 16-bit word of the
;C 0315  1          lowest LBN address in the test range.
;C 0316  1
;C 0317  1
;C 0318  1        12.  LOWER OCTAL WORD OF ENDING LBN ADDRESS (0) 150477?
;C 0319  1
;C 0320  1          Enter in octal the less significant 16-bit word of the
;C 0321  1          highest LBN address in the test range. 150477 is the
;C 0322  1          highest LBN address for an RD52.
;C 0323  1
;C 0324  1
;C 0325  1        13.  HIGHER OCTAL WORD OF ENDING LBN ADDRESS (0) 0?
;C 0326  1
;C 0327  1          Enter in octal the more significant 16 bit word of the
;C 0328  1          highest LBN address in the test range.
;C 0329  1
;C 0330  1
;C 0331  1           Note:
;C 0332  1                The four previous questions are usually software
;C 0333  1                Parameter questions, but since three different disk
;C 0334  1                drives exist on the subsystem, this becomes a  unit
;C 0335  1                by unit question.  It is possible to specify an LBN
;C 0336  1                which is too large since we are dealing with different
;C 0337  1                drives. The program will check for block number bounds,
;C 0338  1                and, if they are exceeded, will assign the maximum
;C 0339  1                bounds for that drive.
;C 0340  1
;C 0341  1
;C 0342  1
;C 0343  1      14.  WRITE ON CUSTOMER DATA AREA ON THIS DISK UNIT (L) ?
;C 0344  1
;C 0345  1          Answering YES will destroy any customer data that is  on
;C 0346  1          the  disk; therefore, the following warning message will
;C 0347  1          appear, followed by a confirmation prompt:
;C 0348  1
;C 0349  1          ** WARNING - CUSTOMER DATA AREA WILL BE OVERWRITTEN! ...
;C 0350  1           CONFIRM (L) ?
;C 0351  1
;C 0352  1          This question will default to NO if the operator has de
;C 0353  1          cided  to  bypass  the  hardware  questions.  Otherwise,
;C 0354  1          there is no default.
```

```
; C 0355  1
; C 0356  1
; C 0357  1
; C 0358  1
; C 0359  1
; C 0360  1
; C 0361  1
; C 0362  1
; C 0363  1
; C 0364  1
; C 0365  1
; C 0366  1
; C 0367  1
; C 0368  1
; C 0369  1
; C 0370  1
; C 0371  1
; C 0372  1
; C 0373  1
; C 0374  1
; C 0375  1
; C 0376  1
; C 0377  1
; C 0378  1
; C 0379  1
; C 0380  1
; C 0381  1
; C 0382  1
; C 0383  1
; C 0384  1
; C 0385  1
; C 0386  1
; C 0387  1
; C 0388  1
; C 0389  1
; C 0390  1
; C 0391  1
; C 0392  1
; C 0393  1
; C 0394  1
; C 0395  1
; C 0396  1
; C 0397  1
; C 0398  1
; C 0399  1
; C 0400  1
```

2.2 SOFTWARE QUESTIONS

Software Parameter Questions

The program will ask the following questions in response to the START, RESTART, and CONTINUE commands.

1. CHANGE SW (L) Y ?

   Answer NO to bypass the following questions in this section. This question should normally be answered NO when the Exerciser is first run. A NO answer will cause the Exerciser to select the default parameters shown with each question below. Then, depending on the errors detected, it may be desirable to change this answer to YES to alter the test parameters and further isolate the problem.

2. ENTER TIME AS HHMM (EXAMPLE: 1305) (D) 0 ?

   Enter the time of day (in 24 hour format). DRS does not ALLOW leading zeros ENTERED FOR numeric values.
   For example, for 14 minutes past midnight, you would enter 14, and for 30 minutes past 3 in the afternoon, enter 1530.

3. HARD ERROR LIMIT (D) 32 ?

   Enter the number of hard errors allowed before a unit is dropped from testing. A number in the range of 1 to 65535 will be accepted.

4. TRANSFER LIMIT IN MEGABYTES (0 FOR QUICK PASS) (D) 0 ?

   When the specified number of bytes have been transferred to/from a unit, the unit will be dropped from testing. When all units are dropped, an end-of-pass will be indicated. This is the method used to determine how long the Exerciser is to run.

   The only other way the Exerciser will declare end-of-pass is if all units are dropped because the error limit on each is exceeded. However, the operator can always abort the program at any time by typing CONTROL-C.

```
; C 0401  1
; C 0402  1
; C 0403  1
; C 0404  1
; C 0405  1
; C 0406  1
; C 0407  1
; C 0408  1
; C 0409  1
; C 0410  1
; C 0411  1
; C 0412  1
; C 0413  1
; C 0414  1
; C 0415  1
; C 0416  1
; C 0417  1
; C 0418  1
; C 0419  1
; C 0420  1
; C 0421  1
; C 0422  1
; C 0423  1
; C 0424  1
; C 0425  1
; C 0426  1
; C 0427  1
; C 0428  1
; C 0429  1
; C 0430  1
; C 0431  1
; C 0432  1
; C 0433  1
; C 0434  1
; C 0435  1
; C 0436  1
; C 0437  1
; C 0438  1
; C 0439  1
; C 0440  1
; C 0441  1
; C 0442  1
; C 0443  1
; C 0444  1
; C 0445  1
; C 0446  1
; C 0447  1
; C 0448  1
; C 0449  1
; C 0450  1
; C 0451  1
```

5. PERCENTAGE OF 'FIXED DISK' OPERATIONS OUT OF TOTAL OPERATIONS (D) 99 ?

In order to maintain typical usage for the devices of this exercise, a certain percentage of operations must be directed to the RD51/52s (the rest go to the RX50s). It turns out that this percentage is very high (as indicated by the 99% figure given as the default). It may be desirable in some cases to direct more activity to the RX50s. This is easily done by directing a smaller percentage of the operations to the RD51/52s. The numbers associated with usage are adjusted internally by the program according to drive type and percentage.

6. CLEAR STATISTICAL TABLES AFTER PRINTING (L) N ?

Answering YES causes the statistical fields to be cleared to zero after the report is printed (either at end of pass, or at operator request). Otherwise, cumulative totals are maintained.

7. REWRITE BLOCKS WHEN "FORCED ERROR" DETECTED ON READS (L) Y ?

On encountering a bad block on the RD51 or RD52 disk (during either a read or a write operation), the RQDX or RUX50 controller will revector the logical block to another physical location on the disk. This operation is transparent to the user. However, if the revectoring was done subsequent to a write operation (i.e the write operation detected the bad block), the data is flagged with a "Forced Error" code, signifying that the data at the revectored location is suspect. The controller returns an error code whenever the block is re-read. Answer 'Yes' to the question to force a WRITE operation on the same block whenever a "Forced Error" flag is detected on a read. This is to avoid the same error code (the "Forced Error") being reported for the same block repeatedly. The re-write will, however, take place only if writes are enabled for the particular disk unit.

8. HALT ON BAD-BLOCK HARD ERRORS (#s 35, 38) (L) Y ?

When the Exerciser is run with the DRS "Halt on Error" switch set (eg. START/FLAGS:HOE), the Exerciser halts on on encountering ANY error. If it is desired that the testing continue on a bad-block error, even with the HOE switch set, answer No to the question.

9. HALT ON OTHER HARD ERRORS (#s 31-34, 36-37, 39-45) (L) Y ?

This question is similar to question 8, but refers to non-bad block type of Hard Errors.

```
; C 0452  1          10.  HALT ON SOFT ERRORS (#s 50-54) (L) N ?
; C 0453  1
; C 0454  1               This question  is similar to  question 8,  but refers to
; C 0455  1               Soft Errors.
; C 0456  1
; C 0457  1          11.  COUNT EACH RETRY AS A SEPARATE SOFT ERROR (L) N ?
; C 0458  1
; C 0459  1               On encountering any error on a read/write,  the control-
; C 0460  1               ler  retries the operation  a number  of times.  If  the
; C 0461  1               operation is eventually successful,  this is reported as
; C 0462  1               a Soft Error.  The error log packet contains  the number
; C 0463  1               of  retries performed  before the operation was success-
; C 0464  1               ful.  Normally, the whole sequence of retries is classi-
; C 0465  1               fied as one Soft Error. Answer Yes to the question if it
; C 0466  1               is  desired to  count each internal  retry attempt  as a
; C 0467  1               separate Soft Error.
; C 0468  1
; C 0469  1          12.  RANDOM SEEK MODE (L) Y ?
; C 0470  1
; C 0471  1               Answer YES to cause block numbers to be chosen randomly.
; C 0472  1               Answer  NO to cause block numbers to be selected sequen-
; C 0473  1               tially.
; C 0474  1
; C 0475  1          13.  UNITS TO BE SELECTED AT RANDOM (NO, IMPLIES SEQUENTIAL) (L) N ?
; C 0476  1
; C 0477  1               This  question is  optionally asked if the answer to the
; C 0478  1               previous  question is N[o].  The  selection of units for
; C 0479  1               sequential  operations is affected by the answer to this
; C 0480  1               question.  If the default  answer is chosen (N[o]), then
; C 0481  1               units  shall be  selected in a  predetermined manner  in
; C 0482  1               accordance  with the typical  seek time margins for each
; C 0483  1               drive.  If the alternate answer is chosen (Y[es]),  then
; C 0484  1               the  units will be chosen  at random  in accordance with
; C 0485  1               the percentages specified in Software question 4.
; C 0486  1
; C 0487  1          14.  READ-COMPARES PERFORMED AT THE CONTROLLER (L) Y ?
; C 0488  1
; C 0489  1               Answering YES causes all read commands  to  include  the
; C 0490  1               "compare"  modifier.   This  essentially forces the con-
; C 0491  1               troller to perform two read operations on the same  disk
; C 0492  1               address, and to compare the results.
; C 0493  1
; C 0494  1               The following message will appear after the operator has
; C 0495  1               answered this question:
; C 0496  1
; C 0497  1
; C 0498  1          15.  RUNNING UNDER THE A.P.T. MONITOR (L) N ?
; C 0499  1
; C 0500  1               THIS QUESTION SHOULD BE ANSWERED N (DEFAULT) IN THE FIELD.
; C 0501  1               IT ENABLES THE PROGRAM TO KNOW THAT IT IS RUNNING UNDER A
; C 0502  1               SPECIAL (AUTOMATED PRODUCT TEST) MONITOR.
```

```
; C 0503  1
; C 0504  1
; C 0505  1
; C 0506  1
; C 0507  1
; C 0508  1
; C 0509  1
; C 0510  1
; C 0511  1
; C 0512  1
; C 0513  1
; C 0514  1
; C 0515  1
; C 0516  1
; C 0517  1
; C 0518  1
; C 0519  1
; C 0520  1
; C 0521  1
; C 0522  1
; C 0523  1
; C 0524  1
; C 0525  1
; C 0526  1
; C 0527  1
; C 0528  1
; C 0529  1
; C 0530  1
; C 0531  1
; C 0532  1
; C 0533  1
; C 0534  1
; C 0535  1
; C 0536  1
; C 0537  1
; C 0538  1
; C 0539  1
; C 0540  1
; C 0541  1
; C 0542  1
; C 0543  1
; C 0544  1
; C 0545  1
; C 0546  1
; C 0547  1
; C 0548  1
; C 0549  1
; C 0550  1
; C 0551  1
```

THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED DISK UNITS.

16. WRITE-COMPARES PERFORMED AT THE CONTROLLER (L) N ?

Answering YES causes all write I/O requests to be changed to write-compare. After each write, the controller will read the data and compare it to data re-obtained from the host.

17. CHECK ALL WRITES AT HOST BY READING (L) Y ?

This question will only be asked if the previous question was answered NO. Answering YES causes all writes to be checked by the host by reading the data immediately after the write operation. This option consumes extra CPU time, and doubles the amount of storage required for writes. Therefore, it is only recommended when drive write-compare operations are suspect.

18. USER-DEFINED DATA PATTERN (L) N ?

An answer of YES allows the operator to define his/her own data pattern to be used in all write operations. A NO answer will allow the operator to select a pre-defined data pattern in the next question.

19. SELECT PRE-DEFINED DATA PATTERN (0 FOR SEQUENTIAL SELECTION) (D) 0 ?

There are 21 pre-defined data patterns available, selected as 1 to 21 (see section 4.9). A zero answer will cause patterns 1 to 21 to be sequentially selected for each write. (Note that pattern 1 consists entirely of random numbers).

20. NUMBER OF WORDS IN DATA PATTERN (16 MAXIMUM) (D) 16 ?
PATTERN VALUES (O) ?

These questions will only be asked if the operator has decided to define his/her own data pattern. The actual bit patterns will be entered as octal (PDP-11).

```
; C 0552  1
; C 0553  1
; C 0554  1
; C 0555  1
; C 0556  1
; C 0557  1
; C 0558  1
; C 0559  1
; C 0560  1
; C 0561  1
; C 0562  1
; C 0563  1
; C 0564  1
; C 0565  1
; C 0566  1
; C 0567  1
; C 0568  1
; C 0569  1
; C 0570  1
; C 0571  1
; C 0572  1
; C 0573  1
; C 0574  1
; C 0575  1
; C 0576  1
; C 0577  1
; C 0578  1
; C 0579  1
; C 0580  1
; C 0581  1
; C 0582  1
; C 0583  1
; C 0584  1
; C 0585  1
; C 0586  1
; C 0587  1
; C 0588  1
; C 0589  1
; C 0590  1
; C 0591  1
; C 0592  1
; C 0593  1
; C 0594  1
; C 0595  1
; C 0596  1
; C 0597  1
; C 0598  1
```

3.0 ERROR TYPES

This program has four types of error classifications; system fatal, drive fatal, hard and soft.

SYSTEM FATAL ERRORS

System fatal errors are used to indicate that an error was detected by the Diagnostic Supervisor in relation to loading/controlling the diagnostic process.

The content of each error is such that it should be self explanatory. However, the messages utilize some terms that are specific to the disk subsystem, and may require some getting use to.

DRIVE FATAL ERRORS

Drive fatal errors are a result of:

an error that is considered fatal to the drive, but testing will continue.

HARD ERRORS

Hard errors are a result of:

1. retries of a soft error or *
2. a non-recoverable error
3. a soft error if retries are not set.

* Note: Retries are executed in the controller

SOFT ERRORS

Soft errors are media related errors. All soft errors will be retried by the controller.

Note: Soft errors are retrieved from the controller via the error log capabilities of MSCP.

```
; C 0599  1
; C 0600  1
; C 0601  1
; C 0602  1
; C 0603  1
; C 0604  1
; C 0605  1
; C 0606  1
; C 0607  1
; C 0608  1
; C 0609  1
; C 0610  1
; C 0611  1
; C 0612  1
; C 0613  1
; C 0614  1
; C 0615  1
; C 0616  1
; C 0617  1
; C 0618  1
; C 0619  1
; C 0620  1
; C 0621  1
; C 0622  1
; C 0623  1
; C 0624  1
; C 0625  1
; C 0626  1
; C 0627  1
; C 0628  1
; C 0629  1
; C 0630  1
; C 0631  1
; C 0632  1
; C 0633  1
; C 0634  1
; C 0635  1
; C 0636  1
; C 0637  1
; C 0638  1
; C 0639  1
; C 0640  1
; C 0641  1
; C 0642  1
; C 0643  1
; C 0644  1
; C 0645  1
; C 0646  1
; C 0647  1
```

3.1 ERROR INFORMATION

All general error messages will include the type of error (system-fatal, drive-fatal, hard, soft) and a unit number. If the error applies to a controller, then only the first unit number of the controller will be given. (The user will know the other unit numbers when subsequent "drop unit" messages are printed).

Basic error messages provide more details about the error. The Exerciser will print all basic error messages, along with the disk address, if applicable. In some cases where a drive-fatal error applies to a controller, the controller's IP address will be printed.

Extended error messages will be used to print the relevant fields of command and end message packets, status codes, SA register contents, and error log messages. All values will be in octal (PDP-11).

The error messages in this section do not include errors detected and printed by the Diagnostic Supervisor.

3.2 INITIALIZATION ERRORS

Two kinds of errors will be reported to the operator during the Initialization Test. The System-fatal error is too many units specified. A system-fatal error will cause the Exerciser to abort.

Drive-fatal errors only affect the unit(s) involved. Testing will continue on all other units. This class of errors includes, but is not limited to, the following:

1. Register Existence Test failure (no drive present)
2. Vector Test failure
3. BR Level Test failure
4. Initialization sequence failure
5. Online failed
6. Access failed

3.3 EXERCISER ERRORS

Most errors reported during this test will originate from MSCP end message packets. The status code field will be converted to text and printed as part of a basic error message. Any subcode value will follow if extended error messages are enabled.

The following list represents some of the error conditions reported via MSCP:

1. Disk unit went offline (a sub-code may follow detailing the reason)
2. Compare error
3. Data error (a sub-code may follow)
4. Drive error (a sub-code may follow)
5. Host buffer access error
6. Media format error (a sub-code may follow)

3.4 ERROR LOG MESSAGES

The contents of the error-log messages received from the controller are printed as received, and should be deciphered using the MSCP specs.

3.5 MSCP ERRORS

An MSCP error occurs when the host receives an Invalid Command End Message from the controller. In such cases, the host will print out the erroneous command followed by the reason for the error. If extended printouts are enabled, then the entire contents of the end message will be displayed in octal without interpretation of the data.

3.6 SAMPLE ERROR STATEMENT

The errors listed by the exerciser are usually very descriptive and are self explanatory. The following is an example error statement. This error statement is the extended error message.

```
(example)                                (comments)

DISK XXX                                 !DISK UNIT NUMBER
INVALID COMMAND                          !MAJOR STATUS CODE RECEIVED BACK
SUB-CODE XXXX                            !SUB-CODE OF GIVEN COMMAND
COMMAND: READ                            !COMMAND GIVEN TO DRIVE
LBN: XXXXX                               !LOGICAL BLOCK NUMBER GIVEN
BYTE COUNT IN COMMAND XXXXX              !NUMBER OF BYTES WANTED TO READ
ACTUAL # OF BYTES TRANSFERRED XXXXX !NUMBER OF BYTES ACTUALLY READ
```

The status code in an end messages is broken into two pieces. The first 5 bits represent the major status which is given by the "invalid command" message. The 11 remaining bits represent the sub-code, which tells in greater detail the error in the controller. The LBN is the logical block on the disk the controller was trying to read. The byte count refers to the number of bytes the controller was going to read off the LBN. The actual number of bytes transferred refers to the number of bytes read before the error.

```
; C 0726  1
; C 0727  1
; C 0728  1
; C 0729  1
; C 0730  1
; C 0731  1
; C 0732  1
; C 0733  1
; C 0734  1
; C 0735  1
; C 0736  1
; C 0737  1
; C 0738  1
; C 0739  1
; C 0740  1
; C 0741  1
; C 0742  1
; C 0743  1
; C 0744  1
; C 0745  1
; C 0746  1
; C 0747  1
; C 0748  1
; C 0749  1
; C 0750  1
; C 0751  1
; C 0752  1
; C 0753  1
; C 0754  1
; C 0755  1
; C 0756  1
; C 0757  1
; C 0758  1
; C 0759  1
```

4.0 PERFORMANCE AND PROGRESS REPORTS

A summary report is printed at the end of each pass of the Exerciser or upon demand by the operator. The fields may be cleared to zero after the report is printed depending on the operator's response to this option in the software questions. Any units added to the test cycle will also begin with cleared statistics.

Errors are grouped into two basic categories: hard and soft. Each is sub divided into four more categories, depending on the most probable classification for that error.

The sub categories are:

1. disk related errors
2. seek (or format) related errors
3. controller or drive related errors
4. host (the CPU) related errors.

All numeric values are in decimal radix.

UNT #	TYPE	# OF READS	BYTS READ	# OF WRITES	BYTES WRITTEN	-- HRD ERS -- DAT	HRD SEK	HRD DRV	HRD HST	-- SFT ERS -- DAT	SFT SEK	SFT DRV	SFT HST
X	XXXX	XXXX	XXXX	XXXXX	XXXXXX	X	X	X	X	X	X	X	X
.	..	..	..	...	....	.	.	.	.	.	.	.	.
.	..	..	..	...	....	.	.	.	.	.	.	.	.

```
; C 0760  1
; C 0761  1
; C 0762  1
; C 0763  1
; C 0764  1
; C 0765  1
; C 0766  1
; C 0767  1
; C 0768  1
; C 0769  1
; C 0770  1
; C 0771  1
; C 0772  1
; C 0773  1
; C 0774  1
; C 0775  1
; C 0776  1
; C 0777  1
; C 0778  1
; C 0779  1
; C 0780  1
; C 0781  1
; C 0782  1
; C 0783  1
; C 0784  1
; C 0785  1
; C 0786  1
; C 0787  1
; C 0788  1
; C 0789  1
; C 0790  1
; C 0791  1
; C 0792  1
; C 0793  1
; C 0794  1
; C 0795  1
; C 0796  1
; C 0797  1
; C 0798  1
; C 0799  1
; C 0800  1
; C 0801  1
; C 0802  1
; C 0803  1
; C 0804  1
; C 0805  1
; C 0806  1
; C 0807  1
; C 0808  1
; C 0809  1
```

5.0 TEST SUMMARY

This exerciser consists of two parts: the initialization subtest, and the performance exerciser. The operator is not able to select which of these two parts he/she wishes to run; they both must be executed.

5.1 INITIALIZATION SUBTEST

The purpose of this subtest is to verify the hardware configuration as specified by the operator, and to bring each unit online. The Initialization Subtest will always precede the execution of any other test.

First, the presence of each drive register will be verified, along with a check on the BR level specified by the operator. Then, an initialization will be issued to each controller configured for testing. When the initialization sequence has been completed, an attempt will be made to bring each unit online. If this succeeds, one or two MSCP reads will be issued to the inner-most LBN of each selected disk to to ensure that each disk drive can seek and be read.

Any drive-fatal or hard errors encountered during this test will cause the appropriate unit(s) to be dropped. If basic error messages are enabled, then the program will print out the specific reason for dropping the unit(s). Henceforth, the failed unit(s) will not be tested unless the operator intervenes (adds unit(s) or restarts Exerciser).

Upon successful completion of the Initialization Subtest, the program will begin executing the Exerciser.

5.2 EXERCISER

The purpose of this subtest is to exercise the disk drives in a manner similar to the typical usage under standard operating systems. Execution of this test should give an indication of the operating performance of the disk drive subunits. This test will utilize random disk addresses, random word counts, and data patterns, all subject to the limits and specifications made by the operator. All protected disks will be subject to read-only operations, while unprotected disks may be read or written, depending on the answers given to the software parameter questions. End-of-pass will be declared when the specified number of bytes have been transferred for all the disks taken as a whole.

```
; C 0810  1          If a read/write error occure during this test,  then the
; C 0811  1          controller will initiate an  appropriate number of re-
; C 0812  1          tries.  If all retries fail, then a hard error will be repor-
; C 0813  1          ted to the host,  an error message  will  be displayed on the
; C 0814  1          console  terminal and  the error will be tallied for the sum-
; C 0815  1          mary report. The unit will be dropped if the hard error count
; C 0816  1          has exceeded the specified limit.
; C 0817  1
; C 0818  1
; C 0819  1
; C 0820  1     5.3  DROP UNIT SUMMARY
; C 0821  1          ----------------
; C 0822  1          During the Initialization Subtest, individual units will
; C 0823  1          be dropped from  the test  sequence if they are unable  to be
; C 0824  1          brought online or the operator specified drive does not match
; C 0825  1          the hardware.
; C 0826  1
; C 0827  1          During  the Exercise,  the program will  drop a unit for
; C 0828  1          one of  three reasons.  The normal path is for  each unit  to
; C 0829  1          complete the transfer of N megabytes, where N is specified by
; C 0830  1          the  operator  during SW  questioning  and be  soft-dropped.
; C 0831  1          Otherwise,  a unit will be hard-dropped if the number of hard
; C 0832  1          errors encountered exceeds  the operator-specified limit,  or
; C 0833  1          if a fatal error is detected. Units hard-dropped may later be
; C 0834  1          added to the  test cycle.  However,  statistics for the hard-
; C 0835  1          added  unit  will be  cleared to  zero;  if a transfer  limit
; C 0836  1          was specified,  in which case the unit was soft-dropped,  the
; C 0837  1          statistics may  or may not be cleared  depending on the oper-
; C 0838  1          ators answer to Software question 12.
```

```
; C 0839  1
; C 0840  1
; C 0841  1
; C 0842  1
; C 0843  1
; C 0844  1
; C 0845  1
; C 0846  1
; C 0847  1
; C 0848  1
; C 0849  1
; C 0850  1
; C 0851  1
; C 0852  1
; C 0853  1
; C 0854  1
; C 0855  1
; C 0856  1
; C 0857  1
; C 0858  1
; C 0859  1
; C 0860  1
; C 0861  1
; C 0862  1
; C 0863  1
; C 0864  1
; C 0865  1
; C 0866  1
; C 0867  1
; C 0868  1
; C 0869  1
; C 0870  1
; C 0871  1
; C 0872  1
; C 0873  1
; C 0874  1
; C 0875  1
; C 0876  1
; C 0877  1
; C 0878  1
; C 0879  1
; C 0880  1
; C 0881  1
; C 0882  1
; C 0883  1
; C 0884  1
; C 0885  1
; C 0886  1
; C 0887  1
; C 0888  1
```

6.0 ERROR CODES

This section describes the error codes generated by this exerciser.

SYSTEM FATAL ERRORS

1	More then 4 units specified	

DRIVE FATAL ERRORS

10	Controller couldn't be addressed at the address given.	Wrong IP address selected
11	Controller didn't interrupt at the interrupt vector given.	Wrong vector address selected.
12	Controller didn't interrupt at the BR level given.	Wrong BR level selected.
13	Init sequence failed.	Either one of the four initialization steps did not receive the correct response from the Controller, or one of the steps timed-out.
14	Fatal Controller error.	The error bit (bit 15) in the SA register was set.
15	Failed to bring unit on-line.	On-line response had an error code. (see also #s 22 and 23.)
16	Write protect conflict.	The unit was hardware write protected and write operations were requested on the unit.
17	Access to either the inner or the outer track failed.	Innermost or outermost track's header may be corrupted.
18	Unit went off line.	
19	Drive type not known.	The version of the Exerciser being run does not support this disk type.

20	Failed to send 'Set Controller Characteristics' command.	Either the unit is off line or the Diagnostic is corrupted because of any problems with its RAM.
21	Controller returned wrong 'end code' for the 'Set Controller Characteristics' command.	Problem with the Controller microcode or the port/DMA interface.
22	Failed to send 'On line' command	Either the unit is off line or the diagnostic is corrupted because of any problems with its RAM.
23	Controller returned wrong 'end code' for the 'On line' command.	Problem with the Controller's microcode or the port/DMA interface.
24	Drive went to the Available' state.	

HARD ERRORS

31	Controller received an invalid command.	The diagnostic is corrupted because of any problems with its RAM, or there is a problem with the Controller microcode (RAM or ROM) or there is problem with the port/DMA interface.
32	Command aborted by the Controller.	Command timed out in the Controller.
35	Media format error.	
36	Drive write protected.	
37	Controller read or write compare error.	
38	Data error.	CRC error in the data field of a disk block.
39	Host buffer access error	
40	Controller error.	Difficult to categorize without looking at the error sub-code or any associated error log message.

```
; C 0942  1
; C 0943  1
; C 0944  1          41  Drive error.                        See 040.
; C 0945  1
; C 0946  1          42  Host write compare error.           Error detected when Host
; C 0947  1                                                  CPU compared the data
; C 0948  1                                                  written and read back. May
; C 0949  1                                                  be a problem with the Host
; C 0950  1                                                  or Controller RAM.
; C 0951  1
; C 0952  1          43 Message from internal diagnostics    See 040.
; C 0953  1
; C 0954  1          44  Duplicate unit number detected      ...
; C 0955  1              by the Controller.
; C 0956  1
; C 0957  1          45  Unknown end code received.         Problem with the Control-
; C 0958  1                                                  ler microcode or the port/
; C 0959  1                                                  DMA interface.
; C 0960  1
; C 0961  1
; C 0962  1
; C 0963  1
; C 0964  1          SOFT ERRORS
; C 0965  1          ----------
; C 0966  1
; C 0967  1          50  Controller error.                  See error-log packet for
; C 0968  1                                                  details as the exact cause
; C 0969  1                                                  may not be evident.
; C 0970  1
; C 0971  1          51  Host memory access error.          See 050.
; C 0972  1
; C 0973  1          52  Disk transfer error.               See 050.
; C 0974  1
; C 0975  1          53  'Standard   Disk  Interconnect'    See 050.
; C 0976  1              error.
; C 0977  1
; C 0978  1          54  'Small Disk' error.                See 050.
; C 0979  1
; C 0980  1
; C 0981  1
; C 0982  1
; C 0983  1          DUP ERRORS
; C 0984  1          ----------
; C 0985  1
; C 0986  1          60 Unable to load local controller DUP media.
; C 0987  1
; C 0988  1          61 (Not used)
; C 0989  1
; C 0990  1          62 Illegal unit number.
; C 0991  1
; C 0992  1          63 Illegal relative or physical block.
; C 0993  1
; C 0994  1          64 Device error.
```

```
;  C 0995  1
;  C 0996  1                65 Zero length message.
;  C 0997  1
;  C 0998  1                66 Unknown DUP status code.
;  C 0999  1
;  C 1000  1                67 Invalid command.
```

```
; C 1001  1
; C 1002  1
; C 1003  1
; C 1004  1
; C 1005  1
; C 1006  1
; C 1007  1
; C 1008  1
; C 1009  1
; C 1010  1
; C 1011  1
; C 1012  1
; C 1013  1
; C 1014  1
; C 1015  1
; C 1016  1
; C 1017  1
; C 1018  1
; C 1019  1
; C 1020  1
```

DUP ERRORS (CONTINUED)
-- --------

68 No region available.

69 No region suitable.

70 Program not known.

71 Load failure.

72 Standalone.

73 Unknown DUP status code.

```
;  C 1021  1              7.0  DATA PATTERNS
;  C 1022  1                   -- ---- ---
;  C 1023  1
;  C 1024  1
;  C 1025  1                          HEX      OCTAL                BINARY
;  C 1026  1                          ---      -----                ------
;  C 1027  1
;  C 1028  1          Pattern 1             R A N D O M   N U M B E R S
;  C 1029  1
;  C 1030  1          Pattern 2       0000     000000         0 000 000 000 000 000
;  C 1031  1
;  C 1032  1          Pattern 3       FFFF     177777         1 111 111 111 111 111
;  C 1033  1
;  C 1034  1          Pattern 4       8888     105613         1 000 101 110 001 011
;  C 1035  1
;  C 1036  1          Pattern 5       3333     031463         0 011 001 100 110 011
;  C 1037  1
;  C 1038  1          Pattern 6       3091     030221         0 011 000 010 010 001
;  C 1039  1
;  C 1040  1          Pattern 7       0001     000001         0 000 000 000 000 001
;  C 1041  1                          0003     000003         0 000 000 000 000 011
;  C 1042  1                          0007     000007         0 000 000 000 000 111
;  C 1043  1                          000F     000017         0 000 000 000 001 111
;  C 1044  1                          001F     000037         0 000 000 000 011 111
;  C 1045  1                          003F     000077         0 000 000 000 111 111
;  C 1046  1                          007F     000177         0 000 000 001 111 111
;  C 1047  1                          00FF     000377         0 000 000 011 111 111
;  C 1048  1                          01FF     000777         0 000 000 111 111 111
;  C 1049  1                          03FF     001777         0 000 001 111 111 111
;  C 1050  1                          07FF     003777         0 000 011 111 111 111
;  C 1051  1                          0FFF     007777         0 000 111 111 111 111
;  C 1052  1                          1FFF     017777         0 001 111 111 111 111
;  C 1053  1                          3FFF     037777         0 011 111 111 111 111
;  C 1054  1                          7FFF     077777         0 111 111 111 111 111
;  C 1055  1                          FFFF     177777         1 111 111 111 111 111
;  C 1056  1
;  C 1057  1          Pattern 8       FFFE     177776         1 111 111 111 111 110
;  C 1058  1                          FFFC     177774         1 111 111 111 111 100
;  C 1059  1                          FFF8     177770         1 111 111 111 111 000
;  C 1060  1                          FFF0     177760         1 111 111 111 110 000
;  C 1061  1                          FFE0     177740         1 111 111 111 100 000
;  C 1062  1                          FFC0     177700         1 111 111 111 000 000
;  C 1063  1                          FF80     177600         1 111 111 110 000 000
;  C 1064  1                          FF00     177400         1 111 111 100 000 000
;  C 1065  1                          FE00     177000         1 111 111 000 000 000
;  C 1066  1                          FC00     176000         1 111 110 000 000 000
;  C 1067  1                          F800     174000         1 111 100 000 000 000
;  C 1068  1                          F000     170000         1 111 000 000 000 000
;  C 1069  1                          E000     160000         1 110 000 000 000 000
;  C 1070  1                          C000     140000         1 100 000 000 000 000
;  C 1071  1                          8000     100000         1 000 000 000 000 000
;  C 1072  1                          0000     000000         0 000 000 000 000 000
```

```
; C 1073  1          Pattern 9       0000    000000      0 000 000 000 000 000
; C 1074  1                          0000    000000      0 000 000 000 000 000
; C 1075  1                          0000    000000      0 000 000 000 000 000
; C 1076  1                          FFFF    177777      1 111 111 111 111 111
; C 1077  1                          FFFF    177777      1 111 111 111 111 111
; C 1078  1                          FFFF    177777      1 111 111 111 111 111
; C 1079  1                          0000    000000      0 000 000 000 000 000
; C 1080  1                          0000    000000      0 000 000 000 000 000
; C 1081  1                          FFFF    177777      1 111 111 111 111 111
; C 1082  1                          FFFF    177777      1 111 111 111 111 111
; C 1083  1                          0000    000000      0 000 000 000 000 000
; C 1084  1                          FFFF    177777      1 111 111 111 111 111
; C 1085  1                          0000    000000      0 000 000 000 000 000
; C 1086  1                          FFFF    177777      1 111 111 111 111 111
; C 1087  1                          0000    000000      0 000 000 000 000 000
; C 1088  1                          FFFF    177777      1 111 111 111 111 111
; C 1089  1
; C 1090  1          Pattern 10      B6D9    133331      1 011 011 011 011 001
; C 1091  1
; C 1092  1          Pattern 11      5555    052525      0 101 010 101 010 101
; C 1093  1                          5555    052525      0 101 010 101 010 101
; C 1094  1                          5555    052525      0 101 010 101 010 101
; C 1095  1                          AAAA    125252      1 010 101 010 101 010
; C 1096  1                          AAAA    125252      1 010 101 010 101 010
; C 1097  1                          AAAA    125252      1 010 101 010 101 010
; C 1098  1                          5555    052525      0 101 010 101 010 101
; C 1099  1                          5555    052525      0 101 010 101 010 101
; C 1100  1                          AAAA    125252      1 010 101 010 101 010
; C 1101  1                          AAAA    125252      1 010 101 010 101 010
; C 1102  1                          5555    052525      0 101 010 101 010 101
; C 1103  1                          AAAA    125252      1 010 101 010 101 010
; C 1104  1                          5555    052525      0 101 010 101 010 101
; C 1105  1                          AAAA    125252      1 010 101 010 101 010
; C 1106  1                          5555    052525      0 101 010 101 010 101
; C 1107  1                          AAAA    125252      1 010 101 010 101 010
```

```
; C 1108  1          Pattern 12      2D2D    026455      0 010 110 100 101 101
; C 1109  1                          2D2D    026455      0 010 110 100 101 101
; C 1110  1                          2D2D    026455      0 010 110 100 101 101
; C 1111  1                          D2D2    151322      1 101 001 011 010 010
; C 1112  1                          D2D2    151322      1 101 001 011 010 010
; C 1113  1                          D2D2    151322      1 101 001 011 010 010
; C 1114  1                          2D2D    026455      0 010 110 100 101 101
; C 1115  1                          2D2D    026455      0 010 110 100 101 101
; C 1116  1                          D2D2    151322      1 101 001 011 010 010
; C 1117  1                          D2D2    151322      1 101 001 011 010 010
; C 1118  1                          2D2D    026455      0 010 110 100 101 101
; C 1119  1                          2D2D    026455      0 010 110 100 101 101
; C 1120  1                          D2D2    151322      1 101 001 011 010 010
; C 1121  1                          2D2D    026455      0 010 110 100 101 101
; C 1122  1                          D2D2    151322      1 101 001 011 010 010
; C 1123  1                          2D2D    026455      0 010 110 100 101 101
; C 1124  1                          D2D2    151322      1 101 001 011 010 010
; C 1125  1                          2D2D    026455      0 010 110 100 101 101
; C 1126  1                          D2D2    151322      1 101 001 011 010 010
; C 1127  1                          2D2D    026455      0 010 110 100 101 101
; C 1128  1
; C 1129  1          Pattern 13      6DB6    066666      0 110 110 110 110 110
; C 1130  1
; C 1131  1          Pattern 14      0001    000001      0 000 000 000 000 001
; C 1132  1                          0002    000002      0 000 000 000 000 010
; C 1133  1                          0004    000004      0 000 000 000 000 100
; C 1134  1                          0008    000010      0 000 000 000 001 000
; C 1135  1                          0010    000020      0 000 000 000 010 000
; C 1136  1                          0020    000040      0 000 000 000 100 000
; C 1137  1                          0040    000100      0 000 000 001 000 000
; C 1138  1                          0080    000200      0 000 000 010 000 000
; C 1139  1                          0100    000400      0 000 000 100 000 000
; C 1140  1                          0200    001000      0 000 001 000 000 000
; C 1141  1                          0400    002000      0 000 010 000 000 000
; C 1142  1                          0800    004000      0 000 100 000 000 000
; C 1143  1                          1000    010000      0 001 000 000 000 000
; C 1144  1                          2000    020000      0 010 000 000 000 000
; C 1145  1                          4000    040000      0 100 000 000 000 000
; C 1146  1                          8000    100000      1 000 000 000 000 000
```

```
; C 1147  1          Pattern 15      FFFE    177776      1 111 111 111 111 110
; C 1148  1                          FFFD    177775      1 111 111 111 111 101
; C 1149  1                          FFFB    177773      1 111 111 111 111 011
; C 1150  1                          FFF7    177767      1 111 111 111 110 111
; C 1151  1                          FFEF    177757      1 111 111 111 101 111
; C 1152  1                          FFDF    177737      1 111 111 111 011 111
; C 1153  1                          FFBF    177677      1 111 111 110 111 111
; C 1154  1                          FF7F    177577      1 111 111 101 111 111
; C 1155  1                          FEFF    177377      1 111 111 011 111 111
; C 1156  1                          FDFF    176777      1 111 110 111 111 111
; C 1157  1                          FBFF    175777      1 111 101 111 111 111
; C 1158  1                          F7FF    173777      1 111 011 111 111 111
; C 1159  1                          EFFF    167777      1 110 111 111 111 111
; C 1160  1                          DFFF    157777      1 101 111 111 111 111
; C 1161  1                          BFFF    137777      1 011 111 111 111 111
; C 1162  1                          7FFF    077777      0 111 111 111 111 111
; C 1163  1
; C 1164  1          Pattern 16      B6D9    133331      1 011 011 011 011 001
; C 1165  1                          B6D9    133331      1 011 011 011 011 001
; C 1166  1                          B6D9    133331      1 011 011 011 011 001
; C 1167  1                          DB6C    155554      1 101 101 101 101 100
; C 1168  1                          DB6C    155554      1 101 101 101 101 100
; C 1169  1                          DB6C    155554      1 101 101 101 101 100
; C 1170  1                          B6D9    133331      1 011 011 011 011 001
; C 1171  1                          B6D9    133331      1 011 011 011 011 001
; C 1172  1                          DB6C    155554      1 101 101 101 101 100
; C 1173  1                          DB6C    155554      1 101 101 101 101 100
; C 1174  1                          B6D9    133331      1 011 011 011 011 001
; C 1175  1                          DB6C    155554      1 101 101 101 101 100
; C 1176  1                          B6D9    133331      1 011 011 011 011 001
; C 1177  1                          DB6C    155554      1 101 101 101 101 100
; C 1178  1                          B6D9    133331      1 011 011 011 011 001
; C 1179  1                          DB6C    155554      1 101 101 101 101 100
```

```
;C 1180  1          Pattern 17      (LBN)*   (LBN)          (LBN)
;C 1181  1                          8D36     106466   1 000 110 100 110 110
;C 1182  1                          8D36     106466   1 000 110 100 110 110
;C 1183  1                          72C9     071311   0 111 001 011 001 001
;C 1184  1                          72C9     071311   0 111 001 011 001 001
;C 1185  1                          72C9     071311   0 111 001 011 001 001
;C 1186  1                          8D36     106466   1 000 110 100 110 110
;C 1187  1                          8D36     106466   1 000 110 100 110 110
;C 1188  1                          8D36     106466   1 000 110 100 110 110
;C 1189  1                          8D36     106466   1 000 110 100 110 110
;C 1190  1                          72C9     071311   0 111 001 011 001 001
;C 1191  1                          72C9     071311   0 111 001 011 001 001
;C 1192  1                          72C9     071311   0 111 001 011 001 001
;C 1193  1                          72C9     071311   0 111 001 011 001 001
;C 1194  1                          72C9     071311   0 111 001 011 001 001
;C 1195  1                          8D36     106466   1 000 110 100 110 110
;C 1196  1                          8D36     106466   1 000 110 100 110 110
;C 1197  1                          8D36     106466   1 000 110 100 110 110
;C 1198  1                          8D36     106466   1 000 110 100 110 110
;C 1199  1                          8D36     106466   1 000 110 100 110 110
;C 1200  1                          8D36     106466   1 000 110 100 110 110
;C 1201  1
;C 1202  1              * This word position contains the number of the locical
;C 1203  1                block to be written.
;C 1204  1
;C 1205  1
;C 1206  1
;C 1207  1          Pattern 18      8D36     106466   1 000 110 100 110 110
;C 1208  1                          (LBN)    (LBN)          (LBN)
;C 1209  1                          72C9     071311   0 111 001 011 001 001
;C 1210  1                          8D36     106466   1 000 110 100 110 110
;C 1211  1                          8D36     106466   1 000 110 100 110 110
;C 1212  1                          8D36     106466   1 000 110 100 110 110
;C 1213  1                          72C9     071311   0 111 001 011 001 001
;C 1214  1                          72C9     071311   0 111 001 011 001 001
;C 1215  1                          72C9     071311   0 111 001 011 001 001
;C 1216  1                          72C9     071311   0 111 001 011 001 001
;C 1217  1                          8D36     106466   1 000 110 100 110 110
;C 1218  1                          8D36     106466   1 000 110 100 110 110
;C 1219  1                          8D36     106466   1 000 110 100 110 110
;C 1220  1                          8D36     106466   1 000 110 100 110 110
;C 1221  1                          8D36     106466   1 000 110 100 110 110
;C 1222  1                          72C9     071311   0 111 001 011 001 001
;C 1223  1                          72C9     071311   0 111 001 011 001 001
;C 1224  1                          72C9     071311   0 111 001 011 001 001
;C 1225  1                          72C9     071311   0 111 001 011 001 001
;C 1226  1                          72C9     071311   0 111 001 011 001 001
;C 1227  1                          72C9     071311   0 111 001 011 001 001
```

```
;  C 1228  1          Pattern 19      (LBN)     (LBN)          (LBN)
;  C 1229  1                          8999      134631   1 011 100 110 011 001
;  C 1230  1                          8999      134631   1 011 100 110 011 001
;  C 1231  1                          4666      043146   0 100 011 001 100 110
;  C 1232  1                          4666      043146   0 100 011 001 100 ı10
;  C 1233  1                          4666      043146   0 100 011 001 100 110
;  C 1234  1                          8999      134631   1 011 100 110 011 001
;  C 1235  1                          8999      134631   1 011 100 110 011 001
;  C 1236  1                          8999      134631   1 011 100 110 011 001
;  C 1237  1                          8999      134631   1 011 100 110 011 001
;  C 1238  1                          4666      043146   0 100 011 001 100 110
;  C 1239  1                          4666      043146   0 100 011 001 100 110
;  C 1240  1                          4666      043146   0 100 011 001 100 110
;  C 1241  1                          4666      043146   0 100 011 001 100 110
;  C 1242  1                          4666      043146   0 100 011 001 100 110
;  C 1243  1                          8999      134631   1 011 100 110 011 001
;  C 1244  1                          8999      134631   1 011 100 110 011 001
;  C 1245  1                          8999      134631   1 011 100 110 011 001
;  C 1246  1                          8999      134631   1 011 100 110 011 001
;  C 1247  1                          8999      134631   1 011 100 110 011 001
;  C 1248  1                          8999      134631   1 011 100 110 011 001
;  C 1249  1
;  C 1250  1          Pattern 20      8999      134631   1 011 100 110 011 001
;  C 1251  1                          (LBN)     (LBN)          (LBN)
;  C 1252  1                          4666      043146   0 100 011 001 100 110
;  C 1253  1                          8999      134631   1 011 100 110 011 001
;  C 1254  1                          8999      134631   1 011 100 110 011 001
;  C 1255  1                          8999      134631   1 011 100 110 011 001
;  C 1256  1                          4666      043146   0 100 011 001 100 110
;  C 1257  1                          4666      043146   0 100 011 001 100 110
;  C 1258  1                          4666      043146   0 100 011 001 100 110
;  C 1259  1                          4666      043146   0 100 011 001 100 110
;  C 1260  1                          8999      134631   1 011 100 110 011 001
;  C 1261  1                          8999      134631   1 011 100 110 011 001
;  C 1262  1                          8999      134631   1 011 100 110 011 001
;  C 1263  1                          8999      134631   1 011 100 110 011 001
;  C 1264  1                          8999      134631   1 011 100 110 011 001
;  C 1265  1                          4666      043146   0 100 011 001 100 110
;  C 1266  1                          4666      043146   0 100 011 001 100 110
;  C 1267  1                          4666      043146   0 100 011 001 100 110
;  C 1268  1                          4666      043146   0 100 011 001 100 110
;  C 1269  1                          4666      043146   0 1C0 011 001 100 110
;  C 1270  1                          4666      043146   0 100 011 001 100 110
;  C 1271  1
;  C 1272  1          Pattern 21      (LBN)     (LBN)          (LBN)
;  C 1273  1
;     1274  1      )%
```

```
ZRQAM1          RD/RX EXERCISER                                 4-Apr-1985 12:40:26    VAX-11 Bliss-16 V4.1 582              SEQ 0048
V02.2           PROGRAM HEADER                                  4-Apr-1985 12:33:21    DISK$USER2:[POWERS.ZRQ]ZRQAG0.BL1;16  Page  31
                                                                                                                              (29)

;    1275  1    %sbttl 'PROGRAM HEADER
;    1276  1
;    1277  1    library 'ZRQAG0.L16';                           ! RDRX EXERCISER GLOBAL LIBRARY
;    1278  1
;    1279  1    !ZZZ require 'BLSMAC.REQ';                      ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    1280  1    require 'HSAXA0.BLB';                           ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    3021  1
;    3022  1    literal
;    3023  1        DS$NBR_OF_TESTS = 1;                        ! NUMBER OF TESTS IN THIS DIAGNOSTIC
;    3024  1
;    3025  1    EQUALS;
;    3026  1
;    3027  1    POINTER (ALL);
;    3028  1
;    3029  1    !+
;    3030  1    ! THE PROGRAM HEADER IS THE INTERFACE BETWEEN
;    3031  1    ! THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
;    3032  1    !-
;    3033  1
;    3034  1    !ZZZ HEADER (%ascii'ZRQA', %ascii'G', %ascii'0', 32000, 1, PRI00);      !ZZZ NEED POSITIVE NUMBER
;    3035  1    HEADER (%ascii'ZRQA', %ascii'G', %ascii'0', 32000, 1, PRI00,1);         !ZZZ FINAL 1 = NO TESTING ON TRAPS (SAVE TI
ME)
;    3036  1
```

```
;    3037  1      %sbttl 'DISPATCH TABLE'
;    3038  1
;    3039  1      !+
;    3040  1      ! THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
;    3041  1      ! IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
;    3042  1      !-
;    3043  1
;    3044  1      DISPATCH (DS$NBR_OF_TESTS);
```

```
;    3045  1     %sbttl  GLOBAL DATA SECTION'
;    3046  1
;    3047  1     !+
;    3048  1     ! THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
;    3049  1     ! IN MORE THAN ONE TEST.
;    3050  1     !-
;    3051  1
;    3052  1     psect
;    3053  1         global = $FFF$ (read, write, noexecute, global, concatenate);
;    3054  1
;    3055  1     global
;    3056  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    3057  1                         ! RUN-TIME CONTROLLER STATUS TABLES
;    3058  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    3059  1                         ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    3060  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    3061  1                         ! DRIVER CONTROLLER TABLES
;    3062  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    3063  1                         ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    3064  1         RDRX_ADDR : ref rdrx field (RC_REG),
;    3065  1                         ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    3066  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;    3067  1                         ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    3068  1
;    3069  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                          !ZZZ
;    3070  1                         !CONTAINS LO. HI LBN FIELDS FOR SEQUENTIAL       !ZZZ
;    3071  1                         !I/O TRANSFER FOR EACH UNIT.                     !ZZZ
;    3072  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    3073  1                         ! STATISTICS TABLES
;    3074  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    3075  1                         ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    3076  1
;    3077  1         DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;    3078  1                         !INFO FROM RECEIVE * SEND CMDS                   ZZZ
;    3079  1         TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED] INITIAL (BYTE (REP    !ZZZ
;    3080  1                 MAX_UNITS OF (1))),         !CURRENT TRACK DIRECTION     ZZZ
;    3081  1         RDM_CNT : WORD INITIAL (RDM_LEN),   !NO OF RANDOM NOS   \\KEEP   ZZZ
;    3082  1         RANDOM : VECTOR [RDM_LEN, WORD],    !RANDOM NO. TABLE   //TOGETHER  ZZZ
;    3083  1
;    3084  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    3085  1                         ! STATISTICS TABLE FOR CONTROLLER ERRORS
;    3086  1         MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;    3087  1                         ! MSCP PACKET POOL
;    3088  1         IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    3089  1                         ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;    3090  1         PKT_USE : vector [PKT_CNT, byte, signed],
;    3091  1                         ! MSCP PACKET POOL ALLOCATION TABLE
;    3092  1         RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;    3093  1                         ! RETURN PACKET POOL
;    3094  1         RP_USE : vector [RP_CNT, byte, signed],
;    3095  1                         ! RETURN PACKET POOL ALLOCATION TABLE
;    3096  1         RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
;    3097  1         RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
```

```
;    3098  1                                  ! CURRENT RETURN PACKET  DDRESS
;    3099  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EF_FIELDS),
;    3100  1                                  ! ERROR-LOG PACKET SAVE AREA
;    3101  1          BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
;    3102  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;    3103  1          IODQ : vector [IODQ_LEN, byte],                      ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;    3104  1          IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
;    3105  1          IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
;    3106  1          ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
;    3107  1          EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
;    3108  1          DUP_FLAGS : WORD,                                    !DUP FLAGS       ZZZ
;    3109  1          CCTLR : word,                                        ! NUMBER OF "CURRENT" CONTROLLER
;    3110  1          CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    3111  1          CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
;    3112  1          CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;    3113  1          DUR : vector [MAX_UNITS, byte],                      ! DROP UNIT REASON
;    3114  1          QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;    3115  1          FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
;    3116  1          BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
;    3117  1          ST_CODE : word,                                      ! CURRENT STATUS CODE
;    3118  1          SB_CODE : word,                                      ! CURRENT SUB-CODE
;    3119  1          STEP : word,                                         ! CURRENT STEP IN HARD_INIT
;    3120  1          OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
;    3121  1          SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
;    3122  1          CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
;    3123  1          NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
;    3124  1          CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
;    3125  1          CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
;    3126  1          TEMP1 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;    3127  1          TEMP2 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;    3128  1          CREDIT_BAL : word,                                   ! CREDIT BALANCE
;    3129  1          NEXT_PKT_USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;    3130  1          HOURS : byte,                                        ! TIME OF DAY (HOURS)
;    3131  1          MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
;    3132  1          CLK_TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
;    3133  1          FERO_LBN : word,                                     !LO LBN ADR OF THE "FORCED ERROR" BLOCK          ZZZ
;    3134  1          FER1_LBN : word,                                     !HI LBN ADR OF THE "FORCED ERROR" BLOCK          ZZZ
;    3135  1          CLK_PRESENT : byte,                                  ! FLAG INDICATES IF LINE-CLOCK PRESENT
;    3136  1          HOE_FLAG : byte,                                     ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;    3137  1
;    3138  1          S_PATTERN : WORD,                  !PATTERN FOR DUP WRITES          ZZZ
;    3139  1          S_DUPPKT : WORD,                   !DBN BYTE COUNTER                ZZZ
;    3140  1          P_INDEX : SIGNED WORD,             !CURRENT MESSAGE PACKET INDEX    ZZZ
;    3141  1          RD_COUNT : WORD INITIAL (0),       ! NUMBER OF WINCHESTER UNITS     ZZZ
;    3142  1          BRLEVEL : WORD,                    !BUS REQUEST LEVEL FROM OPERATOR ZZZ
;    3143  1          D_FAIL : BYTE,                     !SIGNIFIES DUP TYPE ERROR        ZZZ
;    3144  1          FORCED_ERROR : byte,                                 ! "FORCED ERROR" DETECTED IN LAST READ
;    3145  1          FER_LBN : word,                                      ! LBN OF THE "FORCED ERROR" BLOCK
;    3146  1          FER_BC : word,                                       ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    3147  1          INIT_OCCURED : byte initial (byte (FALSE)),          ! EXERCISER INITIALIZATION COMPLETE
;    3148  1          ADDR_VECT_OK : byte initial (byte (FALSE));          ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    3149  1
;    3150  1      ERRTBL;
```

```
;    3151  1      %sbttl 'GLOBAL TEXT SECTION'
;    3152  1
;    3153  1      !+
;    3154  1      ! THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
;    3155  1      ! MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
;    3156  1      ! MORE THAN ONE TEST.
;    3157  1      !-
;    3158  1
;    3159  1      global bind
;    3160  1      !
;    3161  1      ! HARDWARE DIALOG
;    3162  1      !
;    3163  1          PTCH1 = uplit (%ascic'                                                ');  !AREA TO PATCH IF NEEDED     ZZZ
;    3164  1          PTCH2 = uplit (%ascic'                                                ');  !                            ZZZ
;    3165  1          PTCH3 = uplit (%ascic'                                                ');  !                            ZZZ
;    3166  1          PTCH4 = uplit (%ascic'                                                ');  !                            ZZZ

;    3167  1          PTCH5 = uplit (%ascic'                                                ');  !                            ZZZ
;    3168  1          HWQ1 = uplit (%ascic'IP address'),
;    3169  1          HWQ2 = uplit (%ascic'Vector'),
;    3170  1          HWQ3 = uplit (%ascic'BR Level (usually 4-RQDX 5-RUX50)'),                 !ZZZ
;    3171  1          HWQ4 = uplit (%ascic'Drive number'),                                      !ZZZ
;    3172  1          HWQ5 = uplit (%ascic'Test entire customer area of this disk'),            !ZZZ
;    3173  1          HWQ6A = uplit (%ascic'Lower octal word of beginning LBN address'),        !ZZZ
;    3174  1          HWQ6B = uplit (%ascic'Higher octal word of beginning LBN address'),       !ZZZ
;    3175  1          HWQ7A = uplit (%ascic'Lower octal word of ending LBN address'),           !ZZZ
;    3176  1          HWQ7B = uplit (%ascic'Higher octal word of ending LBN address'),          !ZZZ
;    3177  1          HWQ8 = uplit (%ascic'Write on customer data area of this disk unit'), !ZZZ
;    3178  1          HWQ9 = uplit (%ascic'** WARNING - CUSTOMER DATA AREA MAY BE OVERWRITTEN! ... CONFIRM'),
;    3179  1          HWQ10 = uplit (%ascic'Also run DUP exercisor'),                           !ZZZ
;    3180  1          HWQ11 = uplit (%ascic'Write on diagnostic area'),                         !ZZZ
;    3181  1      !
;    3182  1      ! SOFTWARE DIALOG
;    3183  1      !
;    3184  1          SWQ1  = uplit (%ascic'Hard error limit'),                                                    !ZZZ
;    3185  1          SWQ2  = uplit (%ascic'Transfer limit in megabytes (0 for quick pass)'),                      !ZZZ
;    3186  1          SWQ4  = uplit (%ascic'Random seek mode'),
;    3187  1          SWQ7  = uplit (%ascic'Read-compares performed at the controller'),
;    3188  1          SWQ9  = uplit (%ascic'Write-compares performed at the controller'),
;    3189  1          SWQ10 = uplit (%ascic'Check all writes at host by reading'),
;    3190  1          SWQ11 = uplit (%ascic'User-defined data pattern'),
;    3191  1          SWQ12 = uplit (%ascic'Select pre-defined data pattern (0 for sequential selection)'),
;    3192  1          SWQ13 = uplit (%ascic'Number of words in data pattern (16 maximum)'),
;    3193  1          SWQ14 = uplit (%ascic'Pattern value (no leading zeros allowed)'),
;    3194  1          SWQ15 = uplit (%ascic'Clear statistical tables after printing'),
;    3195  1          SWQ17 = uplit (%ascic'Percentage of "Fixed Disk" operations out of total operations'),
;    3196  1          SWQ19 = uplit (%ascic'Units to be selected at random (No, implies sequential)'),
;    3197  1          SWQ20 = uplit (%ascic'Rewrite blocks when "Forced Error" detected on reads'),
;    3198  1          SWQ21 = uplit (%ascic'Halt on other hard errors (@s 31-34, 36-37, 39-45)'),                  !ZZZ
;    3199  1          SWQ22 = uplit (%ascic'Halt on soft errors (@s 50-54)'),                                      !ZZZ
;    3200  1          SWQ23 = uplit (%ascic'Halt on bad-block hard errors (@s 35, 38)'),                           !ZZZ
;    3201  1          SWQ24 = uplit (%ascic'Enter time as HHMM (example: 1305)'),                                  !ZZZ
;    3202  1          SWQ25 = uplit (%ascic'Count each retry as a seperate soft error'),                           !ZZZ
;    3203  1          SWQ26 = uplit (%ascic'Running under the A.P.T. Monitor'),                                    !ZZZ
```

```
;    3204  1          SWM1  = uplit (%asciz'The remaining questions only apply to unprotected disk units'),          !ZZZ
;    3205  1          NULL  = uplit (%asciz''),
;    3206  1
;    3207  1      !++
;    3208  1      !  THE FOLLOWING DBMs ARE DEBUG MESSAGES, AND SHOULD BE REMOVED BEFORE
;    3209  1      !  RELEASING THE PROGRAM.  THEY INCLUDE THE NAMES OF EACH ROUTINE, PLUS
;    3210  1      !  FORMAT STATEMENTS FOR PRINTING OUT OTHER INFORMATION.
;    3211  1      !--
;    3212  1
;    3213  1          DBM5   = uplit (%asciz'%N%A** Drop unit %D2'),
;    3214  1          DBM12  = uplit (%asciz'%N%A** PROC_RETPKT: Conn ID %O6%A received'),
;    3215  1          DBM15  = uplit (%asciz'%N%A** Multi-drive test'),
;    3216  1          DBM18  = uplit (%asciz'%N%A** FATAL_ERROR: RETPKT not available'),
;    3217  1          DBM19  = uplit (%asciz'%N%A** FSET_UPAR: Can''t find disk %D3%A in CST %D1'),
;    3218  1          DBM20  = uplit (%asciz'%N%A** Bad conn ID %O6%A received from %O6'),
;    3219  1          DBM21  = uplit (%asciz'%N%A** Message type %O2%A received in MSCP packet'),
;    3220  1          DBM22  = uplit (%asciz'%N%A** SEQUEN: RETPKT not available'),
;    3221  1          DBM23  = uplit (%asciz'%N%A** Error in SET_CTLR_CHAR'),
;    3222  1          DBM25  = uplit (%asciz'%N%A** Ctlr timeout = %D3%A. seconds'),
;    3223  1          DBM26  = uplit (%asciz'%N%A** Error in UNIT_INIT'),
;    3224  1          DBM27  = uplit (%asciz'%N%A** UNIT_INIT: RETPKT has bad ENDCODE'),
;    3225  1          DBM28A = uplit (%asciz'%N%A** Unit size (Lo) = %D5%A.'),
;    3226  1          DBM28B = uplit (%asciz'%N%A** Unit size (Hi) = %D5%A.'),
;    3227  1          DBM29  = uplit (%asciz'%N%A** ACCESS: RETPKT has bad ENDCODE'),
;    3228  1          DBM32  = uplit (%asciz'%N%A** QIO_UNIT: CST %D1%A no unit selected'),
;    3229  1          DBM101 = uplit (%asciz'%N%A** Unit # is: %O6'),
;    3230  1          DBM104 = uplit (%asciz'%N%A** Removable disk is selected'),
;    3231  1          DBM105 = uplit (%asciz'%N%A** Fixed disk is selected'),
;    3232  1          DBM107 = uplit (%asciz'%N%A** Illegal function: %O6'),
;    3233  1          DBM108 = uplit (%asciz'%N%A** Command ref # %O6%A/%O6%A (Oct) not sent by Host'),
;    3234  1          DBM109 = uplit (%asciz'%N%A** Unknown Error Log format %O3%A received'),
;    3235  1      !   DBM110 = uplit (%asciz'%N%A** Error-Log save area full'),
;    3236  1          DBM111 = uplit (%asciz'%N%A** Op-code %O3%A, End-code %O3%A for ref # %O6%A/%O6%A (8)'),
;    3237  1          DBM112 = uplit (%asciz'%N%A** Cmd-bc %O6%A/%O6%A Rsp-bc %O6%A/%O6%A for %O6%A/%O6%A (8)'),
;    3238  1          DBM120 = uplit (%asciz'%N%A** Response already received for cmd %O6%A/%O6%A (8)'),
;    3239  1          DBM121 = uplit (%asciz'%N%A** Failure to send command after # %O6%A/%O6%A (8)'),
;    3240  1      !
;    3241  1      ! DROP UNIT MESSAGES
;    3242  1      !
;    3243  1          DU_MSG = uplit (%asciz'%N%AUNIT%D2%A DROPPED - '),
;    3244  1          DU_RSN = uplit (
;    3245  1                   uplit (%asciz'%AUSER COMMAND%N'),
;    3246  1                   uplit (%asciz'%ACONFIGURATION ERROR%N'),
;    3247  1                   uplit (%asciz'%AINIT ERROR%N'),
;    3248  1                   uplit (%asciz'%ATRANSFER LIMIT REACHED%N'),
;    3249  1                   uplit (%asciz'%AERROR LIMIT REACHED%N'),
;    3250  1                   uplit (%asciz'%AUNRECOVERABLE DRIVE ERROR%N'),
;    3251  1                   uplit (%asciz'%AUNRECOVERABLE CONTROLLER ERROR%N'),
;    3252  1                   uplit (%asciz'%AFAILED TO COME ONLINE%N'),
;    3253  1                   uplit (%asciz'%AFAILED TO ACCESS EITHER FIRST OR LAST TRACK DURING INIT%N'),
;    3254  1                   uplit (%asciz'%ADISK WRITE PROTECTED%N'),
;    3255  1                   uplit (%asciz'%ACOMMAND TIME OUT%N')) : vector [11],
;    3256  1      !
```

```
;    3257  1      ! SYSTEM MESSAGES (PRINTF)
;    3258  1      !
;    3259  1          MSG_01 = uplit (%asciz'%N%APOWER DELAY - WAITING'),
;    3260  1          MSG_02 = uplit (%asciz'%N%AFUNCTIONAL TEST STARTED'),
;    3261  1          MSG_03 = uplit (%asciz'%N%N%AEXERCISER STARTED%N'),
;    3262  1      !
;    3263  1      ! REPORT MESSAGES (PRINTS)
;    3264  1      !
;    3265  1          RPT1  = uplit (%asciz'%N%N%AUNT DSK%S8%A# OF    # BYTES    # OF     # BYTES'),
;    3266  1          RPT2  = uplit (%asciz'%A  --HARD ERRORS-- --SOFT ERRORS--'),
;    3267  1          RPT3  = uplit (%asciz'%N%A #    #  TYPE  READS     READ   WRITES   WRITTEN'),
;    3268  1          RPT4  = uplit (%asciz'%A  SEK DAT DRV HST SEK DAT DRV HST'),
;    3269  1          RPT5  = uplit (%asciz'%N%A--- --- ----  -----  --------- ------  ---------'),
;    3270  1          RPT6  = uplit (%asciz'%A --- --- --- --- --- --- --- ---'),
;    3271  1          RPT7  = uplit (%asciz'%N%D2%D4%S2%T'),
;    3272  1          RPT8  = uplit (%asciz'%D4%Z3%D3%A,%Z3%A,%Z3'),
;    3273  1          RPT9  = uplit (%asciz'%D4%D4%D4%D4%D4%D4%D4%D4'),
;    3274  1          RPT10 = uplit (%asciz'%N%A .    . CNTR      .  .........    .  .........'),
;    3275  1          RPT11 = uplit (%asciz'%A   .   .%D4%A   .   .   .%D4%A   .'),
;    3276  1          RPT12 = uplit (%asciz'%A   .   .%D4%A   .   .   .%D4%A   .'),
;    3277  1          RPT13 = UPLIT(%ASCIZ'%N%N%AUNIT  DISK            # OF    # BLKS       # OF    # BLKS '),
;    3278  1          RPT14 = UPLIT(%ASCIZ'%N%A  #     #     TYPE   READS     READ      WRITES  WRITTEN '),
;    3279  1          RPT15 = UPLIT(%ASCIZ'%N%A----  ----  -------  ------  ------    ------  ------  '),
;    3280  1          RPT16 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBN I/O  %D6%S3%D6%S5%D6%S3%D6'),
;    3281  1      !ZZZ    RPT17  = uplit (%asciz'%N%D2%D4%A  RD52'),
;    3282  1      !ZZZ    RPT18 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBNRD52  %D6%S3%D6%S5%D6%S3%D6'),
;    3283  1      !ZZZ    RPT19  = uplit (%asciz'%N%D2%D4%A  ????'),
;    3284  1
;    3285  1      !
;    3286  1      ! GENERAL ERROR MESSAGES
;    3287  1      !
;    3288  1      !       SYSTEM FATAL (ERRSF)
;    3289  1      !
;    3290  1          EGS_01 = uplit (%asciz'TOO MANY UNITS'),
;    3291  1          EGS_02 = uplit (%asciz'NOT ENOUGH FREE MEMORY FOR ALLOCATING READ/WRITE BUFFERS'),
;    3292  1      !
;    3293  1      !       DRIVE FATAL (ERRDF)
;    3294  1      !
;    3295  1          EGD_10 = uplit (%asciz'REGISTER EXISTENCE TEST FAILED'),
;    3296  1          EGD_11 = uplit (%asciz'VECTOR TEST FAILED'),
;    3297  1          EGD_12 = uplit (%asciz'BR LEVEL TEST FAILED'),
;    3298  1          EGD_13 = uplit (%asciz'INIT SEQUENCE FAILED'),
;    3299  1          EGD_14 = uplit (%asciz'FATAL CONTROLLER ERROR'),
;    3300  1          EGD_15 = uplit (%asciz'ONLINE FAILED'),
;    3301  1          EGD_16 = uplit (%asciz'WRITE-PROTECT CONFLICT'),
;    3302  1          EGD_17 = uplit (%asciz'ACCESS FAILED'),
;    3303  1          EGD_18 = uplit (%asciz'FATAL I/O ERROR'),
;    3304  1      !   EGD_19 = uplit (%asciz'CONTROLLER TIMEOUT'),
;    3305  1          EGD_19 = uplit (%asciz'DISK TYPE UNKNOWN TO EXERCISER'),
;    3306  1          EGD_20 = uplit (%asciz'FAILED TO SEND SET-CONTROLLER-CHARACTERISTICS COMMAND'),
;    3307  1          EGD_21 = uplit (%asciz'SET-CONTROLLER-CHARACTERISTICS RESPONSE HAS BAD ENDCODE OR FLAGS IN ERROR'),
;    3308  1          EGD_22 = uplit (%asciz'FAILED TO SEND ON-LINE COMMAND'),
;    3309  1          EGD_23 = uplit (%asciz'ON-LINE RESPONSE HAS BAD ENDCODE'),
```

```
;    3310  1          EGD_24 = uplit (%asciz'ON-LINE RESPONSE HAS UNKNOWN DEVICE'),
;    3311  1      !
;    3312  1      !       HARD or SOFT (ERRHRD or ERRSOFT)
;    3313  1      !
;    3314  1          EGH_30 = uplit (%asciz'I/O REQUEST FAILED'),
;    3315  1      !
;    3316  1      ! BASIC ERROR MESSAGES (PRINTB)
;    3317  1      !
;    3318  1      !       SYSTEM FATAL (ERRSF)
;    3319  1      !
;    3320  1          EBS_01 = uplit (%asciz'%AMORE THAN %D2%A UNITS SPECIFIED'),
;    3321  1      !
;    3322  1      !       DRIVE FATAL (ERRDF)
;    3323  1      !
;    3324  1          EBD_10 = uplit (%asciz'%A* NO RESPONSE AT ADDRESS %O6'),
;    3325  1          EBD_12 = uplit (%asciz'%A* INCORRECT BR LEVEL FOR DRIVE %O6'),
;    3326  1          EBD_13 = uplit (%asciz'%A* STEP %D1%A READ ERROR'),
;    3327  1          EBD_14 = uplit (%asciz'%A* BAD SA CODE FROM DRIVE %O6'),
;    3328  1          EBD_18 = uplit (%asciz'%A* DISK%D2%A WENT OFFLINE'),
;    3329  1          EBD_19 = uplit (%asciz'%A* DRIVE %O6%A NOT PROCESSING COMMAND PACKETS'),
;    3330  1          EBD_24 = uplit (%asciz'%A* DISK%D2%A WENT TO THE "AVAILABLE" STATE'),
;    3331  1      !
;    3332  1      !
;    3333  1      !       HARD or SOFT (ERRHRD or ERRSOFT)
;    3334  1      !
;    3335  1      EH_0 = UPLIT (%ASCIZ' - UNRECOGNIZED MESSAGE TYPE'),                    !ZZZ
;    3336  1      EH_1 = UPLIT (%ASCIZ' - UNRECOGNIZED CONNECTION ID'),                   !ZZZ
;    3337  1      EH_2 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN MESSAGE'),                  !ZZZ
;    3338  1      EH_3 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN PACKET'),                   !ZZZ
;    3339  1      EH_4 = UPLIT (%ASCIZ' - UNRECOGNIZED CRN'),                             !ZZZ
;    3340  1      EH_5 = UPLIT (%ASCIZ' - UNRECOGNIZED OPCODE'),                          !ZZZ
;    3341  1      EH_6 = UPLIT (%ASCIZ' - MSCP STATUS CODE ERR'),                         !ZZZ
;    3342  1      EH_7 = UPLIT (%ASCIZ' - DUP STATUS CODE ERR'),                          !ZZZ
;    3343  1      EH_8 = UPLIT (%ASCIZ' - UNRECOGNIZED STATUS CODE'),                     !ZZZ
;    3344  1      EH_9 = UPLIT (%ASCIZ' - LBN HOST COMPARE ERR'),                         !ZZZ
;    3345  1      EH_10 = UPLIT (%ASCIZ' - DBN HOST COMPARE ERR'),                        !ZZZ
;    3346  1      EH_12 = UPLIT (%ASCIZ' - UNABLE TO LOAD DUP MEDIA'),                    !ZZZ
;    3347  1      EH_13 = UPLIT (%ASCIZ' - ERR IN DUP PKT WHEN USING CTLR LC PROG'),      !ZZZ
;    3348  1      !
;    3349  1          ERR_00  = uplit (%asciz'%A* DISK%D2'),
;    3350  1          ERR_COD = uplit (
;    3351  1                    uplit (%asciz'%AINVALID COMMAND'),
;    3352  1                    uplit (%asciz'%ACOMMAND ABORTED'),
;    3353  1                    uplit (%asciz'%AUNIT OFFLINE'),
;    3354  1                    uplit (%asciz'%ATRANSITION TO AVAILABLE STATE'),
;    3355  1                    uplit (%asciz'%AMEDIA FORMAT ERROR'),
;    3356  1                    uplit (%asciz'%AWRITE-PROTECTED'),
;    3357  1                    uplit (%asciz'%ADEVICE COMPARE ERROR'),
;    3358  1                    uplit (%asciz'%ADATA ERROR'),
;    3359  1                    uplit (%asciz'%AHOST BUFFER ACCESS ERROR'),
;    3360  1                    uplit (%asciz'%ACONTROLLER ERROR'),
;    3361  1                    uplit (%asciz'%ADRIVE ERROR'),
;    3362  1                    uplit (%asciz'%AMESSAGE FROM INTERNAL DIAGNOSTICS'),
```

```
;    3363  1                    uplit (%asciz'%AHOST COMPARE ERROR'),
;    3364  1                    uplit (%asciz'%ACOMMAND TIMEOUT')) : vector [14],
;    3365  1        !
;    3366  1        !       ERROR LOG MESSAGE (ERRSOFT)
;    3367  1        !
;    3368  1            ELG_00  = uplit (%asciz'%AERROR LOG MESSAGE RECEIVED:%N'),
;    3369  1            ELG_FMT = uplit (
;    3370  1                    uplit (%asciz'%A* CONTROLLER ERROR%N'),
;    3371  1                    uplit (%asciz'%A* HOST MEMORY ACCESS ERROR%N'),
;    3372  1                    uplit (%asciz'%A* DISK%D2%A - DISK TRANSFER ERROR%N'),
;    3373  1                    uplit (%asciz'%A* DISK%D2%A - "STANDARD DISK INTERCONNECT" ERROR%N'),
;    3374  1                    uplit (%asciz'%A* DISK%D2%A - "SMALL DISK" ERROR%N')) : vector [5],
;    3375  1        !
;    3376  1        ! EXTENDED ERROR MESSAGES (PRINTX)
;    3377  1        !
;    3378  1            EX_SA  = uplit (%asciz'%N%A* SA: %O6'),
;    3379  1            EX_SC  = uplit (%asciz'%N%A* STATUS CODE: %O2'),
;    3380  1            EX_SBO = uplit (%asciz'%O4'),
;    3381  1            EX_SB  = uplit (%asciz'%N%A* SUB_CODE: '),
;    3382  1            EX_CMD = uplit (%asciz'%N%A* COMMAND: '),
;    3383  1            EX_RD  = uplit (%asciz'%AREAD'),
;    3384  1            EX_WRT = uplit (%asciz'%AWRITE'),
;    3385  1            EX_CMP = uplit (%asciz'%A-COMPARE'),
;    3386  1            EX_ONL = uplit (%asciz'%AONLINE'),
;    3387  1            EX_ACC = uplit (%asciz'%AACCESS'),
;    3388  1            EX_OP  = uplit (%asciz'%O3'),
;    3389  1            EX_BB  = uplit (%asciz'%N%A* BAD BLOCK (Host replaceable): %D5%A, (OCT %O6%A)'),
;    3390  1            EX_BB1 = uplit (%asciz'%N%A* 1st BAD BLOCK (Host replaceable): %D5%A, (OCT %O6%A)'),
;    3391  1            EX_BBU = uplit (%asciz'%N%A* BAD BLOCK REPORTED (Replaced): %D%A, (OCT %O6%A)'),
;    3392  1            EX_LBN = uplit (%asciz'%N%A* LBN: %D5%A, (OCT %O6%A)'),
;    3393  1            EX_PBN = uplit (%asciz'%N%A* PBN: %D5%A, (OCT %O6%A)'),
;    3394  1            EX_LBR = uplit (%asciz'%N%A* LBN: (READ) %D5%A, (OCT %O6%A)'),
;    3395  1            EX_LBW = uplit (%asciz'%N%A* LBN: (WRITE) %D5%A, (OCT %O6%A)'),
;    3396  1            EX_RBN = uplit (%asciz'%N%A* REPLACEMENT BLOCK NO. %D5%A, (OCT %O6%A)'),
;    3397  1            EX_CBC = uplit (%asciz'%N%A* BYTE COUNT IN COMMAND: %D5%A.'),
;    3398  1            EX_CBR = uplit (%asciz'%N%A* BYTE COUNT IN READ COMMAND: %D5%A.'),
;    3399  1            EX_CBW = uplit (%asciz'%N%A* BYTE COUNT IN WRITE COMMAND: %D5%A.'),
;    3400  1            EX_BC  = uplit (%asciz'%N%A* ACTUAL # OF BYTES TRANSFERRED: %D5%A.'),
;    3401  1            EX_BD  = uplit (%asciz'%N%A* I/O BUFFER ADDRESS (32 bits): %O6%A %O6'),
;    3402  1            EX_BDR = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR READ (32 bits): %O6%A %O6'),
;    3403  1            EX_BDW = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR WRITE (32 bits): %O6%A %O6'),
;    3404  1            EX_RP  = uplit (%asciz'%N%ACONTENTS OF COMMAND/RESPONSE PACKET SAVE AREA:%N'),
;    3405  1            EX_WRD = uplit (%asciz'%A %O6'),
;    3406  1            EX_TIM = uplit (%asciz'%N%A* TIME: %Z2%A:%Z2%A HOURS%N'),
;    3407  1
;    3408  1
;    3409  1            XX13 = UPLIT (%ASCIZ'%N%A * DISK : %D2'),                          !ZZZ
;    3410  1            XX23 = UPLIT (%ASCIZ'%N%ADBN: %D5%A, (OCT %O6%A)'),                !ZZZ
;    3411  1            XX32 = UPLIT (%ASCIZ'%N%ABYTE NUMBER: %D3'),                       !ZZZ
;    3412  1            XX33 = UPLIT (%ASCIZ'%N%ARANDOM WRITTEN WORD :%B16'),              !ZZZ
;    3413  1            XX34 = UPLIT (%ASCIZ'%N%ARANDOM READ WORD bin:%B16%A oct:%O6'),    !ZZZ
;    3414  1
;    3415  1        !
```

```
;    3416  1        ! CONFIGURATION ERROR MESSAGES (PRINTF)
;    3417  1        !
;    3418  1            CER_01 = uplit (%asciz'%N%ADUPLICATE UNIT:%D2%A AT IP: %O6'),
;    3419  1            CER_02 = uplit (%asciz'%N%AMORE THAN %D1%A DIFFERENT IP ADDRESSES'),
;    3420  1        !
;    3421  1        ! ERROR/EVENT SUB CODES (PRINTX)
;    3422  1        !
;    3423  1            SC_SDI = uplit (%asciz'%ASPIN-DOWN IGNORED'),
;    3424  1            SC_CON = uplit (%asciz'%ASTILL CONNECTED'),
;    3425  1            SC_DUP = uplit (%asciz'%ADUPLICATE UNIT NUMBER'),
;    3426  1            SC_ONL = uplit (%asciz'%AALREADY ONLINE'),
;    3427  1            SC_SON = uplit (%asciz'%ASTILL ONLINE'),
;    3428  1            SC_UNK = uplit (%asciz'%AUNIT UNKNOWN OR ONLINE TO ANOTHER CONTROLLER'),
;    3429  1            SC_VOL = uplit (%asciz'%ANO VOLUME MOUNTED OR DRIVE DISABLED BY SWITCH'),
;    3430  1            SC_IOP = uplit (%asciz'%AUNIT INOPERATIVE (RD51/52 write fault)'),
;    3431  1            SC_DIS = uplit (%asciz'%AUNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTICS'),
;    3432  1            SC_FER = uplit (%asciz'%A"FORCED ERROR" DETECTED WHILE ACCESSING FCT OR RCT'),
;    3433  1            SC_FE2 = uplit (%asciz'%ASECTOR HAD BEEN WRITTEN WITH "FORCED ERROR" MODIFIER'),
;    3434  1            SC_ISH = uplit (%asciz'%AFCT OR RCT UNREADABLE - INVALID SECTOR HEADER'),
;    3435  1            SC_IS2 = uplit (%asciz'%AHEADER COMPARE ERROR (Valid header not found)'),
;    3436  1            SC_DST = uplit (%asciz'%AFCT OR RCT UNREADABLE - DATA SYNC TIMEOUT'),
;    3437  1            SC_DS2 = uplit (%asciz'%ADATA SYNC NOT FOUND (Data sync timeout)'),
;    3438  1            SC_ECC = uplit (%asciz'%AFCT OR RCT UNREADABLE - UNCORRECTABLE ECC ERROR'),
;    3439  1            SC_ECD = uplit (%asciz'%AUNCORRECTABLE ECC ERROR'),
;    3440  1            SC_RCT = uplit (%asciz'%ARCT CORRUPTED'),
;    3441  1            SC_FUL = uplit (%asciz'%ANO REPLACEMENT BLOCK AVAILABLE (RCT full)'),
;    3442  1            SC_576 = uplit (%asciz'%ADISK NOT FORMATTED WITH 512 BYTE SECTORS'),
;    3443  1            SC_FCT = uplit (%asciz'%ADISK NOT FORMATTED OR FCT CORRUPTED'),
;    3444  1            SC_EC1 = uplit (%asciz'%AONE SYMBOL ECC ERROR'),
;    3445  1            SC_EC2 = uplit (%asciz'%ATWO SYMBOL ECC ERROR'),
;    3446  1            SC_EC3 = uplit (%asciz'%ATHREE SYMBOL ECC ERROR'),
;    3447  1            SC_EC4 = uplit (%asciz'%AFOUR SYMBOL ECC ERROR'),
;    3448  1            SC_EC5 = uplit (%asciz'%AFIVE SYMBOL ECC ERROR'),
;    3449  1            SC_EC6 = uplit (%asciz'%ASIX SYMBOL ECC ERROR'),
;    3450  1            SC_EC7 = uplit (%asciz'%ASEVEN SYMBOL ECC ERROR'),
;    3451  1            SC_EC8 = uplit (%asciz'%AEIGHT SYMBOL ECC ERROR'),
;    3452  1            SC_EC9 = uplit (%asciz'%ACORRECTABLE ERROR IN ECC FIELD'),
;    3453  1            SC_SWP = uplit (%asciz'%AUNIT SOFTWARE WRITE PROTECTED'),
;    3454  1            SC_HWP = uplit (%asciz'%AUNIT HARDWARE WRITE PROTECTED'),
;    3455  1            SC_ODA = uplit (%asciz'%AODD TRANSFER ADDRESS'),
;    3456  1            SC_ODB = uplit (%asciz'%AODD BYTE COUNT'),
;    3457  1            SC_NXM = uplit (%asciz'%ANON-EXISTENT HOST MEMORY'),
;    3458  1            SC_PAR = uplit (%asciz'%AHOST MEMORY PARITY ERROR'),
;    3459  1            SC_CTO = uplit (%asciz'%ACOMMAND TIMOUT OR RETRY LIMIT EXCEEDED'),
;    3460  1            SC_SDS = uplit (%asciz'%ASERIALIZER/DESERIALIZER OVERRUN OR UNDERRUN'),
;    3461  1            SC_EDC = uplit (%asciz'%A"ERROR DETECTION CODE" ERROR'),
;    3462  1            SC_IDS = uplit (%asciz'%AINCONSISTENT INTERNAL DATA STRUCTURE'),
;    3463  1            SC_SRT = uplit (%asciz'%ADRIVE COMMAND TIMEOUT (No response or seek incomplete)'),
;    3464  1            SC_SRI = uplit (%asciz'%ACONTROLLER DETECTED TRANSMISSION OR PROTOCOL ERROR'),
;    3465  1            SC_POE = uplit (%asciz'%APOSITION ERROR (Mis-seek)'),
;    3466  1            SC_RDY = uplit (%asciz'%ALOST READ/WRITE READY DURING/BETWEEN TRANSFERS'),
;    3467  1            SC_CLK = uplit (%asciz'%ADRIVE CLOCK DROPOUT'),
;    3468  1            SC_RSP = uplit (%asciz'%ALOST RECEIVER READY BETWEEN SECTORS'),
```

```
;    3469  1          SC_SUR = uplit (%asciz'%ADRIVE DETECTED ERROR'),
;    3470  1          SC_PSP = uplit (%asciz'%ACONTROLLER DETECTED PULSE OR STATE PARITY ERROR'),
;    3471  1      !
;    3472  1      ! CONTROLLER GENERIC ERROR CODES
;    3473  1      !
;    3474  1          CNTR_ERR = uplit (
;    3475  1                     uplit (%asciz'%ACONTROLLER TIMEOUT'),
;    3476  1                     uplit (%asciz'%AENVELOPE/PACKET READ ERROR (Parity or timeout)'),
;    3477  1                     uplit (%asciz'%AENVELOPE/PACKET WRITE ERROR (Parity or timeout)'),
;    3478  1                     uplit (%asciz'%ACONTROLLER ROM AND RAM PARITY ERROR'),
;    3479  1                     uplit (%asciz'%ACONTROLLER RAM PARITY ERROR'),
;    3480  1                     uplit (%asciz'%ACONTROLLER ROM PARITY ERROR'),
;    3481  1                     uplit (%asciz'%ARING READ ERROR (Parity or timeout)'),
;    3482  1                     uplit (%asciz'%ARING WRITE ERROR (Parity or timeout)'),
;    3483  1                     uplit (%asciz'INTERRUPT MASTER FAILURE'),
;    3484  1                     uplit (%asciz'%AHOST ACCESS TIMEOUT (Higher level protocol dependent)'),
;    3485  1                     uplit (%asciz'%ACREDIT LIMIT EXCEEDED'),
;    3486  1                     uplit (%asciz'%AQ-BUS MASTER ERROR'),
;    3487  1                     uplit (%asciz'%ACONTROLLER FATAL ERROR'),
;    3488  1                     uplit (%asciz'%AINSTRUCTION LOOP TIMEOUT'),
;    3489  1                     uplit (%asciz'%AILLEGAL VIRTUAL CIRCUIT ID'),
;    3490  1                     uplit (%asciz'%AINTERRUPT VECTOR ILLEGAL'),
;    3491  1                     uplit (%asciz'%AMAINTENANCE READ/WRITE INVALID REGION IDENTIFIER'),
;    3492  1                     uplit (%asciz'%AMAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER'),
;    3493  1                     uplit (%asciz'%ACONTROLLER RAM ERROR (Non-parity)'),
;    3494  1                     uplit (%asciz'%AINIT SEQUENCE ERROR'),
;    3495  1                     uplit (%asciz'%AHIGHER LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
;    3496  1                     uplit (%asciz'%APURGE/POLL HARDWARE FAILURE'),
;    3497  1                     uplit (%asciz'%AMAPPING REGISTER READ FAILURE (Parity or timeout)')) : vector [23],
;    3498  1      !
;    3499  1      ! RD/RX CONTROLLER DEPENDENT ERRORS CODES
;    3500  1      !
;    3501  1          RDRX_ERR = uplit (
;    3502  1                     uplit (%asciz'%AT11 CPU FAILURE'),
;    3503  1                     uplit (%asciz'%ANON-PARITY RAM ERROR'),
;    3504  1                     uplit (%asciz'%ASTATE MACHINE FAILURE - T11 ADDRESS REGISTER'),
;    3505  1                     uplit (%asciz'%ASTATE MACHINE FAILURE - Q-BUS ADDRESS REGISTER'),
;    3506  1                     uplit (%asciz'%ASTATE MACHINE FAILURE - CRC REGISTER'),
;    3507  1                     uplit (%asciz'%ASTATE MACHINE FAILURE - SERIALIZER/DESERIALIZER REGISTER'),
;    3508  1                     uplit (%asciz'%ASTATE MACHINE FAILURE - WRONG HARDWARE VERSION')) : vector [7],
;    3509  1      !
;    3510  1      ! PRINTOUTS THAT FAKE THE DRS ERROR MESSAGES
;    3511  1      !
;    3512  1          DF_MSG  = uplit (%asciz'%N%AZRQA DEV FTL  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;    3513  1          HRD_MSG = uplit (%asciz'%N%AZRQA HRD ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;    3514  1          SFT_MSG = uplit (%asciz'%N%AZRQA SFT ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06%N'),
;    3515  1          HRD_SUB = uplit (%asciz'%N%AI/O REQUEST FAILED%N'),
;    3516  1
;    3517  1
;    3518  1      !
;    3519  1      ! MISCELLANEOUS
;    3520  1      !
;    3521  1          SPACE4   = uplit (%asciz'%S4'),
```

```
;    3522  1          CRLF      = uplit (%asciz'%N'),
;    3523  1          DASH      = uplit (%asciz'%A - '),
;    3524  1          ASTERISK  = uplit (%asciz'%A* ');
```

```
;    3525  1      %sbttl 'DEFAULT HARDWARE P-TABLE'
;    3526  1
;    3527  1      !+
;    3528  1      ! THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
;    3529  1      ! THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
;    3530  1      ! IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
;    3531  1      ! AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
;    3532  1      !-
;    3533  1
;    3534  1      BGNHW (DFPTBL);
;    3535  1
;    3536  1      global
;    3537  1          HWPT_IP_ADDR : word initial (INIT_IP_ADDR),          ! IP ADDRESS
;    3538  1          HWPT_VECTOR : word initial (INIT_INTR_VECT),         ! VECTOR ADDRESS
;    3539  1          HWPT_BR_LEVEL : word initial (INIT_BR_LEVEL),        ! BR LEVEL
;    3540  1          HWPT_DISK : WORD INITIAL (%o'000200'),               !PROTECT, WHOLE DISK, NO DUP   ZZZ
;    3541  1                                                               ! DK 0        ZZZ
;    3542  1          HWPTS0_LBN : word initial (0),                       ! STARTING TRACK LO ZZZ
;    3543  1          HWPTS1_LBN : word initial (0),                       ! STARTING TRACK HI ZZZ
;    3544  1          HWPTE0_LBN : word initial (%o'177777'),              ! ENDING TRACK LO   ZZZ
;    3545  1          HWPTE1_LBN : word initial (0),                       ! ENDING TRACK HI   ZZZ
;    3546  1          NAME_LO    : WORD INITIAL (%o'020040'),              !DISK TYPE          ZZZ
;    3547  1          NAME_HI    : WORD INITIAL (%o'020040');              !DISK TYPE          ZZZ
;    3548  1
;    3549  1      ENDHW;
```

```
;    3550  1     %sbttl 'SOFTWARE P-TABLE'
;    3551  1
;    3552  1     !+
;    3553  1     ! THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
;    3554  1     ! PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
;    3555  1     ! SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
;    3556  1     ! AT RUN TIME.
;    3557  1     !-
;    3558  1
;    3559  1     BGNSW (SFPTBL);
;    3560  1
;    3561  1     global
;    3562  1         SWP_ERROR : word initial (32),                              ! HARD ERROR LIMIT FOR DROPPING UNIT
;    3563  1         SWP_XFER : WORD INITIAL (0),                                !XFER LIMIT. DEFAULT = QUICK PASS          !ZZZ
;    3564  1         SWP_FLAGS : word initial (SWF_RDM or SWF_CRC or SWF_HWC or SWF_FER  ! FLAGS (SEE DOCUMENTATION)        !ZZZ
;    3565  1                                  or SWF_HRD or SWF_BLK),            !                                      !ZZZ
;    3566  1         SWP_DPAT : word initial (0),                                ! DATA PATTERN NUMBER
;    3567  1         SWP_RAT : word initial (99),                                ! RD51/52 OPERATION RATIO
;    3568  1         SWP_TIME : word initial (0),                                ! START TIME (HHMM)
;    3569  1         DUPROUND : WORD INITIAL (11),               !NO OF I/Os PER DBN TEST ZZZ
;    3570  1
;    3571  1     !       THE NEXT TWO LOCATIONS SHOULD BE TOGETHER
;    3572  1
;    3573  1         SWP_UCNT : word initial (MAX_UDP_CNT),                      ! USER DATA PATTERN COUNT
;    3574  1         SWP_UDPAT : vector [MAX_UDP_CNT, word];                     ! USER DATA PATTERN
;    3575  1
;    3576  1     ENDSW;
```

```
;    3577  1     %sbttl 'PROTECTION TABLE'
;    3578  1
;    3579  1     !+
;    3580  1     ! THIS TABLE IS USED BY THE RUNTIME SERVICES
;    3581  1     ! TO PROTECT THE LOAD MEDIA.
;    3582  1     !
;    3583  1
;    3584  1     BGNPROT (0, -1, 6);
;    3585  1
;    3586  1     !1ST ARG =      OFFSET INTO P-TABLE FOR CSR ADDRESS
;    3587  1     !2ND ARG =      OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;    3588  1     !3RD ARG =      OFFSET INTO P-TABLE FOR DRIVE NUMBER
;    3589  1
;    3590  1     ENDPROT;
;    3591  1     end
;    3592  1
;    3593  0     eludom


                                        .TITLE  ZRQAM1 RD/RX EXERCISER
                                        .IDENT  /V02.2/
                                        .ENABL  AMA


000000                                  .PSECT  $CODE$,  RO
000000      132     122     121     L$NAME::.ASCII  /ZRQ/
000003      101                             .ASCII  /A/
000004      000                             .BYTE   0
000005      000                             .BYTE   0
000006      000                             .BYTE   0
000007      000                             .BYTE   0
000010                              L$REV::
000010      107                             .ASCII  /G/
000011      060                             .ASCII  /0/
000012   000000G                    L$UNIT::.WORD   T$PTHV
000014   076400                     L$TIML::.WORD   76400
000016   000000G                    L$HPCP::.WORD   L$HARD
000020   000000G                    L$SPCP::.WORD   L$SOFT
000022   023104'                    L$HPTP::.WORD   L$HW
000024   023134'                    L$SPTP::.WORD   L$SW
000026   000000G                    L$LADP::.WORD   L$LAST
000030   000000                     L$STA::  .WORD   0
000032   000000                     L$CO::   .WORD   0
000034   000001                     L$DTYP::.WORD   1
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$MREV::
000050      004                             .BYTE   4
000051      000                             .BYTE   0
000052   000000                     L$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$AUTO
000112  023216'                 L$PRT:: .WORD   L$PROT
000114  000001                  L$TEST::.WORD   1
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  040     040     040     P.AAA:  .ASCII  /   /
000141  040     040     040             .ASCII  /   /
000144  040     040     040             .ASCII  /   /
000147  040     040     040             .ASCII  /   /
000152  040     040     040             .ASCII  /   /
000155  040     040     040             .ASCII  /   /
000160  040     040     040             .ASCII  /   /
000163  040     040     040             .ASCII  /   /
000166  040     040     040             .ASCII  /   /
000171  040     040     040             .ASCII  /   /
000174  040     040     040             .ASCII  /   /
000177  040     040     040             .ASCII  /   /
000202  040     040     040             .ASCII  /   /
000205  040     040     000             .ASCII  /   /<00>
000210  040     040     040     P.AAB:  .ASCII  /   /
000213  040     040     040             .ASCII  /   /
000216  040     040     040             .ASCII  /   /
000221  040     040     040             .ASCII  /   /
000224  040     040     040             .ASCII  /   /
000227  040     040     040             .ASCII  /   /
000232  040     040     040             .ASCII  /   /
000235  040     040     040             .ASCII  /   /
000240  040     040     040             .ASCII  /   /
000243  040     040     040             .ASCII  /   /
000246  040     040     040             .ASCII  /   /
000251  040     040     040             .ASCII  /   /
000254  040     040     040             .ASCII  /   /
```

M5


```
000257  040  040  000                .ASCII  /  /<00>
000262  040  040  040      P.AAC:    .ASCII  /  /
000265  040  040  040                .ASCII  /  /
000270  040  040  040                .ASCII  /  /
000273  040  040  040                .ASCII  /  /
000276  040  040  040                .ASCII  /  /
000301  040  040  040                .ASCII  /  /
000304  040  040  040                .ASCII  /  /
000307  040  040  040                .ASCII  /  /
000312  040  040  040                .ASCII  /  /
000315  040  040  040                .ASCII  /  /
000320  040  040  040                .ASCII  /  /
000323  040  040  040                .ASCII  /  /
000326  040  040  040                .ASCII  /  /
000331  040  040  000                .ASCII  /  /<00>
000334  040  040  040      P.AAD:    .ASCII  /  /
000337  040  040  040                .ASCII  /  /
000342  040  040  040                .ASCII  /  /
000345  040  040  040                .ASCII  /  /
000350  040  040  040                .ASCII  /  /
000353  040  040  040                .ASCII  /  /
000356  040  040  040                .ASCII  /  /
000361  040  040  040                .ASCII  /  /
000364  040  040  040                .ASCII  /  /
000367  040  040  040                .ASCII  /  /
000372  040  040  040                .ASCII  /  /
000375  040  040  040                .ASCII  /  /
000400  040  040  040                .ASCII  /  /
000403  040  040  000                .ASCII  /  /<00>
000406  040  040  040      P.AAE:    .ASCII  /  /
000411  040  040  040                .ASCII  /  /
000414  040  040  040                .ASCII  /  /
000417  040  040  040                .ASCII  /  /
000422  040  040  040                .ASCII  /  /
000425  040  040  040                .ASCII  /  /
000430  040  040  040                .ASCII  /  /
000433  040  040  040                .ASCII  /  /
000436  040  040  040                .ASCII  /  /
000441  040  040  040                .ASCII  /  /
000444  040  040  040                .ASCII  /  /
000447  040  040  040                .ASCII  /  /
000452  040  040  040                .ASCII  /  /
000455  040  040  000                .ASCII  /  /<00>
000460  111  120  040      P.AAF:    .ASCII  /IP /
000463  141  144  144                .ASCII  /add/
000466  162  145  163                .ASCII  /res/
000471  163  000  000                .ASCII  /s/<00><00>
000474  126  145  143      P.AAG:    .ASCII  /Vec/
000477  164  157  162                .ASCII  /tor/
000502  000  000                     .ASCII  <00><00>
000504  102  122  040      P.AAH:    .ASCII  /BR /
000507  114  145  166                .ASCII  /Lev/
000512  145  154  040                .ASCII  /el /
```

```
000515  133  165  163                .ASCII  /[ue/
000520  165  141  154                .ASCII  /uel/
000523  154  171  040                .ASCII  /ly /
000526  064  055  122                .ASCII  /4-R/
000531  121  104  130                .ASCII  /QDX/
000534  040  065  055                .ASCII  / 5-/
000537  122  125  130                .ASCII  /RUX/
000542  065  060  135                .ASCII  /50]/
000545  000                        .ASCII  <00>
000546  104  162  151  P.AAI:        .ASCII  /Dri/
000551  166  145  040                .ASCII  /ve /
000554  156  165  155                .ASCII  /num/
000557  142  145  162                .ASCII  /ber/
000562  000  000                    .ASCII  <00><00>
000564  124  145  163  P.AAJ:        .ASCII  /Tes/
000567  164  040  145                .ASCII  /t e/
000572  156  164  151                .ASCII  /nti/
000575  162  145  040                .ASCII  /re /
000600  143  165  163                .ASCII  /cus/
000603  164  157  155                .ASCII  /tom/
000606  145  162  040                .ASCII  /er /
000611  141  162  145                .ASCII  /are/
000614  141  040  157                .ASCII  /a o/
000617  146  040  164                .ASCII  /f t/
000622  150  151  163                .ASCII  /his/
000625  040  144  151                .ASCII  / di/
000630  163  153  000                .ASCII  /sk/<00>
000633  000                        .ASCII  <00>
000634  114  157  167  P.AAK:        .ASCII  /Low/
000637  145  162  040                .ASCII  /er /
000642  157  143  164                .ASCII  /oct/
000645  141  154  040                .ASCII  /al /
000650  167  157  162                .ASCII  /wor/
000653  144  040  157                .ASCII  /d o/
000656  146  040  142                .ASCII  /f b/
000661  145  147  151                .ASCII  /egi/
000664  156  156  151                .ASCII  /nni/
000667  156  147  040                .ASCII  /ng /
000672  114  102  116                .ASCII  /LBN/
000675  040  141  144                .ASCII  / ad/
000700  144  162  145                .ASCII  /dre/
000703  163  163  000                .ASCII  /ss/<00>
000706  110  151  147  P.AAL:        .ASCII  /Hig/
000711  150  145  162                .ASCII  /her/
000714  040  157  143                .ASCII  / oc/
000717  164  141  154                .ASCII  /tal/
000722  040  167  157                .ASCII  / wo/
000725  162  144  040                .ASCII  /rd /
000730  157  146  040                .ASCII  /of /
000733  142  145  147                .ASCII  /beg/
000736  151  156  156                .ASCII  /inn/
000741  151  156  147                .ASCII  /ing/
000744  040  114  102                .ASCII  / LB/
```

```
000747  116  040  141          .ASCII  /N a/
000752  144  144  162          .ASCII  /ddr/
000755  145  163  163          .ASCII  /ess/
000760  000  000               .ASCII  <00><00>
000762  114  157  167  P.AAM:   .ASCII  /Low/
000765  145  162  040          .ASCII  /er /
000770  157  143  164          .ASCII  /oct/
000773  141  154  040          .ASCII  /al /
000776  167  157  162          .ASCII  /wor/
001001  144  040  157          .ASCII  /d o/
001004  146  040  145          .ASCII  /f e/
001007  156  144  151          .ASCII  /ndi/
001012  156  147  040          .ASCII  /ng /
001015  114  102  116          .ASCII  /LBN/
001020  040  141  144          .ASCII  / ad/
001023  144  162  145          .ASCII  /dre/
001026  163  163  000          .ASCII  /ss/<00>
001031  000                    .ASCII  <00>
001032  110  151  147  P.AAN:   .ASCII  /Hig/
001035  150  145  162          .ASCII  /her/
001040  040  157  143          .ASCII  / oc/
001043  164  141  154          .ASCII  /tal/
001046  040  167  157          .ASCII  / wo/
001051  162  144  040          .ASCII  /rd /
001054  157  146  040          .ASCII  /of /
001057  145  156  144          .ASCII  /end/
001062  151  156  147          .ASCII  /ing
001065  040  114  102          .ASCII  / LB/
001070  116  040  141          .ASCII  /N a/
001073  144  144  162          .ASCII  /ddr/
001076  145  163  163          .ASCII  /ess/
001101  000                    .ASCII  <00>
001102  127  162  151  P.AAO:   .ASCII  /Wri/
001105  164  145  040          .ASCII  /te /
001110  157  156  040          .ASCII  /on /
001113  143  165  163          .ASCII  /cus/
001116  164  157  155          .ASCII  /tom/
001121  145  162  040          .ASCII  /er /
001124  144  141  164          .ASCII  /dat/
001127  141  040  141          .ASCII  /a a/
001132  162  145  141          .ASCII  /rea/
001135  040  157  146          .ASCII  / of/
001140  040  164  150          .ASCII  / th/
001143  151  163  040          .ASCII  /is /
001146  144  151  163          .ASCII  /dis/
001151  153  040  165          .ASCII  /k u/
001154  156  151  164          .ASCII  /nit/
001157  000                    .ASCII  <00>
001160  052  052  040  P.AAP:   .ASCII  /** /
001163  127  101  122          .ASCII  /WAR/
001166  116  111  116          .ASCII  /NIN/
001171  107  040  055          .ASCII  /G -/
001174  040  103  125          .ASCII  / CU/
```

```
001177  123  124  117          .ASCII  /STO/
001202  115  105  122          .ASCII  /MER/
001205  040  104  101          .ASCII  / DA/
001210  124  101  040          .ASCII  /TA /
001213  101  122  105          .ASCII  /ARE/
001216  101  040  115          .ASCII  /A M/
001221  101  131  040          .ASCII  /AY /
001224  102  105  040          .ASCII  /BE /
001227  117  126  105          .ASCII  /OVE/
001232  122  127  122          .ASCII  /RWR/
001235  111  124  124          .ASCII  /ITT/
001240  105  116  041          .ASCII  /EN!/
001243  040  056  056          .ASCII  / ../
001246  056  040  103          .ASCII  /. C/
001251  117  116  106          .ASCII  /ONF/
001254  111  122  115          .ASCII  /IRM/
001257  000                    .ASCII  <00>
001260  101  154  163  P.AAQ:   .ASCII  /Als/
001263  157  040  162          .ASCII  /o r/
001266  165  156  040          .ASCII  /un /
001271  104  125  120          .ASCII  /DUP/
001274  040  145  170          .ASCII  / ex/
001277  145  162  143          .ASCII  /erc/
001302  151  163  145          .ASCII  /ise/
001305  162  000  000          .ASCII  /r/<00><00>
001310  127  162  151  P.AAR:   .ASCII  /Wri/
001313  164  145  040          .ASCII  /te /
001316  157  156  040          .ASCII  /on /
001321  144  151  141          .ASCII  /dis/
001324  147  156  157          .ASCII  /gno/
001327  163  164  151          .ASCII  /sti/
001332  143  040  141          .ASCII  /c a/
001335  162  145  141          .ASCII  /rea/
001340  000  000               .ASCII  <00><00>
001342  110  141  162  P.AAS:   .ASCII  /Har/
001345  144  040  145          .ASCII  /d e/
001350  162  162  157          .ASCII  /rro/
001353  162  040  154          .ASCII  /r l/
001356  151  155  151          .ASCII  /imi/
001361  164  000  000          .ASCII  /t/<00><00>
001364  124  162  141  P.AAT:   .ASCII  /Tra/
001367  156  163  146          .ASCII  /nsf/
001372  145  162  040          .ASCII  /er /
001375  154  151  155          .ASCII  /lim/
001400  151  164  040          .ASCII  /it /
001403  151  156  040          .ASCII  /in /
001406  155  145  147          .ASCII  /meg/
001411  141  142  171          .ASCII  /aby/
001414  164  145  163          .ASCII  /tes/
001417  040  050  060          .ASCII  / (0/
001422  040  146  157          .ASCII  / fo/
001425  162  040  161          .ASCII  /r q/
001430  165  151  143          .ASCII  /uic/
```

```
001433  153  040  160                .ASCII  /k p/
001436  141  163  163                .ASCII  /sse/
001441  051  000  000                .ASCII  /)/<00><00>
001444  122  141  156  P.AAU:        .ASCII  /Ran/
001447  144  157  155                 ASCII  /dom/
001452  040  163  145                .ASCII  / se/
001455  145  153  040                .ASCII  /ek /
001460  155  157  144                .ASCII  /mod/
001463  145  000  000                .ASCII  /e/<00><00>
001466  122  145  141  P.AAV:        .ASCII  /Rea/
001471  144  055  143                .ASCII  /d-c/
001474  157  155  160                .ASCII  /omp/
001477  141  162  145                .ASCII  /are/
001502  163  040  160                .ASCII  /s p/
001505  145  162  146                .ASCII  /erf/
001510  157  162  155                .ASCII  /orm/
001513  145  144  040                .ASCII  /ed /
001516  141  164  040                .ASCII  /at /
001521  164  150  145                .ASCII  /the/
001524  040  143  157                .ASCII  / co/
001527  156  164  162                .ASCII  /ntr/
001532  157  154  154                .ASCII  /oll/
001535  145  162  000                .ASCII  /er/<00>
001540  127  162  151  P.AAW:        .ASCII  /Wri/
001543  164  145  055                .ASCII  /te-/
001546  143  157  155                .ASCII  /com/
001551  160  141  162                .ASCII  /par/
001554  145  163  040                .ASCII  /es /
001557  160  145  162                .ASCII  /per/
001562  146  157  162                .ASCII  /for/
001565  155  145  144                .ASCII  /med/
001570  040  141  164                .ASCII  / at/
001573  040  164  150                .ASCII  / th/
001576  145  040  143                .ASCII  /e c/
001601  157  156  164                .ASCII  /ont/
001604  162  157  154                .ASCII  /rol/
001607  154  145  162                .ASCII  /ler/
001612  000  000                     .ASCII  <00><00>
001614  103  150  145  P.AAX:        .ASCII  /Che/
001617  143  153  040                .ASCII  /ck /
001622  141  154  154                .ASCII  /all/
001625  040  167  162                .ASCII  / wr/
001630  151  164  145                .ASCII  /ite/
001633  163  040  141                .ASCII  /s a/
001636  164  040  150                .ASCII  /t h/
001641  157  163  164                .ASCII  /ost/
001644  040  142  171                .ASCII  / by/
001647  040  162  145                .ASCII  / re/
001652  141  144  151                .ASCII  /adi/
001655  156  147  000                .ASCII  /ng/<00>
001660  125  163  145  P.AAY:        .ASCII  /Use/
001663  162  055  144                .ASCII  /r-d/
001666  145  146  151                .ASCII  /efi/
```

```
001671  156  145  144              .ASCII  /ned/
001674  040  144  141              .ASCII  / da/
001677  164  141  040              .ASCII  /ta /
001702  160  141  164              .ASCII  /pat/
001705  164  145  162              .ASCII  /ter/
001710  156  000                   .ASCII  /n/<00>
001712  123  145  154    P.AAZ:    .ASCII  /Sel/
001715  145  143  164              .ASCII  /ect/
001720  040  160  162              .ASCII  / pr/
001723  145  055  144              .ASCII  /e-d/
001726  145  146  151              .ASCII  /efi/
001731  156  145  144              .ASCII  /ned/
001734  040  144  141              .ASCII  / da/
001737  164  141  040              .ASCII  /ta /
001742  160  141  164              .ASCII  /pat/
001745  164  145  162              .ASCII  /ter/
001750  156  040  050              .ASCII  /n (/
001753  060  040  146              .ASCII  /0 f/
001756  157  162  040              .ASCII  /or /
001761  163  145  161              .ASCII  /seq/
001764  165  145  156              .ASCII  /uen/
001767  164  151  141              .ASCII  /tia/
001772  154  040  163              .ASCII  /l s/
001775  145  154  145              .ASCII  /ele/
002000  143  164  151              .ASCII  /cti/
002003  157  156  051              .ASCII  /on)/
002006  000  000                   .ASCII  <00><00>
002010  116  165  155    P.ABA:    .ASCII  /Num/
002013  142  145  162              .ASCII  /ber/
002016  040  157  146              .ASCII  / of/
002021  040  167  157              .ASCII  / wo/
002024  162  144  163              .ASCII  /rds/
002027  040  151  156              .ASCII  / in/
002032  040  144  141              .ASCII  / da/
002035  164  141  040              .ASCII  /ta /
002040  160  141  164              .ASCII  /pat/
002043  164  145  162              .ASCII  /ter/
002046  156  040  050              .ASCII  /n (/
002051  061  066  040              .ASCII  /16 /
002054  155  141  170              .ASCII  /max/
002057  151  155  165              .ASCII  /imu/
002062  155  051  000              .ASCII  /m)/<00>
002065  000                       .ASCII  <00>
002066  120  141  164    P.ABB:    .ASCII  /Pat/
002071  164  145  162              .ASCII  /ter/
002074  156  040  166              .ASCII  /n v/
002077  141  154  165              .ASCII  /alu/
002102  145  040  050              .ASCII  /e (/
002105  156  157  040              .ASCII  /no /
002110  154  145  141              .ASCII  /lea/
002113  144  151  156              .ASCII  /din/
002116  147  040  172              .ASCII  /g z/
002121  145  162  157              .ASCII  /ero/
```

```
002124  163  040  141          .ASCII  /e e/
002127  154  154  157          .ASCII  /llo/
002132  167  145  144          .ASCII  /wed/
002135  051  000  000          .ASCII  /)/<00><00>
002140  103  154  145  P.ABC:   .ASCII  /Cle/
002143  141  162  040          .ASCII  /ar /
002146  163  164  141          .ASCII  /sta/
002151  164  151  163          .ASCII  /tis/
002154  164  151  143          .ASCII  /tic/
002157  141  154  040          .ASCII  /al /
002162  164  141  142          .ASCII  /tab/
002165  154  145  163          .ASCII  /les/
002170  040  141  146          .ASCII  / af/
002173  164  145  162          .ASCII  /ter/
002176  040  160  162          .ASCII  / pr/
002201  151  156  164          .ASCII  /int/
002204  151  156  147          .ASCII  /ing/
002207  000                    .ASCII  <00>
002210  120  145  162  P.ABD:   .ASCII  /Per/
002213  143  145  156          .ASCII  /cen/
002216  164  141  147          .ASCII  /tag/
002221  145  040  157          .ASCII  /e o/
002224  146  040  042          .ASCII  /f "/
002227  106  151  170          .ASCII  /Fix/
002232  145  144  040          .ASCII  /ed /
002235  104  151  163          .ASCII  /Dis/
002240  153  042  040          .ASCII  /k" /
002243  157  160  145          .ASCII  /ope/
002246  162  141  164          .ASCII  /rat/
002251  151  157  156          .ASCII  /ion/
002254  163  040  157          .ASCII  /s o/
002257  165  164  040          .ASCII  /ut /
002262  157  146  040          .ASCII  /of /
002265  164  157  164          .ASCII  /tot/
002270  141  154  040          .ASCII  /al /
002273  157  160  145          .ASCII  /ope/
002276  162  141  164          .ASCII  /rat/
002301  151  157  156          .ASCII  /ion/
002304  163  000                .ASCII  /s/<00>
002306  125  156  151  P.ABE:   .ASCII  /Uni/
002311  164  163  040          .ASCII  /ts /
002314  164  157  040          .ASCII  /to /
002317  142  145  040          .ASCII  /be /
002322  163  145  154          .ASCII  /sel/
002325  145  143  164          .ASCII  /ect/
002330  145  144  040          .ASCII  /ed /
002333  141  164  040          .ASCII  /at /
002336  162  141  156          .ASCII  /ren/
002341  144  157  155          .ASCII  /dom/
002344  040  050  116          .ASCII  / (N/
002347  157  054  040          .ASCII  /o, /
002352  151  155  160          .ASCII  /imp/
002355  154  151  145          .ASCII  /lie/
```

```
002360  163  040  163          .ASCII  /s s/
002363  145  161  165          .ASCII  /equ/
002366  145  156  164          .ASCII  /ent/
002371  151  141  154          .ASCII  /ial/
002374  051  000               .ASCII  /)/<00>
002376  122  145  167  P.ABF:   .ASCII  /Rew/
002401  162  151  164          .ASCII  /rit/
002404  145  040  142          .ASCII  /e b/
002407  154  157  143          .ASCII  /loc/
002412  153  163  040          .ASCII  /ks /
002415  167  150  145          .ASCII  /whe/
002420  156  040  042          .ASCII  /n "/
002423  106  157  162          .ASCII  /For/
002426  143  145  144          .ASCII  /ced/
002431  040  105  162          .ASCII  / Er/
002434  162  157  162          .ASCII  /ror/
002437  042  040  144          .ASCII  /" d/
002442  145  164  145          .ASCII  /ete/
002445  143  164  145          .ASCII  /cte/
002450  144  040  157          .ASCII  /d o/
002453  156  040  162          .ASCII  /n r/
002456  145  141  144          .ASCII  /ead/
002461  163  000  000          .ASCII  /s/<00><00>
002464  110  141  154  P.ABG:   .ASCII  /Hal/
002467  164  040  157          .ASCII  /t o/
002472  156  040  157          .ASCII  /n o/
002475  164  150  145          .ASCII  /the/
002500  162  040  150          .ASCII  /r h/
002503  141  162  144          .ASCII  /ard/
002506  040  145  162          .ASCII  / er/
002511  162  157  162          .ASCII  /ror/
002514  163  040  050          .ASCII  /s (/
002517  043  163  040          .ASCII  /#s /
002522  063  061  055          .ASCII  /31-/
002525  063  064  054          .ASCII  /34,/
002530  040  063  066          .ASCII  / 36/
002533  055  063  067          .ASCII  /-37/
002536  054  040  063          .ASCII  /, 3/
002541  071  055  064          .ASCII  /9-4/
002544  065  051  000          .ASCII  /5)/<00>
002547  000                    .ASCII  <00>
002550  110  141  154  P.ABH:   .ASCII  /Hal/
002553  164  040  157          .ASCII  /t o/
002556  156  040  163          .ASCII  /n s/
002561  157  146  164          .ASCII  /oft/
002564  040  145  162          .ASCII  / er/
002567  162  157  162          .ASCII  /ror/
002572  163  040  050          .ASCII  /s (/
002575  043  163  040          .ASCII  /#s /
002600  065  060  055          .ASCII  /50-/
002603  065  064  051          .ASCII  /54)/
002606  000  000               .ASCII  <00><00>
002610  110  141  154  P.ABI:   .ASCII  /Hal/
```

```
002613  164  040  157                .ASCII  /t o/
002616  156  040  142                .ASCII  /n b/
002621  141  144  055                .ASCII  /ad-/
002624  142  154  157                .ASCII  /blo/
002627  143  153  040                .ASCII  /ck /
002632  150  141  162                .ASCII  /har/
002635  144  040  145                .ASCII  /d e/
002640  162  162  157                .ASCII  /rro/
002643  162  163  040                .ASCII  /rs /
002646  050  043  163                .ASCII  /(#s/
002651  040  063  065                .ASCII  / 35/
002654  054  040  063                .ASCII  /, 3/
002657  070  051  000                .ASCII  /8)/<00>
002662  105  156  164  P.ABJ:        .ASCII  /Ent/
002665  145  162  040                .ASCII  /er /
002670  164  151  155                .ASCII  /tim/
002673  145  040  141                .ASCII  /e a/
002676  163  040  110                .ASCII  /s H/
002701  110  115  115                .ASCII  /HMM/
002704  040  050  145                .ASCII  / (e/
002707  170  141  155                .ASCII  /xam/
002712  160  154  145                .ASCII  /ple/
002715  072  040  061                .ASCII  /: 1/
002720  063  060  065                .ASCII  /305/
002723  051  000  000                .ASCII  /)/<00><00>
002726  103  157  165  P.ABK:        .ASCII  /Cou/
002731  156  164  040                .ASCII  /nt /
002734  145  141  143                .ASCII  /eac/
002737  150  040  162                .ASCII  /h r/
002742  145  164  162                .ASCII  /etr/
002745  171  040  141                .ASCII  /y a/
002750  163  040  141                .ASCII  /s a/
002753  040  163  145                .ASCII  / se/
002756  160  145  162                .ASCII  /per/
002761  141  164  145                .ASCII  /ate/
002764  040  163  157                .ASCII  / so/
002767  146  164  040                .ASCII  /ft /
002772  145  162  162                .ASCII  /err/
002775  157  162  000                .ASCII  /or/<00>
003000  122  165  156  P.ABL:        .ASCII  /Run/
003003  156  151  156                .ASCII  /nin/
003006  147  040  165                .ASCII  /g u/
003011  156  144  145                .ASCII  /nde/
003014  162  040  164                .ASCII  /r t/
003017  150  145  040                .ASCII  /he /
003022  101  056  120                .ASCII  /A.P/
003025  056  124  056                .ASCII  /.T./
003030  040  115  157                .ASCII  / Mo/
003033  156  151  164                .ASCII  /nit/
003036  157  162  000                .ASCII  /or/<00>
003041  000                        .ASCII  <00>
003042  124  150  145  P.ABM:        .ASCII  /The/
003045  040  162  145                .ASCII  / re/
```

```
003050  155     141     151             .ASCII  /mei/
003053  156     151     156             .ASCII  /nin/
003056  147     040     161             .ASCII  /g q/
003061  165     145     163             .ASCII  /uee/
003064  164     151     157             .ASCII  /tio/
003067  156     163     040             .ASCII  /ns /
003072  157     156     154             .ASCII  /onl/
003075  171     040     141             .ASCII  /y a/
003100  160     160     154             .ASCII  /ppl/
003103  171     040     164             .ASCII  /y t/
003106  157     040     165             .ASCII  /o u/
003111  156     160     162             .ASCII  /npr/
003114  157     164     145             .ASCII  /ote/
003117  143     164     145             .ASCII  /cte/
003122  144     040     144             .ASCII  /d d/
003125  151     163     153             .ASCII  /isk/
003130  040     165     156             .ASCII  / un/
003133  151     164     163             .ASCII  /its/
003136  000     000                     .ASCII  <00><00>
003140  000     000             P.ABN:  .ASCII  <00><00>
003142  045     116     045     P.ABO:  .ASCII  /%N%/
003145  101     052     052             .ASCII  /A**/
003150  040     104     162             .ASCII  / Dr/
003153  157     160     040             .ASCII  /op /
003156  165     156     151             .ASCII  /uni/
003161  164     040     045             .ASCII  /t %/
003164  104     062     000             .ASCII  /D2/<00>
003167  000                             .ASCII  <00>
003171  045     116     045     P.ABP:  .ASCII  /%N%/
003173  101     052     052             .ASCII  /A**/
003176  040     120     122             .ASCII  / PR/
003201  117     103     137             .ASCII  /OC_/
003204  122     105     124             .ASCII  /RET/
003207  120     113     124             .ASCII  /PKT/
003212  072     040     103             .ASCII  /: C/
003215  157     156     156             .ASCII  /onn/
003220  040     111     104             .ASCII  / ID/
003223  040     045     117             .ASCII  / %O/
003226  066     045     101             .ASCII  /6%A/
003231  040     162     145             .ASCII  / re/
003234  143     145     151             .ASCII  /cei/
003237  166     145     144             .ASCII  /ved/
003242  000     000                     .ASCII  <00><00>
003244  045     116     045     P.ABQ:  .ASCII  /%N%/
003247  101     052     052             .ASCII  /A**/
003252  040     115     165             .ASCII  / Mu/
003255  154     164     151             .ASCII  /lti/
003260  055     144     162             .ASCII  /-dr/
003263  151     166     145             .ASCII  /ive/
003266  040     164     145             .ASCII  / te/
003271  163     164     000             .ASCII  /st/<00>
003274  045     116     045     P.ABR:  .ASCII  /%N%/
003277  101     052     052             .ASCII  /A**/
```

```
003302  040  106  101                .ASCII  / FA/
003305  124  101  114                .ASCII  /TAL/
003310  137  105  122                .ASCII  /_ER/
003313  122  117  122                .ASCII  /ROR/
003316  072  040  122                .ASCII  /: R/
003321  105  124  120                .ASCII  /ETP/
003324  113  124  040                .ASCII  /KT /
003327  156  157  164                .ASCII  /not/
003332  040  141  166                .ASCII  / av/
003335  141  151  154                .ASCII  /ail/
003340  141  142  154                .ASCII  /abl/
003343  145  000  000                .ASCII  /e/<00><00>
003346  045  116  045      P.ABS:    .ASCII  /%N%/
003351  101  052  052                .ASCII  /A◆◆/
003354  040  106  123                .ASCII  / FS/
003357  105  124  137                .ASCII  /ET_/
003362  125  120  101                .ASCII  /UPA/
003365  122  072  040                .ASCII  /R: /
003370  103  141  156                .ASCII  /Can/
003373  047  164  040                .ASCII  /'t /
003376  146  151  156                .ASCII  /fin/
003401  144  040  144                .ASCII  /d d/
003404  151  163  153                .ASCII  /isk/
003407  040  045  104                .ASCII  / %D/
003412  063  045  101                .ASCII  /3%A/
003415  040  151  156                .ASCII  / in/
003420  040  103  123                .ASCII  / CS/
003423  124  040  045                .ASCII  /T %/
003426  104  061  000                .ASCII  /D1/<00>
003431  000                         .ASCII  <00>
003432  045  116  045      P.ABT:    .ASCII  /%N%/
003435  101  052  052                .ASCII  /A◆◆/
003440  040  102  141                .ASCII  / Ba/
003443  144  040  143                .ASCII  /d c/
003446  157  156  156                .ASCII  /onn/
003451  040  111  104                .ASCII  / ID/
003454  040  045  117                .ASCII  / %O/
003457  066  045  101                .ASCII  /6%A/
003462  040  162  145                .ASCII  / re/
003465  143  145  151                .ASCII  /cei/
003470  166  145  144                .ASCII  /ved/
003473  040  146  162                .ASCII  / fr/
003476  157  155  040                .ASCII  /om /
003501  045  117  066                .ASCII  /%06/
003504  000  000                     .ASCII  <00><00>
003506  045  116  045      P.ABU:    .ASCII  /%N%/
003511  101  052  052                .ASCII  /A◆◆/
003514  040  115  145                .ASCII  / Me/
003517  163  163  141                .ASCII  /ssa/
003522  147  145  040                .ASCII  /ge /
003525  164  171  160                .ASCII  /typ/
003530  145  040  045                .ASCII  /e %/
003533  117  062  045                .ASCII  /02%/
```

```
003536  101  040  162            .ASCII  /A r/
003541  145  143  145            .ASCII  /ece/
003544  151  166  145            .ASCII  /ive/
003547  144  040  151            .ASCII  /d i/
003552  156  040  115            .ASCII  /n M/
003555  123  103  120            .ASCII  /SCP/
003560  040  160  141            .ASCII  / pa/
003563  143  153  145            .ASCII  /cke/
003566  164  000                 .ASCII  /t/<00>
003570  045  116  045  P.ABV:    .ASCII  /%N%/
003573  101  052  052            .ASCII  /A**/
003576  040  123  105            .ASCII  / SE/
003601  121  125  105            .ASCII  /QUE/
003604  116  072  040            .ASCII  /N: /
003607  122  105  124            .ASCII  /RET/
003612  120  113  124            .ASCII  /PKT/
003615  040  156  157            .ASCII  / no/
003620  164  040  141            .ASCII  /t a/
003623  166  141  151            .ASCII  /vai/
003626  154  141  142            .ASCII  /lab/
003631  154  145  000            .ASCII  /le/<00>
003634  045  116  045  P.ABW:    .ASCII  /%N%/
003637  101  052  052            .ASCII  /A**/
003642  040  105  162            .ASCII  / Er/
003645  162  157  162            .ASCII  /ror/
003650  040  151  156            .ASCII  / in/
003653  040  123  105            .ASCII  / SE/
003656  124  137  103            .ASCII  /T_C/
003661  124  114  122            .ASCII  /TLR/
003664  137  103  110            .ASCII  /_CH/
003667  101  122  000            .ASCII  /AR/<00>
003672  045  116  045  P.ABX:    .ASCII  /%N%/
003675  101  052  052            .ASCII  /A**/
003700  040  103  164            .ASCII  / Ct/
003703  154  162  040            .ASCII  /lr /
003706  164  151  155            .ASCII  /tim/
003711  145  157  165            .ASCII  /eou/
003714  164  040  075            .ASCII  /t =/
003717  040  045  104            .ASCII  / %D/
003722  063  045  101            .ASCII  /3%A/
003725  056  040  163            .ASCII  /. s/
003730  145  143  157            .ASCII  /eco/
003733  156  144  163            .ASCII  /nds/
003736  000  000                 .ASCII  <00><00>
003740  045  116  045  P.ABY:    .ASCII  /%N%/
003743  101  052  052            .ASCII  /A**/
003746  040  105  162            .ASCII  / Er/
003751  162  157  162            .ASCII  /ror/
003754  040  151  156            .ASCII  / in/
003757  040  125  116            .ASCII  / UN/
003762  111  124  137            .ASCII  /IT_/
003765  111  116  111            .ASCII  /INI/
003770  124  000                 .ASCII  /T/<00>
```

```
003772  045  116  045  P.ABZ:  .ASCII  /%N%/
003775  101  052  052         .ASCII  /A**/
004000  040  125  116         .ASCII  / UN/
004003  111  124  137         .ASCII  /IT_/
004006  111  116  111         .ASCII  /INI/
004011  124  072  040         .ASCII  /T: /
004014  122  105  124         .ASCII  /RET/
004017  120  113  124         .ASCII  /PKT/
004022  040  150  141         .ASCII  / ha/
004025  163  040  142         .ASCII  /s b/
004030  141  144  040         .ASCII  /ad /
004033  105  116  104         .ASCII  /END/
004036  103  117  104         .ASCII  /COD/
004041  105  000  000         .ASCII  /E/<00><00>
004044  045  116  045  P.ACA:  .ASCII  /%N%/
004047  101  052  052         .ASCII  /A**/
004052  040  125  156         .ASCII  / Un/
004055  151  164  040         .ASCII  /it /
004060  163  151  172         .ASCII  /siz/
004063  145  040  050         .ASCII  /e (/
004066  114  157  051         .ASCII  /Lo)/
004071  040  075  040         .ASCII  / = /
004074  045  104  065         .ASCII  /%D5/
004077  045  101  056         .ASCII  /%A./
004102  000  000             .ASCII  <00><00>
004104  045  116  045  P.ACB:  .ASCII  /%N%/
004107  101  052  052         .ASCII  /A**/
004112  040  125  156         .ASCII  / Un/
004115  151  164  040         .ASCII  /it /
004120  163  151  172         .ASCII  /siz/
004123  145  040  050         .ASCII  /e (/
004126  110  151  051         .ASCII  /Hi)/
004131  040  075  040         .ASCII  / = /
004134  045  104  065         .ASCII  /%D5/
004137  045  101  056         .ASCII  /%A./
004142  000  000             .ASCII  <00><00>
004144  045  116  045  P.ACC:  .ASCII  /%N%/
004147  101  052  052         .ASCII  /A**/
004152  040  101  103         .ASCII  / AC/
004155  103  105  123         .ASCII  /CES/
004160  123  072  040         .ASCII  /S: /
004163  122  105  124         .ASCII  /RET/
004166  120  113  124         .ASCII  /PKT/
004171  040  150  141         .ASCII  / ha/
004174  163  040  142         .ASCII  /s b/
004177  141  144  040         .ASCII  /ad /
004202  105  116  104         .ASCII  /END/
004205  103  117  104         .ASCII  /COD/
004210  105  000             .ASCII  /E/<00>
004212  045  116  045  P.ACD:  .ASCII  /%N%/
004215  101  052  052         .ASCII  /A**/
004220  040  121  111         .ASCII  / QI/
004223  117  137  125         .ASCII  /O U/
```

M6


```
004226  116     111     124             .ASCII  /NIT/
004231  072     040     103             .ASCII  /: C/
004234  123     124     040             .ASCII  /ST /
004237  045     104     061             .ASCII  /%D1/
004242  045     101     040             .ASCII  /%A /
004245  156     157     040             .ASCII  /no /
004250  165     156     151             .ASCII  /uni/
004253  164     040     163             .ASCII  /t s/
004256  145     154     145             .ASCII  /ele/
004261  143     164     145             .ASCII  /cte/
004264  144     000                     .ASCII  /d/<00>
004266  045     116     045     P.ACE:  .ASCII  /%N%/
004271  101     052     052             .ASCII  /A**/
004274  040     125     156             .ASCII  / Un/
004277  151     164     040             .ASCII  /it /
004302  043     040     151             .ASCII  /# i/
004305  163     072     040             .ASCII  /s: /
004310  045     117     066             .ASCII  /%06/
004313  000                             .ASCII  <00>
004314  045     116     045     P.ACF:  .ASCII  /%N%/
004317  101     052     052             .ASCII  /A**/
004322  040     122     145             .ASCII  / Re/
004325  155     157     166             .ASCII  /mov/
004330  141     142     154             .ASCII  /abl/
004333  145     040     144             .ASCII  /e d/
004336  151     163     153             .ASCII  /isk/
004341  040     151     163             .ASCII  / is/
004344  040     163     145             .ASCII  / se/
004347  154     145     143             .ASCII  /lec/
004352  164     145     144             .ASCII  /ted/
004355  000                             .ASCII  <00>
004356  045     116     045     P.ACG:  .ASCII  /%N%/
004361  101     052     052             .ASCII  /A**/
004364  040     106     151             .ASCII  / Fi/
004367  170     145     144             .ASCII  /xed/
004372  040     144     151             .ASCII  / di/
004375  163     153     040             .ASCII  /sk /
004400  151     163     040             .ASCII  /is /
004403  163     145     154             .ASCII  /sel/
004406  145     143     164             .ASCII  /ect/
004411  145     144     000             .ASCII  /ed/<00>
004414  045     116     045     P.ACH:  .ASCII  /%N%/
004417  101     052     052             .ASCII  /A**/
004422  040     111     154             .ASCII  / Il/
004425  154     145     147             .ASCII  /leg/
004430  141     154     040             .ASCII  /al /
004433  146     165     156             .ASCII  /fun/
00443x  143     164     151             .ASCII  /cti/
004441  157     156     072             .ASCII  /on:/
004444  040     045     117             .ASCII  / %O/
004447  066     000     000             .ASCII  /6/<00><00>
004452  045     116     045     P.ACI:  .ASCII  /%N%/
004455  101     052     052             .ASCII  /A**/
```

```
004460  040  103  157                .ASCII  / Co/
004463  155  155  141                .ASCII  /mma/
004466  156  144  040                .ASCII  /nd /
004471  162  145  146                .ASCII  /ref/
004474  040  043  040                .ASCII  / # /
004477  045  117  066                .ASCII  /%06/
004502  045  101  057                .ASCII  /%A/<57>
004505  045  117  066                .ASCII  /%06/
004510  045  101  040                .ASCII  /%A /
004513  050  117  143                .ASCII  /(Oc/
004516  164  051  040                .ASCII  /t) /
004521  156  157  164                .ASCII  /not/
004524  040  163  145                .ASCII  / se/
004527  156  164  040                .ASCII  /nt /
004532  142  171  040                .ASCII  /by /
004535  110  157  163                .ASCII  /Hos/
004540  164  000                     .ASCII  /t/<00>
004542  045  116  045     P.ACJ:     .ASCII  /%N%/
004545  101  052  052                .ASCII  /A**/
004550  040  125  156                .ASCII  / Un/
004553  153  156  157                .ASCII  /kno/
004556  167  156  040                .ASCII  /wn /
004561  105  162  162                .ASCII  /Err/
004564  157  162  040                .ASCII  /or /
004567  114  157  147                .ASCII  /Log/
004572  040  146  157                .ASCII  / fo/
004575  162  155  141                .ASCII  /rma/
004600  164  040  045                .ASCII  /t %/
004603  117  063  045                .ASCII  /03%/
004606  101  040  162                .ASCII  /A r/
004611  145  143  145                .ASCII  /ece/
004614  151  166  145                .ASCII  /ive/
004617  144  000  000                .ASCII  /d/<00><00>
004622  045  116  045     P.ACK:     .ASCII  /%N%/
004625  101  052  052                .ASCII  /A**/
004630  040  117  160                .ASCII  / Op/
004633  055  143  157                .ASCII  /-co/
004636  144  145  040                .ASCII  /de /
004641  045  117  063                .ASCII  /%03/
004644  045  101  054                .ASCII  /%A,/
004647  040  105  156                .ASCII  / En/
004652  144  055  143                .ASCII  /d-c/
004655  157  144  145                .ASCII  /ode/
004660  040  045  117                .ASCII  / %O/
004663  063  045  101                .ASCII  /3%A/
004666  040  146  157                .ASCII  / fo/
004671  162  040  162                .ASCII  /r r/
004674  145  146  040                .ASCII  /ef /
004677  043  040  045                .ASCII  /# %/
004702  117  066  045                .ASCII  /06%/
004705  101  057  045                .ASCII  /A/<57>/%/
004710  117  066  045                .ASCII  /06%/
004713  101  040  050                .ASCII  /A (/
```

```
004716  070     051     000             .ASCII  /8)/<00>
004721  000                             .ASCII  <00>
004722  045     116     045     P.ACL:  .ASCII  /%N%/
004725  101     052     052             .ASCII  /A**/
004730  040     103     155             .ASCII  / Cm/
004733  144     055     142             .ASCII  /d-b/
004736  143     040     045             .ASCII  /c %/
004741  117     066     045             .ASCII  /O6%/
004744  101     057     045             .ASCII  /A/<57>/%/
004747  117     066     045             .ASCII  /O6%/
004752  101     040     122             .ASCII  /A R/
004755  163     160     055             .ASCII  /sp-/
004760  142     143     040             .ASCII  /bc /
004763  045     117     066             .ASCII  /%O6/
004766  045     101     057             .ASCII  /%A/<57>
004771  045     117     066             .ASCII  /%O6/
004774  045     101     040             .ASCII  /%A /
004777  146     157     162             .ASCII  /for/
005002  040     045     117             .ASCII  / %O/
005005  066     045     101             .ASCII  /6%A/
005010  057     045     117             .ASCII  <57>/%O/
005013  066     045     101             .ASCII  /6%A/
005016  040     050     070             .ASCII  / (8/
005021  051     000     000             .ASCII  /)/<00><00>
005024  045     116     045     P.ACM:  .ASCII  /%N%/
005027  101     052     052             .ASCII  /A**/
005032  040     122     145             .ASCII  / Re/
005035  163     160     157             .ASCII  /spo/
005040  156     163     145             .ASCII  /nse/
005043  040     141     154             .ASCII  / al/
005046  162     145     141             .ASCII  /rea/
005051  144     171     040             .ASCII  /dy /
005054  162     145     143             .ASCII  /rec/
005057  145     151     166             .ASCII  /eiv/
005062  145     144     040             .ASCII  /ed /
005065  146     157     162             .ASCII  /for/
005070  040     143     155             .ASCII  / cm/
005073  144     040     045             .ASCII  /d %/
005076  117     066     045             .ASCII  /O6%/
005101  101     057     045             .ASCII  /A/<57>/%/
005104  117     066     045             .ASCII  /O6%/
005107  101     040     050             .ASCII  /A (/
005112  070     051     000             .ASCII  /8)/<00>
005115  000                             .ASCII  <00>
005116  045     116     045     P.ACN:  .ASCII  /%N%/
005121  101     052     052             .ASCII  /A**/
005124  040     106     141             .ASCII  / Fa/
005127  151     154     165             .ASCII  /ilu/
005132  162     145     040             .ASCII  /re /
005135  164     157     040             .ASCII  /to /
005140  163     145     156             .ASCII  /sen/
005143  144     040     143             .ASCII  /d c/
005146  157     155     155             .ASCII  /omm/
```

```
005151  141  156  144            .ASCII  /and/
005154  040  141  146            .ASCII  / af/
005157  164  145  162            .ASCII  /ter/
005162  040  043  040            .ASCII  / # /
005165  045  117  066            .ASCII  /%06/
005170  045  101  057            .ASCII  /%A/<57>
005173  045  117  066            .ASCII  /%06/
005176  045  101  040            .ASCII  /%A /
005201  050  070  051            .ASCII  /(8)/
005204  000  000                 .ASCII  <00><00>
005206  045  116  045  P.ACO:    .ASCII  /%N%/
005211  101  125  116            .ASCII  /AUN/
005214  111  124  045            .ASCII  /IT%/
005217  104  062  045            .ASCII  /D2%/
005222  101  040  104            .ASCII  /A D/
005225  122  117  120            .ASCII  /ROP/
005230  120  105  104            .ASCII  /PED/
005233  040  055  040            .ASCII  / - /
005236  000  000                 .ASCII  <00><00>
005240  045  101  125  P.ACQ:    .ASCII  /%AU/
005243  123  105  122            .ASCII  /SER/
005246  040  103  117            .ASCII  / CO/
005251  115  115  101            .ASCII  /MMA/
005254  116  104  045            .ASCII  /ND%/
005257  116  000  000            .ASCII  /N/<00><00>
005262  045  101  103  P.ACR:    .ASCII  /%AC/
005265  117  116  106            .ASCII  /ONF/
005270  111  107  125            .ASCII  /IGU/
005273  122  101  124            .ASCII  /RAT/
005276  111  117  116            .ASCII  /ION/
005301  040  105  122            .ASCII  / ER/
005304  122  117  122            .ASCII  /ROR/
005307  045  116  000            .ASCII  /%N/<00>
005312  045  101  111  P.ACS:    .ASCII  /%AI/
005315  116  111  124            .ASCII  /NIT/
005320  040  105  122            .ASCII  / ER/
005323  122  117  122            .ASCII  /ROR/
005326  045  116  000            .ASCII  /%N/<00>
005331  000                     .ASCII  <00>
005332  045  101  124  P.ACT:    .ASCII  /%AT/
005335  122  101  116            .ASCII  /RAN/
005340  123  106  105            .ASCII  /SFE/
005343  122  040  114            .ASCII  /R L/
005346  111  115  111            .ASCII  /IMI/
005351  124  040  122            .ASCII  /T R/
005354  105  101  103            .ASCII  /EAC/
005357  110  105  104            .ASCII  /HED/
005362  045  116  000            .ASCII  /%N/<00>
005365  000                     .ASCII  <00>
005366  045  101  105  P.ACU:    .ASCII  /%AE/
005371  122  122  117            .ASCII  /RRO/
005374  122  040  114            .ASCII  /R L/
005377  111  115  111            .ASCII  /IMI/
```

```
005402  124  040  122                .ASCII  /T R/
005405  105  101  103                .ASCII  /EAC/
005410  110  105  104                .ASCII  /HED/
005413  045  116  000                .ASCII  /%N/<00>
005416  045  101  125  P.ACV:        .ASCII  /%AU/
005421  116  122  105                .ASCII  /NRE/
005424  103  117  126                .ASCII  /COV/
005427  105  122  101                .ASCII  /ERA/
005432  102  114  105                .ASCII  /BLE/
005435  040  104  122                .ASCII  / DR/
005440  111  126  105                .ASCII  /IVE/
005443  040  105  122                .ASCII  / ER/
005446  122  117  122                .ASCII  /ROR/
005451  045  116  000                .ASCII  /%N/<00>
005454  045  101  125  P.ACW:        .ASCII  /%AU/
005457  116  122  105                .ASCII  /NRE/
005462  103  117  126                .ASCII  /COV/
005465  105  122  101                .ASCII  /ERA/
005470  102  114  105                .ASCII  /BLE/
005473  040  103  117                .ASCII  / CO/
005476  116  124  122                .ASCII  /NTR/
005501  117  114  114                .ASCII  /OLL/
005504  105  122  040                .ASCII  /ER /
005507  105  122  122                .ASCII  /ERR/
005512  117  122  045                .ASCII  /OR%/
005515  116  000  000                .ASCII  /N/<00><00>
005520  045  101  106  P.ACX:        .ASCII  /%AF/
005523  101  111  114                .ASCII  /AIL/
005526  105  104  040                .ASCII  /ED /
005531  124  117  040                .ASCII  /TO /
005534  103  117  115                .ASCII  /COM/
005537  105  040  117                .ASCII  /E O/
005542  116  114  111                .ASCII  /NLI/
005545  116  105  045                .ASCII  /NE%/
005550  116  000                     .ASCII  /N/<00>
005552  045  101  106  P.ACY:        .ASCII  /%AF/
005555  101  111  114                .ASCII  /AIL/
005560  105  104  040                .ASCII  /ED /
005563  124  117  040                .ASCII  /TO /
005566  101  103  103                .ASCII  /ACC/
005571  105  123  123                .ASCII  /ESS/
005574  040  105  111                .ASCII  / FI/
005577  124  110  105                .ASCII  /THE/
005602  122  040  106                .ASCII  /R F/
005605  111  122  123                .ASCII  /IRS/
005610  124  040  117                .ASCII  /T O/
005613  122  040  114                .ASCII  /R L/
005616  101  123  124                .ASCII  /AST/
005621  040  124  122                .ASCII  / TR/
005624  101  103  113                .ASCII  /ACK/
005627  040  104  125                .ASCII  / DU/
005632  122  111  116                .ASCII  /RIN/
005635  107  040  111                .ASCII  /G I/
```

```
005640      116     111     124                 .ASCII  /NIT/
005643      045     116     000                 .ASCII  /%N/<00>
005646      045     101     104     P.ACZ:      .ASCII  /%AD/
005651      111     123     113                 .ASCII  /ISK/
005654      040     127     122                 .ASCII  / WR/
005657      111     124     105                 .ASCII  /ITE/
005662      040     120     122                 .ASCII  / PR/
005665      117     124     105                 .ASCII  /OTE/
005670      103     124     105                 .ASCII  /CTE/
005673      104     045     116                 .ASCII  /D%N/
005676      000     000                         .ASCII  <00><00>
005700      045     101     103     P.ADA:      .ASCII  /%AC/
005703      117     115     115                 .ASCII  /OMM/
005706      101     116     104                 .ASCII  /AND/
005711      040     124     111                 .ASCII  / TI/
005714      115     105     040                 .ASCII  /ME /
005717      117     125     124                 .ASCII  /OUT/
005722      045     116     000                 .ASCII  /%N/<00>
005725      000                                 .ASCII  <00>
005726  005240'                     P.ACP:      .WORD   P.ACQ
005730  005262'                                 .WORD   P.ACR
005732  005312'                                 .WORD   P.ACS
005734  005332'                                 .WORD   P.ACT
005736  005366'                                 .WORD   P.ACU
005740  005416'                                 .WORD   P.ACV
005742  005454'                                 .WORD   P.ACW
005744  005520'                                 .WORD   P.ACX
005746  005552'                                 .WORD   P.ACY
005750  005646'                                 .WORD   P.ACZ
005752  005700'                                 .WORD   P.ADA
005754      045     116     045     P.ADB:      .ASCII  /%N%/
005757      101     120     117                 .ASCII  /APO/
005762      127     105     122                 .ASCII  /WER/
005765      040     104     105                 .ASCII  / DE/
005770      114     101     131                 .ASCII  /LAY/
005773      040     055     040                 .ASCII  / - /
005776      127     101     111                 .ASCII  /WAI/
006001      124     111     116                 .ASCII  /TIN/
006004      107     000                         .ASCII  /G/<00>
006006      045     116     045     P.ADC:      .ASCII  /%N%/
006011      101     106     125                 .ASCII  /AFU/
006014      116     103     124                 .ASCII  /NCT/
006017      111     117     116                 .ASCII  /ION/
006022      101     114     040                 .ASCII  /AL /
006025      124     105     123                 .ASCII  /TES/
006030      124     040     123                 .ASCII  /T S/
006033      124     101     122                 .ASCII  /TAR/
006036      124     105     104                 .ASCII  /TED/
006041      000                                 .ASCII  <00>
006042      045     116     045     P.ADD:      .ASCII  /%N%/
006045      116     045     101                 .ASCII  /N%A/
006050      105     130     105                 .ASCII  /EXE/
006053      122     103     111                 .ASCII  /RCI/
```

```
006056  123  105  122                .ASCII  /SER/
006061  040  123  124                .ASCII  / ST/
006064  101  122  124                .ASCII  /ART/
006067  105  104  045                .ASCII  /ED%/
006072  116  0C0                    .ASCII  /N/<00>
006074  045  116  045     P.ADE:     .ASCII  /%N%/
006077  116  045  101                .ASCII  /N%A/
006102  125  116  124                .ASCII  /UNT/
006105  040  104  123                .ASCII  / DS/
006110  113  045  123                .ASCII  /K%S/
006113  070  045  101                .ASCII  /8%A/
006116  043  040  117                .ASCII  /# O/
006121  106  040  040                .ASCII  /F  /
006124  040  043  040                .ASCII  / # /
006127  102  131  124                .ASCII  /BYT/
006132  105  123  040                .ASCII  /ES /
006135  040  040  043                .ASCII  /  #/
006140  040  117  106                .ASCII  / OF/
006143  040  040  040                .ASCII  /   /
006146  040  043  040                .ASCII  / # /
006151  102  131  124                .ASCII  /BYT/
006154  105  123  000                .ASCII  /ES/<00>
006157  000                        .ASCII  <00>
006160  045  101  040     P.ADF:     .ASCII  /%A /
006163  040  055  055                .ASCII  / --/
006166  110  101  122                .ASCII  /HAR/
006171  104  040  105                .ASCII  /D E/
006174  122  122  117                .ASCII  /RRO/
006177  122  123  055                .ASCII  /RS-/
006202  055  040  055                .ASCII  /- -/
006205  055  123  117                .ASCII  /-SO/
006210  106  124  040                .ASCII  /FT /
006213  105  122  122                .ASCII  /ERR/
006216  117  122  123                .ASCII  /ORS/
006221  055  055  000                .ASCII  /--/<00>
006224  045  116  045     P.ADG:     .ASCII  /%N%/
006227  101  040  043                .ASCII  /A #/
006232  040  040  040                .ASCII  /   /
006235  043  040  040                .ASCII  /#  /
006240  124  131  120                .ASCII  /TYP/
006243  105  040  040                .ASCII  /E  /
006246  122  105  101                .ASCII  /REA/
006251  104  123  040                .ASCII  /DS /
006254  040  040  040                .ASCII  /   /
006257  040  122  105                .ASCII  / RE/
006262  101  104  040                .ASCII  /AD /
006265  040  040  127                .ASCII  /  W/
006270  122  111  124                .ASCII  /RIT/
006273  105  123  040                .ASCII  /ES /
006276  040  040  127                .ASCII  /  W/
006301  122  111  124                .ASCII  /RIT/
006304  124  105  116                .ASCII  /TEN/
006307  000                        .ASCII  <00>
```

```
006310  045  101  040  P.ADH:  .ASCII  /%A /
006313  040  123  105         .ASCII  / SE/
006316  113  040  104         .ASCII  /K D/
006321  101  124  040         .ASCII  /AT /
006324  104  122  126         .ASCII  /DRV/
006327  040  110  123         .ASCII  / HS/
006332  124  040  123         .ASCII  /T S/
006335  105  113  040         .ASCII  /EK /
006340  104  101  124         .ASCII  /DAT/
006343  040  104  122         .ASCII  / DR/
006346  126  040  110         .ASCII  /V H/
006351  123  124  000         .ASCII  /ST/<00>
006354  045  116  045  P.ADI:  .ASCII  /%N%/
006357  101  055  055         .ASCII  /A--/
006362  055  040  055         .ASCII  /- -/
006365  055  055  040         .ASCII  /-- /
006370  055  055  055         .ASCII  /---/
006373  055  040  040         .ASCII  /-  /
006376  055  055  055         .ASCII  /---/
006401  055  055  040         .ASCII  /-- /
006404  040  055  055         .ASCII  / --/
006407  055  055  055         .ASCII  /---/
006412  055  055  055         .ASCII  /---/
006415  055  040  055         .ASCII  /- -/
006420  055  055  055         .ASCII  /---/
006423  055  055  040         .ASCII  /-- /
006426  040  055  055         .ASCII  / --/
006431  055  055  055         .ASCII  /---/
006434  055  055  055         .ASCII  /---/
006437  055  000  000         .ASCII  /-/<00><00>
006442  045  101  040  P.ADJ:  .ASCII  /%A /
006445  055  055  055         .ASCII  /---/
006450  040  055  055         .ASCII  / --/
006453  055  040  055         .ASCII  /- -/
006456  055  055  040         .ASCII  /-- /
006461  055  055  055         .ASCII  /---/
006464  040  055  055         .ASCII  / --/
006467  055  040  055         .ASCII  /- -/
006472  055  055  040         .ASCII  /-- /
006475  055  055  055         .ASCII  /---/
006500  040  055  055         .ASCII  / --/
006503  055  000  000         .ASCII  /-/<00><00>
006506  045  116  045  P.ADK:  .ASCII  /%N%/
006511  104  062  045         .ASCII  /D2%/
006514  104  064  045         .ASCII  /D4%/
006517  123  062  045         .ASCII  /S2%/
006522  124  000              .ASCII  /T/<00>
006524  045  104  064  P.ADL:  .ASCII  /%D4/
006527  045  132  063         .ASCII  /%Z3/
006532  045  104  063         .ASCII  /%D3/
006535  045  101  054         .ASCII  /%A,/
006540  045  132  063         .ASCII  /%Z3/
006543  045  101  054         .ASCII  /%A,/
```

```
006546  045  132  063          .ASCII  /%Z3/
006551  000                    .ASCII  <00>
006552  045  104  064  P.ADM:  .ASCII  /%D4/
006555  045  104  064          .ASCII  /%D4/
006560  045  104  064          .ASCII  /%D4/
006563  045  104  064          .ASCII  /%D4/
006566  045  104  064          .ASCII  /%D4/
006571  045  104  064          .ASCII  /%D4/
006574  045  104  064          .ASCII  /%D4/
006577  045  104  064          .ASCII  /%D4/
006602  000  000               .ASCII  <00><00>
006604  045  116  045  P.ADN:  .ASCII  /%N%/
006607  101  040  056          .ASCII  /A ./
006612  040  040  040          .ASCII  /   /
006615  056  040  040          .ASCII  /.  /
006620  103  116  124          .ASCII  /CNT/
006623  122  040  040          .ASCII  /R  /
006626  040  040  040          .ASCII  /   /
006631  040  056  040          .ASCII  / . /
006634  040  056  056          .ASCII  / ../
006637  056  056  056          .ASCII  /.../
006642  056  056  056          .ASCII  /.../
006645  056  040  040          .ASCII  /.  /
006650  040  040  040          .ASCII  /   /
006653  040  056  040          .ASCII  / . /
006656  040  056  056          .ASCII  / ../
006661  056  056  056          .ASCII  /.../
006664  056  056  056          .ASCII  /.../
006667  056  000  000          .ASCII  /./<00><00>
006672  045  101  040  P.ADO:  .ASCII  /%A /
006675  040  040  056          .ASCII  /  ./
006700  040  040  040          .ASCII  /   /
006703  056  045  104          .ASCII  /.%D/
006706  064  045  101          .ASCII  /4%A/
006711  040  040  040          .ASCII  /   /
006714  056  040  040          .ASCII  /.  /
006717  040  056  040          .ASCII  / . /
006722  040  040  056          .ASCII  /  ./
006725  045  104  064          .ASCII  /%D4/
006730  045  101  040          .ASCII  /%A /
006733  040  040  056          .ASCII  /  ./
006736  000  000               .ASCII  <00><00>
006740  045  101  040  P.ADP:  .ASCII  /%A /
006743  040  040  056          .ASCII  /  ./
006746  040  040  040          .ASCII  /   /
006751  056  045  104          .ASCII  /.%D/
006753  064  045  101          .ASCII  /4%A/
006757  040  040  040          .ASCII  /   /
006762  056  040  040          .ASCII  /.  /
006765  040  056  040          .ASCII  / . /
006770  040  040  056          .ASCII  /  ./
006773  045  104  064          .ASCII  /%D4/
006776  045  101  040          .ASCII  /%A /
```

```
007001  040  040  056          .ASCII  / ./
007004  000  000               .ASCII  <00><00>
007006  045  116  045  P.ADQ:   .ASCII  /%N%/
007011  116  045  101          .ASCII  /N%A/
007014  125  116  111          .ASCII  /UNI/
007017  124  040  040          .ASCII  /T  /
007022  104  111  123          .ASCII  /DIS/
007025  113  040  040          .ASCII  /K  /
007030  040  040  040          .ASCII  /   /
007033  040  040  040          .ASCII  /   /
007036  040  040  040          .ASCII  /   /
007041  040  043  040          .ASCII  / # /
007044  117  106  040          .ASCII  /OF /
007047  040  040  043          .ASCII  /  #/
007052  040  102  114          .ASCII  / BL/
007055  113  123  040          .ASCII  /KS /
007060  040  040  040          .ASCII  /   /
007063  040  040  040          .ASCII  /   /
007066  043  040  117          .ASCII  /# O/
007071  106  040  040          .ASCII  /F  /
007074  040  040  043          .ASCII  /  #/
007077  040  102  114          .ASCII  / BL/
007102  113  123  040          .ASCII  /KS /
007105  000                    .ASCII  <00>
007106  045  116  045  P.ADR:   .ASCII  /%N%/
007111  101  040  040          .ASCII  /A  /
007114  043  040  040          .ASCII  /#  /
007117  040  040  040          .ASCII  /   /
007122  043  040  040          .ASCII  /#  /
007125  040  040  040          .ASCII  /   /
007130  124  131  120          .ASCII  /TYP/
007133  105  040  040          .ASCII  /E  /
007136  040  122  105          .ASCII  / RE/
007141  101  104  123          .ASCII  /ADS/
007144  040  040  040          .ASCII  /   /
007147  040  040  122          .ASCII  /  R/
007152  105  101  104          .ASCII  /EAD/
007155  040  040  040          .ASCII  /   /
007160  040  040  040          .ASCII  /   /
007163  127  122  111          .ASCII  /WRI/
007166  124  105  123          .ASCII  /TES/
007171  040  040  127          .ASCII  /  W/
007174  122  111  124          .ASCII  /RIT/
007177  124  105  116          .ASCII  /TEN/
007202  040  000               .ASCII  / /<00>
007204  045  116  045  P.ADS:   .ASCII  /%N%/
007207  101  055  055          .ASCII  /A--/
007212  055  055  040          .ASCII  /-- /
007215  040  055  055          .ASCII  / --/
007220  055  055  040          .ASCII  /-- /
007223  040  055  055          .ASCII  / --/
007226  055  055  055          .ASCII  /---/
007231  055  055  040          .ASCII  /-- /
```

```
007234  040  055  055              .ASCII  / - /
007237  055  055  055              .ASCII  /---/
007242  055  040  040              .ASCII  /-  /
007245  040  055  055              .ASCII  / --/
007250  055  055  055              .ASCII  /---/
007253  055  040  040              .ASCII  /-  /
007256  040  040  040              .ASCII  /   /
007261  055  055  055              .ASCII  /---/
007264  055  055  055              .ASCII  /---/
007267  040  040  040              .ASCII  /   /
007272  055  055  055              .ASCII  /---/
007275  055  055  055              .ASCII  /---/
007300  040  040  000              .ASCII  /  /<00>
007303  000                       .ASCII  <00>
007304  045  116  045  P.ADT:      .ASCII  /%N%/
007307  123  061  045              .ASCII  /S1%/
007312  104  062  045              .ASCII  /D2%/
007315  123  064  045              .ASCII  /S4%/
007320  104  062  045              .ASCII  /D2%/
007323  101  040  040              .ASCII  /A  /
007326  040  104  102              .ASCII  / DB/
007331  116  040  111              .ASCII  /N I/
007334  057  117  040              .ASCII  <57>/O /
007337  040  045  104              .ASCII  / %D/
007342  066  045  123              .ASCII  /6%S/
007345  063  045  104              .ASCII  /3%D/
007350  066  045  123              .ASCII  /6%S/
007353  065  045  104              .ASCII  /5%D/
007356  066  045  123              .ASCII  /6%S/
007361  063  045  104              .ASCII  /3%D/
007364  066  000                   .ASCII  /6/<00>
007366  124  117  117  P.ADU:      .ASCII  /TOO/
007371  040  115  101              .ASCII  / MA/
007374  116  131  040              .ASCII  /NY /
007377  125  116  111              .ASCII  /UNI/
007402  124  123  000              .ASCII  /TS/<00>
007405  000                       .ASCII  <00>
007406  116  117  124  P.ADV:      .ASCII  /NOT/
007411  040  105  116              .ASCII  / EN/
007414  117  125  107              .ASCII  /OUG/
007417  110  040  106              .ASCII  /H F/
007422  122  105  105              .ASCII  /REE/
007425  040  115  105              .ASCII  / ME/
007430  115  117  122              .ASCII  /MOR/
007433  131  040  106              .ASCII  /Y F/
007436  117  122  040              .ASCII  /OR /
007441  101  114  114              .ASCII  /ALL/
007444  117  103  101              .ASCII  /OCA/
007447  124  111  116              .ASCII  /TIN/
007452  107  040  122              .ASCII  /G R/
007455  105  101  104              .ASCII  /EAD/
007460  057  127  122              .ASCII  <57>/WR/
007463  111  124  105              .ASCII  /ITE/
```

```
007466  040  102  125            .ASCII  / BU/
007471  106  106  105            .ASCII  /FFE/
007474  122  123  000            .ASCII  /RS/<00>
007477  000                      .ASCII  <00>
007500  122  105  107  P.ADW:    .ASCII  /REG/
007503  111  123  124            .ASCII  /IST/
007506  105  122  040            .ASCII  /ER /
007511  105  130  111            .ASCII  /EXI/
007514  123  124  105            .ASCII  /STE/
007517  116  103  105            .ASCII  /NCE/
007522  040  124  105            .ASCII  / TE/
007525  123  124  040            .ASCII  /ST /
007530  106  101  111            .ASCII  /FAI/
007533  114  105  104            .ASCII  /LED/
007536  000  000                 .ASCII  <00><00>
007540  126  105  103  P.ADX:    .ASCII  /VEC/
007543  124  117  122            .ASCII  /TOR/
007546  040  124  105            .ASCII  / TE/
007551  123  124  040            .ASCII  /ST /
007554  106  101  111            .ASCII  /FAI/
007557  114  105  104            .ASCII  /LED/
007562  000  000                 .ASCII  <00><00>
007564  102  122  040  P.ADY:    .ASCII  /BR /
007567  114  105  126            .ASCII  /LEV/
007572  105  114  040            .ASCII  /EL /
007575  124  105  123            .ASCII  /TES/
007600  124  040  106            .ASCII  /T F/
007603  101  111  114            .ASCII  /AIL/
007606  105  104  000            .ASCII  /ED/<00>
007611  000                      .ASCII  <00>
007612  111  116  111  P.ADZ:    .ASCII  /INI/
007615  124  040  123            .ASCII  /T S/
007620  105  121  125            .ASCII  /EQU/
007623  105  116  103            .ASCII  /ENC/
007626  105  040  106            .ASCII  /E F/
007631  101  111  114            .ASCII  /AIL/
007634  105  104  000            .ASCII  /ED/<00>
007637  000                      .ASCII  <00>
007640  106  101  124  P.AEA:    .ASCII  /FAT/
007643  101  114  040            .ASCII  /AL /
007646  103  117  116            .ASCII  /CON/
007651  124  122  117            .ASCII  /TRO/
007654  114  114  105            .ASCII  /LLE/
007657  122  040  105            .ASCII  /R E/
007662  122  122  117            .ASCII  /RRO/
007665  122  000  000            .ASCII  /R/<00><00>
007670  117  116  114  P.AEB:    .ASCII  /ONL/
007673  111  116  105            .ASCII  /INE/
007676  040  106  101            .ASCII  / FA/
007701  111  114  105            .ASCII  /ILE/
007704  104  000                 .ASCII  /D/<00>
007706  127  122  111  P.AEC:    .ASCII  /WRI/
007711  124  105  055            .ASCII  /TE-/
```

```
007714  120  122  117          .ASCII  /PRO/
007717  124  105  103          .ASCII  /TEC/
007722  124  040  103          .ASCII  /T C/
007725  117  116  106          .ASCII  /ONF/
007730  114  111  103          .ASCII  /LIC/
007733  124  000  000          .ASCII  /T/<00><00>
007736  101  103  103  P.AED:  .ASCII  /ACC/
007741  105  123  123          .ASCII  /ESS/
007744  040  106  101          .ASCII  / FA/
007747  111  114  105          .ASCII  /ILE/
007752  104  000               .ASCII  /D/<00>
007754  106  101  124  P.AEE:  .ASCII  /FAT/
007757  101  114  040          .ASCII  /AL /
007762  111  057  117          .ASCII  /I/<57>/O/
007765  040  105  122          .ASCII  / ER/
007770  122  117  122          .ASCII  /ROR/
007773  000                    .ASCII  <00>
007774  104  111  123  P.AEF:  .ASCII  /DIS/
007777  113  040  124          .ASCII  /K T/
010002  131  120  105          .ASCII  /YPE/
010005  040  125  116          .ASCII  / UN/
010010  113  116  117          .ASCII  /KNO/
010013  127  116  040          .ASCII  /WN /
010016  124  117  040          .ASCII  /TO /
010021  105  130  105          .ASCII  /EXE/
010024  122  103  111          .ASCII  /RCI/
010027  123  105  122          .ASCII  /SER/
010032  000  000               .ASCII  <00><00>
010034  106  101  111  P.AEG:  .ASCII  /FAI/
010037  114  105  104          .ASCII  /LED/
010042  040  124  117          .ASCII  / TO/
010045  040  123  105          .ASCII  / SE/
010050  116  104  040          .ASCII  /ND /
010053  123  105  124          .ASCII  /SET/
010056  055  103  117          .ASCII  /-CO/
010061  116  124  122          .ASCII  /NTR/
010064  117  114  114          .ASCII  /OLL/
010067  105  122  055          .ASCII  /ER-/
010072  103  110  101          .ASCII  /CHA/
010075  122  101  103          .ASCII  /RAC/
010100  124  105  122          .ASCII  /TER/
010103  111  123  124          .ASCII  /IST/
010106  111  103  123          .ASCII  /ICS/
010111  040  103  117          .ASCII  / CO/
010114  115  115  101          .ASCII  /MMA/
010117  116  104  000          .ASCII  /ND/<00>
010122  123  105  124  P.AEH:  .ASCII  /SET/
010125  055  103  117          .ASCII  /-CO/
010130  116  124  122          .ASCII  /NTR/
010133  117  114  114          .ASCII  /OLL/
010136  105  122  055          .ASCII  /ER-/
010141  103  110  101          .ASCII  /CHA/
010144  122  101  103          .ASCII  /RAC/
```

```
010147  124     105     122             .ASCII  /TER/
010152  111     123     124             .ASCII  /IST/
010155  111     103     123             .ASCII  /ICS/
010160  040     122     105             .ASCII  / RE/
010163  123     120     117             .ASCII  /SPO/
010166  116     123     105             .ASCII  /NSE/
010171  040     110     101             .ASCII  / HA/
010174  123     040     102             .ASCII  /S B/
010177  101     104     040             .ASCII  /AD /
010202  105     116     104             .ASCII  /END/
010205  103     117     104             .ASCII  /COD/
010210  105     040     117             .ASCII  /E O/
010213  122     040     106             .ASCII  /R F/
010216  114     101     107             .ASCII  /LAG/
010221  123     040     111             .ASCII  /S I/
010224  116     040     105             .ASCII  /N E/
010227  122     122     117             .ASCII  /RRO/
010232  122     000                     .ASCII  /R/<00>
010234  106     101     111     P.AEI:  .ASCII  /FAI/
010237  114     105     104             .ASCII  /LED/
010242  040     124     117             .ASCII  / TO/
010245  040     123     105             .ASCII  / SE/
010250  116     104     040             .ASCII  /ND /
010253  117     116     055             .ASCII  /ON-/
010256  114     111     116             .ASCII  /LIN/
010261  105     040     103             .ASCII  /E C/
010264  117     115     115             .ASCII  /OMM/
010267  101     116     104             .ASCII  /AND/
010272  000     000                     .ASCII  <00><00>
010274  117     116     055     P.AEJ:  .ASCII  /ON-/
010277  114     111     116             .ASCII  /LIN/
010302  105     040     122             .ASCII  /E R/
010305  105     123     120             .ASCII  /ESP/
010310  117     116     123             .ASCII  /ONS/
010313  105     040     110             .ASCII  /E H/
010316  101     123     040             .ASCII  /AS /
010321  102     101     104             .ASCII  /BAD/
010324  040     105     116             .ASCII  / EN/
010327  104     103     117             .ASCII  /DCO/
010332  104     105     000             .ASCII  /DE/<00>
010335  000                             .ASCII  <00>
010336  117     116     055     P.AEK:  .ASCII  /ON-/
010341  114     111     116             .ASCII  /LIN/
010344  105     040     122             .ASCII  /E R/
010347  105     123     120             .ASCII  /ESP/
010352  117     116     123             .ASCII  /ONS/
010355  105     040     110             .ASCII  /E H/
010360  101     123     040             .ASCII  /AS /
010363  125     116     113             .ASCII  /UNK/
010366  116     117     127             .ASCII  /NOW/
010371  116     040     104             .ASCII  /N D/
010374  105     126     111             .ASCII  /EVI/
010377  103     105     000             .ASCII  /CE/<00>
```

```
010402  111  057  117        P.AEL:  .ASCII  /I/<57>/O/
010405  040  122  105                .ASCII  / RE/
010410  121  125  105                .ASCII  /QUE/
010413  123  124  040                .ASCII  /ST /
010416  106  101  111                .ASCII  /FAI/
010421  114  105  104                .ASCII  /LED/
010424  000  000                     .ASCII  <00><00>
010426  045  101  115        P.AEM:  .ASCII  /%AM/
010431  117  122  105                .ASCII  /ORE/
010434  040  124  110                .ASCII  / TH/
010437  101  116  040                .ASCII  /AN /
010442  045  104  062                .ASCII  /%D2/
010445  045  101  040                .ASCII  /%A /
010450  125  116  111                .ASCII  /UNI/
010453  124  123  040                .ASCII  /TS /
010456  123  120  105                .ASCII  /SPE/
010461  103  111  106                .ASCII  /CIF/
010464  111  105  104                .ASCII  /IED/
010467  000                         .ASCII  <00>
010470  045  101  052        P.AEN:  .ASCII  /%A*/
010473  040  116  117                .ASCII  / NO/
010476  040  122  105                .ASCII  / RE/
010501  123  120  117                .ASCII  /SPO/
010504  116  123  105                .ASCII  /NSE/
010507  040  101  124                .ASCII  / AT/
010512  040  101  104                .ASCII  / AD/
010515  104  122  105                .ASCII  /DRE/
010520  123  123  040                .ASCII  /SS /
010523  045  117  066                .ASCII  /%O6/
010526  000  000                     .ASCII  <00><00>
010530  045  101  052        P.AEO:  .ASCII  /%A*/
010533  040  111  116                .ASCII  / IN/
010536  103  117  122                .ASCII  /COR/
010541  122  105  103                .ASCII  /REC/
010544  124  040  102                .ASCII  /T B/
010547  122  040  114                .ASCII  /R L/
010552  105  126  105                .ASCII  /EVE/
010555  114  040  106                .ASCII  /L F/
010560  117  122  040                .ASCII  /OR /
010563  104  122  111                .ASCII  /DRI/
010566  126  105  040                .ASCII  /VE /
010571  045  117  066                .ASCII  /%O6/
010574  000  000                     .ASCII  <00><00>
010576  045  101  052        P.AEP:  .ASCII  /%A*/
010601  040  123  124                .ASCII  / ST/
010604  105  120  040                .ASCII  /EP /
010607  045  104  061                .ASCII  /%D1/
010612  045  101  040                .ASCII  /%A /
010615  122  105  101                .ASCII  /REA/
010620  104  040  105                .ASCII  /D E/
010623  122  122  117                .ASCII  /RRO/
010626  122  000                     .ASCII  /R/<CO>
010630  045  101  052        P.AEQ:  .ASCII  /%A*/
```

```
010633  040  102  101            .ASCII  / BA/
010636  104  040  123            .ASCII  /D S/
010641  101  040  103            .ASCII  /A C/
010644  117  104  105            .ASCII  /ODE/
010647  040  106  122            .ASCII  / FR/
010652  117  115  040            .ASCII  /OM /
010655  104  122  111            .ASCII  /DRI/
010660  126  105  040            .ASCII  /VE /
010663  045  117  066            .ASCII  /%O6/
010666  000  000                 .ASCII  <00><00>
010670  045  101  052  P.AER:     .ASCII  /%A*/
010673  040  104  111            .ASCII  / DI/
010676  123  113  045            .ASCII  /SK%/
010701  104  062  045            .ASCII  /D2%/
010704  101  040  127            .ASCII  /A W/
010707  105  116  124            .ASCII  /ENT/
010712  040  117  106            .ASCII  / OF/
010715  106  114  111            .ASCII  /FLI/
010720  116  105  000            .ASCII  /NE/<00>
010723  000                      .ASCII  <00>
010724  045  101  052  P.AES:     .ASCII  /%A*/
010727  040  104  122            .ASCII  / DR/
010732  111  126  105            .ASCII  /IVE/
010735  040  045  117            .ASCII  / %O/
010740  066  045  101            .ASCII  /6%A/
010743  040  116  117            .ASCII  / NO/
010746  124  040  120            .ASCII  /T P/
010751  122  117  103            .ASCII  /ROC/
010754  105  123  123            .ASCII  /ESS/
010757  111  116  107            .ASCII  /ING/
010762  040  103  117            .ASCII  / CO/
010765  115  115  101            .ASCII  /MMA/
010770  116  104  040            .ASCII  /ND /
010773  120  101  103            .ASCII  /PAC/
010776  113  105  124            .ASCII  /KET/
011001  123  000  000            .ASCII  /S/<00><00>
011004  045  101  052  P.AET:     .ASCII  /%A*/
011007  040  104  111            .ASCII  / DI/
011012  123  113  045            .ASCII  /SK%/
011015  104  062  045            .ASCII  /D2%/
011020  101  040  127            .ASCII  /A W/
011023  105  116  124            .ASCII  /ENT/
011026  040  124  117            .ASCII  / TO/
011031  040  124  110            .ASCII  / TH/
011034  105  040  042            .ASCII  /E "/
011037  101  126  101            .ASCII  /AVA/
011042  111  114  101            .ASCII  /ILA/
011045  102  114  105            .ASCII  /BLE/
011050  042  040  123            .ASCII  /" S/
011053  124  101  124            .ASCII  /TAT/
011056  105  000                 .ASCII  /E/<00>
011060  040  055  040  P.AEU:     .ASCII  / - /
011063  125  116  122            .ASCII  /UNR/
```

```
011066  105  103  117            .ASCII  /ECO/
011071  107  116  111            .ASCII  /GNI/
011074  132  105  104            .ASCII  /ZED/
011077  040  115  105            .ASCII  / ME/
011102  123  123  101            .ASCII  /SSA/
011105  107  105  040            .ASCII  /GE /
011110  124  131  120            .ASCII  /TYP/
011113  105  000  000            .ASCII  /E/<00><00>
011116  040  055  040  P.AEV:    .ASCII  / - /
011121  125  116  122            .ASCII  /UNR/
011124  105  103  117            .ASCII  /ECO/
011127  107  116  111            .ASCII  /GNI/
011132  132  105  104            .ASCII  /ZED/
011135  040  103  117            .ASCII  / CO/
011140  116  116  105            .ASCII  /NNE/
011143  103  124  111            .ASCII  /CTI/
011146  117  116  040            .ASCII  /ON /
011151  111  104  000            .ASCII  /ID/<00>
011154  040  055  040  P.AEW:    .ASCII  / - /
011157  125  116  122            .ASCII  /UNR/
011162  105  103  117            .ASCII  /ECO/
011165  107  116  111            .ASCII  /GNI/
011170  132  105  104            .ASCII  /ZED/
011173  040  122  105            .ASCII  / RE/
011176  124  125  122            .ASCII  /TUR/
011201  116  040  115            .ASCII  /N M/
011204  105  123  123            .ASCII  /ESS/
011207  101  107  105            .ASCII  /AGE/
011212  000  000                 .ASCII  <00><00>
011214  040  055  040  P.AEX:    .ASCII  / - /
011217  125  116  122            .ASCII  /UNR/
011222  105  103  117            .ASCII  /ECO/
011225  107  116  111            .ASCII  /GNI/
011230  132  105  104            .ASCII  /ZED/
011233  040  122  105            .ASCII  / RE/
011236  124  125  122            .ASCII  /TUR/
011241  116  040  120            .ASCII  /N P/
011244  101  103  113            .ASCII  /ACK/
011247  105  124  000            .ASCII  /ET/<00>
011252  040  055  040  P.AEY:    .ASCII  / - /
011255  125  116  122            .ASCII  /UNR/
011260  105  103  117            .ASCII  /ECO/
011263  107  116  111            .ASCII  /GNI/
011266  132  105  104            .ASCII  /ZED/
011271  040  103  122            .ASCII  / CR/
011274  116  000                 .ASCII  /N/<00>
011276  040  055  040  P.AEZ:    .ASCII  / - /
011301  125  116  122            .ASCII  /UNR/
011304  105  103  117            .ASCII  /ECO/
011307  107  116  111            .ASCII  /GNI/
011312  132  105  104            .ASCII  /ZED/
011315  040  117  120            .ASCII  / OP/
011320  103  117  104            .ASCII  /COD/
```

```
011323  105  000  000                .ASCII  /E/<00><00>
011326  040  055  040      P.AFA:    .ASCII  / - /
011331  115  123  103                .ASCII  /MSC/
011334  120  040  123                .ASCII  /P S/
011337  124  101  124                .ASCII  /TAT/
011342  125  123  040                .ASCII  /US /
011345  103  117  104                .ASCII  /COD/
011350  105  040  105                .ASCII  /E E/
011353  122  122  000                .ASCII  /RR/<00>
011356  040  055  040      P.AFB:    .ASCII  / - /
011361  104  125  120                .ASCII  /DUP/
011364  040  123  124                .ASCII  / ST/
011367  101  124  125                .ASCII  /ATU/
011372  123  040  103                .ASCII  /S C/
011375  117  104  105                .ASCII  /ODE/
011400  040  105  122                .ASCII  / ER/
011403  122  000  000                .ASCII  /R/<00><00>
011406  040  055  040      P.AFC:    .ASCII  / - /
011411  125  116  122                .ASCII  /UNR/
011414  105  103  117                .ASCII  /ECO/
011417  107  116  111                .ASCII  /GNI/
011422  132  105  104                .ASCII  /ZED/
011425  040  123  124                .ASCII  / ST/
011430  101  124  125                .ASCII  /ATU/
011433  123  040  103                .ASCII  /S C/
011436  117  104  105                .ASCII  /ODE/
011441  000                         .ASCII  <00>
011442  040  055  040      P.AFD:    .ASCII  / - /
011445  114  102  116                .ASCII  /LBN/
011450  040  110  117                .ASCII  / HO/
011453  123  124  040                .ASCII  /ST /
011456  103  117  115                .ASCII  /COM/
011461  120  101  122                .ASCII  /PAR/
011464  105  040  105                .ASCII  /E E/
011467  122  122  000                .ASCII  /RR/<00>
011472  040  055  040      P.AFE:    .ASCII  / - /
011475  104  102  116                .ASCII  /DBN/
011500  040  110  117                .ASCII  / HO/
011503  123  124  040                .ASCII  /ST /
011506  103  117  115                .ASCII  /COM/
011511  120  101  122                .ASCII  /PAR/
011514  105  040  105                .ASCII  /E E/
011517  122  122  000                .ASCII  /RR/<00>
011522  040  055  040      P.AFF:    .ASCII  / - /
011525  125  116  101                .ASCII  /UNA/
011530  102  114  105                .ASCII  /BLE/
011533  040  124  117                .ASCII  / TO/
011536  040  114  117                .ASCII  / LO/
011541  101  104  040                .ASCII  /AD /
011544  104  125  120                .ASCII  /DUP/
011547  040  115  105                .ASCII  / ME/
011552  104  111  101                .ASCII  /DIA/
011555  000                         .ASCII  <00>
```

```
011556  040  055  040        P.AFG:  .ASCII  / - /
011561  105  122  122                .ASCII  /ERR/
011564  040  111  116                .ASCII  / IN/
011567  040  104  125                .ASCII  / DU/
011572  120  040  120                .ASCII  /P P/
011575  113  124  040                .ASCII  /KT /
011600  127  110  105                .ASCII  /WHE/
011603  116  040  125                .ASCII  /N U/
011606  123  111  116                .ASCII  /SIN/
011611  107  040  103                .ASCII  /G C/
011614  124  114  122                .ASCII  /TLR/
011617  040  114  103                .ASCII  / LC/
011622  040  120  122                .ASCII  / PR/
011625  117  107  000                .ASCII  /OG/<00>
011630  045  101  052        P.AFH:  .ASCII  /%A*/
011633  040  104  111                .ASCII  / DI/
011636  123  113  045                .ASCII  /SK%/
011641  104  062  000                .ASCII  /D2/<00>
011644  045  101  111        P.AFJ:  .ASCII  /%AI/
011647  116  126  101                .ASCII  /NVA/
011652  114  111  104                .ASCII  /LID/
011655  040  103  117                .ASCII  / CO/
011660  115  115  101                .ASCII  /MMA/
011663  116  104  000                .ASCII  /ND/<00>
011666  045  101  103        P.AFK:  .ASCII  /%AC/
011671  117  115  115                .ASCII  /OMM/
011674  101  116  104                .ASCII  /AND/
011677  040  101  102                .ASCII  / AB/
011702  117  122  124                .ASCII  /ORT/
011705  105  104  000                .ASCII  /ED/<00>
011710  045  101  125        P.AFL:  .ASCII  /%AU/
011713  116  111  124                .ASCII  /NIT/
011716  040  117  106                .ASCII  / OF/
011721  106  114  111                .ASCII  /FLI/
011724  116  105  000                .ASCII  /NE/<00>
011727  000                         .ASCII  <00>
011730  045  101  124        P.AFM:  .ASCII  /%AT/
011733  122  101  116                .ASCII  /RAN/
011736  123  111  124                .ASCII  /SIT/
011741  111  117  116                .ASCII  /ION/
011744  040  124  117                .ASCII  / TO/
011747  040  101  126                .ASCII  / AV/
011752  101  111  114                .ASCII  /AIL/
011755  101  102  114                .ASCII  /ABL/
011760  105  040  123                .ASCII  /E S/
011763  124  101  124                .ASCII  /TAT/
011766  105  000                    .ASCII  /E/<00>
011770  045  101  115        P.AFN:  .ASCII  /%AM/
011773  105  104  111                .ASCII  /EDI/
011776  101  040  106                .ASCII  /A F/
012001  117  122  115                .ASCII  /ORM/
0.2004  101  124  040                .ASCII  /AT /
012007  105  122  122                .ASCII  /ERR/
```

```
012012  117  122  000                .ASCII  /OR/<00>
012015  000                         .ASCII  <00>
012016  045  101  127     P.AFO:     .ASCII  /%AW/
012021  122  111  124                .ASCII  /RIT/
012024  105  055  120                .ASCII  /E-P/
012027  122  117  124                .ASCII  /ROT/
012032  105  103  124                .ASCII  /ECT/
012035  105  104  000                .ASCII  /ED/<00>
012040  045  101  104     P.AFP:     .ASCII  /%AD/
012043  105  126  111                .ASCII  /EVI/
012046  103  105  040                .ASCII  /CE /
012051  103  117  115                .ASCII  /COM/
012054  120  101  122                .ASCII  /PAR/
012057  105  040  105                .ASCII  /E E/
012062  122  122  117                .ASCII  /RRO/
012065  122  000  000                .ASCII  /R/<00><00>
012070  045  101  104     P.AFQ:     .ASCII  /%AD/
012073  101  124  101                .ASCII  /ATA/
012076  040  105  122                .ASCII  / ER/
012101  122  117  122                .ASCII  /ROR/
012104  000  000                    .ASCII  <00><00>
012106  045  101  110     P.AFR:     .ASCII  /%AH/
012111  117  123  124                .ASCII  /OST/
012114  040  102  125                .ASCII  / BU/
012117  106  106  105                .ASCII  /FFE/
012122  122  040  101                .ASCII  /R A/
012125  103  103  105                .ASCII  /CCE/
012130  123  123  040                .ASCII  /SS /
012133  105  122  122                .ASCII  /ERR/
012136  117  122  000                .ASCII  /OR/<00>
012141  000                         .ASCII  <00>
012142  045  101  103     P.AFS:     .ASCII  /%AC/
012145  117  116  124                .ASCII  /ONT/
012150  122  117  114                .ASCII  /ROL/
012153  114  105  122                .ASCII  /LER/
012156  040  105  122                .ASCII  / ER/
012161  122  117  122                .ASCII  /ROR/
012164  000  000                    .ASCII  <00><00>
012166  045  101  104     P.AFT:     .ASCII  /%AD/
012171  122  111  126                .ASCII  /RIV/
012174  105  040  105                .ASCII  /E E/
012177  122  122  117                .ASCII  /RRO/
012202  122  000                    .ASCII  /R/<00>
012204  045  101  115     P.AFU:     .ASCII  /%AM/
012207  105  123  123                .ASCII  /ESS/
012212  101  107  105                .ASCII  /AGE/
012215  040  106  122                .ASCII  / FR/
012220  117  115  040                .ASCII  /OM /
012223  111  116  124                .ASCII  /INT/
012226  105  122  116                .ASCII  /ERN/
012231  101  114  040                .ASCII  /AL /
012234  104  111  101                .ASCII  /DIA/
012237  107  116  117                .ASCII  /GNO/
```

```
012242      123     124     111                 .ASCII  /STI/
012245      103     123     000                 .ASCII  /CS/<00>
012250      045     101     110         P.AFV:  .ASCII  /%AH/
012253      117     123     124                 .ASCII  /OST/
012256      040     103     117                 .ASCII  / CO/
012261      115     120     101                 .ASCII  /MPA/
012264      122     105     040                 .ASCII  /RE /
012267      105     122     122                 .ASCII  /ERR/
012272      117     122     000                 .ASCII  /OR/<00>
012275      000                                 .ASCII  <00>
012276      045     101     103         P.AFW:  .ASCII  /%AC/
012301      117     115     115                 .ASCII  /OMM/
012304      101     116     104                 .ASCII  /AND/
012307      040     124     111                 .ASCII  / TI/
012312      115     105     117                 .ASCII  /MEO/
012315      125     124     000                 .ASCII  /UT/<00>
012320   011644'                        P.AFI:  .WORD   P.AFJ
012322   011666'                                .WORD   P.AFK
012324   011710'                                .WORD   P.AFL
012326   011730'                                .WORD   P.AFM
012330   011770'                                .WORD   P.AFN
012332   012016'                                .WORD   P.AFO
012334   012040'                                .WORD   P.AFP
012336   012070'                                .WORD   P.AFQ
012340   012106'                                .WORD   P.AFR
012342   012142'                                .WORD   P.AFS
012344   012166'                                .WORD   P.AFT
012346   012204'                                .WORD   P.AFU
012350   012250'                                .WORD   P.AFV
012352   012276'                                .WORD   P.AFW
012354      045     101     105         P.AFX:  .ASCII  /%AE/
012357      122     122     117                 .ASCII  /RRO/
012362      122     040     114                 .ASCII  /R L/
012365      117     107     040                 .ASCII  /OG /
012370      115     105     123                 .ASCII  /MES/
012373      123     101     107                 .ASCII  /SAG/
012376      105     040     122                 .ASCII  /E R/
012401      105     103     105                 .ASCII  /ECE/
012404      111     126     105                 .ASCII  /IVE/
012407      104     072     045                 .ASCII  /D:%/
012412      116     000                         .ASCII  /N/<00>
012414      045     101     052         P.AFZ:  .ASCII  /%A*/
012417      040     103     117                 .ASCII  / CO/
012422      116     124     122                 .ASCII  /NTR/
012425      117     114     114                 .ASCII  /OLL/
012430      105     122     040                 .ASCII  /ER /
012433      105     122     122                 .ASCII  /ERR/
012436      117     122     045                 .ASCII  /OR%/
012441      116     000     000                 .ASCII  /N/<00><00>
012444      045     101     052         P.AGA:  .ASCII  /%A*/
012447      040     110     117                 .ASCII  / HO/
012452      123     124     040                 .ASCII  /ST /
012455      115     105     115                 .ASCII  /MEM/
```

```
012460  117     122     131             .ASCII  /ORY/
012463  040     101     103             .ASCII  / AC/
012466  103     105     123             .ASCII  /CES/
012471  123     040     105             .ASCII  /S E/
012474  122     122     117             .ASCII  /RRO/
012477  122     045     116             .ASCII  /R%N/
012502  000     000                     .ASCII  <00><00>
012504  045     101     052     P.AGB:  .ASCII  /%A*/
012507  040     104     111             .ASCII  / DI/
012512  123     113     045             .ASCII  /SK%/
012515  104     062     045             .ASCII  /D2%/
012520  101     040     055             .ASCII  /A -/
012523  040     104     111             .ASCII  / DI/
012526  123     113     040             .ASCII  /SK /
012531  124     122     101             .ASCII  /TRA/
012534  116     123     106             .ASCII  /NSF/
012537  105     122     040             .ASCII  /ER /
012542  105     122     122             .ASCII  /ERR/
012545  117     122     045             .ASCII  /OR%/
012550  116     000                     .ASCII  /N/<00>
012552  045     101     052     P.AGC:  .ASCII  /%A*/
012555  040     104     111             .ASCII  / DI/
012560  123     113     045             .ASCII  /SK%/
012563  104     062     045             .ASCII  /D2%/
012566  101     040     055             .ASCII  /A -/
012571  040     042     123             .ASCII  / "S/
012574  124     101     116             .ASCII  /TAN/
012577  104     101     122             .ASCII  /DAR/
012602  104     040     104             .ASCII  /D D/
012605  111     123     113             .ASCII  /ISK/
012610  040     111     116             .ASCII  / IN/
012613  124     105     122             .ASCII  /TER/
012616  103     117     116             .ASCII  /CON/
012621  116     105     103             .ASCII  /NEC/
012624  124     042     040             .ASCII  /T" /
012627  105     122     122             .ASCII  /ERR/
012632  117     122     045             .ASCII  /OR%/
012635  116     000     000             .ASCII  /N/<00><00>
012640  045     101     052     P.AGD:  .ASCII  /%A*/
012643  040     104     111             .ASCII  / DI/
012646  123     113     045             .ASCII  /SK%/
012651  104     062     045             .ASCII  /D2%/
012654  101     040     055             .ASCII  /A -/
012657  040     042     123             .ASCII  / "S/
012662  115     101     114             .ASCII  /MAL/
012665  114     040     104             .ASCII  /L D/
012670  111     123     113             .ASCII  /ISK/
012673  042     040     105             .ASCII  /" E/
012676  122     122     117             .ASCII  /RRO/
012701  122     045     116             .ASCII  /R%N/
012704  000     000                     .ASCII  <00><00>
012706  012414' .WORD   P.AFZ   P.AFY:
012710  012444' .WORD   P.AGA
```

```
012712  012504'                                .WORD   P.AGB
012714  012552'                                .WORD   P.AGC
012716  012640'                                .WORD   P.AGD
012720     045      116      045      P.AGE:    .ASCII  /%N%/
012723     101      052      040               .ASCII  /A* /
012726     123      101      072               .ASCII  /SA:/
012731     040      045      117               .ASCII  / %O/
012734     066      000                        .ASCII  /6/<00>
012736     045      116      045      P.AGF:    .ASCII  /%N%/
012741     101      052      040               .ASCII  /A* /
012744     123      124      101               .ASCII  /STA/
012747     124      125      123               .ASCII  /TUS/
012752     040      103      117               .ASCII  / CO/
012755     104      105      072               .ASCII  /DE:/
012760     040      045      117               .ASCII  / %O/
012763     062      000      000               .ASCII  /2/<00><00>
012766     045      117      064      P.AGG:    .ASCII  /%04/
012771     000                                 .ASCII  <00>
012772     045      116      045      P.AGH:    .ASCII  /%N%/
012775     101      052      040               .ASCII  /A* /
013000     123      125      102               .ASCII  /SUB/
013003     137      103      117               .ASCII  /_CO/
013006     104      105      072               .ASCII  /DE:/
013011     040      000      000               .ASCII  / /<00><00>
013014     045      116      045      P.AGI:    .ASCII  /%N%/
013017     101      052      040               .ASCII  /A* /
013022     103      117      115               .ASCII  /COM/
013025     115      101      116               .ASCII  /MAN/
013030     104      072      040               .ASCII  /D: /
013033     000                                 .ASCII  <00>
013034     045      101      122      P.AGJ:    .ASCII  /%AR/
013037     105      101      104               .ASCII  /EAD/
013042     000      000                        .ASCII  <00><00>
013044     045      101      127      P.AGK:    .ASCII  /%AW/
013047     122      111      124               .ASCII  /RIT/
013052     105      000                        .ASCII  /E/<00>
013054     045      101      055      P.AGL:    .ASCII  /%A-/
013057     103      117      115               .ASCII  /COM/
013062     120      101      122               .ASCII  /PAR/
013065     105      000      000               .ASCII  /E/<00><00>
013070     045      101      117      P.AGM:    .ASCII  /%AO/
013073     116      114      111               .ASCII  /NLI/
013076     116      105      000               .ASCII  /NE/<00>
013101     000                                 .ASCII  <00>
013102     045      101      101      P.AGN:    .ASCII  /%AA/
013105     103      103      105               .ASCII  /CCE/
013110     123      123      000               .ASCII  /SS/<00>
013113     000                                 .ASCII  <00>
013114     045      117      063      P.AGO:    .ASCII  /%03/
013117     000                                 .ASCII  <00>
013120     045      116      045      P.AGP:    .ASCII  /%N%/
013123     101      052      040               .ASCII  /A* /
013126     102      101      104               .ASCII  /BAD/
```

```
013131  040  102  114            .ASCII  / BL/
013134  117  103  113            .ASCII  /OCK/
013137  040  050  110            .ASCII  / (H/
013142  157  163  164            .ASCII  /ost/
013145  040  162  145            .ASCII  / re/
013150  160  154  141            .ASCII  /ple/
013153  143  145  141            .ASCII  /cea/
013156  142  154  145            .ASCII  /ble/
013161  051  072  040            .ASCII  /); /
013164  045  104  065            .ASCII  /%D5/
013167  045  101  056            .ASCII  /%A./
013172  040  050  117            .ASCII  / (O/
013175  103  124  040            .ASCII  /CT /
013200  045  117  066            .ASCII  /%O6/
013203  045  101  051            .ASCII  /%A)/
013206  000  000                 .ASCII  <00><00>
013210  045  116  045  P.AGQ:     .ASCII  /%N%/
013213  101  052  040            .ASCII  /A* /
013216  061  163  164            .ASCII  /1st/
013221  040  102  101            .ASCII  / BA/
013224  104  040  102            .ASCII  /D B/
013227  114  117  103            .ASCII  /LOC/
013232  113  040  050            .ASCII  /K (/
013235  110  157  163            .ASCII  /Hos/
013240  164  040  162            .ASCII  /t r/
013243  145  160  154            .ASCII  /epl/
013246  141  143  145            .ASCII  /ace/
013251  141  142  154            .ASCII  /abl/
013254  145  051  072            .ASCII  /e):/
013257  040  045  104            .ASCII  / %D/
013262  065  045  101            .ASCII  /5%A/
013265  056  040  050            .ASCII  /. (/
013270  117  103  124            .ASCII  /OCT/
013273  040  045  117            .ASCII  / %O/
013276  066  045  101            .ASCII  /6%A/
013301  051  000  000            .ASCII  /)/<00><00>
013304  045  116  045  P.AGR:     .ASCII  /%N%/
013307  101  052  040            .ASCII  /A* /
013312  102  101  104            .ASCII  /BAD/
013315  040  102  114            .ASCII  / BL/
013320  117  103  113            .ASCII  /OCK/
013323  040  122  105            .ASCII  / RE/
013326  120  117  122            .ASCII  /POR/
013331  124  105  104            .ASCII  /TED/
013334  040  050  122            .ASCII  / (R/
013337  145  160  154            .ASCII  /epl/
013342  141  143  145            .ASCII  /ace/
013345  144  051  072            .ASCII  /d):/
013350  040  045  104            .ASCII  / %D/
013353  045  101  056            .ASCII  /%A./
013356  040  050  117            .ASCII  / (O/
013361  103  124  040            .ASCII  /CT /
013364  045  117  066            .ASCII  /%O6/
```

```
013367  045  101  051                .ASCII  /%A)/
013372  000  000                     .ASCII  <00><00>
013374  045  116  045       P.AGS:    .ASCII  /%N%/
013377  101  052  040                .ASCII  /A* /
013402  114  102  116                .ASCII  /LBN/
013405  072  040  045                .ASCII  /: %/
013410  104  065  045                .ASCII  /D5%/
013413  101  056  040                .ASCII  /A. /
013416  050  117  103                .ASCII  /(OC/
013421  124  040  045                .ASCII  /T %/
013424  117  066  045                .ASCII  /O6%/
013427  101  051  000                .ASCII  /A)/<00>
013432  045  116  045       P.AGT:    .ASCII  /%N%/
013435  101  052  040                .ASCII  /A* /
013440  120  102  116                .ASCII  /PBN/
013443  072  040  045                .ASCII  /: %/
013446  104  065  045                .ASCII  /D5%/
013451  101  056  040                .ASCII  /A. /
013454  050  117  103                .ASCII  /(OC/
013457  124  040  045                .ASCII  /T %/
013462  117  066  045                .ASCII  /O6%/
013465  101  051  000                .ASCII  /A)/<00>
013470  045  116  045       P.AGU:    .ASCII  /%N%/
013473  101  052  040                .ASCII  /A* /
013476  114  102  116                .ASCII  /LBN/
013501  072  040  050                .ASCII  /: (/
013504  122  105  101                .ASCII  /REA/
013507  104  051  040                .ASCII  /D) /
013512  045  104  065                .ASCII  /%D5/
013515  045  101  056                .ASCII  /%A./
013520  040  050  117                .ASCII  / (O/
013523  103  124  040                .ASCII  /CT /
013526  045  117  066                .ASCII  /%O6/
013531  045  101  051                .ASCII  /%A)/
013534  000  000                     .ASCII  <00><00>
013536  045  116  045       P.AGV:    .ASCII  /%N%/
013541  101  052  040                .ASCII  /A* /
013544  114  102  116                .ASCII  /LBN/
013547  072  040  050                .ASCII  /: (/
013552  127  122  111                .ASCII  /WRI/
013555  124  105  051                .ASCII  /TE)/
013560  040  045  104                .ASCII  / %D/
013563  065  045  101                .ASCII  /5%A/
013566  056  040  050                .ASCII  /. (/
013571  117  103  124                .ASCII  /OCT/
013574  040  045  117                .ASCII  / %O/
013577  066  045  101                .ASCII  /6%A/
013602  051  000                     .ASCII  /)/<00>
013604  045  116  045       P.AGW:    .ASCII  /%N%/
013607  101  052  040                .ASCII  /A* /
013612  122  105  120                .ASCII  /REP/
013615  114  101  103                .ASCII  /LAC/
013620  105  115  105                .ASCII  /EME/
```

```
013623  116  124  040                .ASCII  /NT /
013626  102  114  117                .ASCII  /BLO/
013631  103  113  040                .ASCII  /CK /
013634  116  117  056                .ASCII  /NO./
013637  040  045  104                .ASCII  / %D/
013642  065  045  101                .ASCII  /5%A/
013645  056  040  050                .ASCII  /. (/
013650  117  103  124                .ASCII  /OCT/
013653  040  045  117                .ASCII  / %O/
013656  066  045  101                .ASCII  /6%A/
013661  051  000  000                .ASCII  /)/<00><00>
013664  045  116  045    P.AGX:      .ASCII  /%N%/
013667  101  052  040                .ASCII  /A* /
013672  102  131  124                .ASCII  /BYT/
013675  105  040  103                .ASCII  /E C/
013700  117  125  116                .ASCII  /OUN/
013703  124  040  111                .ASCII  /T I/
013706  116  040  103                .ASCII  /N C/
013711  117  115  115                .ASCII  /OMM/
013714  101  116  104                .ASCII  /AND/
013717  072  040  045                .ASCII  /: %/
013722  104  065  045                .ASCII  /D5%/
013725  101  056  000                .ASCII  /A./<00>
013730  045  116  045    P.AGY:      .ASCII  /%N%/
013733  101  052  040                .ASCII  /A* /
013736  102  131  124                .ASCII  /BYT/
013741  105  040  103                .ASCII  /E C/
013744  117  125  116                .ASCII  /OUN/
013747  124  040  111                .ASCII  /T I/
013752  116  040  122                .ASCII  /N R/
013755  105  101  104                .ASCII  /EAD/
013760  040  103  117                .ASCII  / CO/
013763  115  115  101                .ASCII  /MMA/
013766  116  104  072                .ASCII  /ND:/
013771  040  045  104                .ASCII  / %D/
013774  065  045  101                .ASCII  /5%A/
013777  056  000  000                .ASCII  /./<00><00>
014002  045  116  045    P.AGZ:      .ASCII  /%N%/
014005  101  052  040                .ASCII  /A* /
014010  102  131  124                .ASCII  /BYT/
014013  105  040  103                .ASCII  /E C/
014016  117  125  116                .ASCII  /OUN/
014021  124  040  111                .ASCII  /T I/
014024  116  040  127                .ASCII  /N W/
014027  122  111  124                .ASCII  /RIT/
014032  105  040  103                .ASCII  /E C/
014035  117  115  115                .ASCII  /OMM/
014040  101  116  104                .ASCII  /AND/
014043  072  040  045                .ASCII  /: %/
014046  104  065  045                .ASCII  /D5%/
014051  101  056  000                .ASCII  /A./<00>
014054  045  116  045    P.AHA:      .ASCII  /%N%/
014057  101  052  040                .ASCII  /A* /
```

```
014062  101  103  124            .ASCII  /ACT/
014065  125  101  114            .ASCII  /UAL/
014070  040  043  040            .ASCII  / # /
014073  117  106  040            .ASCII  /OF /
014076  102  131  124            .ASCII  /BYT/
014101  105  123  040            .ASCII  /ES /
014104  124  122  101            .ASCII  /TRA/
014107  116  123  106            .ASCII  /NSF/
014112  105  122  122            .ASCII  /ERR/
014115  105  104  072            .ASCII  /ED:/
014120  040  045  104            .ASCII  / %D/
014123  065  045  101            .ASCII  /5%A/
014126  056  000                 .ASCII  /./<00>
014130  045  116  045  P.AMB:    .ASCII  /%N%/
014133  101  052  040            .ASCII  /A* /
014136  111  057  117            .ASCII  /I/<57>/O/
014141  040  102  125            .ASCII  / BU/
014144  106  106  105            .ASCII  /FFE/
014147  122  040  101            .ASCII  /R A/
014152  104  104  122            .ASCII  /DDR/
014155  105  123  123            .ASCII  /ESS/
014160  040  050  063            .ASCII  / (3/
014163  062  040  142            .ASCII  /2 b/
014166  151  164  163            .ASCII  /its/
014171  051  072  040            .ASCII  /): /
014174  045  117  066            .ASCII  /%O6/
014177  045  101  040            .ASCII  /%A /
014202  045  117  066            .ASCII  /%O6/
014205  000                      .ASCII  <00>
014206  045  116  045  P.AMC:    .ASCII  /%N%/
014211  101  052  040            .ASCII  /A* /
014214  111  057  117            .ASCII  /I/<57>/O/
014217  040  102  125            .ASCII  / BU/
014222  106  106  105            .ASCII  /FFE/
014225  122  040  101            .ASCII  /R A/
014230  104  104  122            .ASCII  /DDR/
014233  105  123  123            .ASCII  /ESS/
014236  040  106  117            .ASCII  / FO/
014241  122  040  122            .ASCII  /R R/
014244  105  101  104            .ASCII  /EAD/
014247  040  050  063            .ASCII  / (3/
014252  062  040  142            .ASCII  /2 b/
014255  151  164  163            .ASCII  /its/
014260  051  072  040            .ASCII  /): /
014263  045  117  066            .ASCII  /%O6/
014266  045  101  040            .ASCII  /%A /
014271  045  117  066            .ASCII  /%O6/
014274  000  000                 .ASCII  <00><00>
014276  045  116  045  P.AMD:    .ASCII  /%N%/
014301  101  052  040            .ASCII  /A* /
014304  111  057  117            .ASCII  /I/<57>/O/
014307  040  102  125            .ASCII  / BU/
014312  106  106  105            .ASCII  /FFE/
```

```
014315  122  040  101            .ASCII  /R A/
014320  104  104  122            .ASCII  /DDR/
014323  105  123  123            .ASCII  /ESS/
014326  040  106  117            .ASCII  / FO/
014331  122  040  127            .ASCII  /R W/
014334  122  111  124            .ASCII  /RIT/
014337  105  040  050            .ASCII  /E (/
014342  063  062  040            .ASCII  /32 /
014345  142  151  164            .ASCII  /bit/
014350  163  051  072            .ASCII  /s):/
014353  040  045  117            .ASCII  / %O/
014356  066  045  101            .ASCII  /6%A/
014361  040  045  117            .ASCII  / %O/
014364  066  000                 .ASCII  /6/<00>
014366  045  116  045  P.AHE:    .ASCII  /%N%/
014371  101  103  117            .ASCII  /ACO/
014374  116  124  105            .ASCII  /NTE/
014377  116  124  123            .ASCII  /NTS/
014402  040  117  106            .ASCII  / OF/
014405  040  103  117            .ASCII  / CO/
014410  115  115  101            .ASCII  /MMA/
014413  116  104  057            .ASCII  /ND/<57>
014416  122  105  123            .ASCII  /RES/
014421  120  117  116            .ASCII  /PON/
014424  123  105  040            .ASCII  /SE /
014427  120  101  103            .ASCII  /PAC/
014432  113  105  124            .ASCII  /KET/
014435  040  123  101            .ASCII  / SA/
014440  126  105  040            .ASCII  /VE /
014443  101  122  105            .ASCII  /ARE/
014446  101  072  045            .ASCII  /A:%/
014451  116  000  000            .ASCII  /N/<00><00>
014454  045  101  040  P.AHF:    .ASCII  /%A /
014457  045  117  066            .ASCII  /%06/
014462  000  000                 .ASCII  <00><00>
014464  045  116  045  P.AHG:    .ASCII  /%N%/
014467  101  052  040            .ASCII  /A* /
014472  124  111  115            .ASCII  /TIM/
014475  105  072  040            .ASCII  /E: /
014500  045  132  062            .ASCII  /%Z2/
014503  045  101  072            .ASCII  /%A:/
014506  045  132  062            .ASCII  /%Z2/
014511  045  101  040            .ASCII  /%A /
014514  110  117  125            .ASCII  /HOU/
014517  122  123  045            .ASCII  /RS%/
014522  116  000                 .ASCII  /N/<00>
014524  045  116  045  P.AHH:    .ASCII  /%N%/
014527  101  040  052            .ASCII  /A */
014532  040  104  111            .ASCII  / DI/
014535  123  113  040            .ASCII  /SK /
014540  072  040  045            .ASCII  /: %/
014543  104  062  000            .ASCII  /D2/<00>
014546  045  116  045  P.AHI:    .ASCII  /%N%/
```

```
014551  101     104     102             .ASCII  /ADB/
014554  116     072     040             .ASCII  /N: /
014557  045     104     065             .ASCII  /%D5/
014562  045     10'     056             .ASCII  /%A./
014565  040     050     117             .ASCII  / (O/
014570  103     124     040             .ASCII  /CT /
014573  045     117     066             .ASCII  /%O6/
014576  045     101     051             .ASCII  /%A)/
014601  000                             .ASCII  <00>
014602  045     116     045     P.AHJ:  .ASCII  /%N%/
014605  101     102     131             .ASCII  /ABY/
014610  124     105     040             .ASCII  /TE /
014613  116     125     115             .ASCII  /NUM/
014616  102     105     122             .ASCII  /BER/
014621  072     040     045             .ASCII  /: %/
014624  104     063     000             .ASCII  /D3/<00>
014627  000                             .ASCII  <00>
014630  045     116     045     P.AHK:  .ASCII  /%N%/
014633  101     122     101             .ASCII  /ARA/
014636  116     104     117             .ASCII  /NDO/
014641  115     040     127             .ASCII  /M W/
014644  122     111     124             .ASCII  /RIT/
014647  124     105     116             .ASCII  /TEN/
014652  040     127     117             .ASCII  / WO/
014655  122     104     040             .ASCII  /RD /
014660  072     045     102             .ASCII  /:%B/
014663  061     066     000             .ASCII  /16/<00>
014666  045     116     045     P.AHL:  .ASCII  /%N%/
014671  101     122     101             .ASCII  /ARA/
014674  116     104     117             .ASCII  /NDO/
014677  115     040     122             .ASCII  /M R/
014702  105     101     104             .ASCII  /EAD/
014705  040     127     117             .ASCII  / WO/
014710  122     104     040             .ASCII  /RD /
014713  142     151     156             .ASCII  /bin/
014716  072     045     102             .ASCII  /:%B/
014721  061     066     045             .ASCII  /16%/
014724  101     040     157             .ASCII  /A o/
014727  143     164     072             .ASCII  /ct:/
014732  045     117     066             .ASCII  /%O6/
014735  000                             .ASCII  <00>
014736  045     116     045     P.AHM:  .ASCII  /%N%/
014741  101     104     125             .ASCII  /ADU/
014744  120     114     111             .ASCII  /PLI/
014747  103     101     124             .ASCII  /CAT/
014752  105     040     125             .ASCII  /E U/
014755  116     111     124             .ASCII  /NIT/
014760  072     045     104             .ASCII  /:%D/
014763  062     045     101             .ASCII  /2%A/
014766  040     101     124             .ASCII  / AT/
014771  040     111     120             .ASCII  / IP/
014774  072     040     045             .ASCII  /: %/
014777  117     066     000             .ASCII  /O6/<00>
```

```
015002  045  116  045        P.AHN:  .ASCII  /%N%/
015005  101  115  117                .ASCII  /AMO/
015010  122  105  040                .ASCII  /RE /
015013  124  110  101                .ASCII  /THA/
015016  116  040  045                .ASCII  /N %/
015021  104  061  045                .ASCII  /D1%/
015024  101  040  104                .ASCII  /A D/
015027  111  106  106                .ASCII  /IFF/
015032  105  122  105                .ASCII  /ERE/
015035  116  124  040                .ASCII  /NT /
015040  111  120  040                .ASCII  /IP /
015043  101  104  104                .ASCII  /ADD/
015046  122  105  123                .ASCII  /RES/
015051  123  105  123                .ASCII  /SES/
015054  000  000                    .ASCII  <00><00>
015056  045  101  123        P.AHO:  .ASCII  /%AS/
015061  120  111  116                .ASCII  /PIN/
015064  055  104  117                .ASCII  /-DO/
015067  127  116  040                .ASCII  /WN /
015072  111  107  116                .ASCII  /IGN/
015075  117  122  105                .ASCII  /ORE/
015100  104  000                    .ASCII  /D/<00>
015102  045  101  123        P.AHP:  .ASCII  /%AS/
015105  124  111  114                .ASCII  /TIL/
015110  114  040  103                .ASCII  /L C/
015113  117  116  116                .ASCII  /ONN/
015116  105  103  124                .ASCII  /ECT/
015121  105  104  000                .ASCII  /ED/<00>
015124  045  101  104        P.AHQ:  .ASCII  /%AD/
015127  125  120  114                .ASCII  /UPL/
015132  111  103  101                .ASCII  /ICA/
015135  124  105  040                .ASCII  /TE /
015140  125  116  111                .ASCII  /UNI/
015143  124  040  116                .ASCII  /T N/
015146  125  115  102                .ASCII  /UMB/
015151  105  122  000                .ASCII  /ER/<00>
015154  045  101  101        P.AHR:  .ASCII  /%AA/
015157  114  122  105                .ASCII  /LRE/
015162  101  104  131                .ASCII  /ADY/
015165  040  117  116                .ASCII  / ON/
015170  114  111  116                .ASCII  /LIN/
015173  105  000  000                .ASCII  /E/<00><00>
015176  045  101  123        P.AHS:  .ASCII  /%AS/
015201  124  111  114                .ASCII  /TIL/
015204  114  040  117                .ASCII  /L O/
015207  116  114  111                .ASCII  /NLI/
015212  116  105  000                .ASCII  /NE/<00>
015215  000                         .ASCII  <00>
015216  045  101  125        P.AHT:  .ASCII  /%AU/
015221  116  111  124                .ASCII  /NIT/
015224  040  125  116                .ASCII  / UN/
015227  113  116  117                .ASCII  /KNO/
015232  127  116  040                .ASCII  /WN /
```

```
015235  117  122  040                .ASCII  /OR /
015240  117  116  114                .ASCII  /ONL/
015243  111  116  105                .ASCII  /INE/
015246  040  124  117                .ASCII  / TO/
015251  040  101  116                .ASCII  / AN/
015254  117  124  110                .ASCII  /OTH/
015257  105  122  040                .ASCII  /ER /
015262  103  117  116                .ASCII  /CON/
015265  124  122  117                .ASCII  /TRO/
015270  114  114  105                .ASCII  /LLE/
015273  122  000  000                .ASCII  /R/<00><00>
015276  045  101  116      P.AHU:    .ASCII  /%AN/
015301  117  040  126                .ASCII  /O V/
015304  117  114  125                .ASCII  /OLU/
015307  115  105  040                .ASCII  /ME /
015312  115  117  125                .ASCII  /MOU/
015315  116  124  105                .ASCII  /NTE/
015320  104  040  117                .ASCII  /D O/
015323  122  040  104                .ASCII  /R D/
015326  122  111  126                .ASCII  /RIV/
015331  105  040  104                .ASCII  /E D/
015334  111  123  101                .ASCII  /ISA/
015337  102  114  105                .ASCII  /BLE/
015342  104  040  102                .ASCII  /D B/
015345  131  040  123                .ASCII  /Y S/
015350  127  111  124                .ASCII  /WIT/
015353  103  110  000                .ASCII  /CH/<00>
015356  045  101  125      P.AHV:    .ASCII  /%AU/
015361  116  111  124                .ASCII  /NIT/
015364  040  111  116                .ASCII  / IN/
015367  117  120  105                .ASCII  /OPE/
015372  122  101  124                .ASCII  /RAT/
015375  111  126  105                .ASCII  /IVE/
015400  040  050  122                .ASCII  / (R/
015403  104  065  061                .ASCII  /D51/
015406  057  065  062                .ASCII  <57>/52/
015411  040  167  162                .ASCII  / wr/
015414  151  164  145                .ASCII  /ite/
015417  040  146  141                .ASCII  / fa/
015422  165  154  164                .ASCII  /ult/
015425  051  000  000                .ASCII  /)/<00><00>
015430  045  101  125      P.AHW:    .ASCII  /%AU/
015433  116  111  124                .ASCII  /NIT/
015436  040  104  111                .ASCII  / DI/
015441  123  101  102                .ASCII  /SAB/
015444  114  105  104                .ASCII  /LED/
015447  040  102  131                .ASCII  / BY/
015452  040  106  111                .ASCII  / FI/
015455  105  114  104                .ASCII  /ELD/
015460  040  123  105                .ASCII  / SE/
015463  122  126  111                .ASCII  /RVI/
015466  103  105  040                .ASCII  /CE /
015471  117  122  040                .ASCII  /OR /
```

```
015474  111  116  124          .ASCII  /INT/
015477  105  122  116          .ASCII  /ERN/
015502  101  114  040          .ASCII  /AL /
015505  104  111  101          .ASCII  /DIA/
015510  107  116  117          .ASCII  /GNO/
015513  123  124  111          .ASCII  /STI/
015516  103  123  000          .ASCII  /CS/<00>
015521  000                    .ASCII  <00>
015522  045  101  042  P.AHX:   .ASCII  /%A"/
015525  106  117  122          .ASCII  /FOR/
015530  103  105  104          .ASCII  /CED/
015533  040  105  122          .ASCII  / ER/
015536  122  117  122          .ASCII  /ROR/
015541  042  040  104           ASCII  /" D/
015544  105  124  105          .ASCII  /ETE/
015547  103  124  105          .ASCII  /CTE/
015552  104  040  127          .ASCII  /D W/
015555  110  111  114          .ASCII  /HIL/
015560  105  040  101          .ASCII  /E A/
015563  103  103  105          .ASCII  /CCE/
015566  123  123  111          .ASCII  /SSI/
015571  116  107  040          .ASCII  /NG /
015574  106  103  124          .ASCII  /FCT/
015577  040  117  122          .ASCII  / OR/
015602  040  122  103          .ASCII  / RC/
015605  124  000  000          .ASCII  /T/<00><00>
015610  045  101  123  P.AHY:   .ASCII  /%AS/
015613  105  103  124          .ASCII  /ECT/
015616  117  122  040          .ASCII  /OR /
015621  110  101  104          .ASCII  /HAD/
015624  040  102  105          .ASCII  / BE/
015627  105  116  040          .ASCII  /EN /
015632  127  122  111          .ASCII  /WRI/
015635  124  124  105          .ASCII  /TTE/
015640  116  040  127          .ASCII  /N W/
015643  111  124  110          .ASCII  /ITH/
015646  040  042  106          .ASCII  / "F/
015651  117  122  103          .ASCII  /ORC/
015654  105  104  040          .ASCII  /ED /
015657  105  122  122          .ASCII  /ERR/
015662  117  122  042          .ASCII  /OR"/
015665  040  115  117          .ASCII  / MO/
015670  104  111  106          .ASCII  /DIF/
015673  111  105  122          .ASCII  /IER/
015676  000  000               .ASCII  <00><00>
015700  045  101  106  P.AHZ:   .ASCII  /%AF/
015703  103  124  040          .ASCII  /CT /
015706  117  122  040          .ASCII  /OR /
015711  122  103  124          .ASCII  /RCT/
015714  040  125  116          .ASCII  / UN/
015717  122  105  101          .ASCII  /REA/
015722  104  101  102          .ASCII  /DAB/
015725  114  105  040          .ASCII  /LE /
```

```
015730  055  040  111                .ASCII  /- I/
015733  116  126  101                .ASCII  /NVA/
015736  114  111  104                .ASCII  /LID/
015741  040  123  105                .ASCII  / SE/
015744  103  124  117                .ASCII  /CTO/
015747  122  040  110                .ASCII  /R H/
015752  105  101  104                .ASCII  /EAD/
015755  105  122  000                .ASCII  /ER/<00>
015760  045  101  110  P.AIA:        .ASCII  /%AH/
015763  105  101  104                .ASCII  /EAD/
015766  105  122  040                .ASCII  /ER /
015771  103  117  115                .ASCII  /COM/
015774  120  101  122                .ASCII  /PAR/
015777  105  040  105                .ASCII  /E E/
016002  122  122  117                .ASCII  /RRO/
016005  122  040  050                .ASCII  /R (/
016010  126  141  154                .ASCII  /Val/
016013  151  144  040                .ASCII  /id /
016016  150  145  141                .ASCII  /hea/
016021  144  145  162                .ASCII  /der/
016024  040  156  157                .ASCII  / no/
016027  164  040  146                .ASCII  /t f/
016032  157  165  156                .ASCII  /oun/
016035  144  051  000                .ASCII  /d)/<00>
016040  045  101  106  P.AIB:        .ASCII  /%AF/
016043  103  124  040                .ASCII  /CT /
016046  117  122  040                .ASCII  /OR /
016051  122  103  124                .ASCII  /RCT/
016054  040  125  116                .ASCII  / UN/
016057  122  105  101                .ASCII  /REA/
016062  104  101  102                .ASCII  /DAB/
016065  114  105  040                .ASCII  /LE /
016070  055  040  104                .ASCII  /- D/
016073  101  124  101                .ASCII  /ATA/
016076  040  123  131                .ASCII  / SY/
016101  116  103  040                .ASCII  /NC /
016104  124  111  115                .ASCII  /TIM/
016107  105  117  125                .ASCII  /EOU/
016112  124  000                     .ASCII  /T/<00>
016114  045  101  104  P.AIC:        .ASCII  /%AD/
016117  101  124  101                .ASCII  /ATA/
016122  040  123  131                .ASCII  / SY/
016125  116  103  040                .ASCII  /NC /
016130  116  117  124                .ASCII  /NOT/
016133  040  106  117                .ASCII  / FO/
016136  125  116  104                .ASCII  /UND/
016141  040  050  104                .ASCII  / (D/
016144  141  164  141                .ASCII  /ata/
016147  040  163  171                .ASCII  / sy/
016152  156  143  040                .ASCII  /nc /
016155  164  151  155                .ASCII  /tim/
016160  145  157  165                .ASCII  /eou/
016163  164  051  000                .ASCII  /t)/<00>
```

```
016166  045  101  106        P.AID:  .ASCII  /%AF/
016171  103  124  040                .ASCII  /CT /
016174  117  122  040                .ASCII  /OR /
016177  122  103  124                .ASCII  /RCT/
016202  040  125  116                .ASCII  / UN/
016205  122  105  101                .ASCII  /REA/
016210  104  101  102                .ASCII  /DAB/
016213  114  105  040                .ASCII  /LE /
016216  055  040  125                .ASCII  /- U/
016221  116  103  117                .ASCII  /NCO/
016224  122  122  105                .ASCII  /RRE/
016227  103  124  101                .ASCII  /CTA/
016232  102  114  105                .ASCII  /BLE/
016235  040  105  103                .ASCII  / EC/
016240  103  040  105                .ASCII  /C E/
016243  122  122  117                .ASCII  /RRO/
016246  122  000                    .ASCII  /R/<00>
016250  045  101  125        P.AIE:  .ASCII  /%AU/
016253  116  103  117                .ASCII  /NCO/
016256  122  122  105                .ASCII  /RRE/
016261  103  124  101                .ASCII  /CTA/
016264  102  114  105                .ASCII  /BLE/
016267  040  105  103                .ASCII  / EC/
016272  103  040  105                .ASCII  /C E/
016275  122  122  117                .ASCII  /RRO/
016300  122  000                    .ASCII  /R/<00>
016302  045  101  122        P.AIF:  .ASCII  /%AR/
016305  103  124  040                .ASCII  /CT /
016310  103  117  122                .ASCII  /COR/
016313  122  125  120                .ASCII  /RUP/
016316  124  105  104                .ASCII  /TED/
016321  000                        .ASCII  <00>
016322  045  101  116        P.AIG:  .ASCII  /%AN/
016325  117  040  122                .ASCII  /O R/
016330  105  120  114                .ASCII  /EPL/
016333  101  103  105                .ASCII  /ACE/
016336  115  105  116                .ASCII  /MEN/
016341  124  040  102                .ASCII  /T B/
016344  114  117  103                .ASCII  /LOC/
016347  113  040  101                .ASCII  /K A/
016352  126  101  111                .ASCII  /VAI/
016355  114  101  102                .ASCII  /LAB/
016360  114  105  040                .ASCII  /LE /
016363  050  122  103                .ASCII  /(RC/
016366  124  040  146                .ASCII  /T f/
016371  165  154  154                .ASCII  /ull/
016374  051  000                    .ASCII  /)/<00>
016376  045  101  104        P.AIH:  .ASCII  /%AD/
016401  111  123  113                .ASCII  /ISK/
016404  040  116  117                .ASCII  / NO/
016407  124  040  106                .ASCII  /T F/
016412  117  122  115                .ASCII  /ORM/
016415  101  124  124                .ASCII  /ATT/
```

```
016420  105  104  040                .ASCII  /ED /
016423  127  111  124                .ASCII  /WIT/
016426  110  040  065                .ASCII  /H 5/
016431  061  062  040                .ASCII  /12 /
016434  102  131  124                .ASCII  /BYT/
016437  105  040  123                .ASCII  /E S/
016442  105  103  124                .ASCII  /ECT/
016445  117  122  123                .ASCII  /ORS/
016450  000  000                     .ASCII  <00><00>
016452  045  101  104     P.AII:     .ASCII  /%AD/
016455  111  123  113                .ASCII  /ISK/
016460  040  116  117                .ASCII  / NO/
016463  124  040  106                .ASCII  /T F/
016466  117  122  115                .ASCII  /ORM/
016471  101  124  124                .ASCII  /ATT/
016474  105  104  040                .ASCII  /ED /
016477  117  122  040                .ASCII  /OR /
016502  106  103  124                .ASCII  /FCT/
016505  040  103  117                .ASCII  / CO/
016510  122  122  125                .ASCII  /RRU/
016513  120  124  105                .ASCII  /PTE/
016516  104  000                     .ASCII  /D/<00>
016520  045  101  117     P.AIJ:     .ASCII  /%AO/
016523  116  105  040                .ASCII  /NE /
016526  123  131  115                .ASCII  /SYM/
016531  102  117  114                .ASCII  /BOL/
016534  040  105  103                .ASCII  / EC/
016537  103  040  105                .ASCII  /C E/
016542  122  122  117                .ASCII  /RRO/
016545  122  000  000                .ASCII  /R/<00><00>
016550  045  101  124     P.AIK:     .ASCII  /%AT/
016553  127  117  040                .ASCII  /WO /
016556  123  131  115                .ASCII  /SYM/
016561  102  117  114                .ASCII  /BOL/
016564  040  105  103                .ASCII  / EC/
016567  103  040  105                .ASCII  /C E/
016572  122  122  117                .ASCII  /RRO/
016575  122  000  000                .ASCII  /R/<00><00>
016600  045  101  124     P.AIL:     .ASCII  /%AT/
016603  110  122  105                .ASCII  /HRE/
016606  105  040  123                .ASCII  /E S/
016611  131  115  102                .ASCII  /YMB/
016614  117  114  040                .ASCII  /OL /
016617  105  103  103                .ASCII  /ECC/
016622  040  105  122                .ASCII  / ER/
016625  122  117  122                .ASCII  /ROR/
016630  000  000                     .ASCII  <00><00>
016632  045  101  106     P.AIM:     .ASCII  /%AF/
016635  117  125  122                .ASCII  /OUR/
016640  040  123  131                .ASCII  / SY/
016643  115  102  117                .ASCII  /MBO/
016646  114  040  105                .ASCII  /L E/
016651  103  103  040                .ASCII  /CC /
```

```
016654  105  122  122                .ASCII  /ERR/
016657  117  122  000                .ASCII  /OR/<00>
016662  045  101  106      P.AIN:    .ASCII  /%AF/
016665  111  126  105                .ASCII  /IVE/
016670  040  123  131                .ASCII  / SY/
016673  115  102  117                .ASCII  /MBO/
016676  114  040  105                .ASCII  /L E/
016701  103  103  040                .ASCII  /CC /
016704  105  122  122                .ASCII  /ERR/
016707  117  122  000                .ASCII  /OR/<00>
016712  045  101  123      P.AIO:    .ASCII  /%AS/
016715  111  130  040                .ASCII  /IX /
016720  123  131  115                .ASCII  /SYM/
016723  102  117  114                .ASCII  /BOL/
016726  040  105  103                .ASCII  / EC/
016731  103  040  105                .ASCII  /C E/
016734  122  122  117                .ASCII  /RRO/
016737  122  000  000                .ASCII  /R/<00><00>
016742  045  101  123      P.AIP:    .ASCII  /%AS/
016745  105  126  105                .ASCII  /EVE/
016750  116  040  123                .ASCII  /N S/
016753  131  115  102                .ASCII  /YMB/
016756  117  114  040                .ASCII  /OL /
016761  105  103  103                .ASCII  /ECC/
016764  040  105  122                .ASCII  / ER/
016767  122  117  122                .ASCII  /ROR/
016772  000  000                     .ASCII  <00><00>
016774  045  101  105      P.AIQ:    .ASCII  /%AE/
016777  111  107  110                .ASCII  /IGH/
017002  124  040  123                .ASCII  /T S/
017005  131  115  102                .ASCII  /YMB/
017010  117  114  040                .ASCII  /OL /
017013  105  103  103                .ASCII  /ECC/
017016  040  105  122                .ASCII  / ER/
017021  122  117  122                .ASCII  /ROR/
017024  000  000                     .ASCII  <00><00>
017026  045  101  103      P.AIR:    .ASCII  /%AC/
017031  117  122  122                .ASCII  /ORR/
017034  105  103  124                .ASCII  /ECT/
017037  101  102  114                .ASCII  /ABL/
017042  105  040  105                .ASCII  /E E/
017045  122  122  117                .ASCII  /RRO/
017050  122  040  111                .ASCII  /R I/
017053  116  040  105                .ASCII  /N E/
017056  103  103  040                .ASCII  /CC /
017061  106  111  105                .ASCII  /FIE/
017064  114  104  000                .ASCII  /LD/<00>
017067  000                         .ASCII  <00>
017070  045  101  125      P.AIS:    .ASCII  /%AU/
017073  116  111  124                .ASCII  /NIT/
017076  040  123  117                .ASCII  / SO/
017101  106  124  127                .ASCII  /FTW/
017104  101  122  105                .ASCII  /ARE/
```

```
017107  040     127     122             .ASCII  / WR/
017112  111     124     105             .ASCII  /ITE/
017115  040     120     122             .ASCII  / PR/
017120  117     124     105             .ASCII  /OTE/
017123  103     124     105             .ASCII  /CTE/
017126  104     000                     .ASCII  /D/<00>
017130  045     101     125     P.AIT:  .ASCII  /%AU/
017133  116     111     124             .ASCII  /NIT/
017136  040     110     101             .ASCII  / HA/
017141  122     104     127             .ASCII  /RDW/
017144  101     122     105             .ASCII  /ARE/
017147  040     127     122             .ASCII  / WR/
017152  111     124     105             .ASCII  /ITE/
017155  040     120     122             .ASCII  / PR/
017160  117     124     105             .ASCII  /OTE/
017163  103     124     105             .ASCII  /CTE/
017166  104     000                     .ASCII  /D/<00>
017170  045     101     117     P.AIU:  .ASCII  /%AO/
017173  104     104     040             .ASCII  /DD /
017176  124     122     101             .ASCII  /TRA/
017201  116     123     106             .ASCII  /NSF/
017204  105     122     040             .ASCII  /ER /
017207  101     104     104             .ASCII  /ADD/
017212  122     105     123             .ASCII  /RES/
017215  123     000     000             .ASCII  /S/<00><00>
017220  045     101     117     P.AIV:  .ASCII  /%AO/
017223  104     104     040             .ASCII  /DD /
017226  102     131     124             .ASCII  /BYT/
017231  105     040     103             .ASCII  /E C/
017234  117     125     116             .ASCII  /OUN/
017237  124     000     000             .ASCII  /T/<00><00>
017242  045     101     116     P.AIW:  .ASCII  /%AN/
017245  117     116     055             .ASCII  /ON-/
017250  105     130     111             .ASCII  /EXI/
017253  123     124     105             .ASCII  /STE/
017256  116     124     040             .ASCII  /NT /
017261  110     117     123             .ASCII  /HOS/
017264  124     040     115             .ASCII  /T M/
017267  105     115     117             .ASCII  /EMO/
017272  122     131     000             .ASCII  /RY/<00>
017275  000                             .ASCII  <00>
017276  045     101     110     P.AIX:  .ASCII  /%AH/
017301  117     123     124             .ASCII  /OST/
017304  040     115     105             .ASCII  / ME/
017307  115     117     122             .ASCII  /MOR/
017312  131     040     120             .ASCII  /Y P/
017315  101     122     111             .ASCII  /ARI/
017320  124     131     040             .ASCII  /TY /
017323  105     122     122             .ASCII  /ERR/
017326  117     122     000             .ASCII  /OR/<00>
017331  000                             .ASCII  <00>
017332  045     101     103     P.AIY:  .ASCII  /%AC/
017335  117     115     115             .ASCII  /OMM/
```

```
017340  101  116  104            .ASCII  /AND/
017343  040  124  111            .ASCII  / TI/
017346  115  117  125            .ASCII  /MOU/
017351  124  040  117            .ASCII  /T O/
017354  122  040  122            .ASCII  /R R/
017357  105  124  122            .ASCII  /ETR/
017362  131  040  114            .ASCII  /Y L/
017365  111  115  111            .ASCII  /IMI/
017370  124  040  105            .ASCII  /T E/
017373  130  103  105            .ASCII  /XCE/
017376  105  104  105            .ASCII  /EDE/
017401  104  000  000            .ASCII  /D/<00><00>
017404  045  101  123  P.AIZ:    .ASCII  /%AS/
017407  105  122  111            .ASCII  /ERI/
017412  101  114  111            .ASCII  /ALI/
017415  132  105  122            .ASCII  /ZER/
017420  057  104  105            .ASCII  <57>/DE/
017423  123  105  122            .ASCII  /SER/
017426  111  101  114            .ASCII  /IAL/
017431  111  132  105            .ASCII  /IZE/
017434  122  040  117            .ASCII  /R O/
017437  126  105  122            .ASCII  /VER/
017442  122  125  116            .ASCII  /RUN/
017445  040  117  122            .ASCII  / OR/
017450  040  125  116            .ASCII  / UN/
017453  104  105  122            .ASCII  /DER/
017456  122  125  116            .ASCII  /RUN/
017461  000                      .ASCII  <00>
017462  045  101  042  P.AJA:    .ASCII  /%A"/
017465  105  122  122            .ASCII  /ERR/
017470  117  122  040            .ASCII  /OR /
017473  104  105  124            .ASCII  /DET/
017476  105  103  124            .ASCII  /ECT/
017501  111  117  116            .ASCII  /ION/
017504  040  103  117            .ASCII  / CO/
017507  104  105  042            .ASCII  /DE"/
017512  040  105  122            .ASCII  / ER/
017515  122  117  122            .ASCII  /ROR/
017520  000  000                  .ASCII  <00><00>
017522  045  101  111  P.AJB:    .ASCII  /%AI/
017525  116  103  117            .ASCII  /NCO/
017530  116  123  111            .ASCII  /NSI/
017533  123  124  105            .ASCII  /STE/
017536  116  124  040            .ASCII  /NT /
017541  111  116  124            .ASCII  /INT/
017544  105  122  116            .ASCII  /ERN/
017547  101  114  040            .ASCII  /AL /
017552  104  101  124            .ASCII  /DAT/
017555  101  040  123            .ASCII  /A S/
017560  124  122  125            .ASCII  /TRU/
017563  103  124  125            .ASCII  /CTU/
017566  122  105  000            .ASCII  /RE/<00>
017571  000                      .ASCII  <00>
```

```
017572  045     101     104     P.AJC:  .ASCII  /%AD/
017575  122     111     126             .ASCII  /RIV/
017600  105     040     103             .ASCII  /E C/
017603  117     115     115             .ASCII  /OMM/
017606  101     116     104             .ASCII  /AND/
017611  040     124     111             .ASCII  / TI/
017614  115     105     117             .ASCII  /MEO/
017617  125     124     040             .ASCII  /UT /
017622  050     116     157             .ASCII  /(No/
017625  040     162     145             .ASCII  / re/
017630  163     160     157             .ASCII  /spo/
017633  156     163     145             .ASCII  /nse/
017636  040     157     162             .ASCII  / or/
017641  040     163     145             .ASCII  / se/
017644  145     153     040             .ASCII  /ek /
017647  151     156     143             .ASCII  /inc/
017652  157     155     160             .ASCII  /omp/
017655  154     145     164             .ASCII  /let/
017660  145     051     000             .ASCII  /e)/<00>
017663  000                             .ASCII  <00>
017664  045     101     103     P.AJD:  .ASCII  /%AC/
017667  117     116     124             .ASCII  /ONT/
017672  122     117     114             .ASCII  /ROL/
017675  114     105     122             .ASCII  /LER/
017700  040     104     105             .ASCII  / DE/
017703  124     105     103             .ASCII  /TEC/
017706  124     105     104             .ASCII  /TED
017711  040     124     122             .ASCII  / TR/
017714  101     116     123             .ASCII  /ANS/
017717  115     111     123             .ASCII  /MIS/
017722  123     111     117             .ASCII  /SIO/
017725  116     040     117             .ASCII  /N O/
017730  122     040     120             .ASCII  /R P/
017733  122     117     124             .ASCII  /ROT/
017736  117     103     117             .ASCII  /OCO/
017741  114     040     105             .ASCII  /L E/
017744  122     122     117             .ASCII  /RRO/
017747  122     000     000             .ASCII  /R/<00><00>
017752  045     101     120     P.AJE:  .ASCII  /%AP/
017755  117     123     111             .ASCII  /OSI/
017760  124     111     117             .ASCII  /TIO/
017763  116     040     105             .ASCII  /N E/
017766  122     122     117             .ASCII  /RRO/
017771  122     040     050             .ASCII  /R (/
017774  115     151     163             .ASCII  /Mis/
017777  055     163     145             .ASCII  /-se/
020002  145     153     051             .ASCII  /ek)/
020005  000                             .ASCII  <00>
020006  045     101     114     P.AJF:  .ASCII  /%AL/
020011  117     123     124             .ASCII  /OST/
020014  040     122     105             .ASCII  / RE/
020017  101     104     057             .ASCII  /AD/<57>
020022  127     122     111             .ASCII  /WRI/
```

```
020025  124  105  040                .ASCII  /TE /
020030  122  105  101                .ASCII  /REA/
020033  104  131  040                .ASCII  /DY /
020036  104  125  122                .ASCII  /DUR/
020041  111  116  107                .ASCII  /ING/
020044  057  102  105                .ASCII  <57>/BE/
020047  124  127  105                .ASCII  /TWE/
020052  105  116  040                .ASCII  /EN /
020055  124  122  101                .ASCII  /TRA/
020060  116  123  106                .ASCII  /NSF/
020063  105  122  123                .ASCII  /ERS/
020066  000  000                     .ASCII  <00><00>
020070  045  101  104          P.AJG: .ASCII  /%AD/
020073  122  111  126                .ASCII  /RIV/
020076  105  040  103                .ASCII  /E C/
020101  114  117  103                .ASCII  /LOC/
020104  113  040  104                .ASCII  /K D/
020107  122  117  120                .ASCII  /ROP/
020112  117  125  124                .ASCII  /OUT/
020115  000                         .ASCII  <00>
020116  045  101  114          P.AJM: .ASCII  /%AL/
020121  117  123  124                .ASCII  /OST/
020124  040  122  105                .ASCII  / RE/
020127  103  105  111                .ASCII  /CEI/
020132  126  105  122                .ASCII  /VER/
020135  040  122  105                .ASCII  / RE/
020140  101  104  131                .ASCII  /ADY/
020143  040  102  105                .ASCII  / BE/
020146  124  127  105                .ASCII  /TWE/
020151  105  116  040                .ASCII  /EN /
020154  123  105  103                .ASCII  /SEC/
020157  124  117  122                .ASCII  /TOR/
020162  123  000                     .ASCII  /S/<00>
020164  045  101  104          P.AJI: .ASCII  /%AD/
020167  122  111  126                .ASCII  /RIV/
020172  105  040  104                .ASCII  /E D/
020175  105  124  105                .ASCII  /ETE/
020200  103  124  105                .ASCII  /CTE/
020203  104  040  105                .ASCII  /D E/
020206  122  122  117                .ASCII  /RRO/
020211  122  000  000                .ASCII  /R/<00><00>
020214  045  101  103          P.AJJ: .ASCII  /%AC/
020217  117  116  124                .ASCII  /ONT/
020222  122  117  114                .ASCII  /ROL/
020225  114  105  122                .ASCII  /LER/
020230  040  104  105                .ASCII  / DE/
020233  124  105  103                .ASCII  /TEC/
020236  124  105  104                .ASCII  /TED/
020241  040  120  125                .ASCII  / PU/
020244  114  123  105                .ASCII  /LSE/
020247  040  117  122                .ASCII  / OR/
020252  040  123  124                .ASCII  / ST/
020255  101  124  105                .ASCII  /ATE/
```

```
020260  040  120  101          .ASCII  / PA/
020263  122  111  124          .ASCII  /RIT/
020266  131  040  105          .ASCII  /Y E/
020271  122  122  117          .ASCII  /RRO/
020274  122  000               .ASCII  /R/<00>
020276  045  101  103  P.AJL:   .ASCII  /%AC/
020301  117  116  124          .ASCII  /ONT/
020304  122  117  114          .ASCII  /ROL/
020307  114  105  122          .ASCII  /LER/
020312  040  124  111          .ASCII  / TI/
020315  115  105  117          .ASCII  /MEO/
020320  125  124  000          .ASCII  /UT/<00>
020323  000                    .ASCII  <00>
020324  045  101  105  P.AJM:   .ASCII  /%AE/
020327  116  126  105          .ASCII  /NVE/
020332  114  117  120          .ASCII  /LOP/
020335  105  057  120          .ASCII  /E/<57>/P/
020340  101  103  113          .ASCII  /ACK/
020343  105  124  040          .ASCII  /ET /
020346  122  105  101          .ASCII  /REA/
020351  104  040  105          .ASCII  /D E/
020354  122  122  117          .ASCII  /RRO/
020357  122  040  050          .ASCII  /R (/
020362  120  141  162          .ASCII  /Par/
020365  151  164  171          .ASCII  /ity/
020370  040  157  162          .ASCII  / or/
020373  040  164  151          .ASCII  / ti/
020376  155  145  157          .ASCII  /meo/
020401  165  164  051          .ASCII  /ut)/
020404  000  000               .ASCII  <00><00>
020406  045  101  105  P.AJN:   .ASCII  /%AE/
020411  116  126  105          .ASCII  /NVE/
020414  114  117  120          .ASCII  /LOP/
020417  105  057  120          .ASCII  /E/<57>/P/
020422  101  103  113          .ASCII  /ACK/
020425  105  124  040          .ASCII  /ET /
020430  127  122  111          .ASCII  /WRI/
020433  124  105  040          .ASCII  /TE /
020436  105  122  122          .ASCII  /ERR/
020441  117  122  040          .ASCII  /OR /
020444  050  120  141          .ASCII  /(Pa/
020447  162  151  164          .ASCII  /rit/
020452  171  040  157          .ASCII  /y o/
020455  162  040  164          .ASCII  /r t/
020460  151  155  145          .ASCII  /ime/
020463  157  165  164          .ASCII  /out/
020466  051  000               .ASCII  /)/<00>
020470  045  101  103  P.AJO:   .ASCII  /%AC/
020473  117  116  124          .ASCII  /ONT/
020476  122  117  114          .ASCII  /ROL/
020501  114  105  122          .ASCII  /LER/
020504  040  122  117          .ASCII  / RO/
020507  115  040  101          .ASCII  /M A/
```

```
020512  116  104  040                .ASCII  /ND /
020515  122  101  115                .ASCII  /RAM/
020520  040  120  101                .ASCII  / PA/
020523  122  111  124                .ASCII  /RIT/
020526  131  040  105                .ASCII  /Y E/
020531  122  122  117                .ASCII  /RRO/
020534  122  000                    .ASCII  /R/<00>
020536  045  101  103     P.AJP:     .ASCII  /%AC/
020541  117  116  124                .ASCII  /ONT/
020544  122  117  114                .ASCII  /ROL/
020547  114  105  122                .ASCII  /LER/
020552  040  122  101                .ASCII  / RA/
020555  115  040  120                .ASCII  /M P/
020560  101  122  111                .ASCII  /ARI/
020563  124  131  040                .ASCII  /TY /
020566  105  122  122                .ASCII  /ERR/
020571  117  122  000                .ASCII  /OR/<00>
020574  045  101  103     P.AJQ:     .ASCII  /%AC/
020577  117  116  124                .ASCII  /ONT/
020602  122  117  114                .ASCII  /ROL/
020605  114  105  122                .ASCII  /LER/
020610  040  122  117                .ASCII  / RO/
020613  115  040  120                .ASCII  /M P/
020616  101  122  111                .ASCII  /ARI/
020621  124  131  040                .ASCII  /TY /
020624  105  122  122                .ASCII  /ERR/
020627  117  122  000                .ASCII  /OR/<00>
020632  045  101  122     P.AJR:     .ASCII  /%AR/
020635  111  116  107                .ASCII  /ING/
020640  040  122  105                .ASCII  / RE/
020643  101  104  040                .ASCII  /AD /
020646  105  122  122                .ASCII  /ERR/
020651  117  122  040                .ASCII  /OR /
020654  050  120  141                .ASCII  /(Pa/
020657  162  151  164                .ASCII  /rit/
020662  171  040  157                .ASCII  /y o/
020665  162  040  164                .ASCII  /r t/
020670  151  155  145                .ASCII  /ime/
020673  157  165  164                .ASCII  /out/
020676  051  000                    .ASCII  /)/<00>
020700  045  101  122     P.AJS:     .ASCII  /%AR/
020703  111  116  107                .ASCII  /ING/
020706  040  127  122                .ASCII  / WR/
020711  111  124  105                .ASCII  /ITE/
020714  040  105  122                .ASCII  / ER/
020717  122  117  122                .ASCII  /ROR/
020722  040  050  120                .ASCII  / (P/
020725  141  162  151                .ASCII  /ari/
020730  164  171  040                .ASCII  /ty /
020733  157  162  040                .ASCII  /or /
020736  164  151  155                .ASCII  /tim/
020741  145  157  165                .ASCII  /eou/
020744  164  051  000                .ASCII  /t)/<00>
```

```
020747  000                             .ASCII  <00>
020750  111     116     124     P.AJT:  .ASCII  /INT/
020753  105     122     122             .ASCII  /ERR/
020756  125     120     124             .ASCII  /UPT/
020761  040     115     101             .ASCII  / MA/
020764  123     124     105             .ASCII  /STE/
020767  122     040     106             .ASCII  /R F/
020772  101     111     114             .ASCII  /AIL/
020775  125     122     105             .ASCII  /URE/
021000  000     000                     .ASCII  <00><00>
021002  045     101     110     P.AJU:  .ASCII  /%AH/
021005  117     123     124             .ASCII  /OST/
021010  040     101     103             .ASCII  / AC/
021013  103     105     123             .ASCII  /CES/
021016  123     040     124             .ASCII  /S T/
021021  111     115     105             .ASCII  /IME/
021024  117     125     124             .ASCII  /OUT/
021027  040     050     110             .ASCII  / (H/
021032  151     147     150             .ASCII  /igh/
021035  145     162     040             .ASCII  /er /
021040  154     145     166             .ASCII  /lev/
021043  145     154     040             .ASCII  /el /
021046  160     162     157             .ASCII  /pro/
021051  164     157     143             .ASCII  /toc/
021054  157     154     040             .ASCII  /ol /
021057  144     145     160             .ASCII  /dep/
021062  145     156     144             .ASCII  /end/
021065  145     156     164             .ASCII  /ent/
021070  051     000                     .ASCII  /)/<00>
021072  045     101     103     P.AJV:  .ASCII  /%AC/
021075  122     105     104             .ASCII  /RED/
021100  111     124     040             .ASCII  /IT /
021103  114     111     115             .ASCII  /LIM/
021106  111     124     040             .ASCII  /IT /
021111  105     130     103             .ASCII  /EXC/
021114  105     105     104             .ASCII  /EED/
021117  105     104     000             .ASCII  /ED/<00>
021122  045     101     121     P.AJW:  .ASCII  /%AQ/
021125  055     102     125             .ASCII  /-BU/
021130  123     040     115             .ASCII  /S M/
021133  101     123     124             .ASCII  /AST/
021136  105     122     040             .ASCII  /ER /
021141  105     122     122             .ASCII  /ERR/
021144  117     122     000             .ASCII  /OR/<00>
021147  000                             .ASCII  <00>
021150  045     101     103     P.AJX:  .ASCII  /%AC/
021153  117     116     124             .ASCII  /ONT/
021156  122     117     114             .ASCII  /ROL/
021161  114     105     122             .ASCII  /LER/
021164  040     106     101             .ASCII  / FA/
021167  124     101     114             .ASCII  /TAL/
021172  040     105     122             .ASCII  / ER/
021175  122     117     122             .ASCII  /ROR/
```

```
021200  000     000                     .ASLII  <00><00>
021202  045     101     111     P.AJY:  .ASCII  /%AI/
021205  116     123     124             .ASCII  /NST/
021210  122     125     103             .ASCII  /RUC/
021213  124     111     117             .ASCII  /TIO/
021216  116     040     114             .ASCII  /N L/
021221  117     117     120             .ASCII  /OOP/
021224  040     124     111             .ASCII  / TI/
021227  115     105     117             .ASCII  /MEO/
021232  125     124     000             .ASCII  /UT/<00>
021235  000                             .ASCII  <00>
021236  045     101     111     P.AJZ:  .ASCII  /%AI/
021241  114     114     105             .ASCII  /LLE/
021244  107     101     114             .ASCII  /GAL/
021247  040     126     111             .ASCII  / VI/
021252  122     124     125             .ASCII  /RTU/
021255  101     114     040             .ASCII  /AL /
021260  103     111     122             .ASCII  /CIR/
021263  103     125     111             .ASCII  /CUI/
021266  124     040     111             .ASCII  /T I/
021271  104     000     000             .ASCII  /D/<00><00>
021274  045     101     111     P.AKA:  .ASCII  /%AI/
021277  116     124     105             .ASCII  /NTE/
021302  122     122     125             .ASCII  /RRU/
021305  120     124     040             .ASCII  /PT /
021310  126     105     103             .ASCII  /VEC/
021313  124     117     122             .ASCII  /TOR/
021316  040     111     114             .ASCII  / IL/
021321  114     105     107             .ASCII  /LEG/
021324  101     114     000             .ASCII  /AL/<00>
021327  000                             .ASCII  <00>
021330  045     101     115     P.AKB:  .ASCII  /%AM/
021333  101     111     116             .ASCII  /AIN/
021336  124     105     116             .ASCII  /TEN/
021341  101     116     103             .ASCII  /ANC/
021344  105     040     122             .ASCII  /E R/
021347  105     101     104             .ASCII  /FAD/
021352  057     127     122             .ASCII  <57>/WR/
021355  111     124     105             .ASCII  /ITE/
021360  040     111     116             .ASCII  / IN/
021363  126     101     114             .ASCII  'VAL/
021366  111     104     040             .ASCII  /ID /
021371  122     105     107             .ASCII  /REG/
021374  111     117     116             .ASCII  /TON/
021377  040     111     104             .ASCII  / ID/
021402  105     116     124             .ASCII  /ENT/
021405  111     106     111             .ASCII  /IFI/
021410  105     122     000             .ASCII  /ER/<00>
021413  000                             .ASCII  <00>
021414  045     101     115     P.AKC:  .ASCII  /%AM/
021417  101     111     116             .ASCII  /AIN/
021422  124     105     116             .ASCII  /TEN/
021425  101     116     103             .ASCII  /ANC/
```

```
021430  105  040  127                .ASCII  /E W/
021433  122  111  124                .ASCII  /RIT/
021436  105  040  114                .ASCII  /E L/
021441  117  101  104                .ASCII  /OAD/
021444  040  124  117                .ASCII  / TO/
021447  040  116  117                .ASCII  / NO/
021452  116  055  114                .ASCII  /N-L/
021455  117  101  104                .ASCII  /OAD/
021460  101  102  114                .ASCII  /ABL/
021463  105  040  103                .ASCII  /E C/
021466  117  116  124                .ASCII  /ONT/
021471  122  117  114                .ASCII  /ROL/
021474  114  105  122                .ASCII  /LER/
021477  000                        .ASCII  <00>
021500  045  101  103        P.AKD:  .ASCII  /%AC/
021503  117  116  124                .ASCII  /ONT/
021506  122  117  114                .ASCII  /ROL/
021511  114  105  122                .ASCII  /LER/
021514  040  122  101                .ASCII  / RA/
021517  115  040  105                .ASCII  /M E/
021522  122  122  117                .ASCII  /RRO/
021525  122  040  050                .ASCII  /R (/
021530  116  157  156                .ASCII  /Non/
021533  055  160  141                .ASCII  /-pa/
021536  162  151  164                .ASCII  /rit/
021541  171  051  000                .ASCII  /y)/<00>
021544  045  101  111        P.AKE:  .ASCII  /%AI/
021547  116  111  124                .ASCII  /NIT/
021552  040  123  105                .ASCII  / SE/
021555  121  125  105                .ASCII  /QUE/
021560  116  103  105                .ASCII  /NCE/
021563  040  105  122                .ASCII  / ER/
021566  122  117  122                .ASCII  /ROR/
021571  000                        .ASCII  <00>
021572  045  101  110        P.AKF:  .ASCII  /%AH/
021575  111  107  110                .ASCII  /IGH/
021600  105  122  040                .ASCII  /ER /
021603  114  105  126                .ASCII  /LEV/
021606  105  114  040                .ASCII  /EL /
021611  120  122  117                .ASCII  /PRO/
021614  124  117  103                .ASCII  /TOC/
021617  117  114  040                .ASCII  /OL /
021622  111  116  103                .ASCII  /INC/
021625  117  115  120                .ASCII  /OMP/
021630  101  124  111                .ASCII  /ATI/
021633  102  111  114                .ASCII  /BIL/
021636  111  124  131                .ASCII  /ITY/
021641  040  105  122                .ASCII  / ER/
021644  122  117  122                .ASCII  /ROR/
021647  000                        .ASCII  <00>
021650  045  101  120        P.AKG:  .ASCII  /%AP/
021653  125  122  107                .ASCII  /URG/
021656  105  057  120                .ASCII  /E/<57>/P/
```

```
021661      117      114      114                .ASCII  /OLL/
021664      040      110      101                .ASCII  / HA/
021667      122      104      127                .ASCII  /RDW/
021672      101      122      105                .ASCII  /ARE/
021675      040      106      101                .ASCII  / FA/
021700      111      114      125                .ASCII  /ILU/
021703      122      105      000                .ASCII  /RE/<00>
021706      045      101      115      P.AKH:    .ASCII  /%AM/
021711      101      120      120                .ASCII  /APP/
021714      111      116      107                .ASCII  /ING/
021717      040      122      105                .ASCII  / RE/
021722      107      111      123                .ASCII  /GIS/
021725      124      105      122                .ASCII  /TER/
021730      040      122      105                .ASCII  / RE/
021733      101      104      040                .ASCII  /AD /
021736      106      101      111                .ASCII  /FAI/
021741      114      125      122                .ASCII  /LUR/
021744      105      040      050                .ASCII  /E (/
021747      120      141      162                .ASCII  /Par/
021752      151      164      171                .ASCII  /ity/
021755      040      157      162                .ASCII  / or/
021760      040      164      151                .ASCII  / ti/
021763      155      145      157                .ASCII  /meo/
021766      165      164      051                .ASCII  /ut)/
021771      000                                  .ASCII  <00>
021772   020276'                        P.AJK:    .WORD   P.AJL
021774   020324'                                  .WORD   P.AJM
021776   020406'                                  .WORD   P.AJN
022000   020470'                                  .WORD   P.AJO
022002   020536'                                  .WORD   P.AJP
022004   020574'                                  .WORD   P.AJQ
022006   020632'                                  .WORD   P.AJR
022010   020700'                                  .WORD   P.AJS
022012   020750'                                  .WORD   P.AJT
022014   021002'                                  .WORD   P.AJU
022016   021072'                                  .WORD   P.AJV
022020   021122'                                  .WORD   P.AJW
022022   021150'                                  .WORD   P.AJX
022024   021202'                                  .WORD   P.AJY
022026   021236'                                  .WORD   P.AJZ
022030   021274'                                  .WORD   P.AKA
022032   021330'                                  .WORD   P.AKB
022034   021414'                                  .WORD   P.AKC
022036   021500'                                  .WORD   P.AKD
022040   021544'                                  .WORD   P.AKE
022042   021572'                                  .WORD   P.AKF
022044   021650'                                  .WORD   P.AKG
022046   021706'                                  .WORD   P.AKH
022050      045      101      124      P.AKJ:    .ASCII  /%AT/
022053      061      061      040                .ASCII  /11 /
022056      103      120      125                .ASCII  /CPU/
022061      040      106      101                .ASCII  / FA/
022064      111      114      125                .ASCII  /ILU/
```

```
022067  122  105  000                .ASCII  /RE/<00>
022072  045  101  116      P.AKK:    .ASCII  /%AN/
022075  117  116  055                .ASCII  /ON-/
022100  120  101  122                .ASCII  /PAR/
022103  111  124  131                .ASCII  /ITY/
022106  040  122  101                .ASCII  / RA/
022111  115  040  105                .ASCII  /M E/
022114  122  122  117                .ASCII  /RRO/
022117  122  000  000                .ASCII  /R/<00><00>
022122  045  101  123      P.AKL:    .ASCII  /%AS/
022125  124  101  124                .ASCII  /TAT/
022130  105  040  115                .ASCII  /E M/
022133  101  103  110                .ASCII  /ACH/
022136  111  116  105                .ASCII  /INE/
022141  040  106  101                .ASCII  / FA/
022144  111  114  125                .ASCII  /ILU/
022147  122  105  040                .ASCII  /RE /
022152  055  040  124                .ASCII  /- T/
022155  061  061  040                .ASCII  /11 /
022160  101  104  104                .ASCII  /ADD/
022163  122  105  123                .ASCII  /RES/
022166  123  040  122                 ASCII  /S R/
022171  105  107  111                .ASCII  /EGI/
022174  123  124  105                .ASCII  /STE/
022177  122  000  000                .ASCII  /R/<00><00>
022202  045  101  123      P.AKM:    .ASCII  /%AS/
022205  124  101  124                .ASCII  /TAT/
022210  105  040  115                .ASCII  /E M/
022213  101  103  110                .ASCII  /ACH/
022216  111  116  105                .ASCII  /INE/
022221  040  106  101                .ASCII  / FA/
022224  111  114  125                .ASCII  /ILU/
022227  122  105  040                .ASCII  /RE /
022232  055  040  121                .ASCII  /- Q/
022235  055  102  125                .ASCII  /-BU/
022240  123  040  101                .ASCII  /S A/
022243  104  104  122                .ASCII  /DDR/
022246  105  123  123                .ASCII  /ESS/
022251  040  122  105                .ASCII  / RE/
022254  107  111  123                .ASCII  /GIS/
022257  124  105  122                .ASCII  /TER/
022262  000  000                     .ASCII  <00><00>
022264  045  101  123      P.AKN:    .ASCII  /%AS/
022267  124  101  124                .ASCII  /TAT/
022272  105  040  115                .ASCII  /E M/
022275  101  103  110                .ASCII  /ACH/
022300  111  116  105                .ASCII  /INE/
022303  040  106  101                .ASCII  / FA/
022306  111  114  125                .ASCII  /ILU/
022311  122  105  040                .ASCII  /RE /
022314  055  040  103                .ASCII  /- C/
022317  122  103  040                .ASCII  /RC /
022322  122  105  107                .ASCII  /REG/
```

```
022325  111     123     124             .ASCII  /IST/
022330  105     122     000             .ASCII  /ER/<00>
022333  000                             .ASCII  <00>
022334  045     101     123     P.AKO:  .ASCII  /%AS/
022337  124     101     124             .ASCII  /TAT/
022342  105     040     115             .ASCII  /E M/
022345  101     103     110             .ASCII  /ACH/
022350  111     116     105             .ASCII  /INE/
022353  040     106     101             .ASCII  / FA/
022356  111     114     125             .ASCII  /ILU/
022361  122     105     040             .ASCII  /RE /
022364  055     040     123             .ASCII  /- S/
022367  105     122     111             .ASCII  /ERI/
022372  101     114     111             .ASCII  /ALI/
022375  132     105     122             .ASCII  /ZER/
022400  057     104     105             .ASCII  <57>/DE/
022403  123     105     122             .ASCII  /SER/
022406  111     101     114             .ASCII  /IAL/
022411  111     132     105             .ASCII  /IZE/
022414  122     040     122             .ASCII  /R R/
022417  105     107     111             .ASCII  /EGI/
022422  123     124     105             .ASCII  /STE/
022425  122     000     000             .ASCII  /R/<00><00>
022430  045     101     123     P.AKP:  .ASCII  /%AS/
022433  124     101     124             .ASCII  /TAT/
022436  105     040     115             .ASCII  /E M/
022441  101     103     110             .ASCII  /ACH/
022444  111     116     105             .ASCII  /INE/
022447  040     106     101             .ASCII  / FA/
022452  111     114     125             .ASCII  /ILU/
022455  122     105     040             .ASCII  /RE /
022460  055     040     127             .ASCII  /- W/
022463  122     117     116             .ASCII  /RON/
022466  107     040     110             .ASCII  /G H/
022471  101     122     104             .ASCII  /ARD/
022474  127     101     122             .ASCII  /WAR/
022477  105     040     126             .ASCII  /E V/
022502  105     122     123             .ASCII  /ERS/
022505  111     117     116             .ASCII  /ION/
022510  000     000                     .ASCII  <00><00>
022512  022050'                 P.AKI:  .WORD   P.AKJ
022514  022072'                         .WORD   P.AKK
022516  022122'                         .WORD   P.AKL
022520  022202'                         .WORD   P.AKM
022522  022264'                         .WORD   P.AKN
022524  022334'                         .WORD   P.AKO
022526  022430'                         .WORD   P.AKP
022530  045     116     045     P.AKG:  .ASCII  /%N%/
022533  101     132     122             .ASCII  /AZR/
022536  121     101     040             .ASCII  /QA /
022541  104     105     126             .ASCII  /DEV/
022544  040     106     124             .ASCII  / FT/
022547  114     040     040             .ASCII  /L  /
```

```
022552  045  132  065          .ASCII  /%Z5/
022555  045  101  040          .ASCII  /%A /
022560  117  116  040          .ASCII  /ON /
022563  125  116  111          .ASCII  /UNI/
022566  124  040  045          .ASCII  /T %/
022571  132  062  045          .ASCII  /Z2%/
022574  101  040  124          .ASCII  /A T/
022577  123  124  040          .ASCII  /ST /
022602  060  060  061          .ASCII  /001/
022605  040  123  125          .ASCII  / SU/
022610  102  040  060          .ASCII  /B 0/
022613  060  060  040          .ASCII  /00 /
022616  120  103  072          .ASCII  /PC:/
022621  040  045  117          .ASCII  / %O/
022624  066  000                .ASCII  /6/<00>
022626  045  116  045  P.AKR:   .ASCII  /%N%/
022631  101  132  122          .ASCII  /AZR/
022634  121  101  040          .ASCII  /QA /
022637  110  122  104          .ASCII  /HRD/
022642  040  105  122          .ASCII  / ER/
022645  122  040  040          .ASCII  /R  /
022650  045  132  065          .ASCII  /%Z5/
022653  045  101  040          .ASCII  /%A /
022656  117  116  040          .ASCII  /ON /
022661  125  116  111          .ASCII  /UNI/
022664  124  040  045          .ASCII  /T %/
022667  132  062  045          .ASCII  /Z2%/
022672  101  040  124          .ASCII  /A T/
022675  123  124  040          .ASCII  /ST /
022700  060  060  061          .ASCII  /001/
022703  040  123  125          .ASCII  / SU/
022706  102  040  060          .ASCII  /B 0/
022711  060  060  040          .ASCII  /00 /
022714  120  103  072          .ASCII  /PC:/
022717  040  045  117          .ASCII  / %O/
022722  066  000                .ASCII  /6/<00>
022724  045  116  045  P.AKS:   .ASCII  /%N%/
022727  101  132  122          .ASCII  /AZR/
022732  121  101  040          .ASCII  /QA /
022735  123  106  124          .ASCII  /SFT/
022740  040  105  122          .ASCII  / ER/
022743  122  040  040          .ASCII  /R  /
022746  045  132  065          .ASCII  /%Z5/
022751  045  101  040          .ASCII  /%A /
022754  117  116  040          .ASCII  /ON /
022757  125  116  111          .ASCII  /UNI/
022762  124  040  045          .ASCII  /T %/
022765  132  062  045          .ASCII  /Z2%/
022770  101  040  124          .ASCII  /A T/
022773  123  124  040          .ASCII  /ST /
022776  060  060  061          .ASCII  /001/
023001  040  123  125          .ASCII  / SU/
023004  102  040  060          .ASCII  /B 0/
```

```
023007      060       060       040                    .ASCII  /00 /
023012      120       103       072                    .ASCII  /PC:/
023015      040       045       117                    .ASCII  / %O/
023020      066       045       116                    .ASCII  /6%N/
023023      000                                        .ASCII  <00>
023024      045       116       045          P.AKT:    .ASCII  /%N%/
023027      101       111       057                    .ASCII  /AI/<57>
023032      117       040       122                    .ASCII  /O R/
023035      105       121       125                    .ASCII  /EQU/
023040      105       123       124                    .ASCII  /EST/
023043      040       106       101                    .ASCII  / FA/
023046      111       114       105                    .ASCII  /ILE/
023051      104       045       116                    .ASCII  /D%N/
023054      000       000                              .ASCII  <00><00>
023056      045       123       064          P.AKU:    .ASCII  /%S4/
023061      000                                        .ASCII  <00>
023062      045       116       000          P.AKV:    .ASCII  /%N/<00>
023065      000                                        .ASCII  <00>
023066      045       101       040          P.AKW:    .ASCII  /%A /
023071      055       040       000                    .ASCII  /- /<00>
023074      045       101       052          P.AKX:    .ASCII  /%A*/
023077      040       000       000                    .ASCII  / /<00><00>
023102  000000C                                L$HWLEN::
                                                       .WORD   <<L$NDHW-L$HWLEN>/2>
023104  172150                                 HWPT.IP.ADDR::
                                                       .WORD   -5630
023106  000154                                 HWPT.VECTOR::
                                                       .WORD   154
023110  000004                                 HWPT.BR.LEVEL::
                                                       .WORD   4
023112  000200                                 HWPT.DISK::
                                                       .WORD   200
023114  000000                                 HWPTS0.LBN::
                                                       .WORD   0
023116  000000                                 HWPTS1.LBN::
                                                       .WORD   0
023120  177777                                 HWPTE0.LBN::
                                                       .WORD   -1
023122  000000                                 HWPTE1.LBN::
                                                       .WORD   0
023124  020040                                 NAME.LO::
                                                       .WORD   20040
023126  020040                                 NAME.HI::
                                                       .WORD   20040
023130                                         L$NDHW::.BLKW   1
023132  000000C                                L$SWLEN::
                                                       .WORD   <<L$NDSW-L$SWLEN>/2>
023134  000040                                 SWP.ERROR::
                                                       .WORD   40
023136  000000                                 SWP.XFER::
                                                       .WORD   0
023140  054046                                 SWP.FLAGS::
                                                       .WORD   54046
```

```
023142  000000                  SWP.DPAT::
                                        .WORD   0
023144  000143                  SWP.RAT::
                                        .WORD   143
023146  000000                  SWP.TIME::
                                        .WORD   0
023150  000013                  DUPROUND::
                                        .WORD   13
023152  000020                  SWP.UCNT::
                                        .WORD   20
023154                          SWP.UDPAT::
                                        .BLKW   20
023214                          L$NDSW::.BLKW   1
023216  000000                  L$PROT::.WORD   0
023220  177777                          .WORD   -1
023222  000006                          .WORD   6


000000                                  .PSECT  $FFF$,  D  ,  GBL
000000                          CST::   .BLKW   53
000126                          CST.ADDR::
                                        .BLKW   1
000130                          DCT.:   .BLKW   11
000152                          DCT.ADDR::
                                        .BLKW   1
000154                          RDRX.ADDR::
                                        .BLKW   1
000156                          IRDRX.ADDR::
                                        .BLKW   1
000160                          BST::   .BLKW   10
000200                          TALLY:: .BLKW   154
000530                          T.ADDR::.BLKW   1
000532                          DUPPKT::.BLKW   401
001534                          TRK.SGN::
001534      001                         .BYTE   1
001535      001                         .BYTE   1
001536      001                         .BYTE   1
001537      001                         .BYTE   1
001540  000020                  RDM.CNT::
                                        .WORD   20
001542                          RANDOM::.BLKW   20
001602                          C.ERR.TBL::
                                        .BLKW   1
001604                          MSCP.PKT::
                                        .BLKW   644
003314                          IPKT.ADDR::
                                        .BLKW   1
003316                          PKT.USE::
                                        .BLKW   6
003332                          RETPKT::.BLKW   260
004072                          RP.USE::.BLKW   4
004102                          RP.INDX::
```

```
                        .BLKW   1
004104                  RP.ADDR::
                        .BLKW   1
004106                  ELOG.PKT::
                        .BLKW   655
005640                  BUFF.ADDR::
                        .BLKW   10
005660                  BUFF.OWN::
                        .BLKW   4
005670                  IODQ::  .BLKW   4
005700                  IODQ.IN::
                        .BLKW   1
005702                  IODQ.OUT::
                        .BLKW   1
005704                  ENTRY.REASON::
                        .BLKB   1
005705                  EOP.FLAG::
                        .BLKB   1
005706                  DUP.FLAGS::
                        .BLKW   1
005710                  CCTLR:: .BLKW   1
005712                  CDISK:: .BLKW   1
005714                  CUOFF:: .BLKW   1
005716                  CTLR.CNT::
                        .BLKW   1
005720                  DUR::   .BLKW   2
005724                  QIO::   .BLKB   1
                        .EVEN
005726                  FREE.MEM.ADDR::
                        .BLKW   1
005730                  BYTS.PER.QIO::
                        .BLKW   1
005732                  ST.CODE::
                        .BLKW   1
005734                  SB.CODE::
                        .BLKW   1
005736                  STEP::  .BLKW   1
005740                  OF.RC:: .BLKW   1
005742                  SA.REG::.BLKW   1
005744                  CMD.TIME::
                        .BLKW   1
005746                  NEX::   .BLKW   1
005750                  CRN.LOW::
                        .BLKW   1
005752                  CRN.HIGH::
                        .BLKW   1
005754                  TEMP1:: .BLKW   1
005756                  TEMP2:: .BLKW   1
005760                  CREDIT.BAL::
                        .BLKW   1
005762                  NEXT.PKT.USE::
                        .BLKB   1
005763                  HOURS:: .BLKB   1
```

```
005764                          MINUTES::
                                        .BLKB   1
                                        .EVEN
005766                          CLK.TICKS::
                                        .BLKW   1
005770                          FERO.LBN::
                                        .BLKW   1
005772                          FER1.LBN::
                                        .BLKW   1
005774                          CLK.PRESENT::
                                        .BLKB   1
005775                          HOE.FLAG::
                                        .BLKB   1
005776                          S.PATTERN::
                                        .BLKW   1
006000                          S.DUPPKT::
                                        .BLKW   1
006002                          P.INDEX::
                                        .BLKW   1
006004  000000                  RD.COUNT::
                                        .WORD   0
006006                          BRLEVEL::
                                        .BLKW   1
006010                          D.FAIL::.BLKB   1
006011                          FORCED.ERROR::
                                        .BLKB   1
006012                          FER.LBN::
                                        .BLKW   1
006014                          FER.BC::.BLKW   1
006016                          INIT.OCCURED::
006016  000                             .BYTE   0
006017                          ADDR.VECT.OK::
006017  000                             .BYTE   0


                                .GLOBL  L$RPT, L$INIT, L$CLEAN, L$LAST
                                .GLOBL  L$HARD, L$DU, L$AU, L$AUTO, L$SOFT
                                .GLOBL  T$PTHV, L$DVTYP, L$DESC, T1


000001                          ON==                    1
000002                          OFF==                   2
100000                          BIT15==                 -100000
040000                          BIT14==                 40000
020000                          BIT13==                 20000
010000                          BIT12==                 10000
004000                          BIT11==                 4000
002000                          BIT10==                 2000
001000                          BIT09==                 1000
000400                          BIT08==                 400
000200                          BIT07==                 200
000100                          BIT06==                 100
000040                          BIT05==                 40
```

000020	BIT04==	20
000010	BIT03==	10
000004	BIT02==	4
000002	BIT01==	2
000001	BIT00==	1
001000	BIT9==	1000
000400	BIT8==	400
000200	BIT7==	200
000100	BIT6==	100
000040	BIT5==	40
000020	BIT4==	20
000010	BIT3==	10
000004	BIT2==	4
000002	BIT1==	2
000001	BIT0==	1
000035	EF.NEW==	35
000034	EF.PWR==	34
000040	EF.START==	40
000037	EF.RESTART==	37
000036	EF.CONTINUE==	36
000340	PRI07==	340
000300	PRI06==	300
000240	PRI05==	240
000200	PRI04==	200
000140	PRI03==	140
000100	PRI02==	100
000040	PRI01==	40
000000	PRI00==	0
000004	EVL==	4
000010	LOT==	10
000020	ADR==	20
000040	IDU==	40
000100	ISR==	100
000200	UAM==	200
000400	BOE==	400
001000	PNT==	1000
002000	PRI==	2000
004000	IXE==	4000
010000	IBE==	10000
020000	IER==	20000
040000	LOE==	40000
100000	HOE==	-100000
000126'	L$ERRTBL==	ERRTYP
023134'	L$SW==	L$SWLEN*2
023104'	L$HW==	L$HWLEN*2
000011'	L$DEPC==	L$REV*1
000136'	PTCH1==	P.AAA
000210'	PTCH2==	P.AAB
000262'	PTCH3==	P.AAC
000334'	PTCH4==	P.AAD
000406'	PTCH5==	P.AAE
000460'	HWQ1==	P.AAF
000474'	HWQ2==	P.AAG

000504'	HWQ3--	P.AAH
000546'	HWQ4--	P.AAI
000564'	HWQ5--	P.AAJ
000634'	HWQ6A--	P.AAK
000706'	HWQ6B--	P.AAL
000762'	HWQ7A--	P.AAM
001032'	HWQ7B--	P.AAN
001102'	HWQ8--	P.AAO
001160'	HWQ9--	P.AAP
001260'	HWQ10--	P.AAQ
001310'	HWQ11--	P.AAR
001342'	SWQ1--	P.AAS
001364'	SWQ2--	P.AAT
001444'	SWQ4--	P.AAU
001466'	SWQ7--	P.AAV
001540'	SWQ9--	P.AAW
001614'	SWQ10--	P.AAX
001660'	SWQ11--	P.AAY
001712'	SWQ12--	P.AAZ
002010'	SWQ13--	P.ABA
002066'	SWQ14--	P.ABB
002140'	SWQ15--	P.ABC
002210'	SWQ17--	P.ABD
002306'	SWQ19--	P.ABE
002376'	SWQ20--	P.ABF
002464'	SWQ21--	P.ABG
002550'	SWQ22--	P.ABH
002610'	SWQ23--	P.ABI
002662'	SWQ24--	P.ABJ
002726'	SWQ25--	P.ABK
003000'	SWQ26--	P.ABL
003042'	SWM1--	P.ABM
003140'	NULL--	P.ABN
003142'	DBM5--	P.ABO
003170'	DBM12--	P.ABP
003244'	DBM15--	P.ABQ
003274'	DBM18--	P.ABR
003346'	DBM19--	P.ABS
003432'	DBM20--	P.ABT
003506'	DBM21--	P.ABU
003570'	DBM22--	P.ABV
003634'	DBM23--	P.ABW
003672'	DBM25--	P.ABX
003740'	DBM26--	P.ABY
003772'	DBM27--	P.ABZ
004044'	DBM28A--	P.ACA
004104'	DBM28B--	P.ACB
004144'	DBM29--	P.ACC
004212'	DBM32--	P.ACD
004266'	DBM101--	P.ACE
004314'	DBM104--	P.ACF
004356'	DBM105--	P.ACG
004414'	DBM107--	P.ACH

004452'	DBM108**	P.ACI
004542'	DBM109**	P.ACJ
004622'	DBM111**	P.ACK
004722'	DBM112**	P.ACL
005024'	DBM120**	P.ACM
005116'	DBM121**	P.ACN
005206'	DU.MSG**	P.ACO
005726'	DU.RSN**	P.ACP
005754'	MSG.01**	P.ADB
006006'	MSG.02**	P.ADC
006042'	MSG.03**	P.ADD
006074'	RPT1**	P.ADE
006160'	RPT2**	P.ADF
006224'	RPT3**	P.ADG
006310'	RPT4**	P.ADH
006354'	RPT5**	P.ADI
006442'	RPT6**	P.ADJ
006506'	RPT7**	P.ADK
006524'	RPT8**	P.ADL
006552'	RPT9**	P.ADM
006604'	RPT10**	P.ADN
006672'	RPT11**	P.ADO
006740'	RPT12**	P.ADP
007006'	RPT13**	P.ADQ
007106'	RPT14**	P.ADR
007204'	RPT15**	P.ADS
007304'	RPT16**	P.ADT
007366'	EGS.01**	P.ADU
007406'	EGS.02**	P.ADV
007500'	EGD.10**	P.ADW
007540'	EGD.11**	P.ADX
007564'	EGD.12**	P.ADY
007612'	EGD.13**	P.ADZ
007640'	EGD.14**	P.AEA
007670'	EGD.15**	P.AEB
007706'	EGD.16**	P.AEC
007736'	EGD.17**	P.AED
007754'	EGD.18**	P.AEE
007774'	EGD.19**	P.AEF
010034'	EGD.20**	P.AEG
010122'	EGD.21**	P.AEH
010234'	EGD.22**	P.AEI
010274'	EGD.23**	P.AEJ
010336'	EGD.24**	P.AEK
010402'	EGH.30**	P.AEL
010426'	EBS.01**	P.AEM
010470'	EBD.10**	P.AEN
010530'	EBD.12**	P.AEO
010576'	EBD.13**	P.AEP
010630'	EBD.14**	P.AEQ
010670'	EBD.18**	P.AER
010724'	EBD.19**	P.AES
011004'	EBD.24**	P.AET

```
011060'  EH.0--      P.AEU
011116'  EH.1--      P.AEV
011154'  EH.2--      P.AEW
011214'  EH.3--      P.AEX
011252'  EH.4--      P.AEY
011276'  EH.5--      P.AEZ
011326'  EH.6--      P.AFA
011356'  EH.7--      P.AFB
011406'  EH.8--      P.AFC
011442'  EH.9--      P.AFD
011472'  EH.10--     P.AFE
011522'  EH.12--     P.AFF
011556'  EH.13--     P.AFG
011630'  ERR.OO--    P.AFH
012320'  ERR.COD--   P.AFI
012354'  ELG.OO--    P.AFX
012706'  ELG.FMT--   P.AFY
012720'  EX.SA--     P.AGE
012736'  EX.SC--     P.AGF
012766'  EX.SBO--    P.AGG
012772'  EX.SB--     P.AGH
013014'  EX.CMD--    P.AGI
013034'  EX.RD--     P.AGJ
013044'  EX.WRT--    P.AGK
013054'  EX.CMP--    P.AGL
013070'  EX.ONL--    P.AGM
013102'  EX.ACC--    P.AGN
013114'  EX.OP--     P.AGO
013120'  EX.BB--     P.AGP
013210'  EX.BB1--    P.AGQ
013304'  EX.BBU--    P.AGR
013374'  EX.LBN--    P.AGS
013432'  EX.PBN--    P.AGT
013470'  EX.LBR--    P.AGU
013536'  EX.LBW--    P.AGV
013604'  EX.RBN--    P.AGW
013664'  EX.CBC--    P.AGX
013730'  EX.CBR--    P.AGY
014002'  EX.CBW--    P.AGZ
014054'  EX.BC--     P.AHA
014130'  EX.BD--     P.AHB
014206'  EX.BDR--    P.AHC
014276'  EX.BDW--    P.AHD
014366'  EX.RP--     P.AHE
014454'  EX.WRD--    P.AHF
014464'  EX.TIM--    P.AHG
014524'  XX13--      P.AHH
014546'  XX23--      P.AHI
014602'  XX32--      P.AHJ
014630'  XX33--      P.AHK
014666'  XX34--      P.AHL
014736'  CER.01--    P.AHM
015002'  CER.02--    P.AHN
```

015056'	SC.SDI**	P.AHO
015102'	SC.CON**	P.AHP
015124'	SC.DUP**	P.AHQ
015154'	SC.ONL**	P.AHR
015176'	SC.SON**	P.AHS
015216'	SC.UNK**	P.AHT
015276'	SC.VOL**	P.AHU
015356'	SC.IOP**	P.AHV
015430'	SC.DIS**	P.AHW
015522'	SC.FER**	P.AHX
015610'	SC.FE2**	P.AHY
015700'	SC.ISM**	P.AHZ
015760'	SC.IS2**	P.AIA
016040'	SC.DST**	P.AIB
016114'	SC.DS2**	P.AIC
016166'	SC.ECC**	P.AID
016250'	SC.ECD**	P.AIE
016302'	SC.RCT**	P.AIF
016322'	SC.FUL**	P.AIG
016376'	SC.576**	P.AIH
016452'	SC.FCT**	P.AII
016520'	SC.EC1**	P.AIJ
016550'	SC.EC2**	P.AIK
016600'	SC.EC3**	P.AIL
016632'	SC.EC4**	P.AIM
016662'	SC.EC5**	P.AIN
016712'	SC.EC6**	P.AIO
016742'	SC.EC7**	P.AIP
016774'	SC.EC8**	P.AIQ
017026'	SC.EC9**	P.AIR
017070'	SC.SWP**	P.AIS
017130'	SC.HWP**	P.AIT
017170'	SC.ODA**	P.AIU
017220'	SC.ODB**	P.AIV
017242'	SC.NXM**	P.AIW
017276'	SC.PAR**	P.AIX
017332'	SC.CTO**	P.AIY
017404'	SC.SDS**	P.AIZ
017462'	SC.EDC**	P.AJA
017522'	SC.IDS**	P.AJB
017572'	SC.SRT**	P.AJC
017664'	SC.SRI**	P.AJD
017752'	SC.POE**	P.AJE
020006'	SC.RDY**	P.AJF
020070'	SC.CLK**	P.AJG
020116'	SC.RSP**	P.AJH
020164'	SC.SUR**	P.AJI
020214'	SC.PSP**	P.AJJ
021772'	CNTR.ERR**	P.AJK
022512'	RDRX.ERR**	P.AKI
022530'	DF.MSG**	P.AKQ
022626'	HRD.MSG**	P.AKR
022724'	SFT.MSG**	P.AKS

```
        023024'                 MRD.SUB--       P.AKT
        023056'                 SPACE4--        P.AKU
        023062'                 CRLF--          P.AKV
        023066'                 DASH--          P.AKW
        023074'                 ASTERISK--      P.AKX
        023104'                 DFPTBL--        L$HWLEN*2
        023134'                 SFPTBL--        L$SWLEN*2


;                               PSECT SUMMARY
;
;       Psect Name              Words     Attributes
;        $CODE$                  4938       RO  ,  I  ,  LCL,  REL,  CON
;        $FFF$                   1544       RW  ,  D  ,  GBL,  REL,  CON


;                           Library Statistics
;
;                                    ------- Symbols -------      Pages       Processing
;       File                        Total    Loaded   Percent     Mapped      Time
;
;  DISK$USER2:[POWERS.ZRQ]ZRQAGO.L16;10      407       181        44          21        00:00.1


;                                   COMMAND QUALIFIERS
;       BLISS/PDP11 ZRQAGO.BL1/LIST*ZRQAGO.LS1/OBJECT*ZRQAGO.OB1/SOURCE*PAGE:53
```

```
;    3594  0     module ZRQAM2 (
;    3595  0
;    3596  0     %title 'RD/RX EXERCISER'
;    3597  0                 ident = 'V02.2',
;    3598  0                 addressing_mode (absolute),
;    3599  0                 environment (noeis)
;    3600  0                 ) =
;    3601  0
;    3602  1     begin
;    3603  1
;    3604  1     %sbttl 'DECLARATIONS'
;    3605  1
;    3606  1     library 'ZRQAG0.L16';                          ! RDRX EXERCISER GLOBAL LIBRARY
;    3607  1
;    3608  1     !ZZZ require 'BLSMAC.REQ';                     ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    3609  1     require 'HSAXA0.BLB';                          ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    5350  1
;    5351  1     forward routine
;    5352  1         NEX_TRAP : L$ISR novalue,
;    5353  1         EMS_01 : novalue,
;    5354  1         EMS_TIM : novalue,
;    5355  1         EMS_DBN : NOVALUE,                                                  !ZZZ
;    5356  1         EMS_BLK : NOVALUE,                                                  !ZZZ
;    5357  1         SET_CPAR : novalue,
;    5358  1         SET_UPAR : novalue;
;    5359  1
;    5360  1     external
;    5361  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    5362  1                             ! RUN-TIME CONTROLLER STATUS TABLES
;    5363  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    5364  1                             ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    5365  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    5366  1                             ! DRIVER CONTROLLER TABLES
;    5367  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    5368  1                             ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    5369  1         RDRX_ADDR : ref rdrx field (RC_REG),
;    5370  1                             ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    5371  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;    5372  1                             ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    5373  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                             !ZZZ
;    5374  1                             !CONTAINS LBNS (HI + LO FIELDS) FOR SEQUENTIAL  !ZZZ
;    5375  1                             !I/O TRANSFER FOR EACH UNIT.                    !ZZZ
;    5376  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    5377  1                             ! STATISTICS TABLES
;    5378  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    5379  1                             ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    5380  1         DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;    5381  1                             !INFO FROM RECEIVE AND SEND CMDS                ZZZ
;    5382  1         TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TRACK DIRECTION ZZZ
;    5383  1         RDM_CNT : WORD,                             !NO OF RANDOM NOS   \\KEEP       ZZZ
;    5384  1         RANDOM : VECTOR [RDM_LEN, WORD],            !RANDOM NO TABLE    //TOGETHER   ZZZ
;    5385  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    5386  1                             ! STATISTICS TABLE FOR CONTROLLER ERRORS
```

```
;    5387  1          MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;    5388  1                                 ! MSCP PACKET POOL
;    5389  1          IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    5390  1                                 ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;    5391  1          PKT_USE : vector [PKT_CNT, byte, signed],
;    5392  1                                 ! MSCP PACKET POOL ALLOCATION TABLE
;    5393  1          RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;    5394  1                                 ! RETURN PACKET POOL
;    5395  1          RP_USE : vector [RP_CNT, byte, signed],
;    5396  1                                 ! RETURN PACKET POOL ALLOCATION TABLE
;    5397  1          RP_INDX : word,        ! CURRENT RETURN PACKET INDEX
;    5398  1          RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
;    5399  1                                 ! CURRENT RETURN PACKET ADDRESS
;    5400  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;    5401  1                                 ! ERROR-LOG PACKET SAVE AREA
;    5402  1          BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
;    5403  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;    5404  1          IODQ : vector [IODQ_LEN, byte],                      ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;    5405  1          IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
;    5406  1          IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
;    5407  1          ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
;    5408  1          EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
;    5409  1          DUP_FLAGS : WORD,                                    !DUP FLAGS       ZZZ
;    5410  1          CCTLR : word,                                        ! NUMBER OF "CURRENT" CONTROLLER
;    5411  1          CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    5412  1          CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
;    5413  1          CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;    5414  1          DUR : vector [MAX_UNITS, byte],                      ! DROP UNIT REASON
;    5415  1          QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;    5416  1          FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
;    5417  1          BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
;    5418  1          ST_CODE : word,                                      ! CURRENT STATUS CODE
;    5419  1          SB_CODE : word,                                      ! CURRENT SUB-CODE
;    5420  1          STEP : word,                                         ! CURRENT STEP IN HARD_INIT
;    5421  1          OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
;    5422  1          SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
;    5423  1          CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
;    5424  1          NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
;    5425  1          CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
;    5426  1          CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
;    5427  1          TEMP1 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;    5428  1          TEMP2 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;    5429  1          CREDIT_BAL : word,                                   ! CREDIT BALANCE
;    5430  1          NEXT_PKT_USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;    5431  1          HOURS: byte,                                         ! TIME OF DAY (HOURS)
;    5432  1          MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
;    5433  1          CLK_TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
;    5434  1          FER0_LBN : word,                                     !LO LBN ADR OF THE "FORCED ERROR" BLOCK            ZZZ
;    5435  1          FER1_LBN : word,                                     !HI LBN ADR OF THE "FORCED ERROR" BLOCK            ZZZ
;    5436  1          CLK_PRESENT : byte,                                  ! FLAG INDICATES IF LINE-CLOCK PRESENT
;    5437  1          HOE_FLAG : byte,                                     ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;    5438  1          FORCED_ERROR : byte,                                 ! "FORCED ERROR" DETECTED IN LAST READ
;    5439  1          FER_LBN : word,                                      ! LBN OF THE "FORCED ERROR" BLOCK
```

```
;    5440  1          FER_BC : word,                    ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    5441  1          INIT_OCCURED : byte,              ! EXERCISER INITIALIZATION COMPLETE
;    5442  1          ADDR_VECT_OK : byte,              ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    5443  1          DBM5,
;    5444  1          P_INDEX : SIGNED WORD,            !CURRENT MESSAGE PACKET INDEX       ZZZ
;    5445  1          S_PATTERN : WORD,                 !PATTERN FOR DUP WRITES             ZZZ
;    5446  1          S_DUPPKT : WORD,                  !DBN BYTE COUNTER                   ZZZ
;    5447  1          RD_COUNT : WORD,                  ! NUMBER OF WINCHESTER UNITS        ZZZ
;    5448  1          BRLEVEL : WORD,                   !BUS REQUEST PRIORITY LEVEL         ZZZ
;    5449  1          D_FAIL : BYTE,                    !SIGNIFIES DUP TYPE ERROR           ZZZ
;    5450  1          DBM107,
;    5451  1          DU_MSG,
;    5452  1          DU_RSN : vector [11],
;    5453  1          RPT1,
;    5454  1          RPT2,
;    5455  1          RPT3,
;    5456  1          RPT4,
;    5457  1          RPT5,
;    5458  1          RPT6,
;    5459  1          RPT7,
;    5460  1          RPT8,
;    5461  1          RPT9,
;    5462  1          RPT10,
;    5463  1          RPT11,
;    5464  1          RPT12,
;    5465  1          RPT13,
;    5466  1          RPT14,
;    5467  1          RPT15,
;    5468  1          RPT16,
;    5469  1      !ZZZ    RPT17,
;    5470  1      !ZZZ    RPT18,
;    5471  1      !ZZZ    RPT19,
;    5472  1
;    5473  1          MSG_01,
;    5474  1          EGS_01,
;    5475  1          EBS_01,
;    5476  1          EBD_10,
;    5477  1          EBD_12,
;    5478  1          EBD_13,
;    5479  1          EBD_14,
;    5480  1          EBD_18,
;    5481  1          EBD_19,
;    5482  1          EBD_24,
;    5483  1          ERR_00,
;    5484  1          ERR_COD : vector [14],
;    5485  1          ELG_00,
;    5486  1          ELG_FMT : vector [5],
;    5487  1          EX_TIM,
;    5488  1          XX13,                                                    !ZZZ
;    5489  1          XX23,                                                    !ZZZ
;    5490  1          XX32,                                                    !ZZZ
;    5491  1          XX33,                                                    !ZZZ
;    5492  1          XX34,                                                    !ZZZ
```

```
;    5493  1          EX_SA,
;    5494  1          EX_SC,
;    5495  1          EX_SBO,
;    5496  1          EX_SB,
;    5497  1          EX_RP,
;    5498  1          EX_WRD,
;    5499  1          EX_CMD,
;    5500  1          EX_RD,
;    5501  1          EX_WRT,
;    5502  1          EX_CMP,
;    5503  1          EX_ONL,
;    5504  1          EX_ACC,
;    5505  1          EX_OP,
;    5506  1          EX_BB,
;    5507  1          EX_BB1,
;    5508  1          EX_BBU,
;    5509  1          EX_LBN,
;    5510  1          EX_PBN,
;    5511  1          EX_LBR,
;    5512  1          EX_LBW,
;    5513  1          EX_RBN,
;    5514  1          EX_CBC,
;    5515  1          EX_CBR,
;    5516  1          EX_CBW,
;    5517  1          EX_BC,
;    5518  1          EX_BD,
;    5519  1          EX_BDR,
;    5520  1          EX_BDW,
;    5521  1          SC_SDI,
;    5522  1          SC_CON,
;    5523  1          SC_DUP,
;    5524  1          SC_ONL,
;    5525  1          SC_SON,
;    5526  1          SC_UNK,
;    5527  1          SC_VOL,
;    5528  1          SC_IOP,
;    5529  1          SC_DIS,
;    5530  1          SC_FER,
;    5531  1          SC_FE2,
;    5532  1          SC_ISH,
;    5533  1          SC_IS2,
;    5534  1          SC_DST,
;    5535  1          SC_DS2,
;    5536  1          SC_ECC,
;    5537  1          SC_ECD,
;    5538  1          SC_RCT,
;    5539  1          SC_FUL,
;    5540  1          SC_576,
;    5541  1          SC_FCT,
;    5542  1          SC_SWP,
;    5543  1          SC_HWP,
;    5544  1          SC_EC1,
;    5545  1          SC_EC2,
```

```
;    5546  1          SC_EC3,
;    5547  1          SC_EC4,
;    5548  1          SC_EC5,
;    5549  1          SC_EC6,
;    5550  1          SC_EC7,
;    5551  1          SC_EC8,
;    5552  1          SC_EC9,
;    5553  1          SC_ODA,
;    5554  1          SC_ODB,
;    5555  1          SC_NXM,
;    5556  1          SC_PAR,
;    5557  1          SC_CTO,
;    5558  1          SC_SDS,
;    5559  1          SC_EDC,
;    5560  1          SC_IDS,
;    5561  1          SC_SRT,
;    5562  1          SC_SRI,
;    5563  1          SC_POE,
;    5564  1          SC_RDY,
;    5565  1          SC_CLK,
;    5566  1          SC_RSP,
;    5567  1          SC_SUR,
;    5568  1          SC_PSP,
;    5569  1          CER_01,
;    5570  1          CER_02,
;    5571  1          CNTR_ERR : vector [23],
;    5572  1          RDRX_ERR : vector [7],
;    5573  1          SPACE4,
;    5574  1          CRLF,
;    5575  1          DASH,
;    5576  1          ASTERISK,
;    5577  1          HWQ1,
;    5578  1          HWQ2,
;    5579  1          HWQ3,
;    5580  1          HWQ4,
;    5581  1          HWQ5,
;    5582  1          HWQ6A,
;    5583  1          HWQ6B,
;    5584  1          HWQ7A,
;    5585  1          HWQ7B,
;    5586  1          HWQ8,
;    5587  1          HWQ9,
;    5588  1          HWQ10,                                    !ZZZ
;    5589  1          HWQ11,                                    !ZZZ
;    5590  1          SWQ1,
;    5591  1          SWQ2,
;    5592  1          SWQ4,
;    5593  1          SWQ7,
;    5594  1          SWQ9,
;    5595  1          SWQ10,
;    5596  1          SWQ11,
;    5597  1          SWQ12,
;    5598  1          SWQ13,
```

```
;    5599  1          SWQ14,
;    5600  1          SWQ15,
;    5601  1          SWQ17,
;    5602  1          SWQ19,
;    5603  1          SWQ20,
;    5604  1          SWQ21,
;    5605  1          SWQ22,
;    5606  1          SWQ23,
;    5607  1          SWQ24,
;    5608  1          SWQ25,
;    5609  1          SWQ26,                                          !ZZZ
;    5610  1          EH_0,                                           !ZZZ
;    5611  1          EH_1,                                           !ZZZ
;    5612  1          EH_2,                                           !ZZZ
;    5613  1          EH_3,                                           !ZZZ
;    5614  1          EH_4,                                           !ZZZ
;    5615  1          EH_5,                                           !ZZZ
;    5616  1          EH_6,                                           !ZZZ
;    5617  1          EH_7,                                           !ZZZ
;    5618  1          EH_8,                                           !ZZZ
;    5619  1          EH_9,                                           !ZZZ
;    5620  1          EH_10,                                          !ZZZ
;    5621  1          EH_12,                                          !ZZZ
;    5622  1          EH_13,                                          !ZZZ
;    5623  1          SWM1,
;    5624  1          NULL,
;    5625  1          SWP_FLAGS : word,
;    5626  1          L$HIMEM,
;    5627  1          L$LUN,
;    5628  1          L$UNIT;
;    5629  1      !   O_BRK;
;    5630  1
;    5631  1      own
;    5632  1          TBL_SUC : vector [17] initial (NULL, SC_SDI, SC_CON, NULL, SC_DUP, NULL, NULL,
;    5633  1                    NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SON),
;    5634  1          TBL_OFL : vector [9] initial (SC_UNK, SC_VOL, SC_IOP, NULL, SC_DUP, NULL, NULL,
;    5635  1                    NULL, SC_DIS),
;    5636  1          TBL_MFE : vector [11] initial (SC_FER, NULL, SC_ISH, SC_DST, SC_EC9, SC_576,
;    5637  1                    SC_FCT, SC_ECC, SC_RCT, SC_FUL, SC_EC1),
;    5638  1          TBL_WPT : vector [3] initial (NULL, SC_SWP, SC_HWP),
;    5639  1          TBL_DAT : vector [16] initial (SC_FE2, NULL, SC_IS2, SC_DS2, SC_EC9, NULL, NULL,
;    5640  1                    SC_ECD, SC_EC1, SC_EC2, SC_EC3, SC_EC4, SC_EC5, SC_EC6, SC_EC7, SC_EC8),
;    5641  1          TBL_HST : vector [5] initial (NULL, SC_ODA, SC_ODB, SC_NXM, SC_PAR),
;    5642  1          TBL_CNT : vector [4] initial (SC_CTO, SC_SDS, SC_EDC, SC_IDS),
;    5643  1          TBL_DRV : vector [9] initial (NULL, SC_SRT, SC_SRI, SC_POE, SC_RDY, SC CLK, SC_RSP,
;    5644  1                    SC SUR, SC_PSP);
;    5645  1
;    5646  1
```

```
ZRQAM2          RD/RX EXERCISER                                 4-Apr-1985 12:40:26    VAX-11 Bliss-16 V4.1-582              SEQ 0142
V02.2           TYPE AND DESCRIPTION                            4 Apr-1985 12:33:21    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL1;16  Page 125
                                                                                                                              (37)

;    5647  1     %sbttl 'TYPE AND DESCRIPTION'
;    5648  1
;    5649  1     EQUALS;
;    5650  1
;    5651  1     DEVTYP (%asciz'RQDX or RUX50');                 ! NAME OF DEVICE SUPPORTED BY PROGRAM
;    5652  1     DESCRIPT (%asciz'RD/RX EXERCISER');     ! TEST DESCRIPTION
```

```
;    5653  1     %sbttl 'HARDWARE PARAMETER CODING SECTION'
;    5654  1
;    5655  1     !+
;    5656  1     ! THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5657  1     ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5658  1     ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5659  1     ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5660  1     ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5661  1     ! WITH THE OPERATOR.
;    5662  1     !-
;    5663  1
;    5664  1     BGNHRD;
;    5665  1
;    5666  1     GPRMA (HWQ1, 0, 0, %o'160000', %o'177777', YES, 1);                  ! IP ADDRESS
;    5667  1     GPRMA (HWQ2, 2, 0, %o'4', %o'774', YES, 1);                          ! VECTOR
;    5668  1     GPRMD (HWQ3, 4, 0, %o'377', %o'0', %o'7', YES, 1);                   ! BR LEVEL
;    5669  1     GPRMD (HWQ4, 6, 0, %o'17', %decimal'0', %decimal'15', YES, 1);       ! RDRX DRIVE NUMBER
;    5670  1     GPRML (HWQ10, 6, %o'000040', YES, 1);                                !ALSO RUN DUP EXERCISER     ZZZ
;    5671  1     XFERF (NODU);                                                        !                           ZZZ
;    5672  1     GPRML (HWQ11, 6, %o'000100', YES, 1);                                !WRITE DIAG AREA            ZZZ
;    5673  1     %L (NODU);                                                           !                           ZZZ
;    5674  1     GPRML (HWQ5, 6, %o'000200', YES, 1);                                 !TEST ENTIRE CUSTOMER AREA? ZZZ
;    5675  1     XFERT (TOQ8);                                                        !BR IF YES                  ZZZ
;    5676  1     GPRMD (HWQ6A, 8, 0, %o'177777', %decimal'0', %o'177777', YES, 1);    ! STARTING LBN LO           ZZZ
;    5677  1     GPRMD (HWQ6B, 10, 0, %o'177777', %decimal'0', %o'177777', YES, 1);   ! STARTING LBN HI           ZZZ
;    5678  1     GPRMD (HWQ7A, 12, 0, %o'177777', GP%ATLO (8), %o'177777', YES, 1);   ! ENDING LBN LO           ZZZ
;    5679  1     GPRMD (HWQ7B, 14, 0, %o'177777', %decimal'0', %o'177777', YES, 1);   ! ENDING LBN HI           ZZZ
;    5680  1     %L (TOQ8);                                                           !                           ZZZ
;    5681  1     GPRML (HWQ8, 6, %o'100000', NO, 0);                                  ! WRITE ON CUST DATA AREA
;    5682  1     XFERF (HWDONE);                                                      ! NO - DONE
;    5683  1     GPRML (HWQ9, 6, %o'100000', NO, 1);                                  ! ** WARNING / CONFIRM
;    5684  1     %L (HWDONE);
;    5685  1
;    5686  1     ENDHRD;
```

```
;    5687  1     %sbttl 'SOFTWARE PARAMETER CODING SECTION'
;    5688  1
;    5689  1     !+
;    5690  1     ! THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5691  1     ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5692  1     ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5693  1     ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5694  1     ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5695  1     ! WITH THE OPERATOR.
;    5696  1     !-
;    5697  1
;    5698  1     BGNSFT;
;    5699  1
;    5700  1     !GPRML (SWQ16, 4, SWF_TRC, YES, 1);                    ! ENABLE DIAGNOSTIC TRACE
;    5701  1     GPRMD (SWQ24, 10, D, %o'177777', 0, 2359, YES, 1);     ! START TIME
;    5702  1     GPRMD (SWQ1, 0, D, %o'177777', 0, 65535, YES, 1);      ! ERROR LIMIT
;    5703  1     GPRMD (SWQ2, 2, D, %o'177777', 0, 99, YES, 1);         ! TRANSFER LIMIT
;    5704  1     GPRMD (SWQ17, 8, D, %o'177777', 0, 100, YES, 1);       ! PERCENT OF RD OPERATIONS
;    5705  1     GPRML (SWQ15, 4, SWF_CST, YES, 1);                     ! CLEAR STATISTICAL TABLES ?
;    5706  1     GPRML (SWQ20, 4, SWF_FER, YES, 1);                     ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT SET?
;    5707  1     GPRML (SWQ23, 4, SWF_BLK, YES, 1);                     ! HALT ON BAD-BLOCK TYPE ERRORS WITH 'HOE' FLAG?
;    5708  1     GPRML (SWQ21, 4, SWF_HRD, YES, 1);                     ! HALT ON HARD ERRORS WITH 'HOE' FLAG SET?
;    5709  1     GPRML (SWQ22, 4, SWF_SFT, YES, 1);                     ! HALT ON SOFT ERRORS WITH 'HOE' FLAG SET?
;    5710  1     GPRML (SWQ25, 4, SWF_TRY, YES, 1);                     ! COUNT EACH RETRY AS ANOTHER SOFT-ERROR?
;    5711  1     GPRML (SWQ4, 4, SWF_RDM, YES, 1);                      ! RANDOM SEEK MODE ?
;    5712  1     XFERF (SW1);                                           ! IF NO, DO NEXT QUESTION
;    5713  1     XFER (SW2);                                            !
;    5714  1     $L (SW1);                                              !
;    5715  1     GPRML (SWQ19, 4, SWF_SEQ, YES, 1);                     ! RANDOM OR SEQUENTIAL SELECTION OF DRIVES
;    5716  1     $L (SW2);                                              !
;    5717  1     GPRML (SWQ7, 4, SWF_CRC, YES, 1);                      ! READ-COMPARES AT CONTROLLER ?
;    5718  1     GPRML (SWQ26, 4, SWF_APT, YES, 1);                     !RUNNING UNDER A.P.T. MONITOR?          ZZZ
;    5719  1     DISPLAY (SWM1);                                        ! REMAINING QUESTIONS ONLY APPLY ...
;    5720  1     GPRML (SWQ9, 4, SWF_CWC, YES, 1);                      ! WRITE-COMPARES AT CONTROLLER ?
;    5721  1     XFERF (SW3);                                           ! IF NO, DO NEXT QUESTION
;    5722  1     XFER (SW4);                                            !
;    5723  1     $L (SW3);                                              !
;    5724  1     GPRML (SWQ10, 4, SWF_HWC, YES, 1);                     ! CHECK WRITES AT HOST BY READING ?
;    5725  1     $L (SW4);                                              !
;    5726  1     GPRML (SWQ11, 4, SWF_UDP, YES, 1);                     ! USER-DEFINED DATA PATTERN ?
;    5727  1     XFERF (SW5);                                           ! IF NO, DO NEXT QUESTION
;    5728  1     XFER (SW6);                                            !
;    5729  1     $L (SW5);                                              !
;    5730  1     GPRMD (SWQ12, 6, D, %o'177777', 0, DP_CNT, YES, 1);    ! SELECT PRE-DEFINED DATA PATTERN
;    5731  1     XFER (SW7);                                            ! DONE
;    5732  1     $L (SW6);                                              !
;    5733  1     GPRMD (SWQ13, 12, D, %o'177777', 1, MAX_UDP_CNT, YES, 1);  ! NO. OF WORDS IN USER DATA PATTERN
;    5734  1     GPRMD (SWQ14, 1 , 0, %o'177777', 0, %o'177777', NO, 12);   ! PATTERN VALUES
;    5735  1     $L (SW7);
;    5736  1     ENDSFT;
```

```
;    5737  1
;    5738  1
;    5739  1     %sbttl 'REPORT CODING SECTION'
;    5740  1
;    5741  1
;    5742  1     !+
;    5743  1     ! THE REPORT CODING SECTION CONTAINS THE
;    5744  1     ! "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;    5745  1     !-
;    5746  1
;    5747  1
;    5748  2     BGNRPT;
;    5749  2
;    5750  2     local
;    5751  2         CUR_PRIORITY : word;
;    5752  2
;    5753  2     GETPRI (CUR_PRIORITY);
;    5754  2     !ZZZ SETPRI (PRI04);                                    !ZZZ
;    5755  2     SETPRI (.BRLEVEL);                          !ZZZ
;    5756  2
;    5757  2     PRINTS (RPT1);
;    5758  2     PRINTS (RPT2);
;    5759  2     PRINTS (RPT3);
;    5760  2     PRINTS (RPT4);
;    5761  2     PRINTS (RPT5);
;    5762  2     PRINTS (RPT6);
;    5763  2
;    5764  2     incr CTLR from 0 to MAX_CTLR - 1 do
;    5765  2
;    5766  3         begin
;    5767  3         SET_CPAR (.CTLR);
;    5768  3
;    5769  3         incr DISK from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do
;    5770  3
;    5771  4             begin
;    5772  4             SET_UPAR (.DISK);
;    5773  4
;    5774  4
;    5775  4             if .CST_ADDR [.DISK * OF_DATA, D_PRES] eql PRESENT
;    5776  4             then
;    5777  4
;    5778  5                 begin
; P  5779  5                 PRINTS (RPT7,
;    5780  5                     .L$LUN, .CST_ADDR [.DISK * OF_DATA, D_DISK_NUM], CST [.CTLR, .DISK * OF_NAME_0, D_NAME_0]);
; P  5781  5                 PRINTS (RPT8,
; P  5782  5                     .T_ADDR [TOT_READS_HI], .T_ADDR [TOT_READS_LO],
;    5783  5                     .T_ADDR [MTOT_BYT_RED], .T_ADDR [TOT_BYT_RED_HI], .T_ADDR [TOT_BYT_RED_LO]);
; P  5784  5                 PRINTS (RPT8,
; P  5785  5                     .T_ADDR [TOT_WRITES_HI], .T_ADDR [TOT_WRITES_LO],
;    5786  5                     .T_ADDR [MTOT_BYT_WRT], .T_ADDR [TOT_BYT_WRT_HI], .T_ADDR [TOT_BYT_WRT_LO]);
; P  5787  5                 PRINTS (RPT9,
; P  5788  5                     .T_ADDR [ERR_HRD_SEK], .T_ADDR [ERR_HRD_DAT], .T_ADDR [ERR_HRD_DRV], .T_ADDR [ERR_HRD_HST],
;    5789  5                     .T_ADDR [ERR_SFT_SEK], .T_ADDR [ERR_SFT_DAT], .T_ADDR [ERR_SFT_DRV], .T_ADDR [ERR_SFT_HST]);
```

```
;    5790  4                  end;
;    5791  3              end;
;    5792  3
;    5793  3
;    5794  3          if .CST [.CTLR, STATE] eql PRESENT
;    5795  3          then
;    5796  3
;    5797  4              begin
;    5798  4              PRINTS (RPT10);
;    5799  4              PRINTS (RPT11, .C_ERR_TBL [.CTLR, C_ERR_HRD], .C_ERR_TBL [.CTLR, C_ERR_SFT]);
;    5800  3              end;
;    5801  3
;    5802  3
;    5803  2          end;
;    5804  2
;    5805  2      SETPRI (.CUR_PRIORITY);
;    5806  2
;    5807  2      IF .RD_COUNT NEQ 0                                          !IF THERE IS A WINCHESTER     ZZZ
;    5808  2      THEN                                                        !THEN OUTPUT EXTRA LINES      ZZZ
;    5809  2
;    5810  3      begin                                                       ! PRINTS DUP DATA
;    5811  3      prints(crlf);
;    5812  3      PRINTS(RPT13);
;    5813  3      PRINTS(RPT14);
;    5814  3      PRINTS(RPT15);
;    5815  3      INCR CTLR FROM 0 TO MAX_CTLR-1 DO
;    5816  4              BEGIN
;    5817  4              SET_CPAR(.CTLR);
;    5818  4              INCR DISK FROM (0+OF_UN) TO (3+UNIT_SIZE+OF_UN) BY UNIT_SIZE DO
;    5819  5                      BEGIN
;    5820  5                      SET_UPAR(.DISK);
;    5821  5                      IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_51 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;    5822  5                      THEN
; P  5823  5                  PRINTS (RPT16,
; P  5824  5                      .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;    5825  5                      .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;    5826  5
;    5827  5      !ZZZ            IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_52 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;    5828  5      !ZZZ            THEN
;    5829  5      !ZZZ        PRINTS (RPT18,
;    5830  5      !ZZZ            .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;    5831  5      !ZZZ            .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;    5832  4                      END;
;    5833  3                END;
;    5834  3                   PRINTS (CRLF);
;    5835  2      END;
;    5836  2
;    5837  1      ENDRPT;


                                        .TITLE  ZRQAM2 RD/RX EXERCISER
                                        .IDENT  /V02.2/
                                        .ENABL  AMA
```

```
000000                                          .PSECT  $CODE$,  RO
000000      122     121     104         L$DVTYP::
                                        .ASCII  /RQD/
000003      130     040     157                 .ASCII  /X o/
000006      162     040     122                 .ASCII  /r R/
000011      125     130     065                 .ASCII  /UX5/
000014      060     000                         .ASCII  /0/<00>
000016                                          .BLKB   2
000020      122     104     057         L$DESC::.ASCII  /RD/<57>
000023      122     130     040                 .ASCII  /RX /
000026      105     130     105                 .ASCII  /EXE/
000031      122     103     111                 .ASCII  /RCI/
000034      123     105     122                 .ASCII  /SER/
000037      000                                 .ASCII  <00>
000040                                          .BLKB   2
000042  000000C                         L$HRDLN::
                                        .WORD   <<<L$NDHRD-L$HRDLN>/2>-1>
000044  000031                          GP$1::  .WORD   31
000046  000000G                                 .WORD   HWQ1
000050  160000                                  .WORD   -20000
000052  177777                                  .WORD   -1
000054  001031                          GP$2.:  .WORD   1031
000056  000000G                                 .WORD   HWQ2
000060  000004                                  .WORD   4
000062  000774                                  .WORD   774
000064  002032                          GP$3::  .WORD   2032
000066  000000G                                 .WORD   HWQ3
000070  000377                                  .WORD   377
000072  000000                                  .WORD   0
000074  000007                                  .WORD   7
000076  003052                          GP$4::  .WORD   3052
000100  000000G                                 .WORD   HWQ4
000102  000017                                  .WORD   17
000104  000000                                  .WORD   0
000106  000017                                  .WORD   17
000110  003130                          GP$5::  .WORD   3130
000112  000000G                                 .WORD   HWQ10
000114  000040                                  .WORD   40
000116  000000C                          $NODU:  .WORD   <<<<$LNODU-$NODU>*400>*4>*40>
000120  003130                          GP$6::  .WORD   3130
000122  000000G                                 .WORD   HWQ11
000124  000100                                  .WORD   100
000126  001004                          $LNODU: .WORD   1004
000130  003130                          GP$7::  .WORD   3130
000132  000000G                                 .WORD   HWQ5
000134  000200                                  .WORD   200
000136  000000C                          $TOQ8:  .WORD   <<<<$LTOQ8-$TOQ8>*400>*4>*20>
000140  004032                          GP$8::  .WORD   4032
000142  000000G                                 .WORD   HWQ6A
000144  177777                                 .WORD   -1
000146  000000                                 .WORD   0
```

```
000150  177777                          .WORD   -1
000152  005032          GP$9::          .WORD   5032
000154  000000G                        .WORD   HWQ6B
000156  177777                        .WORD   -1
000160  000000                        .WORD   0
000162  177777                        .WORD   -1
000164  006432          GP$10::         .WORD   6432
000166  000000G                        .WORD   HWQ7A
000170  177777                        .WORD   -1
000172  000004                        .WORD   4
000174  177777                        .WORD   -1
000176  000001                        .WORD   1
000200  007032          GP$11::         .WORD   7032
000202  000000G                        .WORD   HWQ7B
000204  177777                        .WORD   -1
000206  000000                        .WORD   0
000210  177777                        .WORD   -1
000212  001004          $LTOQ8:         .WORD   1004
000214  003120          GP$12::         .WORD   3120
000216  000000G                        .WORD   HWQ8
000220  100000                        .WORD   -100000
000222  000000C         $HWDONE:        .WORD   <<<<$LHWDONE-$HWDONE>*400>*4>*40>
000224  003120          GP$13::         .WORD   3120
000226  000000G                        .WORD   HWQ9
000230  100000                        .WORD   -100000
000232  001004          $LHWDONE:
                                        .WORD   1004
000234                  L$NDHRD::
                                        .BLKW   1
000236  000000C         L$SFTLN::
                                        .WORD   <<<L$NDSFT-L$SFTLN>/2>-1>
000240  005052          GP$14::         .WORD   5052
000242  000000G                        .WORD   SWQ24
000244  177777                        .WORD   -1
000246  000000                        .WORD   0
000250  004467                        .WORD   4467
000252  000052          GP$15::         .WORD   52
000254  000000G                        .WORD   SWQ1
000256  177777                        .WORD   -1
000260  000000                        .WORD   0
000262  177777                        .WORD   -1
000264  001052          GP$16::         .WORD   1052
000266  000000G                        .WORD   SWQ2
000270  177777                        .WORD   -1
000272  000000                        .WORD   0
000274  000143                        .WORD   143
000276  004052          GP$17::         .WORD   4052
000300  000000G                        .WORD   SWQ17
000302  177777                        .WORD   -1
000304  000000                        .WORD   0
000306  000144                        .WORD   144
000310  002130          GP$18::         .WORD   2130
000312  000000G                        .WORD   SWQ15
```

```
000314  000200                          .WORD   200
000316  002130          GP$19::         .WORD   2130
000320  000000G                         .WORD   SWQ20
000322  004000                          .WORD   4000
000324  002130          GP$20::         .WORD   2130
000326  000000G                         .WORD   SWQ23
000330  040000                          .WORD   40000
000332  002130          GP$21::         .WORD   2130
000334  000000G                         .WORD   SWQ21
000336  010000                          .WORD   10000
000340  002130          GP$22::         .WORD   2130
000342  000000G                         .WORD   SWQ22
000344  020000                          .WORD   20000
000346  002130          GP$23::         .WORD   2130
000350  000000G                         .WORD   SWQ25
000352  100000                          .WORD   -100000
000354  002130          GP$24::         .WORD   2130
000356  000000G                         .WORD   SWQ4
000360  000002                          .WORD   2
000362  000000C          $SW1:          .WORD   <<<<$LSW1-$SW1>*400>.4>.40>
000364  000000C          $SW2:          .WORD   <<<$LSW2-$SW2>*400>.4>
000366  001004           $LSW1:         .WORD   1004
000370  002130          GP$25::         .WORD   2130
000372  000000G                         .WORD   SWQ19
000374  001000                          .WORD   1000
000376  001004           $LSW2:         .WORD   1004
000400  002130          GP$26::         .WORD   2130
000402  000000G                         .WORD   SWQ7
000404  000004                          .WORD   4
000406  002130          GP$27::         .WORD   2130
000410  000000G                         .WORD   SWQ26
000412  000001                          .WORD   1
000414  000003          GP$DISP::
                                        .WORD   3
000416  000000G                         .WORD   SWM1
000420  002130          GP$28::         .WORD   2130
000422  000000G                         .WORD   SWQ9
000424  000020                          .WORD   20
000426  000000C          $SW3:          .WORD   <<<<$LSW3-$SW3>*400>.4>.40>
000430  000000C          $SW4:          .WORD   <<<$LSW4-$SW4>*400>.4>
000432  001004           $LSW3:         .WORD   1004
000434  002130          GP$29::         .WORD   2130
000436  000000G                         .WORD   SWQ10
000440  000040                          .WORD   40
000442  001004           $LSW4:         .WORD   1004
000444  002130          GP$30::         .WORD   2130
000446  000000G                         .WORD   SWQ11
000450  000100                          .WORD   100
000452  000000C          $SW5:          .WORD   <<<<$LSW5-$SW5>*400>.4>.40>
000454  000000C          $SW6:          .WORD   <<<$LSW6-$SW6>*400>.4>
000456  001004           $LSW5:         .WORD   1004
000460  003052          GP$31::         .WORD   3052
000462  000000G                         .WORD   SWQ12
```

```
000464  177777                          .WORD   -1
000466  000000                          .WORD   0
000470  000025                          .WORD   25
000472  000000C                 $SW7:   .WORD   <<<$LSW7-$SW7>*400>.4>
000474  001004                  $LSW6:  .WORD   1004
000476  006052                  GP$32:: .WORD   6052
000500  000000G                         .WORD   SWQ13
000502  177777                          .WORD   -1
000504  000001                          .WORD   1
000506  000020                          .WORD   20
000510  007222                  GP$33:: .WORD   7222
000512  000000G                         .WORD   SWQ14
000514  177777                          .WORD   -1
000516  000000                          .WORD   0
000520  177777                          .WORD   -1
000522  000006                          .WORD   6
000524  001004                  $LSW7:  .WORD   1004
000526                          L$NDSFT::
                                        .BLKW   1


000000                                  .PSECT  $OWN$,  D
000000  000000G                 TBL.SUC:.WORD   NULL
000002  000000G                         .WORD   SC.SOI
000004  000000G                         .WORD   SC.CON
000006  000000G                         .WORD   NULL
000010  000000G                         .WORD   SC.DUP
000012  000000G                         .WORD   NULL
000014  000000G                         .WORD   NULL
000016  000000G                         .WORD   NULL
000020  000000G                         .WORD   SC.ONL
000022  000000G                         .WORD   NULL
000024  000000G                         .WORD   NULL
000026  000000G                         .WORD   NULL
000030  000000G                         .WORD   NULL
000032  000000G                         .WORD   NULL
000034  000000G                         .WORD   NULL
000036  000000G                         .WORD   NULL
000040  000000G                         .WORD   SC.SON
000042  000000G                 TBL.OFL:.WORD   SC.UNK
000044  000000G                         .WORD   SC.VOL
000046  000000G                         .WORD   SC.IOP
000050  000000G                         .WORD   NULL
000052  000000G                         .WORD   SC.DUP
000054  000000G                         .WORD   NULL
000056  000000G                         .WORD   NULL
000060  000000G                         .WORD   NULL
000062  000000G                         .WORD   SC.DIS
000064  000000G                 TBL.MFE:.WORD   SC.FER
000066  000000G                         .WORD   NULL
000070  000000G                         .WORD   SC.ISH
000072  000000G                         .WORD   SC.DST
```

```
000074  000000G                         .WORD   SC.EC9
000076  000000G                         .WORD   SC.576
000100  000000G                         .WORD   SC.FCT
000102  000000G                         .WORD   SC.ECC
000104  000000G                         .WORD   SC.RCT
000106  000000G                         .WORD   SC.FUL
000110  000000G                         .WORD   SC.EC1
000112  000000G                 TBL.WPT:.WORD   NULL
000114  000000G                         .WORD   SC.SWP
000116  000000G                         .WORD   SC.HWP
000120  000000G                 TBL.DAT:.WORD   SC.FE2
000122  000000G                         .WORD   NULL
000124  000000G                         .WORD   SC.IS2
000126  000000G                         .WORD   SC.DS2
000130  000000G                         .WORD   SC.EC9
000132  000000G                         .WORD   NULL
000134  000000G                         .WORD   NULL
000136  000000G                         .WORD   SC.ECD
000140  000000G                         .WORD   SC.EC1
000142  000000G                         .WORD   SC.EC2
000144  000000G                         .WORD   SC.EC3
000146  000000G                         .WORD   SC.EC4
000150  000000G                         .WORD   SC.EC5
000152  000000G                         .WORD   SC.EC6
000154  000000G                         .WORD   SC.EC7
000156  000000G                         .WORD   SC.EC8
000160  000000G                 TBL.HST:.WORD   NULL
000162  000000G                         .WORD   SC.ODA
000164  000000G                         .WORD   SC.OD8
000166  000000G                         .WORD   SC.NXM
000170  000000G                         .WORD   SC.PAR
000172  000000G                 TBL.CNT:.WORD   SC.CTO
000174  000000G                         .WORD   SC.SDS
000176  000000G                         .WORD   SC.EDC
000200  000000G                         .WORD   SC.IDS
000202  000000G                 TBL.DRV:.WORD   NULL
000204  000000G                         .WORD   SC.SRT
000206  000000G                         .WORD   SC.SRI
000210  000000G                         .WORD   SC.POE
000212  000000G                         .WORD   SC.RDY
000214  000000G                         .WORD   SC.CLK
000216  000000G                         .WORD   SC.RSP
000220  000000G                         .WORD   SC.SUR
000222  000000G                         .WORD   SC.PSP


                                        .GLOBL  CST, CST.ADDR, DCT, DCT.ADDR, RDRX.ADDR
                                        .GLOBL  IRDRX.ADDR, BST, TALLY, T.ADDR
                                        .GLOBL  DUPPKT, TRK.SGN, RDM.CNT, RANDOM
                                        .GLOBL  C.ERR.TBL, MSCP.PKT, IPKT.ADDR
                                        .GLOBL  PKT.USE, RETPKT, RP.USE, RP.INDX
                                        .GLOBL  RP.ADDR, ELOG.PKT, BUFF.ADDR, BUFF.OWN
                                        .GLOBL  IODQ, IODQ.IN, IODQ.OUT, ENTRY.REASON
```

```
.GLOBL  EOP.FLAG, DUP.FLAGS, CCTLR, CDISK
.GLOBL  CUOFF, CTLR.CNT, DUR, QIO, FREE.MEM.ADDR
.GLOBL  BYTS.PER.QIO, ST.CODE, SB.CODE
.GLOBL  STEP, OF.RC, SA.REG, CMD.TIME
.GLOBL  NEX, CRN.LOW, CRN.HIGH, TEMP1
.GLOBL  TEMP2, CREDIT.BAL, NEXT.PKT.USE
.GLOBL  HOURS, MINUTES, CLK.TICKS, FERO.LBN
.GLOBL  FER1.LBN, CLK.PRESENT, HOE.FLAG
.GLOBL  FORCED.ERROR, FER.LBN, FER.BC
.GLOBL  INIT.OCCURED, ADDR.VECT.OK, DBM5
.GLOBL  P.INDEX, S.PATTERN, S.DUPPKT, RD.COUNT
.GLOBL  BRLEVEL, D.FAIL, DBM107, DU.MSG
.GLOBL  DU.RSN, RPT1, RPT2, RPT3, RPT4
.GLOBL  RPT5, RPT6, RPT7, RPT8, RPT9, RPT10
.GLOBL  RPT11, RPT12, RPT13, RPT14, RPT15
.GLOBL  RPT16, MSG.01, EGS.01, EBS.01
.GLOBL  EBD.10, EBD.12, EBD.13, EBD.14
.GLOBL  EBD.18, EBD.19, EBD.24, ERR.00
.GLOBL  ERR.COD, ELG.00, ELG.FMT, EX.TIM
.GLOBL  XX13, XX23, XX32, XX33, XX34, EX.SA
.GLOBL  EX.SC, EX.SBO, EX.SB, EX.RP, EX.WRD
.GLOBL  EX.CMD, EX.RD, EX.WRT, EX.CMP
.GLOBL  EX.ONL, EX.ACC, EX.OP, EX.BB, EX.BB1
.GLOBL  EX.BBU, EX.LBN, EX.PBN, EX.LBR
.GLOBL  EX.LBW, EX.RBN, EX.CBC, EX.CBR
.GLOBL  EX.CBW, EX.BC, EX.BD, EX.BDR, EX.BDW
.GLOBL  SC.SDI, SC.CON, SC.DUP, SC.ONL
.GLOBL  SC.SON, SC.UNK, SC.VOL, SC.IOP
.GLOBL  SC.DIS, SC.FER, SC.FE2, SC.ISH
.GLOBL  SC.IS2, SC.DST, SC.DS2, SC.ECC
.GLOBL  SC.ECD, SC.RCT, SC.FUL, SC.576
.GLOBL  SC.FCT, SC.SWP, SC.HWP, SC.EC1
.GLOBL  SC.EC2, SC.EC3, SC.EC4, SC.EC5
.GLOBL  SC.EC6, SC.EC7, SC.EC8, SC.EC9
.GLOBL  SC.ODA, SC.ODB, SC.NXM, SC.PAR
.GLOBL  SC.CTO, SC.SDS, SC.EDC, SC.IDS
.GLOBL  SC.SRT, SC.SRI, SC.POE, SC.RDY
.GLOBL  SC.CLK, SC.RSP, SC.SUR, SC.PSP
.GLOBL  CER.01, CER.02, CNTR.ERR, RDRX.ERR
.GLOBL  SPACE4, CRLF, DASH, ASTERISK, HWQ1
.GLOBL  HWQ2, HWQ3, HWQ4, HWQ5, HWQ6A
.GLOBL  HWQ6B, HWQ7A, HWQ7B, HWQ8, HWQ9
.GLOBL  HWQ10, HWQ11, SWQ1, SWQ2, SWQ4
.GLOBL  SWQ7, SWQ9, SWQ10, SWQ11, SWQ12
.GLOBL  SWQ13, SWQ14, SWQ15, SWQ17, SWQ19
.GLOBL  SWQ20, SWQ21, SWQ22, SWQ23, SWQ24
.GLOBL  SWQ25, SWQ26, EH.0, EH.1, EH.2
.GLOBL  EH.3, EH.4, EH.5, EH.6, EH.7, EH.8
.GLOBL  EH.9, EH.10, EH.12, EH.13, SWM1
.GLOBL  NULL, SWP.FLAGS, L$HIMEM, L$LUN
.GLOBL  L$UNIT
```

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

```
        040000                          LOE**           40000
        100000                          HOE**           -100000
        000044'                         L$HARD**        L$HRDLN*2
        000240'                         L$SOFT**        L$SFTLN*2


000530                                  .SBTTL  LRPT REPORT CODING SECTION
                                        .PSECT  $CODE$,  RO

000000  004137  000000G       LRPT:     JSR     R1,$SAVE4                       ;                       5736
000004  104440                          TRAP    40                              ;                       5753
000006  010004                          MOV     R0,R4                           ; *,CUR.PRIORITY
000010  013700  000000G                 MOV     BRLEVEL,R0                      ;                       5755
000014  104441                          TRAP    41
000016  012746  000000G                 MOV     #RPT1,-(SP)                     ;                       5757
000022  012746  000001                  MOV     #1,-(SP)
000026  010600                          MOV     SP,R0                           ; SP,*
000030  104416                          TRAP    16
000032  012716  000000G                 MOV     #RPT2,(SP)                      ;                       5758
000036  012746  000001                  MOV     #1,-(SP)
000042  010600                          MOV     SP,R0                           ; SP,*
000044  104416                          TRAP    16
000046  012716  000000G                 MOV     #RPT3,(SP)                      ;                       5759
000052  012746  000001                  MOV     #1,-(SP)
000056  010600                          MOV     SP,R0                           ; SP,*
000060  104416                          TRAP    16
000062  012716  000000G                 MOV     #RPT4,(SP)                      ;                       5760
000066  012746  000001                  MOV     #1,-(SP)
000072  010600                          MOV     SP,R0                           ; SP,*
000074  104416                          TRAP    16
000076  012716  000000G                 MOV     #RPT5,(SP)                      ;                       5761
000102  012746  000001                  MOV     #1,-(SP)
000106  010600                          MOV     SP,R0                           ; SP,*
000110  104416                          TRAP    16
000112  012716  000000G                 MOV     #RPT6,(SP)                      ;                       5762
000116  012746  000001                  MOV     #1,-(SP)
000122  010600                          MOV     SP,R0                           ; SP,*
000124  104416                          TRAP    16
000126  005002                          CLR     R2                              ; CTLR                  5764
000130  010216          1$:             MOV     R2,(SP)                         ; CTLR,*                5767
000132  004737  000000V                 JSR     PC,SET.CPAR
000136  012703  000003                  MOV     #3,R3                           ; *,DISK                5769
000142  010316          2$:             MOV     R3,(SP)                         ; DISK,*                5772
000144  004737  000000V                 JSR     PC,SET.UPAR
000150  010301                          MOV     R3,R1                           ; DISK,*                5775
000152  006301                          ASL     R1
000154  063701  000000G                 ADD     CST.ADDR,R1
000160  032711  040000                  BIT     #40000,(R1)
000164  001535                          BEQ     3$
000166  010216                          MOV     R2,(SP)                         ; CTLR,*                5780
000170  012746  000053                  MOV     #53,-(SP)
000174  004737  000000G                 JSR     PC,BL$MUL
000200  060300                          ADD     R3,R0                           ; DISK,*
```

```
000202  006300                  ASL     R0
000204  062700  000000G         ADD     #CST,R0
000210  010016                  MOV     R0,(SP)
000212  062716  000012          ADD     #12,(SP)
000216  111146                  MOVB    (R1),-(SP)
000220  042716  177760          BIC     #177760,(SP)
000224  013746  000000G         MOV     L$LUN,-(SP)
000230  012746  000000G         MOV     #RPT7,-(SP)
000234  012746  000004          MOV     #4,-(SP)
000240  010600                  MOV     SP,R0           ; SP,*
000242  104416                  TRAP    16
000244  013700  000000G         MOV     T.ADDR,R0       ;                       5783
000250  016016  000032          MOV     32(R0),(SP)
000254  016046  000034          MOV     34(R0),-(SP)
000260  016046  000036          MOV     36(R0),-(SP)
000264  016046  000016          MOV     16(R0),-(SP)
000270  016046  000020          MOV     20(R0),-(SP)
000274  012746  000000G         MOV     #RPT8,-(SP)
000300  012746  000006          MOV     #6,-(SP)
000304  010600                  MOV     SP,R0           ; SP,*
000306  104416                  TRAP    16
000310  013700  000000G         MOV     T.ADDR,R0       ;                       5786
000314  016016  000040          MOV     40(R0),(SP)
000320  016046  000042          MOV     42(R0),-(SP)
000324  016046  000044          MOV     44(R0),-(SP)
000330  016046  000024          MOV     24(R0),-(SP)
000334  016046  000026          MOV     26(R0),-(SP)
000340  012746  000000G         MOV     #RPT8,-(SP)
000344  012746  000006          MOV     #6,-(SP)
000350  010600                  MOV     SP,R0           ; SP,*
000352  104416                  TRAP    16
000354  013700  000000G         MOV     T.ADDR,R0       ;                       5789
000360  005016                  CLR     (SP)
000362  116016  000055          MOVB    55(R0),(SP)
000366  005046                  CLR     -(SP)
000370  116016  000054          MOVB    54(R0),(SP)
000374  005046                  CLR     -(SP)
000376  116016  000053          MOVB    53(R0),(SP)
000402  005046                  CLR     -(SP)
000404  116016  000052          MOVB    52(R0),(SP)
000410  005046                  CLR     -(SP)
000412  116016  000051          MOVB    51(R0),(SP)
000416  005046                  CLR     -(SP)
000420  116016  000050          MOVB    50(R0),(SP)
000424  005046                  CLR     -(SP)
000426  116016  000047          MOVB    47(R0),(SP)
000432  005046                  CLR     -(SP)
000434  116016  000046          MOVB    46(R0),(SP)
000440  012746  000000G         MOV     #RPT9,-(SP)
000444  012746  000011          MOV     #11,-(SP)
000450  010600                  MOV     SP,R0           ; SP,*
000452  104416                  TRAP    16
000454  062706  000064          ADD     #64,SP          ;                       5778
```

```
000460  062703  000012          3$:     ADD     #12,R3                  ; *,DISK                5769
000464  020327  000041                  CMP     R3,#41                  ; DISK,*
000470  003624                          BLE     2$
000472  010216                          MOV     R2,(SP)                 ; CTLR,*                5794
000474  012746  000126                  MOV     #126,-(SP)
000500  004737  000000G                 JSR     PC,BL$MAL
000504  005726                          TST     (SP)+
000506  005760  000002G                 TST     CST+2(R0)
000512  100026                          BPL     4$
000514  012716  000000G                 MOV     #RPT10,(SP)             ;                       5798
000520  012746  000001                  MOV     #1,-(SP)
000524  010600                          MOV     SP,R0                   ; SP,*
000526  104416                          TRAP    16
000530  010200                          MOV     R2,R0                   ; CTLR,*                5799
000532  006300                          ASL     R0
000534  005016                          CLR     (SP)
000536  116016  000001G                 MOVB    C.ERR.TBL+1(R0),(SP)
000542  005046                          CLR     -(SP)
000544  116016  000000G                 MOVB    C.ERR.TBL(R0),(SP)
000550  012746  000000G                 MOV     #RPT11,-(SP)
000554  012746  000003                  MOV     #3,-(SP)
000560  010600                          MOV     SP,R0                   ; SP,*
000562  104416                          TRAP    16
000564  062706  000010                  ADD     #10,SP                  ;                       5797
000570  005202              4$:         INC     R2                      ; CTLR                  5764
000572  000243                          .WORD CLV!CLC

000574  003002                          BGT     5$
000576  000137  000660'                JMP     1$
000602  010400              5$:         MOV     R4,R0                   ; CUR.PRIORITY,*        5805
000604  104441                          TRAP    41
000606  005737  000000G                 TST     RD.COUNT                ;                       5807
000612  001522                          BEQ     9$
000614  012716  000000G                 MOV     #CRLF,(SP)              ;                       5811
000620  012746  000001                  MOV     #1,-(SP)
000624  010600                          MOV     SP,R0                   ; SP,*
000626  104416                          TRAP    16
000630  012716  000000G                 MOV     #RPT13,(SP)             ;                       5812
000634  012746  000001                  MOV     #1,-(SP)
000640  010600                          MOV     SP,R0                   ; SP,*
000642  104416                          TRAP    16
000644  012716  000000G                 MOV     #RPT14,(SP)             ;                       5813
000650  012746  000001                  MOV     #1,-(SP)
000654  010600                          MOV     SP,R0                   ; SP,*
000656  104416                          TRAP    16
000660  012716  000000G                 MOV     #RPT15,(SP)             ;                       5814
000664  012746  000001                  MOV     #1,-(SP)
000670  010600                          MOV     SP,R0                   ; SP,*
000672  104416                          TRAP    16
000674  005003                          CLR     R3                      ; CTLR                  5815
000676  010316              6$:         MOV     R3,(SP)                 ; CTLR,*                5817
000700  004737  000000V                 JSR     PC,SET.CPAR
000704  012702  000003                  MOV     #3,R2                   ; *,DISK                5818
```

```
000710  010216                          7⌀:     MOV     R2,(SP)                 ; DISK,*                5820
000712  004737  000000V                        JSR     PC,SET.UPAR
000716  010201                                  MOV     R2,R1                   ; DISK,*                5821
000720  006301                                  ASL     R1
000722  063701  000000G                        ADD     CST.ADDR,R1
000726  132711  000020                        BITB    #20,(R1)
000732  001432                                BEQ     8⌀
000734  032711  040000                        BIT     #40000,(R1)
000740  001427                                BEQ     8⌀
000742  013700  000000G                        MOV     T.ADDR,R0               ;                       5825
000746  016016  000056                        MOV     56(R0),(SP)
000752  016046  000060                        MOV     60(R0),-(SP)
000756  016046  000062                        MOV     62(R0),-(SP)
000762  016046  000064                        MOV     64(R0),-(SP)
000766  111146                              MOVB    (R1),-(SP)
000770  042716  177760                        BIC     #177760,(SP)
000774  013746  000000G                        MOV     L⌀LUN,-(SP)
001000  012746  000000G                        MOV     #RPT16,-(SP)
001004  012746  000007                        MOV     #7,-(SP)
001010  010600                                MOV     SP,R0                   ; SP,*
001012  104416                                TRAP    16
001014  062706  000016                        ADD     #16,SP
001020  062702  000012                  8⌀:     ADD     #12,R2                  ; *,DISK                5818
001024  020227  000041                        CMP     R2,#41                  ; DISK,*
001030  003727                                BLE     7⌀
001032  005203                                INC     R3                      ; CTLR                  5815
001034  000243                                .WORD CLV!CLC

001036  003717                                BLE     6⌀
001040  012716  000000G                        MOV     #CRLF,(SP)              ;                       5834
001044  012746  000001                        MOV     #1,-(SP)
001050  010600                                MOV     SP,R0                   ; SP,*
001052  104416                                TRAP    16
001054  062706  000012                        ADD     #12,SP                  ;                       5810
001060  062706  000016                  9⌀:     ADD     #16,SP                  ;                       5736
001064  000207                                RTS     PC

; Routine Size:  283 words,     Routine Base:  $CODE$ + 0530
; Maximum stack depth per invocation:  40 words




                                              .SBTTL  L⌀RPT REPORT CODING SECTION
000000  004737  000530'                L⌀RPT:: JSR     PC,LRPT                 ;                       5835
000004  104425                                TRAP    25
000006  000207                                RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 1616
; Maximum stack depth per invocation:  2 words


;   5838  1
```

```
;    5839  1      %sbttl 'INITIALIZE SECTION'
;    5840  1
;    5841  2      BGNINIT;
;    5842  2
;    5843  2      local
;    5844  2          DELAY_MULT : word,
;    5845  2          FLAG : byte,
;    5846  2          TEMP : word,
;    5847  2          HWPT_REF : ref block [HWPT_LEN, word] field (HWP_FIELDS),
;    5848  2          CLEAR_TABLES : byte,
;    5849  2          SMALLEST_DRIVE : byte,
;    5850  2          BLANKS : WORD INITIAL (%o'020040'),                          !ZZZ
;    5851  2          HWPT_ADDRESS : vector [MAX_UNITS, word];
;    5852  2
;    5853  2      SETPRI (PRI07);                                                  ! NO INTERRUPTS ALLOWED DURING INIT
;    5854  2
;    5855  2      if READEF (EF_NEW)                                               ! IS THIS A NEW PASS?
;    5856  2      then
;    5857  3          begin
;    5858  3          ENTRY_REASON = NEW_PASS;
;    5859  3
;    5860  4          if not BIT_TST (SWP_FLAGS, SWF_CST)
;    5861  3          then
;    5862  3              CLEAR_TABLES = FALSE
;    5863  3          else
;    5864  3              CLEAR_TABLES = TRUE;
;    5865  3
;    5866  2          end;
;    5867  2
;    5868  2      if READEF (EF_START)                                             ! IS THIS A START?
;    5869  2      then
;    5870  3          begin
;    5871  3          BRESET;
;    5872  3          ENTRY_REASON = START;
;    5873  3          CLEAR_TABLES = TRUE;
;    5874  3          ADDR_VECT_OK = FALSE;
;    5875  3          INIT_OCCURED = FALSE;
;    5876  2          end;
;    5877  2
;    5878  2      if READEF (EF_RESTART)                                           ! IS THIS A RESTART?
;    5879  2      then
;    5880  3          begin
;    5881  3          ENTRY_REASON = RESTART;
;    5882  3          CLEAR_TABLES = TRUE;
;    5883  2          end;
;    5884  2
;    5885  2      if READEF (EF_CONTINUE)                                          ! IS THIS A CONTINUE?
;    5886  2      then
;    5887  3          begin
;    5888  3          ENTRY_REASON = CONT;
;    5889  3
;    5890  4          if not BIT_TST (SWP_FLAGS, SWF_CST)
;    5891  3          then
```

```
;    5892  3            CLEAR_TABLES = FALSE
;    5893  3        else
;    5894  3            CLEAR_TABLES = TRUE;
;    5895  3
;    5896  2        end;
;    5897  2
;    5898  2    if READEF (EF_PWR)                                          ! ARE WE HERE BECAUSE OF POWER FAIL
;    5899  2    then
;    5900  3        begin
;    5901  3        ENTRY_REASON = PWR_FAIL;
;    5902  3        ADDR_VECT_OK = FALSE;
;    5903  3        INIT_OCCURED = FALSE;
;    5904  3        CLEAR_TABLES = TRUE;
;    5905  3        PRINTF (MSG_01);                                        ! "POWER DELAY - WAITING"
;    5906  3
;    5907  3        incr COUNT from 0 to 60 do                              ! WAIT APPROX. 60 SECONDS
;    5908  4            begin
;    5909  4            DELAY_MULT = 333;
;    5910  4            DELAY (.DELAY_MULT);
;    5911  4            BREAK;                                              ! BREAK FOR ACT
;    5912  3            end;
;    5913  3
;    5914  2        end;
;    5915  2
;    5916  2    !SETVEC (O_TVEC, O_BRK, PRIO7);                             ! SET ODT TRAP VECTOR
;    5917  2
;    5918  2    !+
;    5919  2    !       MAKE SURE THAT NOT MORE THAN MAX_UNITS HAVE BEEN SPECIFIED.
;    5920  2    !       IF THERE ARE TOO MANY, NOTIFY USER AND RETURN TO SUPERVISOR.
;    5921  2    !       (DIAGNOSTIC IS ABORTED).
;    5922  2    !-
;    5923  2
;    5924  2    if .L$UNIT gtru MAX_UNITS
;    5925  2    then
;    5926  3        begin
;    5927  3        ERRSF (1, EGS_01, EMS_01);
;    5928  3        DOCLN;
;    5929  2        end;
;    5930  2
;    5931  2    !+
;    5932  2    !       THE FOLLOWING CODE IS EXECUTED FOR ALL ENTRY REASONS EXCEPT NEW_PASS.
;    5933  2    !       ALL RUN-TIME CONTROLLER STATUS TABLES (CSTs) ARE CLEARED TO 0, THEN
;    5934  2    !       LOADED WITH CONFIGURATION DATA FROM THE HARDWARE P-TABLES.
;    5935  2    !-
;    5936  2
;    5937  2    if .ENTRY_REASON neq NEW_PASS
;    5938  2    then
;    5939  3        begin
;    5940  3        SMALLEST_DRIVE = 255;                                   ! LARGEST DISK NO. ALLOWED BY MSCP
;    5941  3
;    5942  3        incr COUNT from 0 to ((MAX_CTLR * CST_LEN * 2) - 2) by 2 do
;    5943  3            (CST + .COUNT) = 0;
;    5944  3
```

```
;    5945  3          incr UNIT from 0 to (.L$UNIT - 1) do                                    ! LOOP THROUGH ALL UNITS
;    5946  3
;    5947  3              if (HWPT_ADDRESS [.UNIT] = GPHARD (.UNIT, HWPT_REF)) neqa 0         ! IF HWP TABLE FOUND
;    5948  3              then
;    5949  3
;    5950  3                  if .HWPT_REF [HWP_DISK_NUM] lssu .SMALLEST_DRIVE                ! FIND OUT THE SMALLEST DISK NUMBER
;    5951  3                  then
;    5952  3                      SMALLEST_DRIVE = .HWPT_REF [HWP DISK_NUM];
;    5953  3
;    5954  3          incr UNIT from 0 to (.L$UNIT   1) do                                    ! LOOP THROUGH ALL UNITS
;    5955  3
;    5956  3              if .HWPT_ADDRESS [.UNIT] neqa 0                                     ! IF HWP TABLE FOUND
;    5957  3              then
.    5958  4                  begin
;    5959  4                  FLAG = NOT_FOUND;
;    5960  4                  HWPT_REF = .HWPT_ADDRESS [.UNIT];
;    5961  4
;    5962  4                  incr CTLR from 0 to (MAX_CTLR - 1) do                           ! LOOP THROUGH ALL CSTs
;    5963  4
;    5964  4                      if .CST [.CTLR, IP_ADDR] eqla .HWPT_REF [HWP_IP_ADDR]
;    5965  4                      then
;    5966  4
;    5967  4                          if .CST [.CTLR, (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE
;    5968  4                                  * OF_UN + OF_DATA, D_PRES] eql NOT_PRESENT
;    5969  4                          then
;    5970  5                              begin                                           ! IF EMPTY SLOT FOUND
;    5971  5                              TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;    5972  5                              CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;    5973  5                                                                              ! COPY DISK ADDR AND PROT BIT
;    5974  5                              CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;    5975  5                              CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;    5976  5                              CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;    5977  5
;    5978  5                              IF .HWPT_REF [' '_ENTIRE] EQL TRUE              !ZZZ IF DEFAULT TEST RANGE,
;    5979  5                              THEN HWPT_REF   WP_END_TRK1] = ALL_ONES;        !ZZZ MAKE HI ADDR ALL ONES
;    5980  5
;    5981  5                              CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =
;    5982  5                                      .HWPT_REF [HWP_BEG_TRK];                !ZZZ
;    5983  5                              CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =          !ZZZ
;    5984  5                                      .HWPT_REF [HWP_BEG_TRK1];               !ZZZ
;    5985  5                              CST [.CTLR, .TEMP + OF_END, D_END0] =           !ZZZ
;    5986  5                                      .HWPT_REF [HWP_END_TRK];                !ZZZ
;    5987  5                              CST [.CTLR, .TEMP + OF_END1, D_END1] =          !ZZZ
;    5988  5                                      .HWPT_REF [HWP_END_TRK1];               !ZZZ
;    5989  5
;    5990  5                              CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;    5991  5                              CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;    5992  5
;    5993  5
;    5994  5                              CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;    !ZZZ
;    5995  5                              CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =  !ZZZ
;    5996  5                                  NOT (.HWPT REF [HWP_DISK DUPEX]);           !ZZZ
;    5997  5
```

```
;    5998  5                          CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =     !ZZZ
;    5999  5                                (.HWPT_REF [HWP_DISK_DUPWT]);              !ZZZ
;    6000  5                          CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;      !ZZZ
;    6001  5                          FLAG = FOUND;
;    6002  5                          exitloop;
;    6003  5                          end
;    6004  4                      else
;    6005  5                          begin                                            ! DUPLICATE UNIT
;    6006  5                          PRINTF (CER_01, .HWPT_REF [HWP_DISK_NUM], .HWPT_REF [HWP_IP_ADDR]);
;    6007  5                                                                           ! "DUPLICATE UNIT: X AT IP: XXXXXX"
;    6008  5                          DUR [.UNIT] = DU_CONF;                           ! CONFIGURATION ERROR
;    6009  5                          DODU (.UNIT);                                    ! DROP UNIT
;    6010  5                          FLAG = FOUND;
;    6011  5                          exitloop;
;    6012  4                          end;
;    6013  4
;    6014  4                  if .FLAG eql NOT_FOUND                                   ! IF NO IP MATCH TO EXISTING CST
;    6015  4                  then
;    6016  5                      begin
;    6017  5
;    6018  5                      incr CTLR from 0 to (MAX_CTLR - 1) do                ! LOOP THROUGH EACH CST
;    6019  5
;    6020  5                          if .CST [.CTLR, IP_ADDR] eqle 0                  ! IF EMPTY CST FOUND
;    6021  5                          then
;    6022  6                              begin
;    6023  6                              CST [.CTLR, IP_ADDR] = .HWPT_REF [HWP_IP_ADDR];
;    6024  6                              CST [.CTLR, VEC_ADDR] = .HWPT_REF [HWP_VECTOR];
;    6025  6                              CST [.CTLR, BR_LEV] = .HWPT_REF [HWP_BR_LEVEL];
;    6026  6                              TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE * OF_UN;
;    6027  6                              CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;    6028  6                                                                           ! COPY DISK ADDR AND PROT BIT
;    6029  6                              CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;    6030  6                              CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;    6031  6                              CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;    6032  6
;    6033  6                              IF .HWPT_REF [HWP_ENTIRE] EQL TRUE           !ZZZ IF DEFAULT TEST RANGE,
;    6034  6                              THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;     !ZZZ MAKE HI ADDR ALL ONES
;    6035  6
;    6036  6                              CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =        !ZZZ
;    6037  6                                      .HWPT_REF [HWP_BEG_TRK];             !ZZZ
;    6038  6                              CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =       !ZZZ
;    6039  6                                      .HWPT_REF [HWP_BEG_TRK1];            !ZZZ
;    6040  6                              CST [.CTLR, .TEMP + OF_END, D_END0] =        !ZZZ
;    6041  6                                      .HWPT_REF [HWP_END_TRK];             !ZZZ
;    6042  6                              CST [.CTLR, .TEMP + OF_END1, D_END1] =       !ZZZ
;    6043  6                                      .HWPT_REF [HWP_END_TRK1];            !ZZZ
;    6044  6
;    6045  6                              CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;     !ZZZ BLANK NAME
;    6046  6                              CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;     !ZZZ BLANK NAME
;    6047  6
;    6048  6
;    6049  6                              CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;     !ZZZ
;    6050  6                              CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =   !ZZZ
```

```
;    6051  6                                  NOT (.HWPT_REF [HWP_DISK_DUPEX]);                       !ZZZ
;    6052  6                          CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =                    !ZZZ
;    6053  6                                  (.HWPT_REF [HWP_DISK_DUPWT]);                           !ZZZ
;    6054  6                          CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;                     !ZZZ
;    6055  6                          FLAG = FOUND;
;    6056  6                          exitloop;
;    6057  5                          end;                                                            ! IF EMPTY CST FOUND
;    6058  5
;    6059  5                      if .FLAG eql NOT_FOUND                                              ! IF NO EMPTY CST FOUND
;    6060  5                      then
;    6061  6                          begin
;    6062  6                          PRINTF (CER_02, MAX_CTLR);                                      ! "MORE THAN X IP ADDRESSES."
;    6063  6                          DUR [.UNIT] = DU_CONF;                                          ! CONFIGURATION ERROR
;    6064  6                          DODU (.UNIT);                                                   ! DROP UNIT
;    6065  5                          end;
;    6066  5
;    6067  4                      end;                                                                ! IF NO IP ADDR MATCH IN CST
;    6068  4
;    6069  3                  end;                                                                    ! IF GPHARD RETURNS A HWP TABLE
;    6070  3          !
;    6071  3          ! CONFIGURATON CHECK FOR LEGAL RDRX UNIT MIX BECAUSE WE HAVE DIFFERENT
;    6072  3          ! DRIVES :  THE RD51, RD52, AND RX50.
;    6073  3          ! (NEEDED?)
;    6074  3          !
;    6075  2          end;                                                                            ! END OF "NON NEW_PASS" INIT
;    6076  2
;    6077  2      if .ENTRY_REASON eql NEW_PASS
;    6078  2      then
;    6079  3          begin
;    6080  3
;    6081  3          incr UNIT from 0 to (.L$UNIT - 1) do
;    6082  3              GPHARD (.UNIT, HWPT_REF);                                                   ! DUMMY GPHARDs FOR NEW PASS
;    6083  3
;    6084  3          incr CTLR from 0 to (MAX_CTLR - 1) do
;    6085  4              begin
;    6086  4              CST [.CTLR, U_CNT] = 0;                                                     ! REINITIALIZE UNIT COUNT
;    6087  4
;    6088  4              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF UN) by UNIT_SIZE do
;    6089  4                  CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;                       ! START EACH UNIT AS OFFLINE
;    6090  4
;    6091  3              end;
;    6092  3
;    6093  2          end;
;    6094  2
;    6095  2      if .ENTRY_REASON eql START
;    6096  2      then
;    6097  3          begin
;    6098  3          CTLR_CNT = 0;                                                                   ! NUMBER OF CONFIGURED CONTROLLERS
;    6099  3
;    6100  3          incr CTLR from 0 to (MAX_CTLR - 1) do
;    6101  3
;    6102  3              if .CST [.CTLR, IP ADDR] neqa 0                                             ! IF CONTROLLER IS PRESENT
;    6103  3              then
```

```
;   6104  3                 CTLR_CNT = .CTLR_CNT + 1;                          ! INCREMENT CONTROLLER COUNT
;   6105  3
;   6106  3             MEMORY (FREF_MEM_ADDR);                                ! GET START OF FREE MEMORY
;   6107  3
;   6108  2             end;                                                   ! END OF "START" INITIALIZATION
;   6109  2
;   6110  2         !+
;   6111  2         !       CLEAR STATISTICS TABLES
;   6112  2         !-
;   6113  2
;   6114  2         incr UNITS from 0 to MAX_UNITS - 1 do                      ! CLEAR CURRENT STATISTICS
;   6115  2             incr COUNT from 0 to TALLY_CLEAR - 1 do
;   6116  2                 TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;
;   6117  2
;   6118  2         if .CLEAR_TABLES                                           ! IF CLEAR TABLES ON EVERY PASS
;   6119  2         then
;   6120  2             incr UNITS from 0 to MAX_UNITS   1 do
;   6121  2                 incr COUNT from TALLY_CLEAR to TALLY_LEN - 1 do    ! INITIALIZE TOTALS
;   6122  2                     TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;       !
;   6123  2
;   6124  2         if .CLEAR_TABLES
;   6125  2         then
;   6126  2             incr CTLR from 0 to MAX_CTLR - 1 do
;   6127  3                 begin
;   6128  3                 C_ERR_TBL [.CTLR, C_ERR_HRD] = 0;                  ! INITALIZE CONTROLLER ERRORS
;   6129  3                 C_ERR_TBL [.CTLR, C_ERR_SFT] = 0;                  !
;   6130  2                 end;
;   6131  2
;   6132  2         !+
;   6133  2         !       MISCELLANEOUS INITIALIZATON
;   6134  2         !-
;   6135  2
;   6136  2         incr CTLR from 0 to (MAX_CTLR - 1) do                      ! INIT NO. OF OUTSTANDING QIOs
;   6137  2             QIU [.CTLR] = 0;
;   6138  2
;   6139  2         incr COUNT from 0 to (RP_CNT - 1) do                       ! INITIALIZE RETURN PACKET POOL
;   6140  2             RP_USE [.COUNT] = -1;
;   6141  2
;   6142  2         if .CLK_PRESENT                                            ! STOP CLOCK IF PRESENT
;   6143  2         then
;   6144  2             LINE_CLOCK = 0;
;   6145  2
;   6146  2         IODQ_IN = IODQ_OUT = 0;                                    ! INIT I/O DONE QUEUE POINTERS
;   6147  2         CRN_LOW = CRN_HIGH = 0;                                    ! INIT COMMAND REFERENCE NUMBER
;   6148  2         SETPRI (PRI00);                                            ! SET PROGRAM PRIORITY TO 0
;   6149  2
;   6150  1         ENDINIT;

                                        .GLOBL  L$DLY

                                        .SBTTL  LINIT INITIALIZE SECTION
```

```
000000  004137  000000G            LINIT:  JSR     R1,$SAVE5                       ;                                       5837
000004  162706  000030                    SUB     #30,SP
000010  012746  020040                    MOV     #20040,-(SP)                    ; *,BLANKS
000014  012700  000340                    MOV     #340,R0                         ;                                       5853
000020  104441                            TRAP    41
000022  012700  000035                    MOV     #35,R0                          ;                                       5855
000026  104447                            TRAP    47
000030  103014                            BHIS    2$
000032  112737  000005  000000G           MOVB    #5,ENTRY.REASON                 ;                                       5858
000040  105737  000000G                   TSTB    SWP.FLAGS                       ;                                       5860
000044  100403                            BMI     1$
000046  105066  000012                    CLRB    12(SP)                          ; CLEAR.TABLES                          5862
000052  000403                            BR      2$                              ;                                       5860
000054  112766  000001  000012    1$:     MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5864
000062  012700  000040            2$:     MOV     #40,R0                          ;                                       5868
000066  104447                            TRAP    47
000070  103013                            BHIS    3$
000072  104433                            TRAP    33                              ;                                       5870
000074  112737  000001  000000G           MOVB    #1,ENTRY.REASON                 ;                                       5872
000102  112766  0C0001  000012            MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5873
000110  105037  000000G                   CLRB    ADDR.VECT.OK                    ;                                       5874
000114  105037  000000G                   CLRB    INIT.OCCURED                    ;                                       5875
000120  012700  000037            3$:     MOV     #37,R0                          ;                                       5878
000124  104447                            TRAP    47
000126  103006                            BHIS    4$
000130  112737  000002  000000G           MOVB    #2,ENTRY.REASON                 ;                                       5881
000136  112766  000001  000012            MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5882
000144  012700  000036            4$:     MOV     #36,R0                          ;                                       5885
000150  104447                            TRAP    47
000152  103014                            BHIS    6$
000154  112737  000003  000000G           MOVB    #3,ENTRY.REASON                 ;                                       5888
000162  105737  000000G                   TSTB    SWP.FLAGS                       ;                                       5890
000166  100403                            BMI     5$
000170  105066  000012                    CLRB    12(SP)                          ; CLEAR.TABLES                          5892
000174  000403                            BR      6$                              ;                                       5890
000176  112766  000001  000012    5$:     MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5894
000204  012700  000034            6$:     MOV     #34,R0                          ;                                       5898
000210  104447                            TRAP    47
000212  103043                            BHIS    12$
000214  112737  000004  000000G           MOVB    #4,ENTRY.REASON                 ;                                       5901
000222  105037  000000G                   CLRB    ADDR.VECT.OK                    ;                                       5902
000226  105037  000000G                   CLRB    INIT.OCCURED                    ;                                       5903
000232  112766  000001  000012            MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5904
000240  012746  000000G                   MOV     #MSG.01,-(SP)                   ;                                       5905
000244  012746  000001                    MOV     #1,-(SP)
000250  010600                            MOV     SP,R0                           ; SP,*
000252  104417                            TRAP    17
000254  012702  000075                    MOV     #75,R2                          ; *,COUNT                               5907
000260  012703  000515            7$:     MOV     #515,R3                         ; *,DELAY.MULT                          5909
000264  010301                            MOV     R3,R1                           ; DELAY.MULT,$$TMP2                     5910
000266  001411                    8$:     BEQ     11$
000270  013700  000000G                   MOV     L$DLY,R0                        ; *,$$TMP1
000274  001404                            BEQ     10$
```

```
000276  005066  000024          9$:     CLR     24(SP)                  ; $$TMP
000302  005300                          DEC     R0                      ; $$TMP1
000304  001374                          BNE     9$
000306  005301                  10$:     DEC     R1                      ; $$TMP2
000310  000766                          BR      8$
000312  104422                  11$:     TRAP    22
000314  005302                          DEC     R2                      ; COUNT                                 5907
000316  001360                          BNE     7$
000320  022626                          CMP     (SP)+,(SP)+             ;                                       5900
000322  023727  000000G 000004  12$:     CMP     L$UNIT,#4               ;                                       5924
000330  101405                          BLOS    13$
000332  104454                          TRAP    54                      ;                                       5927
000334  000001                          .WORD   1
000336  000000G                         .WORD   EGS.01
000340  000000V                         .WORD   EMS.01
000342  104444                          TRAP    44
000344  123727  000000G 000005  13$:     CMPB    ENTRY.REASON,#5         ;                                       5937
000352  001002                          BNE     14$
000354  000137  003726'                 JMP     43$
000360  112766  000377  000010  14$:     MOVB    #377,10(SP)             ; *,SMALLEST.DRIVE                      5940
000366  005000                          CLR     R0                      ; COUNT                                 5942
000370  005060  000000G         15$:     CLR     CST(R0)                 ; *(COUNT)                              5943
000374  062700  000002                  ADD     #2,R0                   ; *,COUNT                               5942
000400  020027  000124                  CMP     R0,#124                 ; COUNT,*
000404  003771                          BLE     15$
000406  013704  000000G                 MOV     L$UNIT,R4               ;                                       5945
000412  005003                          CLR     R3                      ; UNIT
000414  000435                          BR      18$
000416  010302                  16$:     MOV     R3,R2                   ; UNIT,*                                5947
000420  006302                          ASL     R2
000422  012700  000022                  MOV     #22,R0
000426  060600                          ADD     SP,R0                   ; HWPT.ADDRESS,*
000430  060002                          ADD     R0,R2
000432  010300                          MOV     R3,R0                   ; UNIT,*
000434  104442                          TRAP    42
000436  010001                          MOV     R0,R1                   ; *,HWPT.REF
000440  010112                          MOV     R1,(R2)                 ; HWPT.REF,*
000442  001421                          BEQ     17$
000444  005002                          CLR     R2                      ;                                       5950
000446  156602  000010                  BISB    10(SP),R2               ; SMALLEST.DRIVE,*
000452  116100  000006                  MOVB    6(R1),R0                ; *(HWPT.REF),*
000456  042700  177760                  BIC     #177760,R0
000462  020002                          CMP     R0,R2
000464  103010                          BHIS    17$
000466  116100  000006                  MOVB    6(R1),R0                ; *(HWPT.REF),*                         5952
000472  042700  177760                  BIC     #177760,R0
000476  105066  000010                  CLRB    10(SP)                  ; SMALLEST.DRIVE
000502  050066  000010                  BIS     R0,10(SP)               ; *,SMALLEST.DRIVE
000506  005203                  17$:     INC     R3                      ; UNIT                                  5945
000510  020304                  18$:     CMP     R3,R4                   ; UNIT,*
000512  002741                          BLT     16$
000514  013766  000000G 000016          MOV     L$UNIT,16(SP)           ;                                       5954
000522  005004                          CLR     R4                      ; UNIT
```

```
000524  000137  003704'                    JMP     41$
000530  010400                      19$:    MOV     R4,R0                   ; UNIT,*                5956
000532  006300                              ASL     R0
000534  012703  000022                      MOV     #22,R3
000540  060603                              ADD     SP,R3                   ; HWPT.ADDRESS,*
000542  060300                              ADD     R3,R0
000544  005710                              TST     (R0)
000546  001002                              BNE     20$
000550  000137  003702'                    JMP     40$
000554  105066  000006              20$:    CLRB    6(SP)                   ; FLAG                  5959
000560  011001                              MOV     (R0),R1                 ; *,HWPT.REF            5960
000562  005066  000002                      CLR     2(SP)                   ; CTLR                  5962
000566  016646  000002              21$:    MOV     2(SP),-(SP)             ; CTLR,*                5964
000572  012746  000126                      MOV     #126.,-(SP)
000576  004737  000000G                     JSR     PC,BL$MUL
000602  022626                              CMP     (SP)+,(SP)+
000604  026011  000000G                     CMP     CST(R0),(R1)            ; *,HWPT.REF
000610  001402                              BEQ     22$
000612  000137  003124'                    JMP     28$
000616  012766  000001  000014      22$:    MOV     #1,14(SP)               ;                       6001
000624  112766  000001  000006              MOVB    #1,6(SP)                ; *,FLAG
000632  012705  000006                      MOV     #6,R5                   ;                       5967
000636  060105                              ADD     R1,R5                   ; HWPT.REF,*
000640  111546                              MOVB    (R5),-(SP)
000642  042716  177760                      BIC     #177760,(SP)
000646  005000                              CLR     R0
000650  156600  000012                      BISB    12(SP),R0               ; SMALLEST.DRIVE,*
000654  160016                              SUB     R0,(SP)
000656  012746  000012                      MOV     #12.,-(SP)
000662  004737  000000G                     JSR     PC,BL$MUL
000666  010066  000010                      MOV     R0,10(SP)
000672  005726                              TST     (SP)+
000674  016616  000004                      MOV     4(SP),(SP)              ; CTLR,*                5968
000700  012746  000053                      MOV     #53.,-(SP)
000704  004737  000000G                     JSR     PC,BL$MUL
000710  010003                              MOV     R0,R3
000712  022626                              CMP     (SP)+,(SP)+
000714  066600  000004                      ADD     4(SP),R0
000720  006300                              ASL     R0
000722  032760  040000  000006G            BIT     #40000,CST+6(R0)
000730  001140                              BNE     27$
000732  016602  000004                      MOV     4(SP),R2                ; *,TEMP                5971
000736  062702  000003                      ADD     #3,R2                   ; *,TEMP
000742  010300                              MOV     R3,R0                   ;                       5972
000744  060200                              ADD     R2,R0                   ; TEMP,*
000746  006300                              ASL     R0
000750  062700  000000G                     ADD     #CST,R0
000754  011510                              MOV     (R5),(R0)
000756  010446                              MOV     R4,-(SP)                ; UNIT,*                5974
000760  000316                              SWAB    (SP)
000762  042716  170377                      BIC     #170377,(SP)
000766  042710  007400                      BIC     #7400,(R0)
000772  052610                              BIS     (SP)+,(R0)
```

```
000774  042710  010000                  BIC     #10000,(R0)
001000  052710  040000                  BIS     #40000,(R0)                                     ;                                       5975
001004  105715                          TSTB    (R5)                                            ;                                       5976
001006  100003                          BPL     23$                                             ;                                       5978
001010  012761  177777  000016          MOV     #-1,16(R1)                      ; *,*(HWPT.REF)
001016  010300                  23$:    MOV     R3,R0                                           ;                                       5979
001020  060200                          ADD     R2,R0                           ; TEMP,*                                                5981
001022  006300                          ASL     R0
001024  016160  000010  000002G          MOV     10(R1),CST+2(R0)                ; *(HWPT.REF),*
001032  010300                          MOV     R3,R0                                           ;                                       5983
001034  060200                          ADD     R2,R0                           ; TEMP,*
001036  006300                          ASL     R0
001040  016160  000012  000004G          MOV     12(R1),CST+4(R0)                ; *(HWPT.REF),*
001046  010300                          MOV     R3,R0                                           ;                                       5985
001050  060200                          ADD     R2,R0                           ; TEMP,*
001052  006300                          ASL     R0
001054  016160  000014  000006G          MOV     14(R1),CST+6(R0)                ; *(HWPT.REF),*
001062  010300                          MOV     R3,R0                                           ;                                       5987
001064  060200                          ADD     R2,R0                           ; TEMP,*
001066  006300                          ASL     R0
001070  016160  000016  000010G          MOV     16(R1),CST+10(R0)               ; *(HWPT.REF),*
001076  010300                          MOV     R3,R0                                           ;                                       5990
001100  060200                          ADD     R2,R0                           ; TEMP,*
001102  006300                          ASL     R0
001104  011660  000012G                 MOV     (SP),CST+12(R0)                 ; BLANKS,*
001110  010300                          MOV     R3,R0                                           ;                                       5991
001112  060200                          ADD     R2,R0                           ; TEMP,*
001114  006300                          ASL     R0
001116  011660  000014G                 MOV     (SP),CST+14(R0)                 ; BLANKS,*
001122  010300                          MOV     R3,R0                                           ;                                       5994
001124  060200                          ADD     R2,R0                           ; TEMP,*
001126  006300                          ASL     R0
001130  062700  000020G                 ADD     #CST+20,R0
001134  105010                          CLRB    (R0)
001136  111546                          MOVB    (R5),-(SP)                                      ;                                       5996
001140  005046                          CLR     -(SP)
001142  032766  000040  000002          BIT     #40,2(SP)
001150  001401                          BEQ     24$
001152  005216                          INC     (SP)
001154  005116                  24$:    COM     (SP)
001156  011646                          MOV     (SP),-(SP)
001160  042710  100000                  BIC     #100000,(R0)
001164  006026                          ROR     (SP)+
001166  103002                          BCC     25$
001170  052710  100000                  BIS     #100000,(R0)
001174  005726                  25$:    TST     (SP)+
001176  111516                          MOVB    (R5),(SP)                                       ;                                       5998
001200  042710  010000                  BIC     #10000,(R0)
001204  032726  000100                  BIT     #100,(SP)+
001210  001402                          BEQ     26$
001212  052710  010000                  BIS     #10000,(R0)
001216  010300                  26$:    MOV     R3,R0                                           ;                                       6000
001220  060200                          ADD     R2,R0                           ; TEMP,*
```

```
001222  006300                          ASL     R0
001224  005060  000022G                 CLR     CST+22(R0)
001230  000430                          BR      29$                             !
001232  011146                  27$:    MOV     (R1),-(SP)                      ; HWPT.REF,*                    5970
001234  111546                          MOVB    (R5),-(SP)                                                      6006
001236  042716  177760                  BIC     #177760,(SP)
001242  012746  000000G                 MOV     #CER.01,-(SP)
001246  012746  000003                  MOV     #3,-(SP)
001252  010600                          MOV     SP,R0                           ; SP,*
001254  104417                          TRAP    17
001256  062706  000010                  ADD     #10,SP
001262  112764  000001  000000G         MOVB    #1,DUR(R4)                      ; *,*(UNIT)                     6008
001270  010400                          MOV     R4,R0                           ; UNIT,*                        6009
001272  104451                          TRAP    51
001274  000406                          BR      29$                             !                               6005
001276  005266  000002          28$:    INC     2(SP)                           ; CTLR                          5962
001302  000243                          .WORD CLV!CLC

001304  003002                          BGT     29$
001306  000137  002414'                 JMP     21$
001312  105766  000006          29$:    TSTB    6(SP)                           ; FLAG                          6014
001316  001402                          BEQ     30$
001320  000137  003702'                 JMP     40$
001324  005066  000014          30$:    CLR     14(SP)                          ; CTLR                          6018
001330  016646  000014          31$:    MOV     14(SP),-(SP)                    ; CTLR,*                        6020
001334  012746  000126                  MOV     #126,-(SP)
001340  004737  000000G                 JSR     PC,BL$MUL
001344  022626                          CMP     (SP)+,(SP)+
001346  005760  000000G                 TST     CST(R0)
001352  001402                          BEQ     32$
001354  000137  003622'                 JMP     37$
001360  011160  000000G          32$:    MOV     (R1),CST(R0)                    ; HWPT.REF,*                    6023
001364  016103  000002                  MOV     2(R1),R3                        ; *(HWPT.REF),*                 6024
001370  042703  177000                  BIC     #177000,R3
001374  042760  000777  000002G         BIC     #777,CST+2(R0)
001402  050360  000002G                 BIS     R3,CST+2(R0)
001406  116160  000004  000004G         MOVB    4(R1),CST+4(R0)                 ; *(HWPT.REF),*                 6025
001414  012705  000006                  MOV     #6,R5                           ;                               6026
001420  060105                          ADD     R1,R5                           ; HWPT.REF,*
001422  111546                          MOVB    (R5),-(SP)
001424  042716  177760                  BIC     #177760,(SP)
001430  005000                          CLR     R0
001432  156600  000012                  BISB    12(SP),R0                       ; SMALLEST.DRIVE,*
001436  160016                          SUB     R0,(SP)
001440  012746  000012                  MOV     #12,-(SP)
001444  004737  000000G                 JSR     PC,BL$MUL
001450  005726                          TST     (SP)+
001452  010002                          MOV     R0,R2                           ; *,TEMP
001454  062702  000003                  ADD     #3,R2                           ; *,TEMP
001460  016616  000016                  MOV     16(SP),(SP)                     ; CTLR,*                        6027
001464  012746  000053                  MOV     #53,-(SP)
001470  004737  000000G                 JSR     PC,BL$MUL
001474  010003                          MOV     R0,R3
```

```
001476  005726                          TST     (SP)+
001500  060200                          ADD     R2,R0                   ; TEMP,+
001502  006300                          ASL     R0
001504  062700  000000G                 ADD     #CST,R0
001510  011510                          MOV     (R5),(R0)
001512  010416                          MOV     R4,(SP)                 ; UNIT,+                        6029
001514  000316                          SWAB    (SP)
001516  042716  170377                  BIC     #170377,(SP)
001522  042710  007400                  BIC     #7400,(R0)
001526  052610                          BIS     (SP)+,(R0)
001530  042710  010000                  BIC     #10000,(R0)             ;                               6030
001534  052710  040000                  BIS     #40000,(R0)             ;                               6031
001540  105715                          TSTB    (R5)                    ;                               6033
001542  100003                          BPL     33$
001544  012761  177777  000016          MOV     #-1,16(R1)              ; *,+(HWPT.REF)                 6034
001552  010300                  33$:    MOV     R3,R0                   ;                               6036
001554  060200                          ADD     R2,R0                   ; TEMP,+
001556  006300                          ASL     R0
001560  016160  000010  000002G         MOV     10(R1),CST+2(R0)        ; +(HWPT.REF),+
001566  010300                          MOV     R3,R0                   ;                               6038
001570  060200                          ADD     R2,R0                   ; TEMP,+
001572  006300                          ASL     R0
001574  016160  000012  000004G         MOV     12(R1),CST+4(R0)        ; +(HWPT.REF),+
001602  010300                          MOV     R3,R0                   ;                               6040
001604  060200                          ADD     R2,R0                   ; TEMP,+
001606  006300                          ASL     R0
001610  016160  000014  000006G         MOV     14(R1),CST+6(R0)        ; +(HWPT.REF),+
001616  010300                          MOV     R3,R0                   ;                               6042
001620  060200                          ADD     R2,R0                   ; TEMP,+
001622  006300                          ASL     R0
001624  016160  000016  000010G         MOV     16(R1),CST+10(R0)       ; +(HWPT.REF),+
001632  010300                          MOV     R3,R0                   ;                               6045
001634  060200                          ADD     R2,R0                   ; TEMP,+
001636  006300                          ASL     R0
001640  011660  000012G                 MOV     (SP),CST+12(R0)         ; BLANKS,+
001644  010300                          MOV     R3,R0                   ;                               6046
001646  060200                          ADD     R2,R0                   ; TEMP,+
001650  006300                          ASL     R0
001652  011660  000014G                 MOV     (SP),CST+14(R0)         ; BLANKS,+
001656  010300                          MOV     R3,R0                   ;                               6049
001660  060200                          ADD     R2,R0                   ; TEMP,+
001662  006300                          ASL     R0
001664  062700  000020G                 ADD     #CST+20,R0
001670  105010                          CLRB    (R0)
001672  111546                          MOVB    (R5),-(SP)              ;                               6051
001674  005046                          CLR     -(SP)
001676  032766  000040  000002          BIT     #40,2(SP)
001704  001401                          BEQ     34$
001706  005216                          INC     (SP)
001710  005116                  34$:    COM     (SP)
001712  011646                          MOV     (SP),-(SP)
001714  042710  100000                  BIC     #100000,(R0)
001720  006026                          ROR     (SP)+
```

```
001722  103002                          BCC     35$
001724  052710  100000                  BIS     #100000,(R0)
001730  005726                  35$:    TST     (SP)+
001732  111516                          MOVB    (R5),(SP)               ;                       6052
001734  042710  010000                  BIC     #10000,(R0)
001740  032726  000100                  BIT     #100,(SP)+
001744  001402                          BEQ     36$
001746  052710  010000                  BIS     #10000,(R0)
001752  010300                  36$:    MOV     R3,R0                   ;
001754  060200                          ADD     R2,R0                   ; TEMP,*                6054
001756  006300                          ASL     R0
001760  005060  000022G                 CLR     CST+22(R0)
001764  112766  000001  000006          MOVB    #1,6(SP)                ; *,FLAG                6055
001772  000410                          BR      39$                     ;                       6022
001774  005266  000014          37$:    INC     14(SP)                  ; CTLR                  6018
002000  000243                          .WORD CLV!CLC

002002  003002                          BGT     38$
002004  000137  003156'                 JMP     31$
002010  105766  000006          38$:    TSTB    6(SP)                   ; FLAG                  6059
002014  001017                  39$:    BNE     40$
002016  012746  000001                  MOV     #1,-(SP)                ;                       6062
002022  012746  000000G                 MOV     #CER.02,-(SP)
002026  012746  000002                  MOV     #2,-(SP)
002032  010600                          MOV     SP,R0                   ; SP,*
002034  104417                          TRAP    17
002036  112764  000001  000000G         MOVB    #1,DUR(R4)              ; *,*(UNIT)             6063
002044  010400                          MOV     R4,R0                   ; UNIT,*                6064
002046  104451                          TRAP    51
002050  062706  000006                  ADD     #6,SP                   ;                       6061
002054  005204                  40$:    INC     R4                      ; UNIT                  5954
002056  020466  000016          41$:    CMP     R4,16(SP)               ; UNIT,*
002062  002002                          BGE     42$
002064  000137  002356'                 JMP     19$
002070  123727  000000G 000005  42$:    CMPB    ENTRY.REASON,#5         ;                       6077
002076  001051                          BNE     48$
002100  013703  000000G         43$:    MOV     L$UNIT,R3               ;                       6081
002104  005004                          CLR     R4                      ; UNIT
002106  000404                          BR      45$
002110  010400                  44$:    MOV     R4,R0                   ; UNIT,*                6082
002112  104442                          TRAP    42
002114  010001                          MOV     R0,R1                   ; *,HWPT.REF
002116  005204                          INC     R4                      ; UNIT                  6081
002120  020403                  45$:    CMP     R4,R3                   ; UNIT,*
002122  002772                          BLT     44$
002124  005003                          CLR     R3                      ; CTLR                  6084
002126  010346                  46$:    MOV     R3,-(SP)                ; CTLR,*                6086
002130  012746  000126                  MOV     #126,-(SP)
002134  004737  000000G                 JSR     PC,BL$MUL
002140  105060  000005G                 CLRB    CST+5(R0)
002144  010316                          MOV     R3,(SP)                 ; CTLR,*                6089
002146  012746  000053                  MOV     #53,-(SP)
002152  004737  000000G                 JSR     PC,BL$MUL
```

```
002156  012701  000003                    MOV     #3,R1                   ; *,OFFSET                      6088
002162  010002                      47$:   MOV     R0,R2                   ;                               6089
002164  060102                             ADD     R1,R2                   ; OFFSET,*
002166  006302                             ASL     R2
002170  042762  020000  000000G            BIC     #20000,CST(R2)
002176  062701  000012                     ADD     #12,R1                  ; *,OFFSET                      6088
002202  020127  000041                     CMP     R1,#41                  ; OFFSET,*
002206  003765                             BLE     47$
002210  062706  000006                     ADD     #6,SP                   ;                               6085
002214  005203                             INC     R3                      ; CTLR                          6084
002216  000243                             .WORD CLV!CLC

002220  003742                             BLE     46$
002222  123727  000000G 000001    48$:    CMPB    ENTRY.REASON,#1         ;                               6095
002230  001017                             BNE     51$
002232  005037  000000G                    CLR     CTLR.CNT                ;                               6098
002236  005000                             CLR     R0                      ; CTLR                          6100
002240  005760  000000G             49$:   TST     CST(R0)                 ; *(CTLR)                       6102
002244  001402                             BEQ     50$
002246  005237  000000G                    INC     CTLR.CNT                ;                               6104
002252  062700  000126              50$:   ADD     #126,R0                 ; *,CTLR                        6100
002256  000243                             .WORD CLV!CLC

002260  003767                             BLE     49$
002262  104431                             TRAP    31
002264  010037  000000G                    MOV     R0,FREE.MEM.ADDR        ;                               6106
002270  005001                      51$:   CLR     R1                      ; UNITS                         6114
002272  005003                      52$:   CLR     R3                      ; COUNT                         6115
002274  010300                      53$:   MOV     R3,R0                   ; COUNT,*                       6116
002276  060100                             ADD     R1,R0                   ; UNITS,*
002300  006300                             ASL     R0
002302  005060  000000G                    CLR     TALLY(R0)
002306  005203                             INC     R3                      ; COUNT                         6115
002310  020327  000006                     CMP     R3,#6                   ; COUNT,*
002314  003767                             BLE     53$
002316  062701  000033                     ADD     #33,R1                  ; *,UNITS                       6114
002322  020127  000121                     CMP     R1,#121                 ; UNITS,*
002326  003761                             BLE     52$
002330  032766  000001  000012            BIT     #1,12(SP)               ; *,CLEAR.TABLES                6118
002336  001436                             BEQ     57$
002340  005001                             CLR     R1                      ; UNITS                         6120
002342  012703  000007              54$:   MOV     #7,R3                   ; *,COUNT                       6121
002346  010300                      55$:   MOV     R3,R0                   ; COUNT,*                       6122
002350  060100                             ADD     R1,R0                   ; UNITS,*
002352  006300                             ASL     R0
002354  005060  000000G                    CLR     TALLY(R0)
002360  005203                             INC     R3                      ; COUNT                         6121
002362  020327  000032                     CMP     R3,#32                  ; COUNT,*
002366  003767                             BLE     55$
002370  062701  000033                     ADD     #33,R1                  ; *,UNITS                       6120
002374  020127  000121                     CMP     R1,#121                 ; UNITS,*
002400  003760                             BLE     54$
002402  032766  000001  000012            BIT     #1,12(SP)               ; *,CLEAR.TABLES                6124
```

```
002410  001411                          BEQ     57$
002412  005000                          CLR     R0                      ; CTLR                  6126
002414  105060  000000G         56$:    CLRB    C.ERR.TBL(R0)           ; *(CTLR)               6128
002420  105060  000001G                 CLRB    C.ERR.TBL+1(R0)         ; *(CTLR)               6129
002424  062700  000002                  ADD     #2,R0                   ; *,CTLR                6126
002430  000243                          .WORD CLV!CLC

002432  003770                          BLE     56$
002434  005000                  57$:    CLR     R0                      ; CTLR                  6136
002436  105060  000000G         58$:    CLRB    QIO(R0)                 ; *(CTLR)               6137
002442  005200                          INC     R0                      ; CTLR                  6136
002444  000243                          .WORD CLV!CLC

002446  003773                          BLE     58$
002450  005000                          CLR     R0                      ; COUNT                 6139
002452  112760  000377  000000G 59$:    MOVB    #377,RP.USE(R0)         ; *,*(COUNT)            6140
002460  005200                          INC     R0                      ; COUNT                 6139
002462  020027  000007                  CMP     R0,#7                   ; COUNT,*
002466  003771                          BLE     59$
002470  132737  000001  000000G         BITB    #1,CLK.PRESENT          ;                       6142
002476  001402                          BEQ     60$
002500  005037  177546                  CLR     @#177546                ;                       6144
002504  005037  000000G         60$:    CLR     IODQ.OUT                ;                       6146
002510  005037  000000G                 CLR     IODQ.IN
002514  005037  000000G                 CLR     CRN.HIGH                ;                       6147
002520  005037  000000G                 CLR     CRN.LOW
002524  005000                          CLR     R0                      ;                       6148
002526  104441                          TRAP    41
002530  062706  000032                  ADD     #32,SP                  ;                       5837
002534  000207                          RTS     PC

; Routine Size:  687 words,     Routine Base:  $CODE$ + 1626
; Maximum stack depth per invocation:  25 words



                                        .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  001626'         L$INIT::JSR     PC,LINIT                ;                       6148
000004  104411                          TRAP    11
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4364
; Maximum stack depth per invocation:  2 words
```

```
;    6151  1     %sbttl 'AUTODROP SECTION'
;    6152  1
;    6153  1     !+
;    6154  1     ! THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;    6155  1     ! THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
;    6156  1     ! SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
;    6157  1     ! DROPPED FROM TESTING.
;    6158  1     !-
;    6159  1
;    6160  2     BGNAUTO;
;    6161  2
;    6162  2     !if BIT_TST (SWP_FLAGS, SWF_TRC)
;    6163  2     !then
;    6164  2     !   PRINTF (DBM3);
;    6165  2
;    6166  2     return;
;    6167  2
;    6168  1     ENDAUTO;


                                        .SBTTL  LAUTO AUTODROP SECTION
000000  000207                  LAUTO:  RTS     PC                              ;  6150

; Routine Size:  1 word,        Routine Base:  $CODE$ + 4374
; Maximum stack depth per invocation:  0 words



                                        .SBTTL  L$AUTO AUTODROP SECTION
                                        .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  004374'       L$AUTO::JSR     PC,LAUTO                        ;  6166
000004  104461                         TRAP    61
000006  000207                         RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4376
; Maximum stack depth per invocation:  2 words
```

```
;    6169  1     %sbttl 'CLEANUP CODING SECTION'
;    6170  1
;    6171  1     !+
;    6172  1     ! THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
;    6173  1     ! AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
;    6174  1     !-
;    6175  1
;    6176  2     BGNCLN;
;    6177  2
;    6178  2     LABEL                                                                          !ZZZ
;    6179  2         LZ1;                                                                       !ZZZ
;    6180  2
;    6181  2     DORPT;
;    6182  2
;    6183  2     !CLRVEC (O_TVEC);                                  ! RETURN ODT TRAP TO DIAGNOSTIC SUPERVISER
;    6184  2
;    6185  2     if .CLK_PRESENT
;    6186  2     then
;    6187  3         begin
;    6188  3         LINE_CLOCK = 0;                                ! STOP THE LINE-CLOCK
;    6189  3     !   CLRVEC (%o'100');                              ! RETURN LINE-CLOCK'S VECTOR TO SUPERVISOR
;    6190  2         end;
;    6191  2
;    6192  2     incr CTLR from 0 to (MAX_CTLR - 1) do              ! FOR EACH CONTROLLER
;    6193  2
;    6194  2         if (RDRX_ADDR = .CST [.CTLR, IP_ADDR]) neqa 0  ! IF CONTROLLER EXISTS
;    6195  2         then
;    6196  3             begin
;    6197  3
;    6198  3             if .ADDR_VECT_OK
;    6199  3             then
;    6200  4     LZ1:        begin                                                              !ZZZ
;    6201  4
;    6202  4                 if .DCT [.CTLR, STAT] eql ONLINE       ! IF CONTROLLER ALIVE
;    6203  4                 then
;    6204  4
;    6205  4                     incr COUNT from 1 to 10000 do
;    6206  5                         begin
;    6207  5                         DELAY (1);
;    6208  5
;    6209  5                         if .DCT [.CTLR, CRING_CNT] eql 0    ! WAIT TILL OUTSTANDING COMMANDS FINISHED
;    6210  5                         then
;    6211  5
;    6212  5                              INCR Z  FROM 0 TO 3  DO                                                    !ZZZ
;    6213  6                                 BEGIN                                                                   !ZZZ
;    6214  6                                 TEMP1 =  (.DCT [.CTLR, RR_BEG]) + 4 * .Z;          !DESCRIPTOR ADDRESS  !ZZZ
;    6215  6                                 TEMP2 = ..TEMP1;                                   !PACKET ADDRESS      !ZZZ
;    6216  6                                 IF ..TEMP2  EQL CRN_LOW                            !CRN                 !ZZZ
;    6217  6                                 THEN                                               !IF THE LAST CRN IS BACK,  !ZZZ
;    6218  6                                    (WRT_RDRX (RCIP, RC_ALL, ALL_ONES); LEAVE LZ1); !THEN STOP WAITING   !ZZZ
;    6219  5                                 END;                                                                    !ZZZ
;    6220  4                         end;
;    6221  4
```

```
;    6222  4                WRT_RDRX (RCIP, RC_ALL, ALL ONES);          ! WRITE IP TO STOP DEVICE
;    6223  3                end;
;    6224  3
;    6225  3            CLRVEC (.CST[.CTLR, VEC_ADDR]);                 ! RETURN CONTROLLER'S TRAP VECTOR TO SUPERVISOR
;    6226  2            end;
;    6227  2
;    6228  1     ENDCLN;


                                        .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  004137  000000G         LCLEAN: JSR     R1,$SAVE5                       ;                               6168
000004  005746                          TST     -(SP)
000006  104424                          TRAP    24                              ;                               6179
000010  132737  000001  000000G         BITB    #1,CLK.PRESENT                  ;                               6185
000016  001402                          BEQ     1$
000020  005037  177546                  CLR     @#177546                        ;                               6188
000024  005005                  1$:     CLR     R5                              ; CTLR                          6192
000026  010546                  2$:     MOV     R5,-(SP)                        ; CTLR,*                        6194
000030  012746  000126                  MOV     #126,-(SP)
000034  004737  000000G                 JSR     PC,BL$MUL
000040  010003                          MOV     R0,R3
000042  022626                          CMP     (SP)+,(SP)+
000044  016337  000000G 000000G         MOV     CST(R3),RDRX.ADDR
000052  001477                          BEQ     13$
000054  132737  000001  000000G         BITB    #1,ADDR.VECT.OK                 ;                               6198
000062  001466                          BEQ     12$
000064  010546                          MOV     R5,-(SP)                        ; CTLR,*                        6202
000066  012746  000022                  MOV     #22,-(SP)
000072  004737  000000G                 JSR     PC,BL$MUL
000076  022626                          CMP     (SP)+,(SP)+
000100  005760  000000G                 TST     DCT(R0)
000104  100051                          BPL     11$
000106  012704  023420                  MOV     #23420,R4                       ; *,COUNT                       6205
000112  012702  000001          3$:     MOV     #1,R2                           ; *,$$TMP2                      6207
000116  001410                  4$:     BEQ     7$
000120  013701  000000G                 MOV     L$DLY,R1                        ; *,$$TMP1
000124  001403                          BEQ     6$
000126  005016                  5$:     CLR     (SP)                            ; $$TMP
000130  005301                          DEC     R1                              ; $$TMP1
000132  001375                          BNE     5$
000134  005302                  6$:     DEC     R2                              ; $$TMP2
000136  000767                          BR      4$
000140  105760  000000G         7$:     TSTB    DCT(R0)                         ;                               6209
000144  001027                          BNE     10$
000146  005001                          CLR     R1                              ; Z                             6212
000150  016037  000004G 000000G  8$:     MOV     DCT+4(R0),TEMP1                 ;                               6214
000156  060137  000000G                 ADD     R1,TEMP1                        ; Z,*
000162  017737  000000G 000000G         MOV     @TEMP1,TEMP2                    ;                               6215
000170  027727  000000G 000000G         CMP     @TEMP2,#CRN.LOW                 ;                               6216
000176  001005                          BNE     9$
000200  012702  177777                  MOV     #-1,R2                          ; *,RC.REG                      6218
000204  010277  000000G                 MOV     R2,@RDRX.ADDR                   ; RC.REG,*
000210  000413                          BR      12$
```

```
000212  062701  000004          9$:     ADD     #4,R1                   ; *,Z                   6212
000216  020127  000014                  CMP     R1,#14                  ; Z,*
000222  003752                          BLE     8$
000224  005304                  10$:    DEC     R4                      ; COUNT                 6205
000226  001331                          BNE     3$
000230  012700  177777          11$:    MOV     #-1,R0                  ; *,RC.REG              6222
000234  010077  000000G                 MOV     R0,@RDRX.ADDR           ; RC.REG,*
000240  016300  000002G         12$:    MOV     CST-2(R3),R0            ;                       6225
000244  042700  177000                  BIC     #177000,R0
000250  104436                          TRAP    36
000252  005205                  13$:    INC     R5                      ; CTLR                  6192
000254  000243                          .WORD CLV!CLC

000256  003663                          BLE     2$
000260  005726                          TST     (SP)+                   ;                       6168
000262  000207                          RTS     PC

; Routine Size:  90 words,      Routine Base:  $CODE$ + 4406
; Maximum stack depth per invocation:  10 words



                                        .SBTTL  L$CLEAN CLEANUP CODING SECTION
000000  004737  004406'         L$CLEAN::
                                        JSR     PC,LCLEAN               ;                       6226
000004  104412                          TRAP    12
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4672
; Maximum stack depth per invocation:  2 words
```

```
;    6229  1     %sbttl 'DROP UNIT SECTION'
;    6230  1
;    6231  1     !+
;    6232  1     ! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;    6233  1     ! TO NO LONGER BE TESTED.
;    6234  1     !-
;    6235  1
;    6236  2     BGNDU;
;    6237  2
;    6238  2     local
;    6239  2         UNIT : word,                                                    ! UNIT NUMBER
;    6240  2         PRINT : byte initial (byte (FALSE));                            ! NO PRINTING
;    6241  2
;    6242  2     label
;    6243  2         SEARCH;
;    6244  2
;    6245  3     begin
;    6246  3
;    6247  3     register
;    6248  3         INPUT = 0;                                                      ! R0 = UNIT NO.
;    6249  3
;    6250  3     UNIT = .INPUT;                                                      ! GET UNIT NUMBER
;    6251  2     end;                                                                ! UNDECLARE R0
;    6252  2
;    6253  2     !ZZZif BIT_TST (SWP_FLAGS, SWF_TRC)
;    6254  2     !ZZZthen
;    6255  2     !ZZZ    PRINTF (DBMS, .UNIT);
;    6256  2
;    6257  2     SEARCH :                                                            ! SEARCH BLOCK
;    6258  3     begin
;    6259  3
;    6260  3     incr CTLR from 0 to (MAX_CTLR - 1) do                               ! FOR EACH CNTR
;    6261  3
;    6262  3         incr OFFSET from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    6263  3
;    6264  3             if (.CST [.CTLR, .OFFSET + OF_DATA, D_UNIT] eql .UNIT) and  ! IF UNIT MATCHES
;    6265  4                (.CST [.CTLR, .OFFSET + OF_DATA, D_PRES] eql PRESENT)
;    6266  3             then
;    6267  4                 begin
;    6268  4
;    6269  4                 if (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE) or  ! IF UNIT ALIVE
;    6270  4                    (.DUR [.UNIT] eql DU_ONLINE) or
;    6271  5                    (.DUR [.UNIT] eql DU_PROTECT)
;    6272  4                 then
;    6273  5                     begin
;    6274  5                     PRINT = TRUE;                                      ! O.K. TO PRINT
;    6275  5
;    6276  5                     if (.CST [.CTLR, U_CNT] gtru 0) and
;    6277  6                        (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;    6278  5                     then
;    6279  5                         CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] - 1;  ! DECREMENT COUNT
;    6280  5
;    6281  5                     if (.CST [.CTLR, U_CNT] eql 0) and
```

```
;   6282  6                       (.CST [.CTLR, .OFFSET + OF DATA, D_STAT] eql ONLINE)
;   6283  5                   then
;   6284  5                       EOP_FLAG = TRUE;                                    ! ALL UNITS OFFLINE
;   6285  5
;   6286  5                   CST [.CTLR, .OFFSET + OF DATA, D_STAT] = OFFLINE;       ! MARK UNIT OFFLINE
;   6287  4                   end;                                                    ! IF UNIT ALIVE
;   6288  4
;   6289  4               leave SEARCH;                                               ! EXIT SEARCH BLOCK
;   6290  3               end;                                                        ! IF UNIT FOUND
;   6291  3
;   6292  2       end;
;   6293  2
;   6294  2       if .PRINT or                                                        ! IF OK TO PRINT
;   6295  2          (.DUR [.UNIT] eql DU_CONF) or
;   6296  2          (.DUR [.UNIT] eql DU_INIT) or
;   6297  2          (.DUR [.UNIT] eql DU_ONLINE) or
;   6298  3          (.DUR [.UNIT] eql DU_PROTECT)
;   6299  2       then
;   6300  3           begin
;   6301  3           PRINTF (DU_MSG, .UNIT);                                       ! "UNIT XX DROPPED"
;   6302  3           PRINTF (.DU_RSN [.DUR [.UNIT]]);                              ! REASON
;   6303  2           end;
;   6304  2
;   6305  1       ENDDU;


                                .SBTTL  LDU DROP UNIT SECTION
000000  004137  000000G    LDU:  JSR     R1,$SAVE5                 ;                 6228
000004  024646                  CMP     -(SP),-(SP)
000006  105066  000002           CLRB    2(SP)                     ; PRINT
000012  010001                   MOV     R0,R1                     ; INPUT,UNIT      6250
000014  005005                   CLR     R5                        ; CTLR            6260
000016  010546              1$:  MOV     R5,-(SP)                  ; CTLR,*          6264
000020  012746  000053           MOV     #53,-(SP)
000024  004737  000000G          JSR     PC,BL$MUL
000030  010066  000004           MOV     R0,4(SP)
000034  012703  000003           MOV     #3,R3                     ; *,OFFSET        6262
000040  010300              2$:  MOV     R3,R0                     ; OFFSET,*        6264
000042  066600  000004           ADD     4(SP),R0
000046  006300                   ASL     R0
000050  012702  000000G          MOV     #CST,R2
000054  060002                   ADD     R0,R2
000056  010104                   MOV     R1,R4                     ; UNIT,*
000060  011200                   MOV     (R2),R0
000062  000300                   SWAB    R0
000064  042700  177760           BIC     #177760,R0
000070  020004                   CMP     R0,R4
000072  001055                   BNE     8$
000074  032712  040000           BIT     #40000,(R2)               ;                 6265
000100  001452                   BEQ     8$
000102  005004                   CLR     R4                        ;                 6269
000104  032712  020000           BIT     #20000,(R2)
000110  001402                   BEQ     3$
```

```
000112  005204                          INC     R4
000114  000410                          BR      4$
000116  126127  000000G 000007   3$:     CMPB    DUR(R1),#7              ; *(UNIT),*             6270
000124  001404                          BEQ     4$
000126  126127  000000G 000011           CMPB    DUR(R1),#11             ; *(UNIT),*             6271
000134  001032                          BNE     7$
000136  112766  000001  000006   4$:     MOVB    #1,6(SP)                ; *,PRINT               6274
000144  010516                          MOV     R5,(SP)                 ; CTLR,*                6276
000146  012746  000126                   MOV     #126.,-(SP)
000152  004737  000000G                  JSR     PC,BL$MUL
000156  005726                          TST     (SP)+
000160  062700  000004G                  ADD     #CST+4,R0
000164  105760  000001                   TSTB    1(R0)
000170  001404                          BEQ     5$
000172  006004                          ROR     R4                      ;                       6277
000174  105660  000001                   SBCB    1(R0)                   ;                       6279
000200  001006                          BNE     6$                      ;                       6281
000202  032712  020000   5$:     BIT     #20000,(R2)             ;                       6282
000206  001403                          BEQ     6$
000210  112737  000001  000000G          MOVB    #1,EOP.FLAG             ;                       6284
000216  042712  020000   6$:     BIC     #20000,(R2)             ;                       6286
000222  022626                   7$:     CMP     (SP)+,(SP)+             ;                       6267
000224  000411                          BR      9$
000226  062703  000012   8$:     ADD     #12,R3                  ; *,OFFSET              6262
000232  020327  000041                   CMP     R3,#41                  ; OFFSET,*
000236  003700                          BLE     2$
000240  022626                          CMP     (SP)+,(SP)+
000242  005205                          INC     R5                      ; CTLR                  6260
000244  000243                          .WORD CLV!CLC

000246  003663                          BLE     1$
000250  032766  000001  000002   9$:     BIT     #1,2(SP)                ; *,PRINT               6294
000256  001020                          BNE     10$
000260  126127  000000G 000001           CMPB    DUR(R1),#1              ; *(UNIT),*             6295
000266  001414                          BEQ     10$
000270  126127  000000G 000002           CMPB    DUR(R1),#2              ; *(UNIT),*             6296
000276  001410                          BEQ     10$
000300  126127  000000G 000007           CMPB    DUR(R1),#7              ; *(UNIT),*             6297
000306  001404                          BEQ     10$
000310  126127  000000G 000011           CMPB    DUR(R1),#11             ; *(UNIT),*             6298
000316  001024                          BNE     11$
000320  010146                  10$:     MOV     R1,-(SP)                ; UNIT,*                6301
000322  012746  000000G                  MOV     #DU.MSG,-(SP)
000326  012746  000002                   MOV     #2,-(SP)
000332  010600                          MOV     SP,R0                   ; SP,*
000334  104417                          TRAP    17
000336  116101  000000G                  MOVB    DUR(R1),R1              ; *(UNIT),*             6302
000342  042701  177400                   BIC     #177400,R1
000346  006301                          ASL     R1
000350  016116  000000G                  MOV     DU.RSN(R1),(SP)
000354  012746  000001                   MOV     #1,-(SP)
000360  010600                          MOV     SP,R0                   ; SP,*
000362  104417                          TRAP    17
```

```
000364  062706  000010                          ADD     #10,SP                          ;
000370  022626                          11$:    CMP     (SP)+,(SP)+                     ;                6300
000372  000207                                  RTS     PC                              ;                6228

; Routine Size:  126 words,     Routine Base:  $CODE$ + 4702
; Maximum stack depth per invocation:  14 words



                                                .SBTTL  L$DU DROP UNIT SECTION
000000  004737  004702'                  L$DU::  JSR     PC,LDU                          ;                6303
000004  104453                                  TRAP    53
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5276
; Maximum stack depth per invocation:  2 words
```

```
;    6306  1     %sbttl  ADD UNIT SECTION
;    6307  1
;    6308  1     !+
;    6309  1     ! THE ADD UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;    6310  1     !  TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;    6311  1     !  TO THE TEST CYCLE.
;    6312  1     !-
;    6313  1
;    6314  2     BGNAU;
;    6315  2
;    6316  2     local
;    6317  2         STINDX : word,
;    6318  2         ENDIDX : word;
;    6319  2
;    6320  2     register
;    6321  2         UNIT = 0;                                   ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
;    6322  2
;    6323  3     if BIT_TST (SWP FLAGS, SWF CST)
;    6324  2     then
;    6325  3         begin                                       ! IF CLEAR STAT. TABLES TRUE....
;    6326  3         STINDX = .UNIT * TALLY_LEN;                 ! ZERO OUT
;    6327  3         ENDIDX = .STINDX + TALLY_LEN - 1;           !    ADDED
;    6328  3
;    6329  3         incr COUNT from .STINDX to .ENDIDX do       !    UNIT'S
;    6330  3             TALLY [.COUNT] = 0;                     !    STATISTICS
;    6331  3
;    6332  2         end;
;    6333  2
;    6334  1     ENDAU;


                                        .SBTTL  LAU ADD UNIT SECTION
000000  004137  000000G         LAU:    JSR     R1,$SAVE2                   ;                       6305
000004  105737  000000G                 TSTB    SWP.FLAGS                   ;                       6323
000010  100023                         BPL     3$
000012  010046                         MOV     R0, (SP)                    ; UNIT,*                6326
000014  012746  000033                 MOV     #33,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  010002                         MOV     R0,R2                       ; STINDX,ENDIDX         6327
000026  062702  000032                 ADD     #32,R2                      ; *,ENDIDX
000032  010001                         MOV     R0,R1                       ; STINDX,COUNT          6329
000034  005301                         DEC     R1                          ; COUNT
000036  000404                         BR      2$
000040  010100                 1$:     MOV     R1,R0                       ; COUNT,*               6330
000042  006300                         ASL     R0
000044  005060  000000G                 CLR     TALLY(R0)
000050  005201                 2$:     INC     R1                          ; COUNT                 6329
000052  020102                         CMP     R1,R2                       ; COUNT,ENDIDX
000054  003771                         BLE     1$
000056  022626                         CMP     (SP)+,(SP)+                 ;                       6325
000060  000207                 3$:     RTS     PC                          ;                       6305

; Routine Size:  25 words,    Routine Base:  $CODE$ + 5306
```

```
; Maximum stack depth per invocation:  6 words



                                        .SBTTL  L$AU ADD UNIT SECTION
000000  004737  005306'          L$AU::  JSR     PC,LAU                  ;                       6332
000004  104452                          TRAP    52
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 5370
; Maximum stack depth per invocation:  2 words
```

```
;    6335  1     %sbttl 'NON-EXISTENT MEMORY TRAP HANDLER'
;    6336  1
;    6337  1     !+
;    6338  1     !       THIS TRAP HANDLER IS VECTORED FROM LOCATION 4 FOR ALL UNIBUS TIMEOUT
;    6339  1     !       ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO REFERENCE A NON-EXISTENT
;    6340  1     !       MEMORY LOCATION. ITS MAIN PURPOSE IS TO SET A FLAG FOR THE RDRX
;    6341  1     !       REGISTER EXISTENCE TEST, INDICATING THE ABSENCE OF A DEVICE REGISTER.
;    6342  1     !-
;    6343  1
;    6344  2     BGNSRV (NEX_TRAP);
;    6345  2
;    6346  2     NEX = TRUE;                                     ! NEX TRAP OCCURRED
;    6347  2
;    6348  1     ENDSRV;


                                        .SBTTL  NEX.TRAP NON-EXISTENT MEMORY TRAP HANDLER
000000  012737  000001  000000G  NEX.TRAP::
                                        MOV     #1,NEX                                  ;                       6346
000006  000002                          RTI                                             ;                       6344

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5400
; Maximum stack depth per invocation:  0 words
```

```
;    6349  1     %sbttl 'TIME OF DAY'
;    6350  1
;    6351  1     !+
;    6352  1     !       THIS INTERRUPT SERVICE ROUTINE KEEPS TRACK OF THE TIME-OF DAY
;    6353  1     !-
;    6354  1
;    6355  2     BGNSRV (TIME);
;    6356  2
;    6357  2     CLK_TICKS = .CLK_TICKS + 1;              ! INCREMENT CLOCK-TICKS
;    6358  2
;    6359  2     if .CLK_TICKS gequ 3600
;    6360  2     then
;    6361  3         begin
;    6362  3         MINUTES = .MINUTES + 1;              ! UPDATE MINUTE COUNT
;    6363  3         CLK_TICKS = 0;
;    6364  2         end;
;    6365  2
;    6366  2     if .MINUTES gequ 60
;    6367  2     then
;    6368  3         begin
;    6369  3         HOURS = .HOURS + 1;                  ! UPDATE HOUR COUNT
;    6370  3         MINUTES = 0;
;    6371  2         end;
;    6372  2
;    6373  2     if .HOURS gequ 24
;    6374  2     then
;    6375  2         HOURS = 0;                           ! RATIONALIZE HOURS
;    6376  2
;    6377  1     ENDSRV;


                                        .SBTTL  TIME TIME OF DAY
000000  005237  000000G                 TIME::  INC     CLK.TICKS                       ;
000004  023727  000000G 007020                 CMP     CLK.TICKS,#7020                 ;
000012  103404                                 BLO     1$                              ;
000014  105237  000000G                         INCB    MINUTES                         ;
000020  005037  000000G                         CLR     CLK.TICKS                       ;
000024  123727  000000G 000074         1$:     CMPB    MINUTES,#74                     ;
000032  103404                                 BLO     2$                              ;
000034  105237  000000G                         INCB    HOURS                           ;
000040  105037  000000G                         CLRB    MINUTES                         ;
000044  123727  000000G 000030         2$:     CMPB    HOURS,#30                       ;
000052  103402                                 BLO     3$                              ;
000054  105037  000000G                         CLRB    HOURS                           ;
000060  000002                         3$:     RTI                                     ;

; Routine Size:  25 words,      Routine Base:  $CODE$ + 5410
; Maximum stack depth per invocation:  0 words
```

```
;    6378  1      %sbttl 'GLOBAL ROUTINES'
;    6379  1
;    6380  1      global routine SET_CPAR (CTLR) : novalue =
;    6381  1
;    6382  1      !+
;    6383  1      !       THIS ROUTINE SETS UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    6384  1      !       FOR THE GIVEN CONTROLLER NUMBER.
;    6385  1      !
;    6386  1      !       INPUTS:
;    6387  1      !               CTLR - CONTROLLER NUMBER
;    6388  1      !
;    6389  1      !       IMPLICIT OUTPUTS:
;    6390  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    6391  1      !               CST_ADDR - ADDRESS OF CONTROLLER'S STATUS TABLE
;    6392  1      !               DCT_ADDR - ADDRESS OF CONTROLLER'S DRIVER TABLE
;    6393  1      !               RDRX_ADDR - ADDRESS OF CONTROLLER'S IP REGISTER
;    6394  1      !-
;    6395  1
;    6396  2          begin
;    6397  2          CCTLR = .CTLR;                              ! SET CURRENT CONTROLLER NUMBER
;    6398  2          CST_ADDR = CST + (.CTLR * CST_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER'S CST
;    6399  2          DCT_ADDR = DCT + (.CTLR * DCT_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER'S DCT
;    6400  2          RDRX_ADDR = .CST_ADDR [IP_ADDR];            ! GET CONTROLLER'S DEVICE ADDRESS
;    6401  1          end;


                                        .SBTTL  SET.CPAR GLOBAL ROUTINES
000000  010146                     SET.CPAR::
                                        MOV     R1,-(SP)                        ;                       6380
000002  016601  000004                  MOV     4(SP),R1                        ; CTLR,*                6397
000006  010137  000000G                 MOV     R1,CCTLR
000012  010146                          MOV     R1,-(SP)                        ;                       6398
000014  012746  000126                  MOV     #126,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  062700  000000G                 ADD     #CST,R0
000030  010037  000000G                 MOV     R0,CST.ADDR
000034  010116                          MOV     R1,(SP)                         ;                       6399
000036  012746  000022                  MOV     #22,-(SP)
000042  004737  000000G                 JSR     PC,BL$MUL
000046  062700  000000G                 ADD     #DCT,R0
000052  010037  000000G                 MOV     R0,DCT.ADDR
000056  017737  000000G 000000G         MOV     @CST.ADDR,RDRX.ADDR             ;                       6400
000064  062706  000006                  ADD     #6,SP                           ;                       6396
000070  012601                          MOV     (SP)+,R1                        ;                       6380
000072  000207                          RTS     PC

; Routine Size:  30 words,      Routine Base:  $CODE$ + 5472
; Maximum stack depth per invocation:  5 words
```

```
;    6402  1     global routine SET_UPAR (OFFSET) : novalue =
;    6403  1     !+
;    6404  1     !       THIS ROUTINE SETS UP THE COMMONLY-USED UNIT-RELATED DATA ITEMS FOR
;    6405  1     !       THE CURRENT CONTROLLER AND GIVEN CST OFFSET.
;    6406  1     !
;    6407  1     !       INPUTS:
;    6408  1     !               OFFSET - WORD OFFSET INTO CURRENT CONTROLLER'S CST WHICH
;    6409  1     !                        DESCRIBES A UNIT
;    6410  1     !
;    6411  1     !       IMPLICIT INPUTS:
;    6412  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    6413  1     !
;    6414  1     !       IMPLICIT OUTPUTS:
;    6415  1     !               CUOFF - CURRENT UNIT'S CST OFFSET
;    6416  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    6417  1     !               L$LUN - CURRENT UNIT NUMBER (DRS UNIT NUMBER)
;    6418  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    6419  1     !-
;    6420  2         begin
;    6421  2         CUOFF = .OFFSET;
;    6422  2         CDISK = .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM];
;    6423  2         L$LUN = .CST_ADDR [.OFFSET + OF_DATA, D_UNIT];
;    6424  2         T_ADDR = TALLY + (.L$LUN * TALLY_LEN * 2);
;    6425  1         end;


                                .SBTTL  SET.UPAR GLOBAL ROUTINES
000000  010146                 SET.UPAR::
                                MOV     R1,-(SP)                        ;                               6402
000002  016637  000004  000000G        MOV     4(SP),CUOFF             ; OFFSET,*                      6421
000010  016600  000004                 MOV     4(SP),R0                ; CUOFF,*                       6422
000014  006300                         ASL     R0
000016  063700  000000G                ADD     CST.ADDR,R0
000022  111037  000000G                MOVB    (R0),CDISK
000026  042737  177760  000000G        BIC     #177760,CDISK
000034  011001                         MOV     (R0),R1                 ;                               6423
000036  000301                         SWAB    R1
000040  042701  177760                 BIC     #177760,R1
000044  010137  000000G                MOV     R1,L$LUN
000050  010146                         MOV     R1,-(SP)                ; L$LUN,*                       6424
000052  012746  000066                 MOV     #66,-(SP)
000056  004737  000000G                JSR     PC,BL$MUL
000062  062700  000000G                ADD     #TALLY,R0
000066  010037  000000G                MOV     R0,T.ADDR
000072  022626                         CMP     (SP)+,(SP)+             ;                               6420
000074  012601                         MOV     (SP)+,R1                ;                               6402
000076  000207                         RTS     PC

; Routine Size:  32 words,      Routine Base:  $CODE$ + 5566
; Maximum stack depth per invocation:  4 words
```

```
;    6426  1
;    6427  1     global routine GET_PKT (CTLR) =
;    6428  1
;    6429  1     !+
;    6430  1     !       THIS ROUTINE SEARCHES THE MSCP PACKET POOL ALLOCATION TABLE (PKT_USE)
;    6431  1     !       FOR A FREE MSCP PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;    6432  1     !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED
;    6433  1     !       TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;    6434  1     !
;    6435  1     !       INPUTS:
;    6436  1     !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;    6437  1     !-
;    6438  1
;    6439  2         begin
;    6440  2
;    6441  2         local
;    6442  2             index : signed word initial (-1),
;    6443  2             RING_ADDR : word,
;    6444  2             PACKET_OWNED : byte,
;    6445  2             NEXT_PACKET : byte;
;    6446  2
;    6447  2
;    6448  2         NEXT_PACKET = .NEXT_PKT_USE;                    ! NEXT PACKET TO TRY
;    6449  2
;    6450  2         incr COUNT from 0 to (PKT_CNT - 1) do           ! FOR EACH ENTRY IN ALLOCATION TABLE
;    6451  3             begin
;    6452  3             PACKET_OWNED = FALSE;
;    6453  3
;    6454  3             if .PKT_USE [.NEXT_PACKET] lss 0            ! IF ENTRY INDICATES FREE PACKET
;    6455  3
;    6456  3             then
;    6457  4                 begin
;    6458  4                 RING_ADDR = .DCT_ADDR [RR_BEG];         ! FIRST RESPONSE PACKET'S ADDRESS
;    6459  4
;    6460  4                 incr I from 1 to (RRING_LEN + CRING_LEN) do ! FOR EACH PACKET ADDRESS
;    6461  4
;    6462  4                     if (..RING_ADDR eqla .MSCP_PKT [.NEXT_PACKET, PKT_LO]) and
;    6463  5                        (((.RING_ADDR + 2) and ED_OWN) eql ED_OWN)
;    6464  5
;    6465  4                     then
;    6466  5                         begin                           ! CHECK ADDRESS AND OWNERSHIP
;    6467  5                         PACKET_OWNED = TRUE;            ! PACKET OWNED BY CONTROLLER
;    6468  5                         exitloop;
;    6469  5                         end
;    6470  4                     else
;    6471  4                         RING_ADDR = .RING_ADDR + 4;     ! ADDRESS OF NEXT PACKET IN RING
;    6472  4
;    6473  4                 if not .PACKET_OWNED                    ! IF NOT ALREADY USED
;    6474  4
;    6475  4                 then
;    6476  5                     begin
;    6477  5                     PKT_USE [.NEXT_PACKET] = .CTLR;     ! ALLOCATE PACKET TO CONTROLLER
;    6478  5                     index = .NEXT_PACKET;
```

```
;    6479  5
;    6480  5                    incr J from 2 to (PKT_LEN - 1) do       ! ZERO OUT PACKET
;    6481  5                        MSCP_PKT [.NEXT_PACKET, .J, 0, 16, 0] = 0;
;    6482  5
;    6483  5                    exitloop;                               ! DONE
;    6484  5
;    6485  4                    end;
;    6486  4
;    6487  3                end;
;    6488  3
;    6489  3            NEXT_PACKET = .NEXT_PACKET + 1;                 ! TRY NEXT PACKET IN RING
;    6490  3
;    6491  3            if .NEXT_PACKET gequ PKT_CNT
;    6492  3
;    6493  3            then
;    6494  3                NEXT_PACKET = 0;                            ! IF BEYOND ALL PACKETS, START AT THE TOP
;    6495  3
;    6496  2            end;
;    6497  2
;    6498  2        if (.index geq 0) and                               ! IF PACKET FOUND
;    6499  3           (.PKT_USE [.index] geq 0)
;    6500  2        then
;    6501  2
;    6502  3            begin
;    6503  3            MSCP_PKT [.index, MSGLEN] = SZ_GEN;             ! PACKET SIZE - ONLY ONLINE AND SCC CHANGE IT
;    6504  3            MSCP_PKT [.index, CREDITS] = 1;                 ! CREDIT SIZE
;    6505  3            NEXT_PKT_USE = .NEXT_PACKET + 1;                ! NEXT PACKET TO ALLOCATE
;    6506  3
;    6507  3            if.NEXT_PKT_USE gequ PKT_CNT
;    6508  3            then
;    6509  3                NEXT_PKT_USE = 0;                           ! CYCLE BACK TO BEGINNING IF AT END
;    6510  3
;    6511  2            end;
;    6512  2
;    6513  2        return .index;
;    6514  2
;    6515  1        end;


                                        .SBTTL  GET.PKT GLOBAL ROUTINES
000000  004137  000000G           GET.PKT::
                                    JSR     R1,$SAVE5               ;                                   6427
000004  162706  000006             SUB     #6,SP
000010  012704  177777             MOV     #-1,R4                  ; *,INDEX                           6439
000014  113766  000000G 000004     MOVB    NEXT.PKT.USE,4(SP)      ; *,NEXT.PACKET                     6448
000022  012766  000014  000002     MOV     #14,2(SP)               ; *,COUNT                           6450
000030  105016                  1$: CLRB    (SP)                    ; PACKET.OWNED                      6452
000032  005001                      CLR     R1                      ;                                   6454
000034  156601  000004             BISB    4(SP),R1                ; NEXT.PACKET,*
000040  105761  000000G            TSTB    PKT.USE(R1)
000044  002072                      BGE     7$
000046  013700  000000G            MOV     DCT.ADDR,R0             ;                                   6458
000052  016005  000004             MOV     4(R0),R5                ; *,RING.ADDR
```

```
000056  010146                          MOV     R1,-(SP)                ;                                       6462
000060  012746  000106                  MOV     #106,-(SP)
000064  004737  000000G                 JSR     PC,BL$MUL
000070  012702  000010                  MOV     #10,R2                  ; *,I                                   6460
000074  021560  000000G         2$:     CMP     (R5),MSCP.PKT(R0)       ; RING.ADDR,*                           6462
000100  001014                          BNE     3$
000102  012703  000002                  MOV     #2,R3                   ;                                       6463
000106  060503                          ADD     R5,R3                   ; RING.ADDR,*
000110  042703  077777                  BIC     #77777,R3
000114  020327  100000                  CMP     R3,#-100000
000120  001004                          BNE     3$
000122  112766  000001  000004          MOVB    #1,4(SP)                ; *,PACKET.OWNED                        6467
000130  000404                          BR      4$                      ;                                       6466
000132  062705  000004         3$:     ADD     #4,R5                   ; *,RING.ADDR                           6471
000136  005302                          DEC     R2                      ; I                                     6460
000140  001355                          BNE     2$
000142  032766  000001  000004  4$:     BIT     #1,4(SP)                ; *,PACKET.OWNED                        6473
000150  001027                          BNE     6$
000152  116661  000030  000000G         MOVB    30(SP),PKT.USE(R1)      ; CTLR,*                                6477
000160  010104                          MOV     R1,R4                   ; *,INDEX                               6478
000162  010116                          MOV     R1,(SP)                 ;                                       6481
000164  012746  000043                  MOV     #43,-(SP)
000170  004737  000000G                 JSR     PC,BL$MUL
000174  005726                          TST     (SP)+
000176  012702  000002                  MOV     #2,R2                   ; *,J                                   6480
000202  010003         5$:     MOV     R0,R3                   ;                                       6481
000204  060203                          ADD     R2,R3                   ; J,*
000206  006303                          ASL     R3
000210  005063  000000G                 CLR     MSCP.PKT(R3)
000214  005202                          INC     R2                      ; J                                     6480
000216  020227  000042                  CMP     R2,#42                  ; J,*
000222  003767                          BLE     5$
000224  022626                          CMP     (SP)+,(SP)+             ;                                       6476
000226  000414                          BR      9$
000230  022626         6$:     CMP     (SP)+,(SP)+             ;                                       6457
000232  105266  000004         7$:     INCB    4(SP)                   ; NEXT.PACKET                           6489
000236  126627  000004  000014          CMPB    4(SP),#14               ; NEXT.PACKET,*                         6491
000244  103402                          BLO     8$
000246  105066  000004                  CLRB    4(SP)                   ; NEXT.PACKET                           6494
000252  005366  000002         8$:     DEC     2(SP)                   ; COUNT                                 6450
000256  001264                          BNE     1$
000260  005704         9$:     TST     R4                      ; INDEX                                 6498
000262  002435                          BLT     11$
000264  105764  000000G                 TSTB    PKT.USE(R4)             ; *(INDEX)                              6499
000270  002432                          BLT     11$
000272  010446                          MOV     R4,-(SP)                ; INDEX,*                               6503
000274  012746  000106                  MOV     #106,-(SP)
000300  004737  000000G                 JSR     PC,BL$MUL
000304  012760  000040  000006G         MOV     #40,MSCP.PKT+6(R0)
000312  142760  000017  000010G         BICB    #17,MSCP.PKT+10(R0)     ;                                       6504
000320  152760  000001  000010G         BISB    #1,MSCP.PKT+10(R0)
000326  005000                          CLR     R0                      ;                                       6505
000330  156600  000010                  BISB    10(SP),R0               ; NEXT.PACKET,*
```

```
ZRQAM2          RD/RX EXERCISER                          4 Apr-1985 12:40:26    VAX-11 Bliss-16 V4.1 582                SEQ 0190
V02.2           GLOBAL ROUTINES                          4-Apr-1985 12:33:21    DISK$USER2:[POWERS.ZRQ]ZRQAG0.BL1;16    Page 173
                                                                                                                         (50)

000334  005200                                  INC     R0
000336  110037  000000G                         MOVB    R0,NEXT.PKT.USE
000342  120027  000014                          CMPB    R0,#14                  ; NEXT.PKT.USE,*                        6507
000346  103402                                  BLO     10$
000350  105037  000000G                         CLRB    NEXT.PKT.USE            ;                                       6509
000354  022626                          10$:    CMP     (SP)+,(SP)+             ;                                       6502
000356  010400                          11$:    MOV     R4,R0                   ; INDEX,*                               6439
000360  062706  000006                          ADD     #6,SP                   ;                                       6427
000364  000207                                  RTS     PC

; Routine Size:  123 words,     Routine Base:  $CODE$ + 5666
; Maximum stack depth per invocation:  13 words


;   6516  1
;   6517  1
```

```
;    6518  1
;    6519  1
;    6520  1     global routine PUT_PKT (index) : novalue =
;    6521  1
;    6522  1     !+
;    6523  1     !       THE MSCP PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;    6524  1     !       ROUTINE.
;    6525  1     !-
;    6526  1
;    6527  1
;    6528  2         begin
;    6529  2
;    6530  2
;    6531  2         local
;    6532  2             RING_ADDR : word,
;    6533  2             OWNER : word;
;    6534  2
;    6535  2         RING_ADDR = .DCT_ADDR [RR_BEG];                             ! ADDRESS IN FIRST RESPONSE RING
;    6536  2
;    6537  2         incr COUNT from 1 to (RRING_LEN + CRING_LEN) do              ! FOR EACH ADDRESS IN THE RINGS
;    6538  3             begin
;    6539  3
;    6540  3             if .MSCP_PKT [.index, PKT_LO] eqla ..RING_ADDR          ! IF ADDRESS MATCHES
;    6541  3
;    6542  3             then
;    6543  4                 begin
;    6544  4                 OWNER = .RING_ADDR + 2;                             ! ADDRESS OF OWNERSHIP WORD
;    6545  4                 .OWNER = ..OWNER and (not (ED_OWN)) and (not (ED_FLAG)) ;   ! GIVE OWNERSHIP TO HOST
;    6546  3                 end;
;    6547  3
;    6548  3
;    6549  3                 RING_ADDR = .RING_ADDR + 4;                         ! LOOK AT NEXT PACKET ADDRESS IN RING
;    6550  2                 end;
;    6551  2
;    6552  2
;    6553  2             PKT_USE [.index] = -1;
;    6554  2
;    6555  1             end;


                                        .SBTTL  PUT.PKT GLOBAL ROUTINES
000000  004137  000000G         PUT.PKT::
                                JSR     R1,$SAVE4                   ;                       6520
000004  013700  000000G                 MOV     DCT.ADDR,R0         ;                       6535
000010  016001  000004                  MOV     4(R0),R1            ; *,RING.ADDR
000014  016602  000014                  MOV     14(SP),R2           ; INDEX,*               6540
000020  010246                          MOV     R2,-(SP)
000022  012746  000106                  MOV     #106,-(SP)
000026  004737  000000G                 JSR     PC,BL$MUL
000032  012704  000010                  MOV     #10,R4              ; *,COUNT               6537
000036  026011  000000G         1$:     CMP     MSCP.PKT(R0),(R1)   ; *,RING.ADDR           6540
000042  001005                          BNE     2$
000044  012703  000002                  MOV     #2,R3               ; *,OWNER               6544
```

```
000050  060103                          ADD     R1,R3                   ; RING.ADDR,OWNER
000052  042713  140000                  BIC     #140000,(R3)            ; *,OWNER                       6545
000056  062701  000004          2$:     ADD     #4,R1                   ; *,RING.ADDR                   6549
000062  005304                          DEC     R4                      ; COUNT                         6537
000064  001364                          BNE     1$
000066  112762  000377  000000G         MOVB    #377,PKT.USE(R2)        ;                               6553
000074  022626                          CMP     (SP)+,(SP)+             ;                               6528
000076  000207                          RTS     PC                      ;                               6520

; Routine Size:  32 words,      Routine Base:  $CODE$ + 6254
; Maximum stack depth per invocation:  8 words


;   6556  1
;   6557  1
```

```
;    6558  1      routine PUTA_PKT (CTLR) : novalue =
;    6559  1
;    6560  1      !+
;    6561  1      !       THIS ROUTINE DEALLOCATES ALL MSCP PACKETS WHICH HAVE BEEN ALLOCATED
;    6562  1      !       TO A PARTICULAR CONTROLLER.
;    6563  1      !
;    6564  1      !       INPUTS:
;    6565  1      !               CTLR   CONTROLLER NUMBER
;    6566  1      !-
;    6567  1
;    6568  1          incr COUNT from 0 to (PKT_CNT - 1) do       ! FOR EACH ENTRY IN ALLOCATION TABLE
;    6569  1
;    6570  1              if .PKT_USE [.COUNT] eql .CTLR          ! IF PACKET IS ALLOCATED TO GIVEN CONTROLLER
;    6571  1              then
;    6572  1                  PKT_USE [.COUNT] = -1;              ! DEALLOCATE IT


                                        .SBTTL  PUTA.PKT GLOBAL ROUTINES
000000  010146                  PUTA.PKT:
                                        MOV     R1,-(SP)                        ;                                       6558
000002  005000                          CLR     R0                              ; COUNT                                 6568
000004  116001  000000G         1$:     MOVB    PKT.USE(R0),R1                  ; *(COUNT),*                            6570
000010  020166  000004                  CMP     R1,4(SP)                        ; *,CTLR
000014  001003                          BNE     2$
000016  112760  000377  000000G         MOVB    #377,PKT.USE(R0)                ; *,*(COUNT)                            6572
000024  005200                  2$:     INC     R0                              ; COUNT                                 6568
000026  020027  000013                  CMP     R0,#13                          ; COUNT,*
000032  003764                          BLE     1$
000034  012601                          MOV     (SP)+,R1                        ;                                       6558
000036  000207                          RTS     PC

; Routine Size:  16 words.      Routine Base:  $CODE$ + 6354
; Maximum stack depth per invocation:  2 words
```

```
;    6573  1     global routine GET_RETPKT (CTLR) =
;    6574  1
;    6575  1     !+
;    6576  1     !       THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
;    6577  1     !       FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;    6578  1     !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
;    6579  1     !       THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;    6580  1     !
;    6581  1     !       INPUTS:
;    6582  1     !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;    6583  1     !-
;    6584  1
;    6585  2         begin
;    6586  2
;    6587  2         local
;    6588  2             index : signed word initial (-1);       ! ASSUME NONE AVAILABLE
;    6589  2
;    6590  2         incr COUNT from 0 to (RP_CNT - 1) do        ! FOR EACH ENTRY IN TABLE
;    6591  2
;    6592  2             if .RP_USE [.COUNT] lss 0               ! IF FREE RETPKT IS FOUND
;    6593  2             then
;    6594  3                 begin
;    6595  3                 RP_USE [.COUNT] = .CTLR;            ! ALLOCATE RETURN PACKET TO CONTROLLER
;    6596  3                 index = .COUNT;
;    6597  3
;    6598  3                 incr J from 0 to (RP_LEN - 1) do    ! ZERO OUT RETPKT
;    6599  3                     RETPKT [.COUNT, .J, 0, 16, 0) = 0;
;    6600  3
;    6601  3                 exitloop;                           ! DONE
;    6602  2                 end;
;    6603  2
;    6604  2         return .index;                              ! RETURN PACKET INDEX (OR -1) TO CALLER
;    6605  1         end;


                                        .SBTTL  GET.RETPKT GLOBAL ROUTINES
000000  004137  000000G         GET.RETPKT::
                                JSR     R1,$SAVE4                       ;                       6573
000004  012703  177777                  MOV     #-1,R3                  ; *.INDEX               6585
000010  005001                          CLR     R1                      ; COUNT                 6590
000012  105761  000000G         1$:     TSTB    RP.USE(R1)              ; *(COUNT)              6592
000016  002025                          BGE     3$
000020  116661  000014  000000G        MOVB    14(SP),RP.USE(R1)       ; CTLR,*(COUNT)         6595
000026  010103                          MOV     R1,R3                   ; COUNT,INDEX           6596
000030  010146                          MOV     R1,-(SP)                ; COUNT,*               6599
000032  012746  000026                  MOV     #26,-(SP)
000036  004737  000000G                 JSR     PC,BL$MUL
000042  022626                          CMP     (SP)+,(SP)+
000044  005002                          CLR     R2                      ; J                     6598
000046  010004                  2$:     MOV     R0,R4                   ;                       6599
000050  060204                          ADD     R2,R4                   ; J,*
000052  006304                          ASL     R4
000054  005064  000000G                 CLR     RETPKT(R4)
```

```
000060  005202                          INC     R2                      ; J                             6598
000062  020227  000025                  CMP     R2,#25                  ; J,*
000066  003767                          BLE     2$
000070  000404                          BR      4$                      ;                               6594
000072  005201                  3$:     INC     R1                      ; COUNT                         6590
000074  020127  000007                  CMP     R1,#7                   ; COUNT,*
000100  003744                          BLE     1$
000102  010300                  4$:     MOV     R3,R0                   ; INDEX,*                       6585
000104  000207                          RTS     PC                      ;                               6573

; Routine Size:  35 words.      Routine Base:  $CODE$ + 6414
; Maximum stack depth per invocation:  8 words
```

```
;    6606  1      global routine PUT_RETPKT (index) : novalue =
;    6607  1
;    6608  1      !+
;    6609  1      !       THE RETURN PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;    6610  1      !       ROUTINE.
;    6611  1      !-
;    6612  1
;    6613  1          RP_USE [.index] = -1;


                                        .SBTTL  PUT.RETPKT GLOBAL ROUTINES
000000  016600  000002          PUT.RETPKT::
                                MOV     2(SP),R0                        ; INDEX,*                       6613
000004  112760  000377  000000G        MOVB    #377,RP.USE(R0)
000012  000207                  RTS     PC                              ;                               6606

; Routine Size:  6 words,       Routine Base:  $CODE$ + 6522
; Maximum stack depth per invocation:  0 words
```

```
;    6614  1
;    6615  1
;    6616  1     global routine GET_IO_BUFF (ADDR) : novalue =
;    6617  1
;    6618  1
;    6619  1     !+
;    6620  1     !       THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
;    6621  1     !       POOL.
;    6622  1     !
;    6623  1     !       INPUTS:
;    6624  1     !               ADDR - ADDRESS TO STORE THE 2-WORD BUFFER DESCRIPTOR
;    6625  1     !
;    6626  1     !       IMPLICIT INPUTS:
;    6627  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    6628  1     !
;    6629  1     !       OUTPUTS:
;    6630  1     !               THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
;    6631  1     !               WORDS AT "ADDR" AND "ADDR + 2". OTHERWISE, A ZERO IS RETURNED
;    6632  1     !               AT "ADDR" IF NO BUFFERS ARE AVAILABLE.
;    6633  1     !-
;    6634  1
;    6635  1
;    6636  1
;    6637  2         begin
;    6638  2         .ADDR = 0;                                          ! ASSUME FAILURE
;    6639  2
;    6640  2         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do   ! FOR EACH ENTRY IN BUFFER TABLE
;    6641  2
;    6642  2             if .BUFF_OWN [.COUNT] lss 0                     ! IF BUFFER IS FREE
;    6643  2
;    6644  2             then
;    6645  2
;    6646  3                 begin
;    6647  3                 BUFF_OWN [.COUNT] = .CCTLR;                 ! ALLOCATE BUFFER TO CONTROLLER
;    6648  3                 .ADDR = .BUFF_ADDR [.COUNT];                ! RETURN BUFFER DESCRIPTOR
;    6649  3                 exitloop;                                   ! DONE
;    6650  2                 end;
;    6651  2
;    6652  2
;    6653  1         end;                                                ! ROUTINE GET_IO_BUFF


                                        .SBTTL  GET.IO.BUFF GLOBAL ROUTINES
000000  010146                    GET.IO.BUFF::
                                        MOV     R1,-(SP)                ;                                   6616
000002  005076  000004                  CLR     @4(SP)                  ; ADDR                              6638
000006  005001                          CLR     R1                      ; COUNT                             6640
000010  105761  000000G         1$:     TSTB    BUFF.OWN(R1)            ; *(COUNT)                          6642
000014  002011                          BGE     2$
000016  113761  000000G 000000G         MOVB    CCTLR,BUFF.OWN(R1)      ; *,*(COUNT)                        6647
000024  010100                          MOV     R1,R0                   ; COUNT,*                           6648
000026  006300                          ASL     R0
000030  016076  000000G 000004          MOV     BUFF.ADDR(R0),@4(SP)    ; *,ADDR
```

```
000036  000404                          BR      3$                      ;                               6646
000040  005201                  2$:     INC     R1                      ; COUNT                         6640
000042  020127  000007                  CMP     R1,#7                   ; COUNT,#
000046  003760                          BLE     1$
000050  012601                  3$:     MOV     (SP)+,R1                ;                               6616
000052  000207                          RTS     PC

; Routine Size:  22 words,    Routine Base:  $CODE$ + 6536
; Maximum stack depth per invocation:  2 words


;   6654  1
;   6655  1
```

```
;    6656  1     global routine PUT_IO_BUFF (ADDR) : novalue =
;    6657  1
;    6658  1     !+
;    6659  1     !       THIS ROUTINE HANDLES THE DEALLOCATION OF AN I/O BUFFER, RETURNING IT
;    6660  1     !       TO THE BUFFER POOL.
;    6661  1     !
;    6662  1     !       INPUTS:
;    6663  1     !               ADDR - ADDRESS OF THE 2-WORD BUFFER DESCRIPTOR TO BE
;    6664  1     !                       DEALLOCATED
;    6665  1     !-
;    6666  1
;    6667  1         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;    6668  1
;    6669  1             if .BUFF_ADDR [.COUNT] eqla ..ADDR                      ! IF THIS IS THE BUFFER'S ENTRY
;    6670  1             then
;    6671  2                 begin
;    6672  2                 BUFF_OWN [.COUNT] = -1;                             ! DEALLOCATE BUFFER
;    6673  2                 exitloop;                                           ! DONE
;    6674  1                 end;


                                        .SBTTL  PUT.IO.BUFF GLOBAL ROUTINES
000000  010146                  PUT.IO.BUFF::
                                        MOV     R1,-(SP)                        ;                                       6656
000002  005001                          CLR     R1                              ; COUNT                                 6667
000004  010100                  1$:     MOV     R1,R0                           ; COUNT,*                               6669
000006  006300                          ASL     R0
000010  026076  000000G 000004          CMP     BUFF.ADDR(R0),04(SP)            ; *,ADDR
000016  001004                          BNE     2$
000020  112761  000377  000000G         MOVB    #377,BUFF.OWN(R1)               ; *,*(COUNT)                            6672
000026  000404                          BR      3$                              ;                                       6671
000030  005201                  2$:     INC     R1                              ; COUNT                                 6667
000032  020127  000007                  CMP     R1,#7                           ; COUNT,*
000036  003762                          BLE     1$
000040  012601                  3$:     MOV     (SP)+,R1                        ;                                       6656
000042  000207                          RTS     PC

; Routine Size:  18 words,      Routine Base:  $CODE$ + 6612
; Maximum stack depth per invocation:  2 words
```

```
;    6675  1      global routine PUTA_BUFF : novalue =
;    6676  1
;    6677  1      !+
;    6678  1      !       THIS ROUTINE DEALLOCATES ALL I/O BUFFERS WHICH HAVE BEEN ALLOCATED TO
;    6679  1      !       THE CURRENT CONTROLLER (CCTLR).
;    6680  1      !-
;    6681  1
;    6682  1          incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;    6683  1
;    6684  1              if .BUFF_OWN [.COUNT] eql .CCTLR                        ! IF THIS BUFFER ALLOCATED TO CURRENT CONTROLLER
;    6685  1              then
;    6686  1                  BUFF_OWN [.COUNT] = -1;                             ! DEALLOCATE IT


                                        .SBTTL  PUTA.BUFF GLOBAL ROUTINES
000000  010146                  PUTA.BUFF::
                                        MOV     R1,-(SP)                        ;                                       6675
000002  005000                          CLR     R0                              ; COUNT                                 6682
000004  116001  000000G         1$:     MOVB    BUFF.OWN(R0),R1                 ; *(COUNT),*                            6684
000010  020137  000000G                 CMP     R1,CCTLR
000014  001003                          BNE     2$
000016  112760  000377  000000G        MOVB    #377,BUFF.OWN(R0)               ; *,*(COUNT)                            6686
000024  005200                  2$:     INC     R0                              ; COUNT                                 6682
000026  020027  000007                  CMP     R0,#7                           ; COUNT,*
000032  003764                          BLE     1$
000034  012601                          MOV     (SP)+,R1                        ;                                       6675
000036  000207                          RTS     PC

; Routine Size:  16 words,      Routine Base:  $CODE$ + 6656
; Maximum stack depth per invocation:  2 words
```

```
;    6687  1     global routine OUT_IODQ =
;    6688  1
;    6689  1     !+
;    6690  1     !       THIS ROUTINE RETURNS TO THE CALLER THE NEXT RETPKT INDEX TO BE
;    6691  1     !       PROCESSED FROM THE I/O DONE QUEUE (IODQ). THE "OUT" POINTER TO THE
;    6692  1     !       QUEUE IS ALSO UPDATED.
;    6693  1     !
;    6694  1     !       INPUTS:
;    6695  1     !               NONE
;    6696  1     !
;    6697  1     !       OUTPUTS:
;    6698  1     !               THE INDEX OF THE NEXT RETPKT TO BE PROCESSED.
;    6699  1     !-
;    6700  1
;    6701  2         begin
;    6702  2
;    6703  2         local
;    6704  2             index : word;
;    6705  2
;    6706  2         index = .IODQ [.IODQ_OUT];                  ! GET NEXT RETPKT INDEX
;    6707  2         IODQ_OUT = .IODQ_OUT + 1;                   ! ADVANCE "OUT" POINTER
;    6708  2
;    6709  2         if .IODQ_OUT gequ IODQ_LEN                  ! IF BEYOND END OF QUEUE
;    6710  2         then
;    6711  2             IODQ_OUT = 0;                           ! SET POINTER TO BEGINNING OF QUEUE
;    6712  2
;    6713  2         return .index;                              ! RETURN INDEX TO CALLER
;    6714  1         end;


                                .SBTTL  OUT.IODQ GLOBAL ROUTINES
000000  013700  000000G     OUT.IODQ::
                            MOV     IODQ.OUT,R0                 ;                       6706
000004  116000  000000G             MOVB    IODQ(R0),R0         ; *,INDEX
000010  042700  177400              BIC     #177400,R0          ; *,INDEX
000014  005237  000000G             INC     IODQ.OUT            ;                       6707
000020  023727  000000G 000010      CMP     IODQ.OUT,#10        ;                       6709
000026  103402                      BLO     1$
000030  005037  000000G             CLR     IODQ.OUT            ;                       6711
000034  000207              1$:     RTS     PC                  ;                       6687

; Routine Size:  15 words,      Routine Base:  $CODE$ + 6716
; Maximum stack depth per invocation:  0 words
```

```
;    6715  1     global routine IN_IODQ (index) : novalue =
;    6716  1
;    6717  1     !+
;    6718  1     !       THIS ROUTINE INSERTS A RETURN PACKET INDEX INTO THE I/O DONE QUEUE, AND
;    6719  1     !       UPDATES THE IODQ_IN POINTER.
;    6720  1     !-
;    6721  1
;    6722  1         if ((.IODQ_IN + 1) eql .IODQ_OUT) or
;    6723  2            (.IODQ_IN - (IODQ_LEN - 1) eql .IODQ_OUT)
;    6724  1         then
;    6725  1             return
;    6726  1         else
;    6727  2             begin
;    6728  2             IODQ [.IODQ_IN] = .index;              ! LOAD INDEX INTO QUEUE
;    6729  2             IODQ_IN = .IODQ_IN + 1;                ! ADVANCE "IN" POINTER
;    6730  2
;    6731  2             if .IODQ_IN gequ IODQ_LEN              ! IF BEYOND END OF QUEUE
;    6732  2             then
;    6733  2                 IODQ_IN = 0;                       ! CYCLE BACK TO BEGINNING OF QUEUE
;    6734  2
;    6735  1             end;                                   ! IF IODQ IS NOT FULL


                                .SBTTL  IN.IODQ GLOBAL ROUTINES
000000  010146                IN.IODQ::
                                MOV     R1,-(SP)                        ;                       6715
000002  013701  000000G         MOV     IODQ.IN,R1                      ;                       6722
000006  010100                  MOV     R1,R0
000010  005200                  INC     R0
000012  020037  000000G         CMP     R0,IODQ.OUT
000016  001421                  BEQ     1$
000020  010100                  MOV     R1,R0                           ;                       6723
000022  162700  000007          SUB     #7,R0
000026  020037  000000G         CMP     R0,IODQ.OUT
000032  001413                  BEQ     1$                              ;                       6725
000034  116661  000004  000000G MOVB    4(SP),IODQ(R1)                  ; INDEX,*               6728
000042  005237  000000G         INC     IODQ.IN                         ;                       6729
000046  023727  000000G 000010  CMP     IODQ.IN,#10                     ;                       6731
000054  103402                  BLO     1$
000056  005037  000000G         CLR     IODQ.IN                         ;                       6733
000062  012601                1$:       MOV     (SP)+,R1                ;                       6715
000064  000207                  RTS     PC

; Routine Size:  27 words,      Routine Base:  $CODE$ + 6754
; Maximum stack depth per invocation:  2 words
```

```
;   6736  1
;   6737  1
;   6738  1     global routine DROP_CTLR (CTLR, REASON) : novalue =
;   6739  1
;   6740  1
;   6741  1
;   6742  1     !+
;   6743  1     !       THIS ROUTINE DROPS ALL UNITS ASSOCIATED WITH THE CONTROLLER DESIGNATED
;   6744  1     !       BY "CTLR". THE REASON FOR DROPPING THE DEVICE IS LOADED INTO THE DUR
;   6745  1     !       VECTOR FOR EACH ATTACHED UNIT. THIS DATA IS THEN USED BY THE DROP UNIT
;   6746  1     !       SECTION.
;   6747  1     !-
;   6748  1
;   6749  1
;   6750  2         begin
;   6751  2
;   6752  2         local
;   6753  2             UNIT;
;   6754  2
;   6755  2         incr N from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do       ! FOR EACH UNIT
;   6756  2
;   6757  2             if .CST [.CTLR, .N * OF_DATA, D_PRES] eql PRESENT                                       ! IF CONFIGURED
;   6758  2             then
;   6759  3                 begin
;   6760  3                 UNIT = .CST [.CTLR, .N * OF_DATA, D_UNIT];                                          ! DRS UNIT NUMBER
;   6761  3                 DUR [.UNIT] = .REASON;                                                              ! DROP REASON
;   6762  3                 DODU (.UNIT);                                                                       ! DROP UNIT
;   6763  2                 end;
;   6764  2
;   6765  1         end;


                                        .SBTTL  DROP.CTLR GLOBAL ROUTINES
000000  004137  000000G          DROP.CTLR::
                                        JSR     R1,$SAVE3                       ;                       6738
000004  016646  000014                  MOV     14(SP),-(SP)                    ; CTLR,*                6757
000010  012746  000053                  MOV     #53,-(SP)
000014  004737  000000G                 JSR     PC,BL$MUL
000020  010003                          MOV     R0,R3
000022  012702  000003                  MOV     #3,R2                           ; *,N                   6755
000026  010300                  1$:     MOV     R3,R0                           ;                       6757
000030  060200                          ADD     R2,R0                           ; N,*
000032  006300                          ASL     R0
000034  032760  040000  000000G         BIT     #40000,CST(R0)
000042  001412                          BEQ     2$
000044  016001  000000G                 MOV     CST(R0),R1                      ; *,UNIT                6760
000050  000301                          SWAB    R1                              ; UNIT
000052  042701  177760                  BIC     #177760,R1                      ; *,UNIT
000056  116661  000016  000000G         MOVB    16(SP),DUR(R1)                  ; REASON,*(UNIT)        6761
000064  010100                          MOV     R1,R0                           ; UNIT,*                6762
000066  104451                          TRAP    51
000070  062702  000012          2$:     ADD     #12,R2                          ; *,N                   6755
000074  020227  000041                  CMP     R2,#41                          ; N,*
```

```
000100  003752                          BLE     1$
000102  022626                          CMP     (SP)+,(SP)+                     ;                       6750
000104  000207                          RTS     PC                              ;                       6738

; Routine Size:  35 words,      Routine Base:  $CODE$ + 7042
; Maximum stack depth per invocation:  8 words


;   6766  1
;   6767  1
```

```
;    6768  1      global routine DRV_CTLERR (CTLR) : novalue =
;    6769  1
;    6770  1      !+
;    6771  1      !       THIS ROUTINE IS CALLED BY DRV_TIMCHK AND FATAL_ERROR WHENEVER AN
;    6772  1      !       UNRECOVERABLE CONTROLLER ERROR HAS BEEN DETECTED. ITS PURPOSE IS TO
;    6773  1      !       CLEAN UP ALL CONTROLLER-RELATED DATA IN THE "DRIVER" PORTION OF THE
;    6774  1      !       PROGRAM. THIS INCLUDES MARKING THE CONTROLLER OFFLINE, CLEARING THE
;    6775  1      !       C-RING COUNT, AND DEALLOCATING MSCP PACKETS DESCRIBED IN THE RESPONSE
;    6776  1      !       RING.
;    6777  1      !
;    6778  1      !       INPUTS:
;    6779  1      !               CTLR - DYING CONTROLLER NUMBER
;    6780  1      !-
;    6781  1
;    6782  2          begin
;    6783  2
;    6784  2          local
;    6785  2              D_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS);  ! CONTROLLER'S DCT ADDRESS
;    6786  2
;    6787  2          D_ADDR = DCT + (.CTLR * DCT_LEN * 2);                       ! GET CONTROLLER'S DCT ADDR
;    6788  2          D_ADDR [WORD0] = OFFLINE;                                   ! MARK DCT OFFLINE AND CLEAR CRING_CNT
;    6789  2          PUTA_PKT (.CTLR);                                           ! RELEASE ALL PACKETS ALLOCATED TO CONTROLLER
;    6790  2          DROP_CTLR (.CTLR, DU_CFATAL);                               ! DROP ALL UNITS ON THE CONTROLLER
;    6791  1          end;                                                        ! ROUTINE DRV_CTLERR


                                        .SBTTL  DRV.CTLERR GLOBAL ROUTINES
000000  010146                    DRV.CTLERR::
                                        MOV     R1,-(SP)                        ;
000002  016601  000004                  MOV     4(SP),R1                        ; CTLR,*                6768
000006  010146                          MOV     R1,-(SP)                                                6787
000010  012746  000022                  MOV     #22,-(SP)
000014  004737  000000G                 JSR     PC,BL$MUL
000020  062700  000000G                 ADD     #DCT,R0
000024  005010                          CLR     (R0)                            ; D.ADDR
000026  010116                          MOV     R1,(SP)                         ;                       6788
000030  004737  006354'                 JSR     PC,PUTA.PKT                                             6789
000034  010116                          MOV     R1,(SP)                         ;
000036  012746  000006                  MOV     #6,-(SP)                                                6790
000042  004737  007042'                 JSR     PC,DROP.CTLR
000046  062706  000006                  ADD     #6,SP
000052  012601                          MOV     (SP)+,R1                        ;                       6782
000054  000207                          RTS     PC                              ;                       6768

; Routine Size:  23 words,      Routine Base:  $CODE$ + 7150
; Maximum stack depth per invocation:  5 words
```

```
;    6792  1      global routine SEND (index) =
;    6793  1
;    6794  1      !+
;    6795  1      !       IF THE CURRENT RDRX IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
;    6796  1      !       ROUTINE "SENDS" A COMMAND TO THE RDRX BY LOADING THE PACKET
;    6797  1      !       DESCRIPTOR OF AN MSCP PACKET INTO THE COMMAND RING AND READING THE
;    6798  1      !       DEVICE'S IP REGISTER.  IF THE
;    6799  1      !       CURRENT RDRX IN NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
;    6800  1      !       THE CALLER, AND NO ACTION IS TAKEN.
;    6801  1      !
;    6802  1      !       INPUTS:
;    6803  1      !               INDEX - INDEX OF MSCP PACKET CONTAINING THE COMMAND TO
;    6804  1      !                       BE SENT
;    6805  1      !
;    6806  1      !       IMPLICIT INPUTS:
;    6807  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    6808  1      !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;    6809  1      !-
;    6810  1
;    6811  2          begin
;    6812  2
;    6813  2          local
;    6814  2              SLOT_ADDR,
;    6815  2              TEMP : word,
;    6816  2              CUR_PRIORITY : word;
;    6817  2
;    6818  2          if (.DCT_ADDR [CRING_CNT] lssu CRING_LEN) and                   ! IF CRING IS NOT FULL AND
;    6819  3             ((.DCT_ADDR [STAT] eql ONLINE) or                            !   IF DEVICE IS ONLINE OR
;    6820  3              (.MSCP_PKT [.index, OPCODE] eql OP_SCC))                    !   IT IS A SET-CTRL-CHAR COMMAND
;    6821  2          then
;    6822  2
;    6823  4              if (not ((.MSCP_PKT [.index, OPCODE] eql OP_ACC) or (.MSCP_PKT [.index, OPCODE] eql OP_ONL) or
;    6824  4                       (.MSCP_PKT [.index, OPCODE] eql OP_RD) or (.MSCP_PKT [.index, OPCODE] eql OP_SCC) or
;    6825  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_SDD) OR                  !ZZZ
;    6826  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_RCD) OR                  !ZZZ
;    6827  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_GDS) OR                  !ZZZ
;    6828  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ELP) OR                  !ZZZ
;    6829  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ABT) OR                  !ZZZ
;    6830  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ESP) OR                  !ZZZ
;    6831  3                       (.MSCP_PKT [.index, OPCODE] eql OP_WRT)))
;    6832  2              then
;    6833  3                  begin
;    6834  3                  PRINTF (DBM107, .MSCP_PKT [.index, OPCODE]);
;    6835  3                  return FAILURE;
;    6836  3                  end
;    6837  2              else
;    6838  3                  begin
;    6839  3
;    6840  3                  do
;    6841  3                      BREAK                                               ! LOOP TILL CREDIT BALANCE POSITIVE
;    6842  4                  until ((.MSCP_PKT [.index, CMD_TYPE] eql IMM_CMD) and
;    6843  3                         (.CREDIT_BAL gequ 1)) or
;    6844  3                        (.CREDIT_BAL gtru 1);
```

```
;    6845  3
;    6846  3                MSCP_PKT [.index, CRN_LO] = (CRN_LOW = .CRN_LOW + 1);       ! ASSIGN CMD REF NUM
;    6847  3
;    6848  3                if .CRN_LOW eql 0
;    6849  3                then
;    6850  3                    CRN_HIGH = .CRN_HIGH + 1;                               ! CMD REF NUM (HIGH ORDER)
;    6851  3
;    6852  3                MSCP_PKT [.index, CRN_HI] = .CRN_HIGH;                      !
;    6853  3                SLOT_ADDR = .DCT_ADDR [CR_NEXT];                            ! ADDR OF NEXT COMMAND SLOT
;    6854  3
;    6855  3                do                                                          ! WAIT TILL NEXT SLOT HOST OWNED
;    6856  3                    BREAK                                                   !
;    6857  3                until ((.(.SLOT_ADDR + 2) and ED_OWN) eql 0);               !
;    6858  3
;    6859  3                GETPRI (CUR_PRIORITY);                                      ! NO INTERRUPTS WHILE POINTERS UPDATED
;    6860  3      !ZZZ      SETPRI (PRI04);                                             !
;    6861  3                SETPRI (.BRLEVEL);                                          !          ZZZ
;    6862  3
;    6863  3                .SLOT_ADDR = .MSCP_PKT [.index, PKT_LO];                    ! LOAD BUFF DESC (LO) INTO COMMAND SLOT
;    6864  3                SLOT_ADDR = .SLOT_ADDR + 2;                                 ! ADVANCE TO NEXT WORD
;    6865  3                .SLOT_ADDR = .MSCP_PKT [.index, PKT_HI];                    ! LOAD BUFF DESC (HI) INTO COMMAND SLOT
;    6866  3                .SLOT_ADDR = ..SLOT_ADDR and (not (ED_FLAG));               ! CLEAR INTERRUPT FLAG IN CASE SET
;    6867  3                .SLOT_ADDR = ..SLOT_ADDR or ED_OWN;                         ! GIVE OWNERSHIP TO CONTROLLER
;    6868  3                SLOT_ADDR = .SLOT_ADDR + 2;                                 ! ADVANCE TO NEXT COMMAND SLOT
;    6869  3
;    6870  3                if .SLOT_ADDR gtra .DCT_ADDR [CR_END]                       ! IF BEYOND END OF CRING
;    6871  3                then
;    6872  3                    SLOT_ADDR = .DCT_ADDR [CR_BEG];                         ! CYCLE BACK TO BEGINNING
;    6873  3
;    6874  3                DCT_ADDR [CR_NEXT] = .SLOT_ADDR;                            ! RESTORE CR_NEXT POINTER IN DCT
;    6875  3                DCT_ADDR [CRING_CNT] = .DCT_ADDR [CRING_CNT] + 1;           ! INCR # OF COMMANDS IN CRING
;    6876  4                IF (.MSCP_PKT [.INDEX, CONNID] EQL CID_MSCP)                !IF MSCP COMMAND                ZZZ
;    6877  3                THEN (CREDIT_BAL = .CREDIT_BAL - 1);                        !DECR CREDIT BALANCE            ZZZ
;    6878  3                TEMP = .RDRX_ADDR [RCIP, RC_ALL];                           ! READ IP TO FORCE PORT TO POLL
;    6879  3                SETPRI (.CUR_PRIORITY);                                     ! LOWER PRIORITY
;    6880  3                return SUCCESS;
;    6881  3                end
;    6882  3
;    6883  2            else
;    6884  2                return FAILURE;                                             ! IF DEVICE IS NOT ONLINE
;    6885  2
;    6886  1            end;                                                            ! ROUTINE SEND


                                                .SBTTL  SEND GLOBAL ROUTINES
000000  004137  000000G                 SEND::  JSR     R1,$SAVE3                       ;                               6792
000004  005746                                 TST     -(SP)
000006  127727  000000G 000004                 CMPB    @DCT.ADDR,#4                    ;                               6818
000014  103100                                 BHIS    2$
000016  005777  000000G                        TST     @DCT.ADDR                       ;                               6819
000022  100413                                 BMI     1$
000024  016646  000014                         MOV     14(SP),-(SP)                    ; INDEX,*                       6820
000030  012746  000106                         MOV     #106,-(SP)
```

```
000034  004737  000000G                        JSR     PC,BL$MUL
000040  022626                                  CMP     (SP)+,(SP)+
000042  126027  000022G 000004                 CMPB    MSCP.PKT+22(R0),#4
000050  001167                                  BNE     10$
000052  016646  000014                1$:       MOV     14(SP),-(SP)            ; INDEX,*                       6823
000056  012746  000106                          MOV     #106,-(SP)
000062  004737  000000G                        JSR     PC,BL$MUL
000066  010002                                  MOV     R0,R2
000070  022626                                  CMP     (SP)+,(SP)+
000072  005000                                  CLR     R0
000074  156200  000022G                        BISB    MSCP.PKT+22(R2),R0
000100  020027  000020                        CMP     R0,#20
000104  001445                                  BEQ     3$
000106  020027  000011                        CMP     R0,#11
000112  001442                                  BEQ     3$
000114  020027  000041                        CMP     R0,#41                  ;                               6824
000120  001437                                  BEQ     3$
000122  020027  000004                        CMP     R0,#4
000126  001434                                  BEQ     3$
000130  020027  000005                        CMP     R0,#5                   ;                               6826
000134  001431                                  BEQ     3$
000136  020027  000001                        CMP     R0,#1                   ;                               6827
000142  001426                                  BEQ     3$
000144  020027  000003                        CMP     R0,#3                   ;                               6828
000150  001423                                  BEQ     3$
000152  020027  000006                        CMP     R0,#6                   ;                               6829
000156  001420                                  BEQ     3$
000160  020027  000002                        CMP     R0,#2                   ;                               6830
000164  001415                                  BEQ     3$
000166  020027  000042                        CMP     R0,#42                  ;                               6831
000172  001412                                  BEQ     3$
000174  010046                                  MOV     R0,-(SP)                ;                               6834
000176  012746  000000G                        MOV     #DBM107,-(SP)
000202  012746  000002                        MOV     #2,-(SP)
000206  010600                                  MOV     SP,R0                   ; SP,*
000210  104417                                  TRAP    17
000212  062706  000006                        ADD     #6,SP                   ;                               6833
000216  000504                      2$:       BR      10$                     ;                               6823
000220  104422                      3$:       TRAP    22                      ;                               6840
000222  105762  000004G                        TSTB    MSCP.PKT+4(R2)          ;                               6842
000226  001003                                  BNE     4$
000230  005737  000000G                        TST     CREDIT.BAL              ;                               6843
000234  001004                                  BNE     5$
000236  023727  000000G 000001        4$:       CMP     CREDIT.BAL,#1           ;                               6844
000244  101765                                  BLOS    3$
000246  013700  000000G                4$:       MOV     CRN.LOW,R0              ;                               6846
000252  005200                                  INC     R0
000254  010037  000000G                        MOV     R0,CRN.LOW
000260  010062  000012G                        MOV     R0,MSCP.PKT+12(R2)
000264  001002                                  BNE     6$                      ;                               6848
000266  005237  000000G                        INC     CRN.HIGH                ;                               6850
000272  013762  000000G 000014G        6$:       MOV     CRN.HIGH,MSCP.PKT+14(R2)  ;                             6852
000300  013700  000000G                        MOV     DCT.ADDR,R0             ;                               6853
```

```
000304  016001  000020                        MOV     20(R0),R1               ; *,SLOT.ADDR
000310  104422                          7$:    TRAP    22                      ;                       6855
000312  032761  100000  000002                BIT     #-100000,2(R1)          ; *,*(SLOT.ADDR)        6857
000320  001373                                BNE     7$
000322  104440                                TRAP    40                      ;                       6859
000324  010003                                MOV     R0,R3                   ; *,CUR.PRIORITY
000326  013700  000000G                       MOV     @RLEVEL,R0              ;                       6861
000332  104441                                TRAP    41
000334  016221  000000G                       MOV     MSCP.PKT(R2),(R1)+      ; *,SLOT.ADDR           6863
000340  016211  000002G                       MOV     MSCP.PKT+2(R2),(R1)     ; *,SLOT.ADDR           6865
000344  042711  040000                        BIC     #40000,(R1)             ; *,SLOT.ADDR           6866
000350  052721  100000                        BIS     #100000,(R1)+           ; *,SLOT.ADDR           6867
000354  013700  000000G                       MOV     DCT.ADDR,R0             ;                       6870
000360  020160  000012                        CMP     R1,12(R0)               ; SLOT.ADDR,*
000364  101402                                BLOS    8$
000366  016001  000010                        MOV     10(R0),R1               ; *,SLOT.ADDR           6872
000372  010160  000020                  8$:   MOV     R1,20(R0)               ; SLOT.ADDR,*           6874
000376  105210                                INCB    (R0)                    ;                       6875
000400  105762  000011G                       TSTB    MSCP.PKT+11(R2)         ;                       6876
000404  001002                                BNE     9$
000406  005337  000000G                       DEC     @CREDIT.BAL             ;                       6877
000412  017716  000000G                  9$:   MOV     @RDRX.ADDR,(SP)         ; *,RC.REG              6878
000416  010300                                MOV     R3,R0                   ; CUR.PRIORITY,*        6879
000420  104441                                TRAP    41
000422  012700  000001                        MOV     #1,R0                   ;                       6823
000426  000401                                BR      11$                     ;                       6884
000430  005000                           10$:   CLR     R0
000432  005726                           11$:   TST     (SP)+                   ;                       6792
000434  000207                                RTS     PC

; Routine Size:  143 words,     Routine Base:  $CODE$ + 7226
; Maximum stack depth per invocation:  10 words
```

```
ZRQAM2          RD/RX EXERCISER                                 4-Apr-1985 12:40:26    VAX-11 Bliss-16 V4.1-582              SEQ 0210
V02.2           GLOBAL ROUTINES                                 4-Apr-1985 12:33:21    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;16  Page 193
                                                                                                                              (63)

;    6887  1     global routine WAIT : novalue =
;    6888  1
;    6889  1     !+
;    6890  1     !       THE PURPOSE OF THIS ROUTINE IS TO KILL TIME UNTIL AN RDRX INTERRUPT
;    6891  1     !       RESULTS IN A RETURN PACKET INDEX BEING DEPOSITED INTO THE I/O DONE
;    6892  1     !       QUEUE (IODQ).
;    6893  1     !-
;    6894  1
;    6895  1         do
;    6896  1             BREAK                                   ! BREAK FOR ACT
;    6897  1         until .IODQ_IN neq .IODQ_OUT;


                                        .SBTTL  WAIT GLOBAL ROUTINES
000000  104422                      WAIT::
000000                              1$:     TRAP    22                      ;                       6895
000002  023737  000000G 000000G            CMP     IODQ.IN,IODQ.OUT        ;                       6897
000010  001773                            BEQ     1$
000012  000207                            RTS     PC                      ;                       6887

; Routine Size:  6 words,       Routine Base:  $CODE$ + 7664
; Maximum stack depth per invocation:  2 words


;    6898  1
```

```
;    6899  1
;    6900  1     GLOBAL ROUTINE MODULAS (LO_LIMIT, HI_LIMIT) =                                  !ZZZ
;    6901  1                                                                                    !ZZZ
;    6902  1     !+ THE PURPOSE OF THIS ROUTINE IS TO GET A RANDOM NUMBER BETWEEN               !ZZZ
;    6903  1     !  THE LOW AND HIGH LIMITS. THIS SHOULD WORK FOR A 16 BIT WORD.                !ZZZ
;    6904  1     !- THE "MOD" FUNC ONLY WORKS ON 15 BITS.                                       !ZZZ
;    6905  1                                                                                    !ZZZ
;    6906  2     BEGIN                                                                          !ZZZ
;    6907  2     OWN     X : WORD;                   !VARIABLE FOR RANDOM WD TABLE              !ZZZ
;    6908  2     LOCAL   ANSWER : UNSIGNED WORD,     !FINAL ANSWER                              !ZZZ
;    6909  2             SAVESZ : UNSIGNED WORD,     !SAVES SIZE OF WINDOW                      !ZZZ
;    6910  2             SIZE : UNSIGNED WORD;       !SIZE OF WINDOW                            !ZZZ
;    6911  2                                                                                    !ZZZ
;    6912  2                                                                                    !ZZZ
;    6913  2     X = .X + 1;                                                                    !ZZZ
;    6914  2     IF .X GEQ RDM_LEN                                                              !ZZZ
;    6915  2     THEN X = 0;                         !KEEP ROTATING RANDOM NUMBERS USED         !ZZZ
;    6916  2
                        !ZZZ
;    6917  2     SIZE = .HI_LIMIT - .LO_LIMIT;                                                  !ZZZ
;    6918  2     SAVESZ = .HI_LIMIT - .LO_LIMIT;                                                !ZZZ
;    6919  3     IF (.SIZE LEQU %O'077777')          !IF BIT 15 NOT SET                         !ZZZ
;    6920  3     THEN ANSWER = ((.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1))                  !ZZZ
;    6921  3                                         !ONLY 15 BIT WD, SO TAKE RANDOM SAMPLE     !ZZZ
;    6922  2     ELSE                                !16 BIT WD                                 !ZZZ
;    6923  3             BEGIN                                                                  !ZZZ
;    6924  3             SIZE = .SIZE ^ -1;          !MAKES SIZE A 15 BIT LENGTH, OR DIV BY 2   !ZZZ
;    6925  3             ANSWER = (.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1);                !ZZZ
;    6926  3                                         !GIVES 15 BIT RANDOM NUMBER                !ZZZ
;    6927  3             ANSWER = .ANSWER ^ 1;       !BUILD UP TO REGULAR SIZE                  !ZZZ
;    6928  3             ANSWER = .ANSWER + (.RANDOM [.X + 1] AND 1);                           !ZZZ
;    6929  3                                         !RANDOMLY FILL BIT 0                       !ZZZ
;    6930  4             IF (.ANSWER GTRU SAVESZ) !ITS POSSIBLE TO BE 1 LARGER THAN SIZE        !ZZZ
;    6931  3             THEN ANSWER = .SAVESZ;  !SO CHECK.                                     !ZZZ
;    6932  2             END;                                                                   !ZZZ
.    6933  2     RETURN .ANSWER;
;    6934  1     END;                                !END MODULAS ROUTINE                       !ZZZ


007700                                  X:      .BLKW   1


                                                .SBTTL  MODULAS GLOBAL ROUTINES
000000  004137  000000G                 MODULAS::
                                                JSR     R1,$SAVE2                       ;                       6900
000004  005746                                 TST     -(SP)
000006  005237  007700'                         INC     X                               ;                       6913
000012  023727  007700'  000020                 CMP     X,#20                           ;                       6914
000020  002402                                 BLT     1$
000022  005037  007700'                         CLR     X                               ;                       6915
000026  016600  000012                 1$:     MOV     12(SP),R0                       ; HI.LIMIT,*            6917
000032  166600  000014                         SUB     14(SP),R0                       ; LO.LIMIT,*
000036  010001                                 MOV     R0,R1                           ; *,SIZE
000040  010016                                 MOV     R0,(SP)                         ; *,SAVESZ              6918
```

```
000042  013700  007700'                    MOV     X,R0                    ;                               6920
000046  006300                              ASL     R0
000050  020127  077777                      CMP     R1,#77777               ; SIZE,*                        6919
000054  101011                              BHI     2$
000056  016046  000000G                     MOV     RANDOM(R0),-(SP)        ;                               6920
000062  042716  100000                      BIC     #100000,(SP)
000066  010146                              MOV     R1,-(SP)                ; SIZE,*
000070  005216                              INC     (SP)
000072  004737  000000G                     JSR     PC,BL$MOD
000076  000431                              BR      3$                      ;                               6919
000100  006201                      2$:     ASR     R1                      ; SIZE                          6924
000102  016046  000000G                     MOV     RANDOM(R0), (SP)        ;                               6925
000106  042716  100000                      BIC     #100000,(SP)
000112  010146                              MOV     R1,-(SP)                ; SIZE,*
000114  005216                              INC     (SP)
000116  004737  000000G                     JSR     PC,BL$MOD
000122  006300                              ASL     R0                      ; ANSWER                        6927
000124  013701  007700'                    MOV     X,R1                    ;                               6928
000130  006301                              ASL     R1
000132  116102  000002G                     MOVB    RANDOM+2(R1),R2
000136  042702  177776                      BIC     #177776,R2
000142  060200                              ADD     R2,R0                   ; *,ANSWER
000144  012701  000004                      MOV     #4,R1                   ;                               6930
000150  060601                              ADD     SP,R1                   ; SAVESZ,*
000152  020001                              CMP     R0,R1                   ; ANSWER,*
000154  101402                              BLOS    3$
000156  016600  000004                      MOV     4(SP),R0                ; SAVESZ,ANSWER                 6931
000162  062706  000006              3$:     ADD     #6,SP                   ;                               6900
000166  000207                              RTS     PC

; Routine Size:  60 words,    Routine Base:  $CODE$ + 7702
; Maximum stack depth per invocation:  7 words
```

```
;    6935  1     %sbttl 'ERROR MESSAGE SUBROUTINES'
;    6936  1
;    6937  1     routine EMS_SA : novalue =
;    6938  1
;    6939  1     !+
;    6940  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SA_REG" WHICH CONTAINS
;    6941  1     !       THE CONTENTS OF THE SA REGISTER.
;    6942  1     !-
;    6943  1
;    6944  2         begin
;    6945  2
;    6946  2         if .SA_REG eql %o'177777'                                   ! IF CONTROLLER TIME OUT
;    6947  2         then
;    6948  3             begin
;    6949  3             PRINTX (CRLF);
;    6950  3             PRINTX (ASTERISK);
;    6951  3             PRINTX (.CNTR_ERR [0]);
;    6952  3             end
;    6953  2         else
;    6954  2
;    6955  2             if (.SA_REG and %o'003777') lequ 22                     ! IF GENERIC CONTROLLER ERROR
;    6956  2             then
;    6957  3                 begin
;    6958  3                 PRINTX (CRLF);
;    6959  3                 PRINTX (ASTERISK);
;    6960  3                 PRINTX (.CNTR_ERR [.SA_REG and %o'003777']);
;    6961  3                 end
;    6962  2             else
;    6963  2
;    6964  2                 if ((.SA_REG and %o'003777')  400) lequ 6           ! IF RDRX SPECIFIC CONTROLLER ERROR
;    6965  2                 then
;    6966  3                     begin
;    6967  3                     PRINTX (CRLF);
;    6968  3                     PRINTX (ASTERISK);
;    6969  3                     PRINTX (.RDRX_ERR [(.SA_REG and %o'003777') - 400]);
;    6970  3                     end
;    6971  2                 else
;    6972  2                     PRINTX (EX_SA, .SA_REG);                        ! JUST PRINT CONTENTS OF SA
;    6973  2
;    6974  2         EMS_TIM ();                                                 ! TIME
;    6975  1         end;


                                        .SBTTL  EMS.SA ERROR MESSAGE SUBROUTINES
000000  010146                    EMS.SA: MOV     R1,-(SP)                    ;                       6937
000002  013701  000000G                 MOV     SA.REG,R1                   ;                       6946
000006  020127  177777                  CMP     R1,#-1
000012  001023                          BNE     1$
000014  012746  000000G                 MOV     #CRLF,-(SP)                 ;                       6949
000020  012746  000001                  MOV     #1,-(SP)
000024  010600                          MOV     SP,R0                       ; SP,*
000026  104415                          TRAP    15
000030  012716  000000G                 MOV     #ASTERISK,(SP)              ;                       6950
```

```
000034  012746  000001                        MOV     #1,-(SP)
000040  010600                                MOV     SP,R0                   ; SP,*
000042  104415                                TRAP    15
000044  013716  000000G                       MOV     CNTR.ERR,(SP)           ;                       6951
000050  012746  000001                        MOV     #1,-(SP)
000054  010600                                MOV     SP,R0                   ; SP,*
000056  104415                                TRAP    15
000060  000475                                BR      3$                      ;                       6948
000062  010100                          1$:   MOV     R1,R0                   ;                       6955
000064  042700  174000                        BIC     #174000,R0
000070  020027  000026                        CMP     R0,#26
000074  101030                                BHI     2$
000076  012746  000000G                       MOV     #CRLF,-(SP)             ;                       6958
000102  012746  000001                        MOV     #1,-(SP)
000106  010600                                MOV     SP,R0                   ; SP,*
000110  104415                                TRAP    15
000112  012716  000000G                       MOV     #ASTERISK,(SP)          ;                       6959
000116  012746  000001                        MOV     #1,-(SP)
000122  010600                                MOV     SP,R0                   ; SP,*
000124  104415                                TRAP    15
000126  013700  000000G                       MOV     SA.REG,R0               ;                       6960
000132  042700  174000                        BIC     #174000,R0
000136  006300                                ASL     R0
000140  016016  000000G                       MOV     CNTR.ERR(R0),(SP)
000144  012746  000001                        MOV     #1,-(SP)
000150  010600                                MOV     SP,R0                   ; SP,*
000152  104415                                TRAP    15
000154  000437                                BR      3$                      ;                       6957
000156  010100                          2$:   MOV     R1,R0                   ;                       6964
000160  042700  174000                        BIC     #174000,R0
000164  162700  000620                        SUB     #620,R0
000170  020027  000006                        CMP     R0,#6
000174  101031                                BHI     4$
000176  012746  000000G                       MOV     #CRLF,-(SP)             ;                       6967
000202  012746  000001                        MOV     #1,-(SP)
000206  010600                                MOV     SP,R0                   ; SP,*
000210  104415                                TRAP    15
000212  012716  000000G                       MOV     #ASTERISK,(SP)          ;                       6968
000216  012746  000001                        MOV     #1,-(SP)
000222  010600                                MOV     SP,R0                   ; SP,*
000224  104415                                TRAP    15
000226  013700  000000G                       MOV     SA.REG,R0               ;                       6969
000232  042700  174000                        BIC     #174000,R0
000236  006300                                ASL     R0
000240  016016  176340G                       MOV     RDRX.ERR-1440(R0),(SP)
000244  012746  000001                        MOV     #1,-(SP)
000250  010600                                MOV     SP,R0                   ; SP,*
000252  104415                                TRAP    15
000254  005726                          3$:   TST     (SP)+                   ;                       6966
000256  000407                                BR      5$                      ;                       6964
000260  010146                          4$:   MOV     R1,-(SP)                ;                       6972
000262  012746  000000G                       MOV     #EX.SA,-(SP)
000266  012746  000002                        MOV     #2,-(SP)
```

```
000272  010600                          MOV     SP,R0                           ; SP,*
000274  104415                          TRAP    15
000276  004737  000000V          5$:     JSR     PC,EMS.TIM                      ;                               6974
000302  062706  000006                  ADD     #6,SP                           ;                               6944
000306  012601                          MOV     (SP)+,R1                        ;                               6937
000310  000207                          RTS     PC

; Routine Size:  101 words,     Routine Base:  $CODE$ + 10072
; Maximum stack depth per invocation:  7 words
```

```
;    6976  1     routine EMS_SBC : novalue =
;    6977  1
;    6978  1     !+
;    6979  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SB CODE" (SUB-CODE) IF
;    6980  1     !       EITHER THE STATUS CODE (ST_CODE) OR THE SUB-CODE IS NON ZERO. (A
;    6981  1     !       NON-ZERO SUB-CODE ALWAYS HAS SIGNIFICANCE, WHEREAS A ZERO SUB-CODE ONLY
;    6982  1     !       HAS MEANING WITH A NON ZERO STATUS CODE).
;    6983  1     !-
;    6984  1
;    6985  2         begin
;    6986  2
;    6987  2         if (.ST_CODE or .SB_CODE) neq 0                                 ! PRINT SUB CODE ONLY ON ERROR
;    6988  2         then
;    6989  3             begin
;    6990  3             PRINTX (EX_SB);                                             ! SUB CODE :
;    6991  3
;    6992  3             case .ST_CODE from ST_SUC to ST_DRV of
;    6993  3                 set
;    6994  3
;    6995  3                 [ST_SUC]:           if .SB_CODE lequ 16                 ! SUCCESS SUB CODES
;    6996  3                                     then
;    6997  3                                         PRINTX (.TBL_SUC [.SB_CODE]);
;    6998  3
;    6999  3                 [ST_CMD]:           PRINTX (EX_SBO, .SB_CODE / 8);      ! INVALID COMMAND
;    7000  3
;    7001  3                 [ST_ABO]:           ;                                   ! COMMAND ABORTED
;    7002  3
;    7003  3                 [ST_OFL]:           if .SB_CODE lequ 8                  ! UNIT OFFLINE
;    7004  3                                     then
;    7005  3                                         PRINTX (.TBL_OFL [.SB_CODE]);
;    7006  3
;    7007  3                 [ST_AVL]:           ;                                   ! UNIT AVAILABLE
;    7008  3
;    7009  3                 [ST_MFE]:           if .SB_CODE lequ 10                 ! MEDIA FORMAT ERROR
;    7010  3                                     then
;    7011  3                                         PRINTX (.TBL_MFE [.SB_CODE]);
;    7012  3
;    7013  3                 [ST_WPT]:           if (.SB_CODE / 128) lequ 2          ! WRITE PROTECTED
;    7014  3                                     then
;    7015  3                                         PRINTX (.TBL_WPT [(.SB_CODE / 128)]);
;    7016  3
;    7017  3                 [ST_CMP]:           ;                                   ! COMPARE ERROR
;    7018  3
;    7019  3                 [ST_DAT]:           if .SB_CODE lequ 15                 ! DATA ERROR
;    7020  3                                     then
;    7021  3                                         PRINTX (.TBL_DAT [.SB_CODE]);
;    7022  3
;    7023  3                 [ST_HST]:           if .SB_CODE lequ 4                  ! HOST ACCESS ERROR
;    7024  3                                     then
;    7025  3                                         PRINTX (.TBL_HST [.SB_CODE]);
;    7026  3
;    7027  3                 [ST_CNT]:           if .SB_CODE lequ 3                  ! CONTROLLER ERROR
;    7028  3                                     then
```

```
;    7029  3                                  PRINTX (.TBL_CNT [.SB_CODE]);
;    7030  3
;    7031  3              [ST_DRV]:          if .SB_CODE lequ 8                  ! DRIVE ERROR
;    7032  3                                 then
;    7033  3                                     PRINTX (.TBL_DRV [.SB_CODE]);
;    7034  3
;    7035  3              [outrange]:        PRINTX (EX_SBO, .SB CODE);          ! JUST PRINT SUB CODE IF NO MATCH
;    7036  3              tes;
;    7037  3
;    7038  2          end;
;    7039  2
;    7040  1      end;


                                            .SBTTL  EMS.SBC ERROR MESSAGE SUBROUTINES
000000  013700  000000G             EMS.SBC:MOV     ST.CODE,R0                  ;                                   6987
000004  053700  000000G                     BIS     SB.CODE,R0
000010  001001                              BNE     1$
000012  000207                              RTS     PC
000014  012746  000000G             1$:     MOV     #EX.SB,-(SP)                ;                                   6990
000020  012746  000001                      MOV     #1, (SP)
000024  010600                              MOV     SP,R0                       ; SP,*
000026  104415                              TRAP    15
000030  013700  000000G                     MOV     ST.CODE,R0                  ;                                   6992
000034  020027  000013                      CMP     R0,#13
000040  101003                              BHI     3$
000042  006300                              ASL     R0
000044  066007  000000'                      ADD     P.AAA(R0),PC                ; Case dispatch
000050  013716  000000G             3$:     MOV     SB.CODE,(SP)                ;                                   7035
000054  012746  000000G                     MOV     #EX.SBO,-(SP)
000060  012746  000002                      MOV     #2,-(SP)
000064  010600                              MOV     SP,R0                       ; SP,*
000066  104415                              TRAP    15
000070  022626                              CMP     (SP)+,(SP)+
000072  000435                              BR      6$                          ;                                   6992
000074  023727  000000G 000020      4$:     CMP     SB.CODE,#20                 ;                                   6995
000102  101165                              BHI     14$
000104  013700  000000G                     MOV     SB.CODE,R0                  ;                                   6997
000110  006300                              ASL     R0
000112  016016  000000'                      MOV     TBL.SUC(R0),(SP)
000116  012746  000001                      MOV     #1, (SP)
000122  010600                              MOV     SP,R0                       ; SP,*
000124  104415                              TRAP    15
000126  000565                              BR      15$
000130  013716  000000G             5$:     MOV     SB.CODE,(SP)                ;                                   6999
000134  012746  000010                      MOV     #10,-(SP)
000140  004737  000000G                     JSR     PC,BL$DIV
000144  010016                              MOV     R0,(SP)
000146  012746  000000G                     MOV     #EX.SBO, (SP)
000152  012746  000002                      MOV     #2, (SP)
000156  010600                              MOV     SP,R0                       ; SP,*
000160  104415                              TRAP    15
000162  062706  000006                      ADD     #6,SP
```

```
000166  000546                          6$:     BR      16$                             ;
000170  023727  000000G 000010        7$:     CMP     SB.CODE,#10                     ;                6992
000176  101142                                BHI     16$                                              7003
000200  013700  000000G                       MOV     SB.CODE,R0                      ;                7005
000204  006300                                ASL     R0
000206  016016  000042                        MOV     TBL.OFL(R0),(SP)
000212  012746  000001                        MOV     #1,-(SP)
000216  010600                                MOV     SP,R0                           ; SP,*
000220  104415                                TRAP    15
000222  000527                                BR      15$
000224  023727  000000G 000012        8$:     CMP     SB.CODE,#12                     ;                7009
000232  101124                                BHI     16$
000234  013700  000000G                       MOV     SB.CODE,R0                      ;                7011
000240  006300                                ASL     R0
000242  016016  000064                        MOV     TBL.MFE(R0),(SP)
000246  012746  000001                        MOV     #1, (SP)
000252  010600                                MOV     SP,R0                           ; SP,*
000254  104415                                TRAP    15
000256  000511                                BR      15$
000260  013716  000000G                 9$:   MOV     SB.CODE,(SP)                    ;                7013
000264  012746  000200                        MOV     #200,-(SP)
000270  004737  000000G                       JSR     PC,BL$DIV
000274  005726                                TST     (SP)+
000276  020027  000002                        CMP     R0,#2
000302  101100                                BHI     16$
000304  006300                                ASL     R0                              ;                7015
000306  016016  000112                        MOV     TBL.WPT(R0),(SP)
000312  012746  000001                        MOV     #1, (SP)
000316  010600                                MOV     SP,R0                           ; SP,*
000320  104415                                TRAP    15
000322  000467                                BR      15$
000324  023727  000000G 000017        10$:    CMP     SB.CODE,#17                     ;                7019
000332  101064                                BHI     16$
000334  013700  000000G                       MOV     SB.CODE,R0                      ;                7021
000340  006300                                ASL     R0
000342  016016  000120                        MOV     TBL.DAT(R0),(SP)
000346  012746  000001                        MOV     #1,-(SP)
000352  010600                                MOV     SP,R0                           ; SP,*
000354  104415                                TRAP    15
000356  000451                                BR      15$
000360  023727  000000G 000004        11$:    CMP     SB.CODE,#4                      ;                7023
000366  101046                                BHI     16$
000370  013700  000000G                       MOV     SB.CODE,R0                      ;                7025
000374  006300                                ASL     R0
000376  016016  000160                        MOV     TBL.HST(R0),(SP)
000402  012746  000001                        MOV     #1, (SP)
000406  010600                                MOV     SP,R0                           ; SP,*
000410  104415                                TRAP    15
000412  000433                                BR      15$
000414  023727  000000G 000003        12$:    CMP     SB.CODE,#3                      ;                7027
000422  101030                                BHI     16$
000424  013700  000000G                       MOV     SB.CODE,R0                      ;                7029
000430  006300                                ASL     R0
```

```
000432  016016  000172'                        MOV     TBL.CNT(R0),(SP)
000436  012746  000001                        MOV     #1, (SP)
000442  010600                                MOV     SP,R0                   ; SP,*
000444  104415                                TRAP    15
000446  000415                                BR      15$
000450  023727  000000G 000010        13$:    CMP     SB.CODE,#10             ;                       7031
000456  101012                        14$:    BHI     16$
000460  013700  000000G                        MOV     SB.CODE,R0              ;                       7033
000464  006300                                ASL     R0
000466  016016  000202'                        MOV     TBL.DRV(R0),(SP)
000472  012746  000001                        MOV     #1, (SP)
000476  010600                                MOV     SP,R0                   ; SP,*
000500  104415                                TRAP    15
000502  005726                        15$:    TST     (SP)+
000504  022626                        16$:    CMP     (SP)+,(SP)+             ;                       6989
000506  000207                                RTS     PC                      ;                       6976

; Routine Size:  164 words,    Routine Base:  $CODE$ + 10404
; Maximum stack depth per invocation:  7 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:
000000  000024                  2$:    .WORD   24                      ; CASE Table for EMS.SBC-0044   6992
000002  000060                         .WORD   60                      ; [4$]
000004  000434                         .WORD   434                     ; [5$]
000006  000120                         .WORD   120                     ; [16$]
000010  000434                         .WORD   434                     ; [7$]
000012  000154                         .WORD   154                     ; [16$]
000014  000210                         .WORD   210                     ; [8$]
000016  000434                         .WORD   434                     ; [9$]
000020  000254                         .WORD   254                     ; [16$]
000022  000310                         .WORD   310                     ; [10$]
000024  000344                         .WORD   344                     ; [11$]
000026  000400                         .WORD   400                     ; [12$]
                                                                        ; [13$]
```

```
;    7041  1     routine EMS_CMD : novalue =
;    7042  1
;    7043  1     !+
;    7044  1     !       THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
;    7045  1     !       PRESENT) OF THE CURRENT RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
;    7046  1     !       INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
;    7047  1     !
;    7048  1     !       IMPLICIT INPUTS:
;    7049  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    7050  1     !-
;    7051  1
;    7052  2         begin
;    7053  2         PRINTX (EX_CMD);                                        ! "COMMAND: "
;    7054  2
;    7055  2         selectoneu (.RP_ADDR [ENDCOD] and OP_MSK) of
;    7056  2             set
;    7057  2
;    7058  2             [OP_ONL]:       PRINTX (EX_ONL);                    ! ONLINE
;    7059  2
;    7060  2             [OP_ACC]:       PRINTX (EX_ACC);                    ! ACCESS
;    7061  2
;    7062  3             [OP_RD]:        begin
;    7063  3                             PRINTX (EX_RD);                     ! READ
;    7064  3
;    7065  3                             if .RP_ADDR [CMDMOD] neq 0
;    7066  3                             then
;    7067  3                                 PRINTX (EX_CMP);                ! COMPARE
;    7068  3
;    7069  2                             end;
;    7070  2
;    7071  3                 [OP_WRT]:   begin
;    7072  3                             PRINTX (EX_WRT);                    ! WRITE
;    7073  3
;    7074  3                             if .RP_ADDR [CMDMOD] neq 0
;    7075  3                             then
;    7076  3                                 PRINTX (EX_CMP);                ! COMPARE
;    7077  3
;    7078  2                             end;
;    7079  2
;    7080  2                 [otherwise]: PRINTX (EX_OP, .RP_ADDR [ENDCOD]); ! ENDCODE VALUE IF NO MATCH
;    7081  2                 tes;
;    7082  2
;    7083  1         end;                                                    ! ROUTINE EMS_CMD


                                .SBTTL  EMS.CMD ERROR MESSAGE SUBROUTINES
011114                          .PSECT  $CODE$,  RO

000000  004137  000000G    EMS.CMD:JSR     R1,$SAVE2                       ;                       7041
000004  012746  000000G            MOV     @EX.CMD,-(SP)                   ;                       7053
000010  012746  000001             MOV     #1,-(SP)
000014  010600                     MOV     SP,R0                           ; SP,*
000016  104415                     TRAP    15
```

```
000020  013702  000000G                MOV     RP.ADDR,R2              ;                               7055
000024  116201  000014                 MOVB    14(R2),R1
000030  042701  177600                 BIC     #177600,R1
000034  020127  000011                 CMP     R1,#11                  ;                               7058
000040  001007                         BNE     1$
000042  012716  000000G                MOV     #EX.ONL,(SP)
000046  012746  000001                 MOV     #1,-(SP)
000052  010600                         MOV     SP,R0                   ; SP,*
000054  104415                         TRAP    15
000056  000464                         BR      5$
000060  020127  000020        1$:      CMP     R1,#20                  ;                               7060
000064  001007                         BNE     2$
000066  012716  000000G                MOV     #EX.ACC,(SP)
000072  012746  000001                 MOV     #1,-(SP)
000076  010600                         MOV     SP,R0                   ; SP,*
000100  104415                         TRAP    15
000102  000452                         BR      5$
000104  020127  000041        2$:      CMP     R1,#41                  ;                               7062
000110  001022                         BNE     3$
000112  012716  000000G                MOV     #EX.RD,(SP)             ;                               7063
000116  012746  000001                 MOV     #1,-(SP)
000122  010600                         MOV     SP,R0                   ; SP,*
000124  104415                         TRAP    15
000126  013700  000000G                MOV     RP.ADDR,R0              ;                               7065
000132  005760  000012                 TST     12(R0)
000136  001434                         BEQ     5$
000140  012716  000000G                MOV     #EX.CMP,(SP)            ;                               7067
000144  012746  000001                 MOV     #1,-(SP)
000150  010600                         MOV     SP,R0                   ; SP,*
000152  104415                         TRAP    15
000154  000424                         BR      4$
000156  020127  000042        3$:      CMP     R1,#42                  ;                               7071
000162  001024                         BNE     6$
000164  012716  000000G                MOV     #EX.WRT,(SP)            ;                               7072
000170  012746  000001                 MOV     #1,-(SP)
000174  010600                         MOV     SP,R0                   ; SP,*
000176  104415                         TRAP    15
000200  013700  000000G                MOV     RP.ADDR,R0              ;                               7074
000204  005760  000012                 TST     12(R0)
000210  001407                         BEQ     5$
000212  012716  000000G                MOV     #EX.CMP,(SP)            ;                               7076
000216  012746  000001                 MOV     #1,-(SP)
000222  010600                         MOV     SP,R0                   ; SP,*
000224  104415                         TRAP    15
000226  005726                 4$:     TST     (SP)+
000230  005726                 5$:     TST     (SP)+                   ;                               7071
000232  000412                         BR      7$                      ;                               7055
000234  005016                 6$:     CLR     (SP)                    ;                               7080
000236  116216  000014                 MOVB    14(R2),(SP)
000242  012746  000000G                MOV     #EX.OP,-(SP)
000246  012746  000002                 MOV     #2,-(SP)
000252  010600                         MOV     SP,R0                   ; SP,*
000254  104415                         TRAP    15
```

```
000256  022626                          CMP     (SP)+,(SP)+
000260  022626                  7$:     CMP     (SP)+,(SP)+                     ;                       7052
000262  000207                          RTS     PC                              ;                       7041

; Routine Size:  90 words,      Routine Base:  $CODE$ + 11114
; Maximum stack depth per invocation:  9 words
```

```
;    7084  1     GLOBAL ROUTINE EMS_DBN : NOVALUE =                                          !ZZZ
;    7085  1     !+                                                                         !ZZZ
;    7086  1     !       THIS ROUTINE PRINTS THE PRESENT DBN                                !ZZZ
;    7087  1     !                                                                          !ZZZ
;    7088  1     !       IMPLICIT IMPUTS:                                                   !ZZZ
;    7089  1     !               CST_ADDR - ADDRESS OF CONTROLLER STATUS TABLE              !ZZZ
;    7090  1     !-                                                                         !ZZZ
;    7091  1                                                                               !ZZZ
;    7092  2     BEGIN                                                                      !ZZZ
;    7093  2     PRINTB (XX13, .CDISK);                          !"DISK XXX"                !ZZZ
; P  7094  2     PRINTB (XX23, .CST_ADDR [.CUOFF + OF_DBN, D_DBN], .CST_ADDR                !ZZZ
;    7095  2                            [.CUOFF + OF_DBN, D_DBN]);      !"DBN: XXXXXX."  !ZZZ
;    7096  2     PRINTB (XX32, .S_DUPPKT - 2);                   !PRINT BYTE COUNT          !ZZZ
;    7097  2     PRINTB (XX33, .S_PATTERN);                      !PRINT THE PATTERN         !ZZZ
;    7098  2     PRINTB (XX34, .(DUPPKT + .S_DUPPKT), .(DUPPKT + .S_DUPPKT));    !PRINT THE WORD READ    !ZZZ
;    7099  2     EMS_BLK (DUPPKT +2, 256);                       !PRINT WHOLE BLOCK READ    !ZZZ
;    7100  1     END;                                            !IN OCTAL                  !ZZZ


                                        .SBTTL  EMS.DBN ERROR MESSAGE SUBROUTINES
000000  013746  000000G          EMS.DBN::
                                        MOV     CDISK,-(SP)                     ;                       7093
000004  012746  000000G                 MOV     #XX13,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  013700  000000G                 MOV     CUOFF,R0                        ;                       7095
000024  006300                          ASL     R0
000026  063700  000000G                 ADD     CST.ADDR,R0
000032  005016                          CLR     (SP)
000034  116016  000020                  MOVB    20(R0),(SP)
000040  005046                          CLR     -(SP)
000042  116016  000020                  MOVB    20(R0),(SP)
000046  012746  000000G                 MOV     #XX23,-(SP)
000052  012746  000003                  MOV     #3,-(SP)
000056  010600                          MOV     SP,R0                           ; SP,*
000060  104414                          TRAP    14
000062  013716  000000G                 MOV     S.DUPPKT,(SP)                   ;                       7096
000066  162716  000002                  SUB     #2,(SP)
000072  012746  000000G                 MOV     #XX32,-(SP)
000076  012746  000002                  MOV     #2,-(SP)
000102  010600                          MOV     SP,R0                           ; SP,*
000104  104414                          TRAP    14
000106  013716  000000G                 MOV     S.PATTERN,(SP)                  ;                       7097
000112  012746  000000G                 MOV     #XX33,-(SP)
000116  012746  000002                  MOV     #2,-(SP)
000122  010600                          MOV     SP,R0                           ; SP,*
000124  104414                          TRAP    14
000126  013700  000000G                 MOV     S.DUPPKT,R0                     ;                       7098
000132  016016  000000G                 MOV     DUPPKT(R0),(SP)
000136  011646                          MOV     (SP),-(SP)
000140  012746  000000G                 MOV     #XX34,-(SP)
000144  012746  000003                  MOV     #3,-(SP)
```

```
000150  010600                          MOV     SP,R0                           ; SP,*
000152  104414                          TRAP    14
000154  012716  000002G                 MOV     #DUPPKT+2,(SP)                  ;                       7099
000160  012746  000400                  MOV     #400,-(SP)
000164  004737  000000V                 JSR     PC,EMS.BLK
000170  062706  000034                  ADD     #34,SP                          ;                       7092
000174  000207                          RTS     PC                              ;                       7084

; Routine Size:  63 words,      Routine Base:  $CODE$ + 11400
; Maximum stack depth per invocation:  15 words


;   7101  1
;   7102  1
```

```
;    7103  1
;    7104  1     GLOBAL ROUTINE EMS_BLK (ADDR, LENGTH) : NOVALUE =                        !ZZZ
;    7105  1                                                                             !ZZZ
;    7106  1     !+                                                                      !ZZZ
;    7107  1     !       THIS ROUTINE WILL PRINTX A BLOCK OF MEMORY, WHICH IS 'LENGTH'   !ZZZ
;    7108  1     !       WORDS LONG STARTING AT ADDRESS 'ADDR'. PRINTING IS DONE IN OCTAL!ZZZ
;    7109  1     !       8 WDS TO A LINE.                                                !ZZZ
;    7110  1     !-                                                                      !ZZZ
;    7111  1                                                                             !ZZZ
;    7112  2     BEGIN                                                                   !ZZZ
;    7113  2     LITERAL                                                                 !ZZZ
;    7114  2             MASK = %o'7';                                                   !ZZZ
;    7115  2                                                                             !ZZZ
;    7116  2     PRINTX (CRLF);                                                          !ZZZ
;    7117  2             INCR COUNT FROM 1 TO .LENGTH DO          !FOR EACH WD TO PRINT  !ZZZ
;    7118  3                     BEGIN                                                   !ZZZ
;    7119  3                     IF ((.COUNT - 1) AND MASK) EQL 0 !IF START OF NEW LINE  !ZZZ
;    7120  3                     THEN                                                    !ZZZ
;    7121  3                             PRINTX (SPACE4);         !PRINT 4 BLANKS        !ZZZ
;    7122  3                                                                             !ZZZ
;    7123  3                     PRINTX (EX_WRD, ..ADDR);         !PRINTX A WORD         !ZZZ
;    7124  3                     ADDR = .ADDR +2;                 !TO NEXT ADDRESS       !ZZZ
;    7125  3                                                                             !ZZZ
;    7126  4                     IF (((.COUNT AND MASK) EQL 0) OR !END OF LINE OR        !ZZZ
;    7127  4                          (.COUNT EQL .LENGTH))       !WHEN DONE             !ZZZ
;    7128  3                     THEN                                                    !ZZZ
;    7129  3                             PRINTX (CRLF);           !PRINT CR LF           !ZZZ
;    7130  2                     END;                                                    !ZZZ
;    7131  1             END;                                                            !ZZZ


                                        .SBTTL  EMS.BLK ERROR MESSAGE SUBROUTINES
000000  010146                  EMS.BLK::
                                        MOV     R1,-(SP)
000002  012746  000000G                 MOV     #CRLF,-(SP)             ;               7104
000006  012746  000001                  MOV     #1,-(SP)                ;               7116
000012  010600                          MOV     SP,R0
000014  104415                          TRAP    15                      ; SP,*
000016  005001                          CLR     R1
000020  000445                          BR      5$                      ; COUNT         7117
000022  010100                  1$:     MOV     R1,R0
000024  005300                          DEC     R0                      ; COUNT,*       7119
000026  032700  000007                  BIT     #7,R0
000032  001007                          BNE     2$
000034  012716  000000G                 MOV     #SPACE4,(SP)            ;               7121
000040  012746  000001                  MOV     #1,-(SP)
000044  010600                          MOV     SP,R0
000046  104415                          TRAP    15                      ; SP,*
000050  005726                          TST     (SP)+
000052  017616  000012          2$:     MOV     @12(SP),(SP)            ; ADDR,*        7123
000056  012746  000000G                 MOV     #EX.WRD,-(SP)
000062  012746  000002                  MOV     #2,-(SP)
000066  010600                          MOV     SP,R0                   ; SP,*
```

```
000070  104415                          TRAP    15
000072  062766  000002  000016          ADD     #2,16(SP)               ; *,ADDR                        7124
000100  032701  000007                  BIT     #7,R1                   ; *,COUNT                       7126
000104  001403                          BEQ     3$
000106  020166  000014                  CMP     R1,14(SP)               ; COUNT,LENGTH                  7127
000112  001007                          BNE     4$
000114  012716  000000G           3$:    MOV     #CRLF,(SP)              ;                               7129
000120  012746  000001                  MOV     #1,-(SP)
000124  010600                          MOV     SP,R0                   ; SP,*
000126  104415                          TRAP    15
000130  005726                          TST     (SP)+
000132  022626                  4$:     CMP     (SP)+,(SP)+             ;                               7118
000134  005201                  5$:     INC     R1                      ; COUNT                         7117
000136  020166  000010                  CMP     R1,10(SP)               ; COUNT,LENGTH
000142  003727                          BLE     1$
000144  022626                          CMP     (SP)+,(SP)+             ;                               7112
000146  012601                          MOV     (SP)+,R1                ;                               7104
000150  000207                          RTS     PC

; Routine Size:  53 words,      Routine Base:  $CODE$ + 11576
; Maximum stack depth per invocation:  8 words


;   7132  1
;   7133  1
```

```
;    7134  1     routine EMS_LBN : novalue =
;    7135  1
;    7136  1     !+
;    7137  1     !       THIS ROUTINE PRINTS (EXTENDED) ONE OF TWO BLOCK NUMBERS APPEARING IN
;    7138  1     !       THE CURRENT RETURN PACKET. NORMALLY, THE LBN FIELD IS PRINTED; THIS
;    7139  1     !       FIELD WAS COPIED INTO THE RETURN PACKET FROM THE ASSOCIATED COMMAND
;    7140  1     !       PACKET. HOWEVER, IF THE "FLAGS" FIELD OF THE CURRENT RETURN PACKET
;    7141  1     !       INDICATES "BAD BLOCK REPORTED", THEN THE "FIRST BAD BLOCK" FIELD IS
;    7142  1     !       PRINTED.
;    7143  1     !
;    7144  1     !       IMPLICIT INPUTS:
;    7145  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    7146  1     !-
;    7147  1
;    7148  2         begin
;    7149  2
;    7150  2         if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and              ! IF NO BAD BLOCK FOUND
;    7151  3            (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    7152  2         then
;    7153  2             PRINTX (EX_LBN, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);
;    7154  2
;    7155  2         if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and              ! IF BAD BLOCKS FOUND AND REPLACED
;    7156  3            (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    7157  2         then
;    7158  2             PRINTX (EX_BBU, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    7159  2
;    7160  2         if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                  ! IF HOST REPLACEABLE BAD BLOCK FOUND
;    7161  3            (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    7162  2         then
;    7163  2             PRINTX (EX_BB, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    7164  2
;    7165  2         if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                  ! IF MORE THAN 1 HOST REPLACEABLE BAD BLOCK FOUND
;    7166  3            (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    7167  2         then
;    7168  2             PRINTX (EX_BB1, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    7169  1         end;


                                        .SBTTL  EMS.LBN ERROR MESSAGE SUBROUTINES
000000  013700  000000G          EMS.LBN:MOV     RP.ADDR,R0                      ;                                       7150
000004  105760  000015                  TSTB    15(R0)
000010  100417                          BMI     1$
000012  132760  000100  000015          BITB    #100,15(R0)                     ;                                       7151
000020  001013                          BNE     1$
000022  016046  000050                  MOV     50(R0),-(SP)                    ;                                       7153
000026  011646                          MOV     (SP),-(SP)
000030  012746  000000G                 MOV     #EX.LBN,-(SP)
000034  012746  000003                  MOV     #3,-(SP)
000040  010600                          MOV     SP,R0                           ; SP,*
000042  104415                          TRAP    15
000044  062706  000010                  ADD     #10,SP
000050  013700  000000G          1$:     MOV     RP.ADDR,R0                      ;                                       7155
000054  105760  000015                  TSTB    15(R0)
```

```
000060  100417                          BMI     2$
000062  132760  000100  000015          BITB    #100,15(R0)                     ;                       7156
000070  001413                          BEQ     2$
000072  016046  000040                  MOV     40(R0),-(SP)                    ;                       7158
000076  011646                          MOV     (SP),-(SP)
000100  012746  000000G                 MOV     #EX.BBU,-(SP)
000104  012746  000003                  MOV     #3,-(SP)
000110  010600                          MOV     SP,R0                           ; SP,*
000112  104415                          TRAP    15
000114  062706  000010                  ADD     #10,SP
000120  013700  000000G          2$:    MOV     RP.ADDR,R0                      ;                       7160
000124  105760  000015                  TSTB    15(R0)
000130  100017                          BPL     3$
000132  132760  000100  000015          BITB    #100,15(R0)                     ;                       7161
000140  001013                          BNE     3$
000142  016046  000040                  MOV     40(R0),-(SP)                    ;                       7163
000146  011646                          MOV     (SP),-(SP)
000150  012746  000000G                 MOV     #EX.BB,-(SP)
000154  012746  000003                  MOV     #3,-(SP)
000160  010600                          MOV     SP,R0                           ; SP,*
000162  104415                          TRAP    15
000164  062706  000010                  ADD     #10,SP
000170  013700  000000G          3$:    MOV     RP.ADDR,R0                      ;                       7165
000174  105760  000015                  TSTB    15(R0)
000200  100017                          BPL     4$
000202  132760  000100  000015          BITB    #100,15(R0)                     ;                       7166
000210  001413                          BEQ     4$
000212  016046  000040                  MOV     40(R0),-(SP)                    ;                       7168
000216  011646                          MOV     (SP),-(SP)
000220  012746  000000G                 MOV     #EX.BB1,-(SP)
000224  012746  000003                  MOV     #3,-(SP)
000230  010600                          MOV     SP,R0                           ; SP,*
000232  104415                          TRAP    15
000234  062706  000010                  ADD     #10,SP
000240  000207                  4$:     RTS     PC                              ;                       7134

; Routine Size:  81 words,      Routine Base:  $CODE$ + 11750
; Maximum stack depth per invocation:  6 words
```

```
;    7170  1     routine EMS_BC : novalue =
;    7171  1
;    7172  1     !+
;    7173  1     !       THIS ROUTINE PRINTS (EXTENDED) BOTH BYTE COUNT FIELDS OF THE CURRENT
;    7174  1     !       RETURN PACKET: THE BYTE COUNT FROM THE COMMAND PACKET AND THE
;    7175  1     !       ACTUAL NUMBER OF BYTES TRANSFERRED (FROM THE RESPONSE PACKET).
;    7176  1     !
;    7177  1     !       IMPLICIT INPUTS:
;    7178  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    7179  1     !-
;    7180  1
;    7181  2         begin
;    7182  2         PRINTX (EX_CBC, .RP_ADDR [CBCNT_LO]);       ! "BYTE COUNT IN COMMAND: XXXXX,'
;    7183  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);         ! 'ACTUAL # OF BYTES TRANSFERRED: XXXXX."
;    7184  1         end;                                        ! ROUTINE EMS_BC


                                        .SBTTL  EMS.BC ERROR MESSAGE SUBROUTINES
000000  013700  000000G        EMS.BC: MOV     RP.ADDR,R0                      ;                       7182
000004  016046  000044                  MOV     44(R0),-(SP)
000010  012746  000000G                 MOV     #EX.CBC,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104415                          TRAP    15
000024  013700  000000G                 MOV     RP.ADDR,R0                      ;                       7183
000030  016016  000020                  MOV     20(R0),(SP)
000034  012746  000000G                 MOV     #EX.BC,-(SP)
000040  012746  000002                  MOV     #2,-(SP)
000044  010600                          MOV     SP,R0                           ; SP,*
000046  104415                          TRAP    15
000050  062706  000012                  ADD     #12,SP                          ;                       7181
000054  000207                          RTS     PC                              ;                       7170

; Routine Size:  23 words,      Routine Base:  $CODE$ + 12212
; Maximum stack depth per invocation:  7 words
```

```
;    7185  1      routine EMS BD : novalue •
;    7186  1
;    7187  1      !•
;    7188  1      !       THIS ROUTINE PRINTS (EXTENDED) THE TWO WORD I/O BUFFER DESCRIPTOR
;    7189  1      !       APPEARING IN THE CURRENT RETURN PACKET.
;    7190  1      !
;    7191  1      !       IMPLICIT INPUTS:
;    7192  1      !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    7193  1      !
;    7194  1
;    7195  1          PRINTX (EX BD, .RP_ADDR [BUFF 1], .RP_ADDR [BUFF 0]);       ! "I/O BUFFER DESCRIPTOR: XXXXXX XXXXXX"


                                        .SBTTL  EMS.BD ERROR MESSAGE SUBROUTINES
000000  013700  000000G          EMS.BD: MOV     RP.ADDR,R0                      ;                                       7195
000004  016046  000024                  MOV     24(R0), (SP)
000010  016046  000026                  MOV     26(R0),-(SP)
000014  012746  000000G                  MOV     #EX.BD,-(SP)
000020  012746  000003                  MOV     #3,-(SP)
000024  010600                          MOV     SP,R0                           ; SP,•
000026  104415                          TRAP    15
000030  062706  000010                  ADD     #10,SP
000034  000207                          RTS     PC                              ;                                       7185

; Routine Size:  15 words,      Routine Base:  $CODE$ • 12270
; Maximum stack depth per invocation:  6 words
```

```
;    7196  1
;    7197  1
;    7198  1     routine EMS_RP : novalue =
;    7199  1
;    7200  1
;    7201  1     !+
;    7202  1     !       THIS ROUTINE IS RESPONSIBLE FOR PRINTING (EXTENDED) THE RELEVANT FIELDS
;    7203  1     !       OF THE CURRENT RETURN PACKET.
;    7204  1     !-
;    7205  1
;    7206  1
;    7207  2         begin
;    7208  2         EMS_SBC ();                                     ! SUB CODE
;    7209  2         EMS_CMD ();                                     ! COMMAND (AND MODIFIER)
;    7210  2
;    7211  2         if (.RP_ADDR [ENDCOD] and OP_MSK) neq OP_ONL
;    7212  2
;    7213  2         then
;    7214  2             EMS_LBN ();                                 ! LBN OR BAD BLOCK NUMBER
;    7215  2
;    7216  2         if ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD) or
;    7217  3            ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_WRT)
;    7218  3
;    7219  2         then
;    7220  3             begin
;    7221  3             EMS_BC ();                                  ! BYTE COUNTS
;    7222  3             EMS_BD ();                                  ! I/O BUFFER DESCRIPTOR
;    7223  2             end;
;    7224  2
;    7225  2         EMS_TIM ();                                     ! TIME
;    7226  1         end;                                            ! ROUTINE EMS_RP


                                .SBTTL  EMS.RP ERROR MESSAGE SUBROUTINES
000000  010146                 EMS.RP: MOV     R1,-(SP)                ;                       7198
000002  004737  010404'                 JSR     PC,EMS.SBC              ;                       7208
000006  004737  011114'                 JSR     PC,EMS.CMD              ;                       7209
000012  013700  000000G                 MOV     RP.ADDR,R0              ;                       7211
000016  116000  000014                  MOVB    14(R0),R0
000022  042700  177600                  BIC     #177600,R0
000026  020027  000011                  CMP     R0,#11
000032  001402                         BEQ     1$
000034  004737  011750                  JSR     PC,EMS.LBN              ;                       7214
000040  013700  000000G         1$:     MOV     RP.ADDR,R0              ;                       7216
000044  116001  000014                  MOVB    14(R0),R1
000050  042701  177600                  BIC     #177600,R1
000054  020127  000041                  CMP     R1,#41
000060  001407                         BEQ     2$
000062  116000  000014                  MOVB    14(R0),R0               ;                       7217
000066  042700  177600                  BIC     #177600,R0
000072  020027  000042                  CMP     R0,#42
000076  001004                         BNE     3$
000100  004737  012212'         2$:     JSR     PC,EMS.BC               ;                       7221
```

```
000104  004737  012270'                        JSR     PC,EMS.BD               !                       7222
000110  004737  000000V                 38:     JSR     PC,EMS.TIM              !                       7225
000114  012601                                  MOV     (SP)+,R1                !                       7198
000116  000207                                  RTS     PC

; Routine Size:  40 words.      Routine Base:  $CODE$ + 12326
; Maximum stack depth per invocation:  2 words


;   7227  1
;   7228  1
```

N2


```
;    7229  1     global routine EMS_RP1 : novalue =
;    7230  1
;    7231  1     !+
;    7232  1     !       THIS  ROUTINE  IS  CALLED  TO  PRINT  THE  ENTIRE CONTENTS OF THE
;    7233  1     !       RETURN PACKET DESIGNATED BY THE GLOBAL DATUM "RP_ADDR". HOWEVER, THE
;    7234  1     !       PRINTING WILL ONLY OCCUR IF EXTENDED ERROR PRINTING IS ENABLED.
;    7235  1     !-
;    7236  1
;    7237  2         begin
;    7238  2         PRINTX (EX_RP);                         ! "CONTENTS OF RETURN PACKET:"
;    7239  2         EMS_BLK (.RP_ADDR, RP_LEN);             ! PRINT BLOCK OF WORDS
;    7240  1         end;


                                        .SBTTL  EMS.RP1 ERROR MESSAGE SUBROUTINES
000000  012746  000000G          EMS.RP1::
                                        MOV     #EX.RP,-(SP)                    ;                       7238
000004  012746  000001                  MOV     #1,-(SP)
000010  010600                          MOV     SP,R0                           ; SP,*
000012  104415                          TRAP    15
000014  013716  000000G                 MOV     RP.ADDR,(SP)                    ;                       7239
000020  012746  000026                  MOV     #26,-(SP)
000024  004737  011576'                 JSR     PC,EMS.BLK
000030  062706  000006                  ADD     #6,SP                           ;                       7237
000034  000207                          RTS     PC                              ;                       7229

; Routine Size:  15 words,      Routine Base:  $CODE$ + 12446
; Maximum stack depth per invocation:  4 words
```

```
;    7241  1     global routine EMS_EL (index) : novalue =
;    7242  1
;    7243  1     !+
;    7244  1     !       THIS ROUTINE IS CALLED FROM 'SEQUEN' AND 'DATAGM' AND PRINTS THE CONTENTS OF THE
;    7245  1     !       ERROR-LOG PACKET
;    7246  1     !-
;    7247  1
;    7248  2         begin
;    7249  2
;    7250  2         local
;    7251  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;    7252  2             REASON : word,
;    7253  2             DISK_NUM : byte,
;    7254  2             ELOG_CODE : byte,
;    7255  2             ELOG_SUB : word;
;    7256  2
;    7257  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                  ! ERROR LOG PACKET'S ADDRESS
;    7258  2         REASON = .ELOG_ADDR [EL_FORMAT];                               ! FORMAT
;    7259  2         DISK_NUM = .ELOG_ADDR [EL_DK_NUM];                             ! DISK NUMBER
;    7260  2         ELOG_CODE = .ELOG_ADDR [EL_CODE];                              ! CODE
;    7261  2         ELOG_SUB = .ELOG_ADDR [EL_SUBCODE];                            ! SUBCODE
;    7262  2         PRINTP (ELG_00);                                               ! ERROR-LOG MESSAGE RECEIVED
;    7263  2
;    7264  2         if (.REASON eql FORMAT_CNTR) or
;    7265  3            (.REASON eql FORMAT_HOST)
;    7266  2         then
;    7267  2             PRINTB (.ELG_FMT [.REASON])                                ! PRINT BASIC REASON
;    7268  2         else
;    7269  2             PRINTB (.ELG_FMT [.REASON], .DISK_NUM);                    ! PRINT BASIC REASON WITH DISK NUMBER
;    7270  2
;    7271  2         if (.ELOG_CODE gtru 0) and
;    7272  3            (.ELOG_CODE lequ 11)
;    7273  2         then
;    7274  3             begin
;    7275  3             PRINTX (ASTERISK);
;    7276  3             PRINTX (.ERR_COD [.ELOG_CODE - 1]);                        ! CODE
;    7277  3             end
;    7278  2         else
;    7279  2
;    7280  2             if .ELOG_CODE eql ST_DIA                                   ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    7281  2             then
;    7282  3                 begin
;    7283  3                 PRINTX (ASTERISK);
;    7284  3                 PRINTX (.ERR_COD [12]);
;    7285  2                 end;
;    7286  2
;    7287  2         if (.ELOG_CODE eql ST_MFE) and
;    7288  3            (.ELOG_SUB lequ 10)
;    7289  2         then
;    7290  3             begin
;    7291  3             PRINTX (CRLF);
;    7292  3             PRINTX (ASTERISK);
;    7293  3             PRINTX (.TBL_MFE [.ELOG_SUB]);                             ! MEDIA FORMAT ERROR
```

```
;    7294  2              end;
;    7295  2
;    7296  2          if (.ELOG_CODE eql ST_DAT) and
;    7297  3             (.ELOG_SUB lequ 15)
;    7298  2          then
;    7299  3              begin
;    7300  3              PRINTX (CRLF);
;    7301  3              PRINTX (ASTERISK);
;    7302  3              PRINTX (.TBL_DAT [.ELOG_SUB]);                          ! DATA ERROR
;    7303  2              end;
;    7304  2
;    7305  2          if (.ELOG_CODE eql ST_HST) and
;    7306  3             (.ELOG_SUB lequ 4)
;    7307  2          then
;    7308  3              begin
;    7309  3              PRINTX (CRLF);
;    7310  3              PRINTX (ASTERISK);
;    7311  3              PRINTX (.TBL_HST [.ELOG_SUB]);                          ! HOST ACCESS ERROR
;    7312  2              end;
;    7313  2
;    7314  2          if (.ELOG_CODE eql ST_CNT) and
;    7315  3             (.ELOG_SUB lequ 3)
;    7316  2          then
;    7317  3              begin
;    7318  3              PRINTX (CRLF);
;    7319  3              PRINTX (ASTERISK);
;    7320  3              PRINTX (.TBL_CNT [.ELOG_SUB]);                          ! CONTROLLER ERROR
;    7321  2              end;
;    7322  2
;    7323  2          if (.ELOG_CODE eql ST_DRV) and
;    7324  3             (.ELOG_SUB lequ 8)
;    7325  2          then
;    7326  3              begin
;    7327  3              PRINTX (CRLF);
;    7328  3              PRINTX (ASTERISK);
;    7329  3              PRINTX (.TBL_DRV [.ELOG_SUB]);                          ! DRIVE ERROR
;    7330  2              end;
;    7331  2
;    7332  2          if .REASON eql FORMAT_XFER                                  ! IF DISK XFER INVOLVED
;    7333  2          then
;    7334  2
;    7335  2              if .ELOG_ADDR [EL_BLOCK_TYPE] eql TYPE_LBN              ! PRINT PBN OR RBN
;    7336  2              then
;    7337  3                  PRINTX (EX_PBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK])
;    7338  2              else
;    7339  2                  PRINTX (EX_RBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK]);
;    7340  2
;    7341  2          EMS_TIM ();                                                 ! TIME
;    7342  2          EMS_BLK ((.ELOG_ADDR + 2), ((.ELOG_ADDR [EL_MSGLEN] + 1) / 2) + 2); ! PRINTX CONTENTS OF PACKET
;    7343  2          ELOG_ADDR [EL_CONTENTS] = EMPTY;                            ! DECLARE SAVE AREA FREE
;    7344  2
;    7345  1          end;
```

```
                                        .SBTTL  EMS.EL ERROR MESSAGE SUBROUTINES
000000  004137  000000G           EMS.EL::JSR     R1,$SAVE5                       ;                                       7241
000004  005746                           TST     -(SP)
000006  016646  000020                   MOV     20(SP),-(SP)                    ; INDEX,*                               7257
000012  012746  000102                   MOV     #102,-(SP)
000016  004737  000000G                  JSR     PC,BL$MUL
000022  062700  000000G                  ADD     #ELOG.PKT,R0
000026  010001                           MOV     R0,R1                           ; *,ELOG.ADDR
000030  116166  000016  000004           MOVB    16(R1),4(SP)                    ; *(ELOG.ADDR),REASON                   7258
000036  105066  000005                   CLRB    5(SP)                           ; REASON
000042  116105  000012                   MOVB    12(R1),R5                       ; *(ELOG.ADDR),DISK.NUM                 7259
000046  116100  000020                   MOVB    20(R1),R0                       ; *(ELOG.ADDR),*                        7260
000052  042700  177740                   BIC     #177740,R0
000056  105004                           CLRB    R4                              ; ELOG.CODE
000060  050004                           BIS     R0,R4                           ; *,ELOG.CODE
000062  016103  000020                   MOV     20(R1),R3                       ; *(ELOG.ADDR),ELOG.SUB                 7261
000066  006203                           ASR     R3                              ; ELOG.SUB
000070  006203                           ASR     R3                              ; ELOG.SUB
000072  006203                           ASR     R3                              ; ELOG.SUB
000074  006203                           ASR     R3                              ; ELOG.SUB
000076  006203                           ASR     R3                              ; ELOG.SUB
000100  042703  174000                   BIC     #174000,R3                      ; *,ELOG.SUB
000104  012716  000000G                  MOV     #ELG.00,(SP)                    ;                                       7262
000110  012746  000001                   MOV     #1,-(SP)
000114  010600                           MOV     SP,R0                           ; SP,*
000116  104414                           TRAP    14
000120  016602  000006                   MOV     6(SP),R2                        ; REASON,*                              7267
000124  006302                           ASL     R2
000126  005766  000006                   TST     6(SP)                           ; REASON                                7264
000132  001404                           BEQ     1$
000134  026627  000006  000001           CMP     6(SP),#1                        ; REASON,*                              7265
000142  001007                           BNE     2$
000144  016216  000000G           1$:     MOV     ELG.FMT(R2),(SP)                ;                                       7267
000150  012746  000001                   MOV     #1,-(SP)
000154  010600                           MOV     SP,R0                           ; SP,*
000156  104414                           TRAP    14
000160  000411                           BR      3$                              ;                                       7264
000162  005016                  2$:      CLR     (SP)                            ;                                       7269
000164  110516                           MOVB    R5,(SP)                         ; DISK.NUM,*
000166  016246  000000G                  MOV     ELG.FMT(R2),-(SP)
000172  012746  000002                   MOV     #2,-(SP)
000176  010600                           MOV     SP,R0                           ; SP,*
000200  104414                           TRAP    14
000202  005726                           TST     (SP)+
000204  105704                  3$:      TSTB    R4                              ; ELOG.CODE                             7271
000206  001423                           BEQ     4$
000210  120427  000013                   CMPB    R4,#13                          ; ELOG.CODE,*                           7272
000214  101020                           BHI     4$
000216  012716  000000G                  MOV     #ASTERISK,(SP)                  ;                                       7275
000222  012746  000001                   MOV     #1,-(SP)
000226  010600                           MOV     SP,R0                           ; SP,*
000230  104415                           TRAP    15
```

```
000232  005000                          CLR     R0                              ;                               7276
000234  150400                          BISB    R4,#0                           ; ELOG.CODE,*
000236  006300                          ASL     R0
000240  016016  177776G                 MOV     ERR.COD-2(R0),(SP)
000244  012746  000001                  MOV     #1,-(SP)
000250  010600                          MOV     SP,R0                           ; SP,*
000252  104415                          TRAP    15
000254  000417                          BR      5$                              ;                               7274
000256  120427  000037          4$:     CMPB    R4,#37                          ; ELOG.CODE,*                   7280
000262  001015                          BNE     6$
000264  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               7283
000270  012746  000001                  MOV     #1,-(SP)
000274  010600                          MOV     SP,R0                           ; SP,*
000276  104415                          TRAP    15
000300  013716  000030G                 MOV     ERR.COD+30,(SP)                 ;                               7284
000304  012746  000001                  MOV     #1,-(SP)
000310  010600                          MOV     SP,R0                           ; SP,*
000312  104415                          TRAP    15
000314  022626                  5$:     CMP     (SP)+,(SP)+                     ;                               7282
000316  120427  000005          6$:     CMPB    R4,#5                           ; ELOG.CODE,*                   7287
000322  001031                          BNE     7$
000324  020327  000012                  CMP     R3,#12                          ; ELOG.SUB,*                    7288
000330  101026                          BHI     7$
000332  012716  000000G                 MOV     #CRLF,(SP)                      ;                               7291
000336  012746  000001                  MOV     #1,-(SP)
000342  010600                          MOV     SP,R0                           ; SP,*
000344  104415                          TRAP    15
000346  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               7292
000352  012746  000001                  MOV     #1,-(SP)
000356  010600                          MOV     SP,R0                           ; SP,*
000360  104415                          TRAP    15
000362  010300                          MOV     R3,R0                           ; ELOG.SUB,*                    7293
000364  006300                          ASL     R0
000366  016016  000064'                 MOV     TBL.MFE(R0),(SP)
000372  012746  000001                  MOV     #1,-(SP)
000376  010600                          MOV     SP,R0                           ; SP,*
000400  104415                          TRAP    15
000402  062706  000006                  ADD     #6,SP                           ;                               7290
000406  120427  000010          7$:     CMPB    R4,#10                          ; ELOG.CODE,*                   7296
000412  001031                          BNE     8$
000414  020327  000017                  CMP     R3,#17                          ; ELOG.SUB,*                    7297
000420  101026                          BHI     8$
000422  012716  000000G                 MOV     #CRLF,(SP)                      ;                               7300
000426  012746  000001                  MOV     #1,-(SP)
000432  010600                          MOV     SP,R0                           ; SP,*
000434  104415                          TRAP    15
000436  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               7301
000442  012746  000001                  MOV     #1,-(SP)
000446  010600                          MOV     SP,R0                           ; SP,*
000450  104415                          TRAP    15
000452  010300                          MOV     R3,R0                           ; ELOG.SUB,*                    7302
000454  006300                          ASL     R0
000456  016016  000120'                 MOV     TBL.DAT(R0),(SP)
```

```
000462  012746  000001                    MOV     #1,-(SP)
000466  010600                            MOV     SP,R0                   ; SP,*
000470  104415                            TRAP    15
000472  062706  000006                    ADD     #6,SP                   ;                               7299
000476  120427  000011          8$:       CMPB    R4,#11                  ; ELOG.CODE,*                   7305
000502  001031                            BNE     9$
000504  020327  000004                    CMP     R3,#4                   ; ELOG.SUB,*                    7306
000510  101026                            BHI     9$
000512  012716  000000G                   MOV     #CRLF,(SP)              ;                               7309
000516  012746  000001                    MOV     #1,-(SP)
000522  010600                            MOV     SP,R0                   ; SP,*
000524  104415                            TRAP    15
000526  012716  000000G                   MOV     #ASTERISK,(SP)          ;                               7310
000532  012746  000001                    MOV     #1,-(SP)
000536  010600                            MOV     SP,R0                   ; SP,*
000540  104415                            TRAP    15
000542  010300                            MOV     R3,R0                   ; ELOG.SUB,*                    7311
000544  006300                            ASL     R0
000546  016016  000160'                   MOV     TBL.HST(R0),(SP)
000552  012746  000001                    MOV     #1,-(SP)
000556  010600                            MOV     SP,R0                   ; SP,*
000560  104415                            TRAP    15
000562  062706  000006                    ADD     #6,SP                   ;                               7308
000566  120427  000012          9$:       CMPB    R4,#12                  ; ELOG.CODE,*                   7314
000572  001031                            BNE     10$
000574  020327  000003                    CMP     R3,#3                   ; ELOG.SUB,*                    7315
000600  101026                            BHI     10$
000602  012716  000000G                   MOV     #CRLF,(SP)              ;                               7318
000606  012746  000001                    MOV     #1,-(SP)
000612  010600                            MOV     SP,R0                   ; SP,*
000614  104415                            TRAP    15
000616  012716  000000G                   MOV     #ASTERISK,(SP)          ;                               7319
000622  012746  000001                    MOV     #1,-(SP)
000626  010600                            MOV     SP,R0                   ; SP,*
000630  104415                            TRAP    15
000632  010300                            MOV     R3,R0                   ; ELOG.SUB,*                    7320
000634  006300                            ASL     R0
000636  016016  000172'                   MOV     TBL.CNT(R0),(SP)
000642  012746  000001                    MOV     #1,-(SP)
000646  010600                            MOV     SP,R0                   ; SP,*
000650  104415                            TRAP    15
000652  062706  000006                    ADD     #6,SP                   ;                               7317
000656  120427  000013          10$:      CMPB    R4,#13                  ; ELOG.CODE,*                   7323
000662  001031                            BNE     11$
000664  020327  000010                    CMP     R3,#10                  ; ELOG.SUB,*                    7324
000670  101026                            BHI     11$
000672  012716  000000G                   MOV     #CRLF,(SP)              ;                               7327
000676  012746  000001                    MOV     #1,-(SP)
000702  010600                            MOV     SP,R0                   ; SP,*
000704  104415                            TRAP    15
000706  012716  000000G                   MOV     #ASTERISK,(SP)          ;                               7328
000712  012746  000001                    MOV     #1,-(SP)
000716  010600                            MOV     SP,R0                   ; SP,*
```

```
000720  104415                          TRAP    15
000722  010300                          MOV     R3,R0                   ; ELOG.SUB,*                    7329
000724  006300                          ASL     R0
000726  016016  000202'                 MOV     TBL.DRV(R0),(SP)
000732  012746  000001                  MOV     #1,-(SP)
000736  010600                          MOV     SP,R0                   ; SP,*
000740  104415                          TRAP    15
000742  062706  000006                  ADD     #6,SP                   ;                               7326
000746  026627  000010  000002  11$:    CMP     10(SP),#2               ; REASON,*                      7332
000754  001031                          BNE     14$
000756  032761  170000  000060          BIT     #170000,60(R1)          ; *,*(ELOG.ADDR)                7335
000764  001012                          BNE     12$
000766  016116  000056                  MOV     56(R1),(SP)             ; *(ELOG.ADDR),*                7337
000772  011646                          MOV     (SP),-(SP)
000774  012746  000000G                 MOV     #EX.PBN,-(SP)
001000  012746  000003                  MOV     #3,-(SP)
001004  010600                          MOV     SP,R0                   ; SP,*
001006  104415                          TRAP    15
001010  000411                          BR      13$                     ;                               7335
001012  016116  000056          12$:    MOV     56(R1),(SP)             ; *(ELOG.ADDR),*                7339
001016  011646                          MOV     (SP),-(SP)
001020  012746  000000G                 MOV     #EX.RBN,-(SP)
001024  012746  000003                  MOV     #3,-(SP)
001030  010600                          MOV     SP,R0                   ; SP,*
001032  104415                          TRAP    15
001034  062706  000006          13$:    ADD     #6,SP                   ;                               7335
001040  004737  000000V         14$:    JSR     PC,EMS.TIM              ;                               7341
001044  012716  000002                  MOV     #2,(SP)                 ;                               7342
001050  060116                          ADD     R1,(SP)                 ; ELOG.ADDR,*
001052  016146  000002                  MOV     2(R1),-(SP)             ; *(ELOG.ADDR),*
001056  005216                          INC     (SP)
001060  012746  000002                  MOV     #2,-(SP)
001064  004737  000000G                 JSR     PC,BL$DIV
001070  010066  000002                  MOV     R0,2(SP)
001074  062766  000002  000002          ADD     #2,2(SP)
001102  005726                          TST     (SP)+
001104  004737  011576'                 JSR     PC,EMS.BLK
001110  105061  000001                  CLRB    1(R1)                   ; *(ELOG.ADDR)                  7343
001114  062706  000014                  ADD     #14,SP                  ;                               7241
001120  000207                          RTS     PC

; Routine Size:  297 words,     Routine Base:  $CODE$ + 12504
; Maximum stack depth per invocation:  16 words
```

```
;    7346  1     global routine EMS_CMP (ADDR) : novalue =
;    7347  1
;    7348  1     !+
;    7349  1     ! THIS ROUTINE IS CALLED FROM 'HOST_WRT_CHK' AND PRINTS RELEVENT DATA ON A HOST
;    7350  1     ! COMPARE ERROR
;    7351  1     !-
;    7352  1
;    7353  2         begin
;    7354  2
;    7355  2         local
;    7356  2             ORIG_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;    7357  2
;    7358  2         ORIG_ADDR = .ADDR;                                          ! ADDRESS OF THE WRITE RETPKT
;    7359  2         PRINTB (ERR_00, .CDISK);                                    ! "DISK XXX"
;    7360  2         PRINTB (DASH);                                              !
;    7361  2         PRINTB (.ERR_COD [12]);                                     ! " - HOST COMPARE ERROR"
;    7362  2         PRINTX (EX_LBW, .ORIG_ADDR [LBN_LO], .ORIG_ADDR [LBN_LO]);  ! LBN (WRITE)
;    7363  2         PRINTX (EX_LBR, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);      ! LBN (READ)
;    7364  2         PRINTX (EX_CBW, .ORIG_ADDR [CBCNT_LO]);                     ! BYTE COUNT (WRITE)
;    7365  2         PRINTX (EX_BC, .ORIG_ADDR [BCNT_LO]);                       ! BYTE COUNT XMITTED (WRITE)
;    7366  2         PRINTX (EX_CBR, .RP_ADDR [CBCNT_LO]);                       ! BYTE COUNT (READ);
;    7367  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);                         ! BYTE COUNT XMITTED (READ)
;    7368  2         PRINTX (EX_BDW, .ORIG_ADDR [BUFF_1], .ORIG_ADDR [BUFF_0]);  ! BUFFER ADDRESS (WRITE)
;    7369  2         PRINTX (EX_BDR, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);      ! BUFFER ADDRESS (READ)
;    7370  2         EMS_TIM ();                                                 ! TIME
;    7371  1         end;


                                .SBTTL  EMS.CMP ERROR MESSAGE SUBROUTINES
000000  G10146            EMS.CMP::
                            MOV     R1,-(SP)                    ;                                   7346
000002  016601  000004      MOV     4(SP),R1                    ; ADDR,ORIG.ADDR                    7358
000006  013746  000000G     MOV     CDISK,-(SP)                 ;                                   7359
000012  012746  000000G     MOV     #ERR.00,-(SP)
000016  012746  000002      MOV     #2,-(SP)
000022  010600              MOV     SP,R0                       ; SP,*
000024  104414              TRAP    14
000026  012716  000000G     MOV     #DASH,(SP)                  ;                                   7360
000032  012746  000001      MOV     #1,-(SP)
000036  010600              MOV     SP,R0                       ; SP,*
000040  104414              TRAP    14
000042  013716  000030G     MOV     ERR.COD+30,(SP)             ;                                   7361
000046  012746  000001      MOV     #1,-(SP)
000052  010600              MOV     SP,R0                       ; SP,*
000054  104414              TRAP    14
000056  016116  000050      MOV     50(R1),(SP)                 ; *(ORIG.ADDR),*                    7362
000062  011646              MOV     (SP),-(SP)
000064  012746  000000G     MOV     #EX.LBW,-(SP)
000070  012746  000003      MOV     #3,-(SP)
000074  010600              MOV     SP,R0                       ; SP,*
000076  104415              TRAP    15
000100  013700  000000G     MOV     RP.ADDR,R0                  ;                                   7363
000104  016016  000050      MOV     50(R0),(SP)
```

```
000110  011646                          MOV     (SP),-(SP)
000112  012746  000000G                 MOV     #EX.LBR, (SP)
000116  012746  000003                  MOV     #3,-(SP)
000122  010600                          MOV     SP,R0                   ; SP,*
000124  104415                          TRAP    15
000126  016116  000044                  MOV     44(R1),(SP)             ; *(ORIG.ADDR),*        7364
000132  012746  000000G                 MOV     #EX.CBW, (SP)
000136  012746  000002                  MOV     #2,-(SP)
000142  010600                          MOV     SP,R0                   ; SP,*
000144  104415                          TRAP    15
000146  016116  000020                  MOV     20(R1),(SP)             ; *(ORIG.ADDR),*        7365
000152  012746  000000G                 MOV     #EX.BC,-(SP)
000156  012746  000002                  MOV     #2,-(SP)
000162  010600                          MOV     SP,R0                   ; SP,*
000164  104415                          TRAP    15
000166  013700  000000G                 MOV     RP.ADDR,R0              ;                       7366
000172  016016  000044                  MOV     44(R0),(SP)
000176  012746  000000G                 MOV     #EX.CBR,-(SP)
000202  012746  000002                  MOV     #2,-(SP)
000206  010600                          MOV     SP,R0                   ; SP,*
000210  104415                          TRAP    15
000212  013700  000000G                 MOV     RP.ADDR,R0              ;                       7367
000216  016016  000020                  MOV     20(R0),(SP)
000222  012746  000000G                 MOV     #EX.BC,-(SP)
000226  012746  000002                  MOV     #2,-(SP)
000232  010600                          MOV     SP,R0                   ; SP,*
000234  104415                          TRAP    15
000236  016116  000024                  MOV     24(R1),(SP)             ; *(ORIG.ADDR),*        7368
000242  016146  000026                  MOV     26(R1),-(SP)            ; *(ORIG.ADDR),*
000246  012746  000000G                 MOV     #EX.BDW,-(SP)
000252  012746  000003                  MOV     #3,-(SP)
000256  010600                          MOV     SP,R0                   ; SP,*
000260  104415                          TRAP    15
000262  013700  000000G                 MOV     RP.ADDR,R0              ;                       7369
000266  016016  000024                  MOV     24(R0),(SP)
000272  016046  000026                  MOV     26(R0),-(SP)
000276  012746  000000G                 MOV     #EX.BDR,-(SP
000302  012746  000003                  MOV     #3,-(SP)
000306  010600                          MOV     SP,R0                   ; SP,*
000310  104415                          TRAP    15
000312  004737  000000V                 JSR     PC,EMS.TIM              ;                       7370
000316  062706  000062                  ADD     #62,SP                  ;                       7353
000322  012601                          MOV     (SP)+,R1                ;                       7346
000324  000207                          RTS     PC

; Routine Size:  107 words,     Routine Base:  $CODE$ + 13626
; Maximum stack depth per invocation:  28 words
```

```
;    7372  1     global routine EMS_ERR : novalue =
;    7373  1
;    7374  2         begin
;    7375  2
;    7376  2     ! TABLE OF BASIC, HARD ERROR MESSAGE ADDRESSES, INDEXED BY STATUS CODE
;    7377  2     !
;    7378  2         PRINTB (ERR_00, .CDISK);                        ! "DISK XXX"
;    7379  2         PRINTB (DASH);                                  !
;    7380  2
;    7381  2         if (.ST_CODE gtru 0) and                        ! IF STATUS CODE IS WITHIN RANGE
;    7382  3            (.ST_CODE lequ 11)
;    7383  2         then
;    7384  3             PRINTB (.ERR_COD [.ST_CODE   1])            ! PRINTB APPROPRIATE MESSAGE
;    7385  2         else
;    7386  2
;    7387  2             if .ST_CODE eql ST_DIA
;    7388  2             then
;    7389  3                 PRINTB (.ERR_COD [11])                  ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    7390  2             else
;    7391  2                 PRINTB (EX_SC, .ST_CODE);               ! JUST PRINT STATUS CODE WHEN NO MATCH
;    7392  2
;    7393  2         EMS_RP ();                                      ! PRINTX OTHER RETPKT FIELDS
;    7394  2
;    7395  1         end;


                                .SBTTL  EMS.ERR ERROR MESSAGE SUBROUTINES
000000  013746  000000G    EMS.ERR::
                            MOV     CDISK,-(SP)                     ;                       7378
000004  012746  000000G            MOV     #ERR.00,-(SP)
000010  012746  000002             MOV     #2,-(SP)
000014  010600                     MOV     SP,R0                   ; SP,*
000016  104414                     TRAP    14
000020  012716  000000G            MOV     #DASH,(SP)              ;                       7379
000024  012746  000001             MOV     #1,-(SP)
000030  010600                     MOV     SP,R0                   ; SP,*
000032  104414                     TRAP    14
000034  013700  000000G            MOV     ST.CODE,R0              ;                       7381
000040  001413                     BEQ     1$
000042  020027  000013             CMP     R0,#13                  ;                       7382
000046  101010                     BHI     1$
000050  006300                     ASL     R0                      ;                       7384
000052  016016  177776G            MOV     ERR.COD-2(R0),(SP)
000056  012746  000001             MOV     #1,-(SP)
000062  010600                     MOV     SP,R0                   ; SP,*
000064  104414                     TRAP    14
000066  000422                     BR      3$                      ;                       7381
000070  020027  000037     1$:     CMP     R0,#37                  ;                       7387
000074  001007                     BNE     2$
000076  013716  000026G            MOV     ERR.COD+26,(SP)         ;                       7389
000102  012746  000001             MOV     #1,-(SP)
000106  010600                     MOV     SP,R0                   ; SP,*
000110  104414                     TRAP    14
```

```
000112  000410                          BR      3$                              ;                       7387
000114  010016                  2$:     MOV     R0,(SP)                         ;                       7391
000116  012746  000000G                 MOV     #EX.SC, (SP)
000122  012746  000002                  MOV     #2,-(SP)
000126  010600                          MOV     SP,R0                           ; SP,*
000130  104414                          TRAP    14
000132  005726                          TST     (SP)+
000134  004737  012326'                3$:     JSR     PC,EMS.RP               ;                       7393
000140  062706  000012                  ADD     #12,SP                          ;                       7374
000144  000207                          RTS     PC                              ;                       7372

; Routine Size:  51 words.      Routine Base:  $CODE$ + 14154
; Maximum stack depth per invocation:  8 words
```

```
ZRQAM2          RD/RX EXERCISER                          4 Apr 1985 12:40:26    VAX-11 Bliss 16 V4.1 582          SEQ 0244
V02.2           ERROR MESSAGE SUBROUTINES                4 Apr 1985 12:33:21    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL1;16   Page 227
                                                                                                                    (78)

;    7396  1      routine EMS_TIM : novalue =
;    7397  1
;    7398  1      !+
;    7399  1      ! THIS ROUTINE PRINTS THE TIME OF DAY MESSAGE
;    7400  1      !
;    7401  1
;    7402  1      PRINTX (EX_TIM, .HOURS, .MINUTES);


                                                .SBTTL  EMS.TIM ERROR MESSAGE SUBROUTINES
000000  005046                            EMS.TIM:CLR     (SP)                                    ;                       7402
000002  113716  000000G                         MOVB    MINUTES,(SP)
000006  005046                                  CLR     -(SP)
000010  113716  000000G                         MOVB    HOURS,(SP)
000014  012746  000000G                         MOV     #EX.TIM, (SP)
000020  012746  000003                          MOV     #3, (SP)
000024  010600                                  MOV     SP,R0                                   ; SP,*
000026  104415                                  TRAP    15
000030  062706  000010                          ADD     #10,SP
000034  000207                                  RTS     PC                                      ;                       7396

; Routine Size:  15 words,      Routine Base:  $CODE$ + 14322
; Maximum stack depth per invocation:  6 words
```

```
;    7403  1     BGNMSG (EMS 01);


                                        .SBTTL  EMS.01 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.01::JSR     PC,M$EMS.01                     ;                       7403
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14360
; Maximum stack depth per invocation:  2 words


;    7404  2     PRINTB (EBS 01, MAX UNITS);                 ! "MORE THAN XX UNITS SPECIFIED"
;    7405  1     ENDMSG;


                                        .SBTTL  M$EMS.01 ERROR MESSAGE SUBROUTINES
000000  012746  000004            M$EMS.01:
                                        MOV     #4, (SP)                        ;                       7404
000004  012746  000000G                 MOV     #EBS.01, (SP)
000010  012746  000002                  MOV     #2, (SP)
000014  010600                          MOV     SP,R0                           ; SP,R0
000016  104414                          TRAP    14
000020  062706  000006                  ADD     #6,SP                           ;                       7403
000024  000207                          RTS     PC

; Routine Size:  11 words,      Routine Base:  $CODE$ + 14370
; Maximum stack depth per invocation:  5 words
```

```
ZRQAM2          RD/RX EXERCISER                                 4-Apr-1985 12:40:26    VAX 11 Bliss 16 V4.1 582          SEQ 0245
V02.2           ERROR MESSAGE SUBROUTINES                       4-Apr 1985 12:33:21    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL1;16   Page 229
                                                                                                                          (80)

;   7406  1     BGNMSG (EMS_10);


                                                    .SBTTL  EMS.10 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                    EMS.10::JSR     PC,M$EMS.10                     ;                          7406
000004  104423                                      TRAP    23
000006  000207                                      RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14416
; Maximum stack depth per invocation:  2 words


;   7407  2     PRINTB (EBD_10, .RDRX_ADDR + .OF_RC);           ! "NO RESPONSE AT ADDRESS XXXXXX'
;   7408  1     ENDMSG;


                                                    .SBTTL  M$EMS.10 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                  M$EMS.10:
                                                    MOV     RDRX.ADDR,-(SP)                 ;                          7407
000004  063716  000000G                            ADD     OF.RC,(SP)
000010  012746  000000G                            MOV     #EBD.10, (SP)
000014  012746  000002                             MOV     #2,-(SP)
000020  010600                                     MOV     SP,R0                           ; SP,*
000022  104414                                     TRAP    14
000024  062706  000006                             ADD     #6,SP                           ;                          7406
000030  000207                                     RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14426
; Maximum stack depth per invocation:  5 words
```

```
ZRQAM2          RD/RX EXERCISER                                 4-Apr-1985 12:40:26    VAX-11 Bliss-16 V4.1-582              SEQ 0247
V02.2           ERROR MESSAGE SUBROUTINES                       4-Apr-1985 12:33:21    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL1;16  Page 230
                                                                                                                             (81)

;   7409  1     BGNMSG (EMS_12);


                                                .SBTTL  EMS.12 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                 EMS.12::JSR     PC,M$EMS.12                     ;                                    7409
000004  104423                                  TRAP    23
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14460
; Maximum stack depth per invocation:  2 words


;   7410  2     PRINTB (EBD_12, .RDRX_ADDR);                    ! "INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX"
;   7411  1     ENDMSG;


                                                .SBTTL  M$EMS.12 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                 M$EMS.12:
                                                MOV     RDRX.ADDR,-(SP)                 ;                                    7410
000004  012746  000000G                         MOV     #EBD.12,-(SP)
000010  012746  000002                          MOV     #2,-(SP)
000014  010600                                  MOV     SP,R0                           ; SP,*
000016  104414                                  TRAP    14
000020  062706  000006                          ADD     #6,SP                           ;                                    7409
000024  000207                                  RTS     PC

; Routine Size:  11 words,      Routine Base:  $CODE$ + 14470
; Maximum stack depth per invocation:  5 words
```

```
;   7412  1     BGNMSG (EMS_13);


                                          .SBTTL  EMS.13 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.13::JSR     PC,M$EMS.13                     ;                       7412
000004  104423                                TRAP    23
000006  000207                                RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14516
; Maximum stack depth per invocation:  2 words


;   7413  2     PRINTB (EBD_13, .STEP);                 ! "STEP X READ ERROR"
;   7414  2     EMS_SA ();                              ! PRINTX SA CONTENTS
;   7415  1     ENDMSG;


                                          .SBTTL  M$EMS.13 ERROR MESSAGE SUBROUTINES
000000  013746  000000G             M$EMS.13:
                                          MOV     STEP,-(SP)                      ;                       7413
000004  012746  000000G                   MOV     #EBD.13,-(SP)
000010  012746  000002                    MOV     #2,-(SP)
000014  010600                            MOV     SP,R0                           ; SP,*
000016  104414                            TRAP    14
000020  004737  010072'                   JSR     PC,EMS.SA                       ;                       7414
000024  062706  000006                    ADD     #6,SP                           ;                       7412
000030  000207                            RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14526
; Maximum stack depth per invocation:  5 words
```

```
;   7416  1     BGNMSG (EMS_14);


                                          .SBTTL  EMS.14 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.14::JSR     PC,M$EMS.14                     ;                       7416
000004  104423                                TRAP    23
000006  000207                                RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14560
; Maximum stack depth per invocation:  2 words


;   7417  2     PRINTB (EBD_14, .IRDRX_ADDR);                   ! "BAD SA CODE FROM DEVICE XXXXXX"
;   7418  2     EMS_SA ();                                      ! PRINTX SA REGISTER CONTENTS
;   7419  1     ENDMSG;


                                          .SBTTL  M$EMS.14 ERROR MESSAGE SUBROUTINES
000000  013746  000000G             M$EMS.14:
                                          MOV     IRDRX.ADDR, (SP)                ;                       7417
000004  012746  000000G                   MOV     #EBD.14,-(SP)
000010  012746  000002                    MOV     #2,-(SP)
000014  010600                            MOV     SP,R0                           ; SP,*
000016  104414                            TRAP    14
000020  004737  010072'                   JSR     PC,EMS.SA                       ;                       7418
000024  062706  000006                    ADD     #6,SP                           ;                       7416
000030  000207                            RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14570
; Maximum stack depth per invocation:  5 words
```

```
;   7420  1     BGNMSG (EMS_18);


                                        .SBTTL  EMS.18 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.18::JSR     PC,M$EMS.18                     ;                       7420
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14622
; Maximum stack depth per invocation:  2 words


;   7421  2     PRINTB (EBD_18, .CDISK);                ! "DISK XXX WENT OFFLINE"
;   7422  2     EMS_RP ();                              ! PRINTX RELEVANT RETPKT FIELDS
;   7423  1     ENDMSG;


                                        .SBTTL  M$EMS.18 ERROR MESSAGE SUBROUTINES
00000C  013746  000000G           M$EMS.18:
                                        MOV     CDISK,-(SP)                     ;                       7421
000004  012746  000000G                 MOV     @EBD.18,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  004737  012326'                 JSR     PC,EMS.RP                       ;                       7422
000024  062706  000006                  ADD     #6,SP                           ;                       7420
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14632
; Maximum stack depth per invocation:  5 words
```

```
ZRQAM2          RD/RX EXERCISER                                 4-Apr-1985 12:40:26    VAX 11 Bliss '6 '4.1 582                SEQ 0251
V02.2           ERROR MESSAGE SUBROUTINES                       4 Apr-1985 12:33:21    DISK$USER2:[ OW RS.ZRQ]ZRQAG0.BL1;16     Page 234
                                                                                                                                (85)

;   7424  1     BGNMSG (EMS_21);


                                                        .SBTTL  EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                        EMS.21::JSR     PC,M$EMS.21                     ;                               7424
000004  104423                                        TRAP    23
000006  000207                                        RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14664
; Maximum stack depth per invocation:  2 words


;   7425  2     EMS_RP1 ();                                     ! CONTENTS OF RETURN PACKET
;   7426  1     ENDMSG;


                                                        .SBTTL  M$EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  012446'                        M$EMS.21:
                                                        JSR     PC,EMS.RP1                      ;                               7425
000004  000207                                        RTS     PC                              ;                               7424

; Routine Size:  3 words,       Routine Base:  $CODE$ + 14674
; Maximum stack depth per invocation:  1 word
```

```
ZRQAM2          RD/RX EXERCISER                                 4-Apr 1985 12:40:26    VAX-11 Bliss-16 V4.1 582          SEQ 0252
V02.2           ERROR MESSAGE SUBROUTINES                       4-Apr-1985 12:33:21    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL1;16  Page 235
                                                                                                                          (86)

;   7427  1     BGNMSG (EMS_22)                         !CONTENTS OF DUP BUFFER          ZZZ


                                                .SBTTL  EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.22::JSR     PC,M$EMS.22                     ;                                 7427
000004  104423                                  TRAP    23
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14702
; Maximum stack depth per invocation:  2 words


;   7428  2     EMS_DBN ();                                                             !ZZZ
;   7429  1     ENDMSG;                                                                 !ZZZ


                                                .SBTTL  M$EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  011400'                M$EMS.22:
                                                JSR     PC,EMS.DBN                      ;                                 7428
000004  000207                                  RTS     PC                              ;                                 7427

; Routine Size:  3 words,       Routine Base:  $CODE$ + 14712
; Maximum stack depth per invocation:  1 word
```

```
;   7430  1     BGNMSG (EMS_24);


                                        .SBTTL  EMS.24 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.24::JSR     PC,M$EMS.24                     ;                       7430
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14720
; Maximum stack depth per invocation:  2 words


;   7431  2     PRINTB (EBD_24, .CDISK);                ! "DISK XXX WENT TO THE AVAILABLE STATE"
;   7432  2     EMS_RP ();                              ! PRINTX RELEVANT RETPKT FIELDS
;   7433  1     ENDMSG;


                                        .SBTTL  M$EMS.24 ERROR MESSAGE SUBROUTINES
000000  013746  000000G            M$EMS.24:
                                        MOV     CDISK,-(SP)                     ;                       7431
000004  012746  000000G                 MOV     #EBD.24,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  004737  012326'                 JSR     PC,EMS.RP                       ;                       7432
000024  062706  000006                  ADD     #6,SP                           ;                       7430
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14730
; Maximum stack depth per invocation:  5 words
```

```
;   7434  1     BGNMSG (EMS_30);


                                        .SBTTL  EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  C00000V           EMS.30::JSR     PC,M$EMS.30                     ;                       7434
000004  104423                           TRAP    23
000006  000207                           RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14762
; Maximum stack depth per invocation:  2 words


;   7435  2     EMS_ERR ();                                     ! PRINT ALL RELEVANT DATA ON DETECTING AN ERROR
;   7436  1     ENDMSG;


                                        .SBTTL  M$EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  014154'           M$EMS.30:
                                        JSR     PC,EMS.ERR                      ;                       7435
000004  000207                           RTS     PC                              ;                       7434

; Routine Size:  3 words,       Routine Base:  $CODE$ + 14772
; Maximum stack depth per invocation:  1 word


;   7437  1
;   7438  1     end
;   7439  1
;   7440  0     eludom



;                                   OTS external references
                                            .GLOBL  $SAVE5, $SAVE4, $SAVE3, $SAVE2
                                            .GLOBL  BL$DIV, BL$MOD, BL$MUL


;                                   PSECT SUMMARY
;
;       Psect Name                  Words     Attributes
;        $OWN$                        74       RW ,  D  ,  LCL,  REL,  CON
;        $CODE$                     3328       RO ,  I  ,  LCL,  REL,  CON
;        $PLIT$                       12       RO ,  D  ,  LCL,  REL,  CON


;                               Library Statistics
;
;                                        ------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[POWERS.ZRQ]ZRQAG0.L16;10   407       297        72          21        00:00.1
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAGO.BL1/LIST=ZRQAGO.LS1/OBJECT=ZRQAGO.OB1/SOURCE=PAGE:53

; Size:          3155 code + 6741 data words
; Run Time:        02:34.3
; Elapsed Time:    41:36.4
; Lines/CPU Min:    2893
; Lexemes/CPU-Min: 27465
; Memory Used:  712 pages
; Compilation Complete
```

```
;    0001  0     module ZRQAM3 (
;    0002  0
;    0003  0     %title 'RD/RX EXERCISER'
;    0004  0                     ident = 'V02.2',
;    0005  0                     addressing mode (absolute),
;    0006  0                     environment (noeis)
;    0007  0                     ) %
;    0008  0
;    0009  1     begin
;    0010  1
;    0011  1     %sbttl 'DECLARATIONS'
;    0012  1
;    0013  1     library 'ZRQAG0.L16';                   ! RDRX EXERCISER GLOBAL LIBRARY
;    0014  1
;    0015  1     !ZZZ require 'BLSMAC.REQ';              ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    0016  1     require 'HSAXA0.BLB';                   ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    1757  1
;    1758  1     EQUALS;
;    1759  1
;    1760  1     forward routine                         ! ROUTINES APPEAR IN THIS ORDER
;    1761  1         INIT_TEST : novalue,                !    INDENTATION IMPLIES CALLED SUBROUTINE
;    1762  1             DRIVER_INIT : novalue,
;    1763  1                 CTLR_INIT : novalue,
;    1764  1                     INI_CTLR_DAT : novalue,
;    1765  1                     REG_EXIST,
;    1766  1                     VEC_BR_TEST,
;    1767  1                     INT_GEN,
;    1768  1                     HARD_INIT,
;    1769  1                     INI_RRING : novalue,
;    1770  1                     SET_CTLR_CHAR,
;    1771  1                 UNIT_INIT : novalue,
;    1772  1                     DR_ERR : novalue,
;    1773  1             ACCESS : novalue,
;    1774  1         MULTI_DRIVE : novalue,
;    1775  1             MD_INIT : novalue,
;    1776  1                 INIT_IO_BUFF : novalue,
;    1777  1             FATAL_ERROR : novalue,
;    1778  1             QIO_OK,
;    1779  1                 QIO_OUT,
;    1780  1                 QIO_GEN : novalue,
;    1781  1                     GET_RANDOM : novalue,
;    1782  1                     QIO_UNIT : novalue,
;    1783  1                     QIO_FUNC : novalue,
;    1784  1                        DUP : NOVALUE,                          !ZZZ
;    1785  1                              DUPWRTDBN : NOVALUE,              !ZZZ
;    1786  1                              DUPREDDBN : NOVALUE,              !ZZZ
;    1787  1                                   DUPCOMMAND : NOVALUE,        !ZZZ
;    1788  1                                   DUPIDLE : NOVALUE,           !ZZZ
;    1789  1                     QIO_LBN : novalue,
;    1790  1                     QIO_SIZE : novalue,
;    1791  1                     FILL_BUFF : novalue,
;    1792  1                 PROC_RETPKT : novalue,
;    1793  1                     DIO_RETPKT : NOVALUE,                       !ZZZ
```

```
;    1794  1                      DUP_COMPARE : NOVALUE,                                  !ZZZ
;    1795  1                  IO RETPKT : novalue,
;    1796  1                      FSET_UPAR : novalue,
;    1797  1                      HARD_ERROR : novalue,
;    1798  1                          ERR_HRD_RTNE : novalue,
;    1799  1                          ERR_HRD_RTNE_APT : novalue,
;    1800  1                      UPD_IO_TALLY : novalue,
;    1801  1                          OVF_CHK : novalue,
;    1802  1                          ROUND_OUTPUT : novalue,
;    1803  1                      HOST_WRT_CHK,
;    1804  1      !                   ERR_HRD_RTNE : novalue,
;    1805  1      !                   ERR_HRD_RTNE APT : novalue,
;    1806  1                      SWEEP : novalue,
;    1807  1                          RPS_REM,
;    1808  1                  DR_RETPKT : novalue,
;    1809  1              AZINTO : L$ISR novalue,
;    1810  1                  AZINT : novalue,
;    1811  1      !               FATAL_ERROR : novalue,
;    1812  1                      POLL_CRING : novalue,
;    1813  1                      POLL_RRING : novalue,
;    1814  1                          DUP_RSP  : NOVALUE,
 !ZZZ
;    1815  1                          DISK_RSP : novalue,
;    1816  1                      SEQUEN : novalue,
;    1817  1                          SCAN_ERRLOG : novalue,
;    1818  1                              ERR_SOFT_RTNE : novalue,
;    1819  1                              ERR_SOFT_RTNE_APT : novalue,
;    1820  1                              SOFT_ERROR : novalue,
;    1821  1                          DATAGM : novalue;
;    1822  1      !                       ERR_SOFT_RTNE : novalue,
;    1823  1      !                       ERR_SOFT_RTNE_APT : novalue,
;    1824  1      !                       SOFT_ERROR : novalue;
;    1825  1
;    1826  1      external
;    1827  1          CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    1828  1                              ! RUN-TIME CONTROLLER STATUS TABLES
;    1829  1          CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    1830  1                              ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    1831  1          DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    1832  1                              ! DRIVER CONTROLLER TABLES
;    1833  1          DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    1834  1                              ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    1835  1          RDRX_ADDR : ref rdrx field (RC_REG),
;    1836  1                              ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    1837  1          IRDRX_ADDR : ref rdrx field (RC_REG),
;    1838  1                              ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    1839  1          BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                             !ZZZ
;    1840  1                              !BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS    !ZZZ
;    1841  1                              !RANDOM SEEK) MODE                              !ZZZ
;    1842  1          TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    1843  1                              ! STATISTICS TABLES
;    1844  1          T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    1845  1                              ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    1846  1          DUPPKT : BLOCK [257, WORD] FIELD (DP FIELDS),       !BUFFER FOR DUP  ZZZ
```

```
;    1847  1                          !INFO FROM RECEIVE AND SEND COMMANDS                ZZZ
;    1848  1          TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TK DIRECTION    ZZZ
;    1849  1          RDM_CNT : WORD,     !NO. OF RANDOM NOS.                KEEP\\       ZZZ
;    1850  1          RANDOM : VECTOR [RDM_LEN, WORD],    !RAND NO TABLE TOGET//HER       ZZZ
;    1851  1          C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    1852  1                          ! STATISTICS TABLE FOR CONTROLLER ERRORS
;    1853  1          MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;    1854  1                          ! MSCP PACKET POOL
;    1855  1          IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    1856  1                          ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;    1857  1          PKT_USE : vector [PKT_CNT, byte, signed],
;    1858  1                          ! MSCP PACKET POOL ALLOCATION TABLE
;    1859  1          RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP FIELDS),
;    1860  1                          ! RETURN PACKET POOL
;    1861  1          RP USE : vector [RP_CNT, byte, signed],
;    1862  1                          ! RETURN PACKET POOL ALLOCATION TABLE
;    1863  1          RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
;    1864  1          RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
;    1865  1                          ! CURRENT RETURN PACKET ADDRESS
;    1866  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;    1867  1                          ! ERROR-LOG PACKET SAVE AREA
;    1868  1          BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
;    1869  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;    1870  1          IODQ : vector [IODQ LEN, byte],                      ! I/O DONE QUEUE   CIRCULAR QUEUE OF RETPKT INDECES
;    1871  1          IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
;    1872  1          IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
;    1873  1          ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
;    1874  1          EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
;    1875  1          DUP_FLAGS : WORD,                                    !DUP FLAGS                                  ZZZ
;    1876  1          CCTLR : word,                                        ! NUMBER OF "CURRENT" CONTROLLER
;    1877  1          CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    1878  1          CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
;    1879  1          CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;    1880  1          DUR : vector [MAX UNITS, byte],                      ! DROP UNIT REASON
;    1881  1          QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;    1882  1          FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
;    1883  1          BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
;    1884  1          ST_CODE : word,                                      ! CURRENT STATUS CODE
;    1885  1          SB_CODE : word,                                      ! CURRENT SUB-CODE
;    1886  1          STEP : word,                                         ! CURRENT STEP IN HARD_INIT
;    1887  1          OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
;    1888  1          SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
;    1889  1          CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
;    1890  1          NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
;    1891  1          CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
;    1892  1          CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
;    1893  1          TEMP1 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;    1894  1          TEMP2 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;    1895  1          CRFDIT_BAL : word,                                   ! CREDIT BALANCE
;    1896  1          NEXT_PKT USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;    1897  1          HOURS : byte,                                        ! TIME OF DAY (HOURS)
;    1898  1          MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
;    1899  1          CLK TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
```

```
;    1900  1          CLK_PRESENT : byte.                 ! FLAG INDICATES IF LINE CLOCK PRESENT
;    1901  1          MOE_FLAG : byte.                    ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;    1902  1          FORCED_ERROR : byte.                ! "FORCED ERROR" DETECTED IN LAST READ
;    1903  1          FER0_LBN : word.                    !LO LBN ADR OF THE "FORCED ERROR" BLOCK                ZZZ
;    1904  1          FER1_LBN : word.                    !HI LBN ADR OF THE "FORCED ERROR" BLOCK                ZZZ
;    1905  1    !     FER_LBN : word.                     ! LBN OF THE "FORCED ERROR" BLOCK                      ZZZ
;    1906  1          FER_BC : word.                      ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    1907  1          INIT_OCCURED : byte.                ! EXERCISER INITIALIZATION COMPLETE
;    1908  1          ADDR_VECT_OK : byte.                ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    1909  1          S_PATTERN : WORD.                 !PATTERN WRITTEN TO DBNS                               ZZZ
;    1910  1          S_DUPPKT : WORD.                  !DBN BYTE COUNTER                                      ZZZ
;    1911  1          P_INDEX : SIGNED WORD.            !CURRENT MESSAGE PACKET INDEX                          ZZZ
;    1912  1          RD_COUNT : WORD.                  ! NUMBER OF WINCHESTER UNITS                           ZZZ
;    1913  1          BRLEVEL : word.                   ! CURRENT DEVICE'S BR LEVEL                            ZZZ
;    1914  1          D_FAIL : BYTE.                    !SIGNIFIES DUP TYPE ERROR                              ZZZ
;    1915  1          DBM12.
;    1916  1          DBM18.
;    1917  1          DBM19.
;    1918  1          DBM20.
;    1919  1          DBM21.
;    1920  1          DBM22.
;    1921  1          DBM23.
;    1922  1          DBM25.
;    1923  1          DBM26.
;    1924  1          DBM27.
;    1925  1          DBM29.
;    1926  1          DBM108.
;    1927  1          DBM109.
;    1928  1          DBM111.
;    1929  1          DBM112.
;    1930  1          DBM120.
;    1931  1          DBM121.
;    1932  1          EM_0.                                                                                    !ZZZ
;    1933  1          EM_1.                                                                                    !ZZZ
;    1934  1          EM_2.                                                                                    !ZZZ
;    1935  1          EM_3.                                                                                    !ZZZ
;    1936  1          EM_4.                                                                                    !ZZZ
;    1937  1          EM_5.                                                                                    !ZZZ
;    1938  1          EM_6.                                                                                    !ZZZ
;    1939  1          EM_7.                                                                                    !ZZZ
;    1940  1          EM_8.                                                                                    !ZZZ
;    1941  1          EM_9.                                                                                    !ZZZ
;    1942  1          EM_10.                                                                                   !ZZZ
;    1943  1          EM_12.                                                                                   !ZZZ
;    1944  1          EM_13.                                                                                   !ZZZ
;    1945  1          MSG_02.
;    1946  1          MSG_03.
;    1947  1          EGS_02.
;    1948  1          EGD_10.
;    1949  1          EGD_11.
;    1950  1          EGD_12.
;    1951  1          EGD_13.
;    1952  1          EGD_14.
```

```
;    1953  1          EGD_15,
;    1954  1          EGD_16,
;    1955  1          EGD_17,
;    1956  1          EGD_18,
;    1957  1          EGD_19,
;    1958  1          EGD_20,
;    1959  1          EGD_21,
;    1960  1          EGD_22,
;    1961  1          EGD_23,
;    1962  1          EGD_24,
;    1963  1          EGH_30,
;    1964  1          DF_MSG,
;    1965  1          HRD_MSG,
;    1966  1          SFT_MSG,
;    1967  1          HRD_SUB,
;    1968  1          CRLF,
;    1969  1          SWP_ERROR : word,                          ! HARD ERROR LIMIT FOR DROPPING UNIT
;    1970  1          SWP_XFER : word,                           ! TRANSFER LIMIT FOR DROPPING UNIT
;    1971  1          SWP_FLAGS : word,                          ! FLAGS (SEE DOCUMENTATION)
;    1972  1          DUPROUND  : WORD,                          !DUP TESTING RATIO                    ZZZ
;    1973  1          SWP_RAT : word,                            ! RD51/52 OPERATION RATIO
;    1974  1          SWP_DPAT : word,                           ! DATA PATTERN NUMBER
;    1975  1          SWP_UCNT : word,                           ! USER DATA PATTERN COUNT
;    1976  1          SWP_TIME : word,                           ! TIME OF DAY
;    1977  1          SWP_UDPAT : vector [MAX_UDP_CNT, word],    ! USER DATA PATTERN
;    1978  1          L$LUN,
;    1979  1          L$UNIT;
;    1980  1
;    1981  1      psect
;    1982  1          own = $GGG$(read, nowrite, execute, local, concatenate);
;    1983  1
;    1984  1      own
;    1985  1          COMM_AREA : blockvector [MAX_CTLR, COMM_LEN, word] field (COM_FIELDS),
;    1986  1                              ! COMMUNICATIONS AREA BETWEEN HOST AND AZTEC CONTROLLERS
;    1987  1      !!ZZZ    BST : vector [MAX_UNITS, word, signed],
;    1988  1                              ! BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
;    1989  1          DPST : vector [MAX_UNITS, byte],           ! DATA PATTERN SEQUENCE TABLE
;    1990  1          MAX_LBN : vector [MAX_UNITS, word],        ! LARGEST LBN ALLOWED
;    1991  1          STORAGE : vector [MAX_UNITS, word],        ! DUMMY STORAGE
;    1992  1          ICOM_ADDR : ref block [COMM_LEN, word] field (COM_FIELDS),
;    1993  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S COMMUNICATION AREA
;    1994  1          ICST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    1995  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S CST
;    1996  1          IDCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    1997  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    1998  1          INT_ADDR : vector [MAX_CTLR] initial (AZINT0 %(, AZINT1, AZINT2, AZINT3)%),
;    1999  1                              ! INTERRUPT SERVICE ROUTINE ADDRESS TABLE
;    2000  1      !!ZZZ    RDM_CNT : word initial (RDM_LEN),          ! NUMBER OF RANDOM NUMBERS  \ KEEP
;    2001  1      !!ZZZ    RANDOM : vector [RDM_LEN, word],           ! RANDOM NUMBER TABLE       / TOGETHER
;    2002  1          ICTLR : word,                              ! INTERRUPTING CONTROLLING NUMBER
;    2003  1          RW_BALANCE : WORD INITIAL (3),             ! FLAGS TOO MANY READS IN RD/WR RATIO      ZZZ
;    2004  1          MX1 : signed word,                         ! MSCP PKT INDEX FOR FIRST QIO
;    2005  1          MX2 : signed word,                         ! MSCP PKT INDEX FOR SECOND QIO
```

```
;    2006  1          MAD1 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    2007  1                         ! ADDRESS OF MSCP PACKET FOR FIRST QIO
;    2008  1          MAD2 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    2009  1                         ! ADDRESS OF MSCP PACKET FOR SECOND QIO
;    2010  1          LAST_PKT : blockvector [MAX_CTLR, LAST_PKT_LEN, word] field (LAST_PKT_FIELDS),
;    2011  1                         ! SAVE AREA FOR INFO ABOUT LAST RESPONSE PACKET
;    2012  1
;    2013  1          RNDY0 : WORD,                                          !32-BIT RANDOM PATTERN LO WD          ZZZ
;    2014  1          RNDY1 : WORD,                                          !32-BIT RANDOM PATTERN HI WD          ZZZ
;    2015  1          FRAME_CNT : WORD,                                      !WHICH 7-BIT FRAME OF R_STRING IN USE ZZZ
;    2016  1          R_STRING: WORD,                                        !BITS USED IN PATTERN SELECTION       ZZZ
;    2017  1          RNDYIN :  vector [9, word] initial (%o'127102',        !NINE SEED WORDS                       ZZZ
;    2018  1                    %o'143662', %o'036750', %o'121624', %o'023267',                                  !ZZZ
;    2019  1                    %o'036561', %o'063714', %o'560255', %o'134230'),                                 !ZZZ
;    2020  1          RNDMS0 :  vector [8, word] initial (%o'17',            !MASK FOR LOW WORD                     ZZZ
;    2021  1                    %o'377', %o'7777', %o'177777', %o'177777',                                       !ZZZ
;    2022  1                    %o'177777', %o'177777', %o'177777'),                                             !ZZZ
;    2023  1          RNDMS1 :  vector [8, word] initial (%o'0000',          !MASK FOR HIGH WORD                    ZZZ
;    2024  1                    %o'0000', %o'0000', %o'0000', %o'17',                                            !ZZZ
;    2025  1                    %o'377', %o'7777', %o'177777'),                                                  !ZZZ
;    2026  1
;    2027  1          PAT02 : vector [2] initial (1,                                  ! PATTERN 2
;    2028  1                    %o'000000'),
;    2029  1          PAT03 : vector [2] initial (1,                                  ! PATTERN 3
;    2030  1                    %o'177777'),
;    2031  1          PAT04 : vector [2] initial (1,                                  ! PATTERN 4
;    2032  1                    %o'105613'),
;    2033  1          PAT05 : vector [2] initial (1,                                  ! PATTERN 5
;    2034  1                    %o'031463'),
;    2035  1          PAT06 : vector [2] initial (1,                                  ! PATTERN 6
;    2036  1                    %o'030221'),
;    2037  1          PAT07 : vector [17] initial (16,                                ! PATTERN 7
;    2038  1                    %o'000001', %o'000003', %o'000007', %o'000017',
;    2039  1                    %o'000037', %o'000077', %o'000177', %o'000377',
;    2040  1                    %o'000777', %o'001777', %o'003777', %o'007777',
;    2041  1                    %o'017777', %o'037777', %o'077777', %o'177777'),
;    2042  1          PAT08 : vector [17] initial (16,                                ! PATTERN 8
;    2043  1                    %o'177776', %o'177774', %o'177770', %o'177760',
;    2044  1                    %o'177740', %o'177700', %o'177600', %o'177400',
;    2045  1                    %o'177000', %o'176000', %o'174000', %o'170000',
;    2046  1                    %o'160000', %o'140000', %o'100000', %o'000000'),
;    2047  1          PAT09 : vector [17] initial (16,                                ! PATTERN 9
;    2048  1                    rep 3 of (%o'000000'), rep 3 of (%o'177777'),
;    2049  1                    rep 2 of (%o'000000'), rep 2 of (%o'177777'),
;    2050  1                    %o'000000', %o'177777', %o'000000', %o'177777',
;    2051  1                    %o'000000', %o'177777'),
;    2052  1          PAT10 : vector [2] initial (1,                                  ! PATTERN 10
;    2053  1                    %o'133331'),
;    2054  1          PAT11 : vector [17] initial (16,                                ! PATTERN 11
;    2055  1                    rep 3 of (%o'052525'), rep 3 of (%o'125252'),
;    2056  1                    rep 2 of (%o'052525'), rep 2 of (%o'125252'),
;    2057  1                    %o'052525', %o'125252', %o'052525', %o'125252',
;    2058  1                    %o'052525', %o'125252'),
```

```
;    2059  1          PAT12 : vector [21] initial (20,                                ! PATTERN 12
;    2060  1                  rep 3 of (%o'026455'), rep 3 of (%o'151322'),
;    2061  1                  rep 2 of (%o'026455'), rep 2 of (%o'151322'),
;    2062  1                  rep 2 of (%o'026455'),
;    2063  1                  %o'151322', %o'026455', %o'151322', %o'026455',
;    2064  1                  %o'151322', %o'026455', %o'151322', %o'026455'),
;    2065  1          PAT13 : vector [2] initial (1,                                  ! PATTERN 13
;    2066  1                  %o'066666'),
;    2067  1          PAT14 : vector [17] initial (16,                                ! PATTERN 14
;    2068  1                  %o'000001', %o'000002', %o'000004', %o'000010',
;    2069  1                  %o'000020', %o'000040', %o'000100', %o'000200',
;    2070  1                  %o'000400', %o'001000', %o'002000', %o'004000',
;    2071  1                  %o'010000', %o'020000', %o'040000', %o'100000'),
;    2072  1          PAT15 : vector [17] initial (16,                                ! PATTERN 15
;    2073  1                  %o'177776', %o'177775', %o'177773', %o'177767',
;    2074  1                  %o'177757', %o'177737', %o'177677', %o'177577',
;    2075  1                  %o'177377', %o'176777', %o'175777', %o'173777',
;    2076  1                  %o'167777', %o'157777', %o'137777', %o'077777'),
;    2077  1          PAT16 : vector [17] initial (16,                                ! PATTERN 16
;    2078  1                  rep 3 of (%o'133331'), rep 3 of (%o'155554'),
;    2079  1                  rep 2 of (%o'133331'), rep 2 of (%o'155554'),
;    2080  1                  %o'133331', %o'155554', %o'133331', %o'155554',
;    2081  1                  %o'133331', %o'155554'),
;    2082  1          PAT17 : vector [22] initial (21,                                ! PATTERN 17
;    2083  1                  %o'000000', rep 2 of (%o'106466'),
;    2084  1                  rep 3 of (%o'071311'), rep 4 of (%o'106466'),
;    2085  1                  rep 5 of (%o'071311'), rep 6 of (%o'106466')),
;    2086  1          PAT18 : vector [22] initial (21,                                ! PATTERN 18
;    2087  1                  %o'106466', %o'000000', %o'071311',
;    2088  1                  rep 3 of (%o'106466'), rep 4 of (%o'071311'),
;    2089  1                  rep 5 of (%o'106466'), rep 6 of (%o'071311')),
;    2090  1          PAT19 : vector [22] initial (21,                                ! PATTERN 19
;    2091  1                  %o'000000', rep 2 of (%o'134631'),
;    2092  1                  rep 3 of (%o'043146'), rep 4 of (%o'134631'),
;    2093  1                  rep 5 of (%o'043146'), rep 6 of (%o'134631')),
;    2094  1          PAT20 : vector [22] initial (21,                                ! PATTERN 20
;    2095  1                  %o'134631', %o'000000', %o'043146',
;    2096  1                  rep 3 of (%o'134631'), rep 4 of (%o'043146'),
;    2097  1                  rep 5 of (%o'134631'), rep 6 of (%o'043146')),
.    2098  1          PAT21 : vector [2] initial (1,                                  ! PATTERN 21
;    2099  1                  %o'000000'),                                            !    (LBN)
;    2100  1          DPA_TBL : vector [DP_CNT] initial                               ! DATA PATTERN ADDRESS TABLE
;    2101  1                    (RDM_CNT, PAT02, PAT03, PAT04, PAT05,
;    2102  1                    PAT06, PAT07, PAT08, PAT09, PAT10, PAT11,
;    2103  1                    PAT12, PAT13, PAT14, PAT15, PAT16, PAT17,
;    2104  1                    PAT18, PAT19, PAT20, PAT21),
;    2105  1          BST_CNT : word initial (0),                 ! CURRENT SEQUENTIAL BLOCK COUNT
;    2106  1          BST_DEV : word initial (0),                 ! CURRENT SEQUENTIAL BLOCK DEVICE
;    2107  1          CURRENT_VECTOR : word,                      ! CURRENT DEVICE'S VECTOR ADDRESS
;    2108  1      !ZZZ    BRLEVEL : word,                         ! CURRENT DEVICE'S BR LEVEL              ZZZ
;    2109  1          DUOFF   : WORD,                             !DUP OFFSET INTO CST                     ZZZ
;    2110  1          DRS_START,                                  ! START OF THE SUPERVISOR
;    2111  1          APT_MODE : byte initial (byte (FALSE)),     ! FLAG SET IF EXERCISER RUNNING UNDER APT
```

```
;    2112  1          MAIL_BOX_TESTNUM,                ! ADDRESS OF TEST NUMBER LOCATION IN APT MAIL-BOX
;    2113  1          MAIL_BOX_SUBTST,                 ! ADDRESS OF SUB TEST NUMBER LOCATION IN APT MAIL BOX
;    2114  1          COMPARE_DATA : byte,             ! FLAG CLEARED TO BYPASS HOST COMPARES
;    2115  1          DRS_FLAGS: word,                 ! FLAGS USED IN START/RESTART OF THE EXERCISER
;    2116  1          RD_MAX_SEQ_CNT : word,           ! COUNT USED IN SEQUENTIAL ACCESS OPERATIONS
;    2117  1          RX_MAX_SEQ_CNT : word;           !
;    2118  1
;    2119  1      external routine
;    2120  1          NEX_TRAP : L$ISR novalue,
;    2121  1          TIME : L$ISR novalue,
;    2122  1          SET_CPAR : novalue,
;    2123  1          SET_UPAR : novalue,
;    2124  1          OUT_IODQ,
;    2125  1          IN_IODQ : novalue,
;    2126  1          GET_PKT,
;    2127  1          PUT_PKT : novalue,
;    2128  1          GET_RETPKT,
;    2129  1          PUT_RETPKT : novalue,
;    2130  1          GET_IO_BUFF : novalue,
;    2131  1          PUT_IO_BUFF : novalue,
;    2132  1          PUTA_BUFF : novalue,
;    2133  1          SEND,
;    2134  1          WAIT : novalue,
;    2135  1          MODULAS,                                                    !ZZZ
;    2136  1          DROP_CTLR : novalue,
;    2137  1          DRV_CTLERR : novalue,
;    2138  1          EMS_RP1 : novalue,
;    2139  1          EMS_EL : novalue,
;    2140  1          EMS_CMP : novalue,
;    2141  1          EMS_ERR : novalue,
;    2142  1          EMS_10 : novalue,
;    2143  1          EMS_12 : novalue,
;    2144  1          EMS_13 : novalue,
;    2145  1          EMS_14 : novalue,
;    2146  1          EMS_18 : novalue,
;    2147  1          EMS_21 : novalue,
;    2148  1          EMS_22 : NOVALUE,                                                   !ZZZ
;    2149  1          EMS_24 : novalue,
;    2150  1          EMS_30 : novalue;
```

```
;    2151  1     %sbttl 'TEST SECTION
;    2152  1
;    2153  1
;    2154  1     !+
;    2155  1     !       THIS SECTION CONTAINS THE TOP-LEVEL TEST CODE FOR THE RDRX EXERCISER.
;    2156  1     !       THE EXERCISER CONSISTS OF ONE TEST WHICH IS SUBDIVIDED INTO A NUMBER OF
;    2157  1     !       SUBTESTS. ALL SUBTESTS ARE DECLARED WITHIN THIS BLOCK.
;    2158  1     !-
;    2159  1
;    2160  1
;    2161  3     BGNTST;
;    2162  3
;    2163  3     local
;    2164  3         DUMMY_0 : word,
;    2165  3         DUMMY_1 : word;
;    2166  3
;    2167  3
;    2168  3
;    2169  3     EOP_FLAG = TRUE;                                    ! ASSUME NO UNIT AVAILABLE
;    2170  3     COMPARE_DATA = TRUE;                                ! ALLOW HOST COMAPRES IF ASKED FOR
;    2171  3     DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT);          !CLEAR DUP INIT FLAG     ZZZ
;    2172  3     HOE_FLAG = FALSE;                                   ! ASSUME 'HOE' FLAG NOT SET
;    2173  3     FORCED_ERROR = FALSE;                               ! INITIALIZE "FORCED ERROR" FLAG
;    2174  3
;    2175  3
;    2176  3     incr I from 0 to PKT_CNT - 1 do                     ! INITIALIZE PACKET AREA
;    2177  4         begin
;    2178  4
;    2179  4         incr J from 0 to PKT_LEN - 1 do
;    2180  4             MSCP_PKT [.I, .J, 0, 16, 0] = 0;
;    2181  4
;    2182  4         MSCP_PKT [.I, RSP_RECEIVED] = FALSE;
;    2183  3         end;
;    2184  3
;    2185  3     incr I from 0 to RP_CNT - 1 do                      ! INITIALIZE RESPONSE SAVE AREA
;    2186  3         incr J from 0 to RP_LEN - 1 do
;    2187  3             RETPKT [.I, .J, 0, 16, 0] = 0;
;    2188  3
;    2189  3     incr I from 0 to EP_CNT do                          ! INITIALIZE ERROR-LOG SAVE AREA
;    2190  4         begin
;    2191  4
;    2192  4         incr J from 0 to EP_LEN - 1 do
;    2193  4             ELOG_PKT [.I, .J, 0, 16, 0] = 0;
;    2194  4
;    2195  4         ELOG_PKT [.I, EL_CONTENTS] = EMPTY;
;    2196  3         end;
;    2197  3
;    2198  4     if BIT_TST (SWP_FLAGS, SWF_CWC)                     ! NO SIMULTANEOUS CNTR/HOST WRIE CHECKS
;    2199  3     then
;    2200  3         SWP_FLAGS = .SWP_FLAGS and (not SWF_HWC);
;    2201  3
;    2202  4     if BIT_TST (SWP_FLAGS, SWF_RDM)                     ! NO SIMULTANEOUS RANDOM/SEQUENTIAL SELECTS
;    2203  3     then
```

```
;    2204  3          SWP_FLAGS = .SWP_FLAGS and (not SWF_SEQ);
;    2205  3
;    2206  3      if not .INIT_OCCURED
;    2207  3      then
;    2208  4          begin
;    2209  4          DRS_START = .FREE_MEM_ADDR + 2 + (..FREE_MEM_ADDR * 2);          ! START OF SUPERVISOR
;    2210  4
;    2211  4
;    2212  4
;    2213  4      !-  THE FOLLOWING DETERMINES WHETHER THE TEST IS TO BE RUN IN APT MODE:                    !Z
ZZ                                                                                                          !Z
;    2214  4
ZZ                                                                                                          !Z
;    2215  5      IF BIT_TST (SWP_FLAGS, SWF_APT)                                  !IF APT                   !Z
ZZ
;    2216  4          then
;    2217  5              begin
;    2218  5              APT_MODE = TRUE;
;    2219  5              MAIL_BOX_TESTNUM = .DRS_START + %o'62' + %o'6';          ! APT MAIL-BOX IS OFFSET AT OCTAL 62 FROM
;    2220  5              MAIL_BOX_SUBTST = .DRS_START + %o'62' + %o'4';           ! BEGINNING OF SUPERVISOR
;    2221  4              end;
;    2222  4
;    2223  4
;    2224  4          NEX = FALSE;                                                 ! CHECK IF LINE CLOCK PRESENT
;    2225  4          CLK_PRESENT = FALSE;                                         !
;    2226  4          SETVEC (4, NEX_TRAP, PRI07);                                 ! SET TRAP CATCHER ADDRESS
;    2227  4          DUMMY_0 = .LINE_CLOCK;                                       ! TRY TO ADDRESS THE CLOCK
;    2228  4          DUMMY_1 = 0;                                                 ! DUMMY INSTRUCTION
;    2229  4          CLRVEC (4);                                                  ! RETURN LOC 4 TO THE SUPERVISOR
;    2230  4
;    2231  4
;    2232  4          if not .NEX
;    2233  4          then
;    2234  5              begin
;    2235  5              CLK_PRESENT = TRUE;                                      ! SET FLAG IF CLOCK PRESENT
;    2236  5              CLK_TICKS = 0;                                           ! INITIALIZE THE LINE CLOCK TICK COUNT
;    2237  5              HOURS = .SWP_TIME / 100;                                 ! TIME OF DAY (HOURS)
;    2238  5              MINUTES = (.SWP_TIME mod 100) + 1;                       ! TIME OF DAY (MINUTES)
;    2239  5
;    2240  5              while .MINUTES gequ 60 do                                ! NORMALIZE MINUTES
;    2241  6                  begin
;    2242  6                  MINUTES = .MINUTES - 60;
;    2243  6                  HOURS = .HOURS + 1;
;    2244  5                  end;
;    2245  5
;    2246  5              HOURS = ..HOURS mod 24;                                  ! NORMALIZE HOURS
;    2247  4              end;
;    2248  4
;    2249  3          end;
;    2250  3
;    2251  3
;    2252  3      if .CLK_PRESENT
;    2253  3      then
;    2254  4          begin
;    2255  4          SETVEC (%o'100', TIME, PRI06);                               ! LINE-CLOCK VECTOR
;    2256  4          LINE_CLOCK = BIT6;                                           ! START THE CLOCK
```

```
;    2257  3          end;
;    2258  3
;    2259  3      RFLAGS (DRS_FLAGS);                                    ! READ DRS FLAGS INTO LOC DRS_FLAGS
;    2260  3
;    2261  3      if BIT_TST (DRS_FLAGS, HOE) eql HOE
;    2262  3      then
;    2263  3          HOE_FLAG = TRUE;                                   ! SET FLAG IF 'HOE' SET
;    2264  3
;    2265  3
;    2266  3      INIT_TEST ();                                          ! INITIALIZE TEST ENVIRONMENT
;    2267  3
;    2268  3      incr CTLR from 0 to (MAX_CTLR - 1) do                  ! FOR EVERY CONTROLLER
;    2269  3
;    2270  3          if (.CST [.CTLR, STATE] eql ONLINE) and            ! IF CONTROLLER ONLINE
;    2271  3             (.DCT [.CTLR, STAT] eql ONLINE) and
;    2272  4             (.CST [.CTLR, U_CNT] gequ 0)
;    2273  3          then
;    2274  3              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + 4) by UNIT_SIZE do
;    2275  3
;    2276  3                  if .CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE   ! IF AT LEAST ONE UNIT ALIVE
;    2277  3                  then
;    2278  4                      begin
;    2279  4                      EOP_FLAG = FALSE;                      ! NOT END OF PASS
;    2280  4                      exitloop;
;    2281  3                      end;
;    2282  3
;    2283  3      if not .EOP_FLAG
;    2284  3      then
;    2285  3          MULTI_DRIVE ();                                    ! RUN MULTI-DRIVE TEST
;    2286  1      ENDTST;


                                        .TITLE  ZRQAM3 RD/RX EXERCISER
                                        .IDENT  /V02.2/
                                        .ENABL  AMA

000000                                  .PSECT  $GGG$,  RO
000000                          COMM.AREA:
                                        .BLKW   24
000050                          DPST:   .BLKW   2
000054                          MAX.LBN:.BLKW   4
000064                          STORAGE:.BLKW   4
000074                          ICOM.ADDR:
                                        .BLKW   1
000076                          ICST.ADDR:
                                        .BLKW   1
000100                          IDCT.ADDR:
                                        .BLKW   1
000102  000000V                 INT.ADDR:
                                        .WORD   AZINTO
000104                          ICTLR:  .BLKW   1
000106  000003                  RW.BALANCE:
```

```
                        .WORD   3
000110                  MX1:    .BLKW   1
000112                  MX2:    .BLKW   1
000114                  MAD1:   .BLKW   1
000116                  MAD2:   .BLKW   1
000120                  LAST.PKT:
                        .BLKW   3
000126                  RNDY0:  .BLKW   1
000130                  RNDY1:  .BLKW   1
000132                  FRAME.CNT:
                        .BLKW   1
000134                  R.STRING:
                        .BLKW   1
000136  127102          RNDYIN: .WORD   -50676
000140  143662                  .WORD   -34116
000142  036750                  .WORD   36750
000144  121624                  .WORD   -56154
000146  023267                  .WORD   23267
000150  036561                  .WORD   36561
000152  063714                  .WORD   63714
000154  160255                  .WORD   -17523
000156  134230                  .WORD   -43550
000160  000017          RNDMS0: .WORD   17
000162  000377                  .WORD   377
000164  007777                  .WORD   7777
000166  177777                  .WORD   -1
000170  177777                  .WORD   -1
000172  177777                  .WORD   -1
000174  177777                  .WORD   -1
000176  177777                  .WORD   -1
000200  000000          RNDMS1: .WORD   0
000202  000000                  .WORD   0
000204  000000                  .WORD   0
000206  000000                  .WORD   0
000210  000017                  .WORD   17
000212  000377                  .WORD   377
000214  007777                  .WORD   7777
000216  177777                  .WORD   -1
000220  000001          PAT02:  .WORD   1
000222  000000                  .WORD   0
000224  000001          PAT03:  .WORD   1
000226  177777                  .WORD   -1
000230  000001          PAT04:  .WORD   1
000232  105613                  .WORD   -72165
000234  000001          PAT05:  .WORD   1
000236  031463                  .WORD   31463
000240  000001          PAT06:  .WORD   1
000242  030221                  .WORD   30221
000244  000020          PAT07:  .WORD   20
000246  000001                  .WORD   1
000250  000003                  .WORD   3
000252  000007                  .WORD   7
000254  000017                  .WORD   17
```

```
000256  000037                          .WORD   37
000260  000077                          .WORD   77
000262  000177                          .WORD   177
000264  000377                          .WORD   377
000266  000777                          .WORD   777
000270  001777                          .WORD   1777
000272  003777                          .WORD   3777
000274  007777                          .WORD   7777
000276  017777                          .WORD   17777
000300  037777                          .WORD   37777
000302  077777                          .WORD   77777
000304  177777                          .WORD   -1
000306  000020                  PAT08:  .WORD   20
000310  177776                          .WORD    2
000312  177774                          .WORD   -4
000314  177770                          .WORD   -10
000316  177760                          .WORD    20
000320  177740                          .WORD   -40
000322  177700                          .WORD   -100
000324  177600                          .WORD   -200
000326  177400                          .WORD   -400
000330  177000                          .WORD   -1000
000332  176000                          .WORD   -2000
000334  174000                          .WORD   -4000
000336  170000                          .WORD   -10000
000340  160000                          .WORD   -20000
000342  140000                          .WORD   -40000
000344  100000                          .WORD   -100000
000346  000000                          .WORD   0
000350  000020                  PAT09:  .WORD   20
000352  000000                          .WORD   0
000354  000000                          .WORD   0
000356  000000                          .WORD   0
000360  177777                          .WORD   -1
000362  177777                          .WORD    1
000364  177777                          .WORD   -1
000366  000000                          .WORD   0
000370  000000                          .WORD   0
000372  177777                          .WORD   -1
000374  177777                          .WORD   -1
000376  000000                          .WORD   0
000400  177777                          .WORD   -1
000402  000000                          .WORD   0
000404  177777                          .WORD   -1
000406  000000                          .WORD   0
000410  177777                          .WORD   -1
000412  000001                  PAT10:  .WORD   1
000414  133331                          .WORD   -44447
000416  000020                  PAT11:  .WORD   20
000420  052525                          .WORD   52525
000422  052525                          .WORD   52525
000424  052525                          .WORD   52525
000426  125252                          .WORD   -52526
```

```
000430  125252                          .WORD    52526
000432  125252                          .WORD    -52526
000434  052525                          .WORD    52525
000436  052525                          .WORD    52525
000440  125252                          .WORD    -52526
000442  125252                          .WORD    -52526
000444  052525                          .WORD    52525
000446  125252                          .WORD    -52526
000450  052525                          .WORD    52525
000452  125252                          .WORD    -52526
000454  052525                          .WORD    52525
000456  125252                          .WORD    -52526
000460  000024                  PAT12:  .WORD    24
000462  026455                          .WORD    26455
000464  026455                          .WORD    26455
000466  026455                          .WORD    26455
000470  151322                          .WORD    -26456
000472  151322                          .WORD    -26456
000474  151322                          .WORD    -26456
000476  026455                          .WORD    26455
000500  026455                          .WORD    26455
000502  151322                          .WORD    -26456
000504  151322                          .WORD    -26456
000506  026455                          .WORD    26455
000510  026455                          .WORD    26455
000512  151322                          .WORD    -26456
000514  026455                          .WORD    26455
000516  151322                          .WORD    -26456
000520  026455                          .WORD    26455
000522  151322                          .WORD    -26456
000524  026455                          .WORD    26455
000526  151322                          .WORD    -26456
000530  026455                          .WORD    26455
000532  000001                  PAT13:  .WORD    1
000534  066666                          .WORD    66666
000536  000020                  PAT14:  .WORD    20
000540  000001                          .WORD    1
000542  000002                          .WORD    2
000544  000004                          .WORD    4
000546  000010                          .WORD    10
000550  000020                          .WORD    20
000552  000040                          .WORD    40
000554  000100                          .WORD    100
000556  000200                          .WORD    200
000560  000400                          .WORD    400
000562  001000                          .WORD    1000
000564  002000                          .WORD    2000
000566  004000                          .WORD    4000
000570  010000                          .WORD    10000
000572  020000                          .WORD    20000
000574  040000                          .WORD    40000
000576  100000                          .WORD    100000
000600  000020                  PAT15:  .WORD    20
```

```
000602  177776                          .WORD   2
000604  177775                          .WORD   3
000606  177773                          .WORD   5
000610  177767                          .WORD   11
000612  177757                          .WORD   21
000614  177737                          .WORD   41
000616  177677                          .WORD   101
000620  177577                          .WORD   -201
000622  177377                          .WORD   401
000624  176777                          .WORD   1001
000626  175777                          .WORD   2001
000630  173777                          .WORD   -4001
000632  167777                          .WORD   -10001
000634  157777                          .WORD   20001
000636  137777                          .WORD   40001
000640  077777                          .WORD   77777
000642  000020                  PAT16:  .WORD   20
000644  133331                          .WORD   -44447
000646  133331                          .WORD   44447
000650  133331                          .WORD   -44447
000652  155554                          .WORD   -22224
000654  155554                          .WORD   -22224
000656  155554                          .WORD   -22224
000660  133331                          .WORD   -44447
000662  133331                          .WORD   -44447
000664  155554                          .WORD   -22224
000666  155554                          .WORD   -22224
000670  133331                          .WORD   -44447
000672  155554                          .WORD   -22224
000674  133331                          .WORD   -44447
000676  155554                          .WORD   -22224
000700  133331                          .WORD   -44447
000702  155554                          .WORD   -22224
000704  000025                  PAT17:  .WORD   25
000706  000000                          .WORD   0
000710  106466                          .WORD   -71312
000712  106466                          .WORD   -71312
000714  071311                          .WORD   71311
000716  071311                          .WORD   71311
000720  071311                          .WORD   71311
000722  106466                          .WORD   -71312
000724  106466                          .WORD   -71312
000726  106466                          .WORD   71312
000730  106466                          .WORD   -71312
000732  071311                          .WORD   71311
000734  071311                          .WORD   71311
000736  071311                          .WORD   71311
000740  071311                          .WORD   71311
000742  071311                          .WORD   71311
000744  106466                          .WORD   -71312
000746  106466                          .WORD   -71312
000750  106466                          .WORD   71312
000752  106466                          .WORD   71312
```

M5


```
000754  106466                          .WORD    71312
000756  106466                          .WORD    71312
000760  000025          PAT18:          .WORD    25
000762  106466                          .WORD    -71312
000764  000000                          .WORD    0
000766  071311                          .WORD    71311
000770  106466                          .WORD    -71312
000772  106466                          .WORD    -71312
000774  106466                          .WORD    71312
000776  071311                          .WORD    71311
001000  071311                          .WORD    71311
001002  071311                          .WORD    71311
001004  071311                          .WORD    71311
001006  106466                          .WORD    71312
001010  106466                          .WORD    -71312
001012  106466                          .WORD    -71312
001014  106466                          .WORD    71312
001016  106466                          .WORD    -71312
001020  071311                          .WORD    71311
001022  071311                          .WORD    71311
001024  071311                          .WORD    71311
001026  071311                          .WORD    71311
001030  071311                          .WORD    71311
001032  071311                          .WORD    71311
001034  000025          PAT19:          .WORD    25
001036  000000                          .WORD    0
001040  134631                          .WORD    -43147
001042  134631                          .WORD    -43147
001044  043146                          .WORD    43146
001046  043146                          .WORD    43146
001050  043146                          .WORD    43146
001052  134631                          .WORD    -43147
001054  134631                          .WORD    -43147
001056  134631                          .WORD    -43147
001060  134631                          .WORD    43147
001062  043146                          .WORD    43146
001064  043146                          .WORD    43146
001066  043146                          .WORD    43146
001070  043146                          .WORD    43146
001072  043146                          .WORD    43146
001074  134631                          .WORD    -43147
001076  134631                          .WORD    -43147
001100  134631                          .WORD    -43147
001102  134631                          .WORD    -43147
001104  134631                          .WORD    -43147
001106  134631                          .WORD    43147
001110  000025          PAT20:          .WORD    25
001112  134631                          .WORD    43147
001114  000000                          .WORD    0
001116  043146                          .WORD    43146
001120  134631                          .WORD    43147
001122  134631                          .WORD    43147
001124  134631                          .WORD    43147
```

```
001126  043146                          .WORD   43146
001130  043146                          .WORD   43146
001132  043146                          .WORD   43146
001134  043146                          .WORD   43146
001136  134631                          .WORD   -43147
001140  134631                          .WORD    43147
001142  134631                          .WORD   -43147
001144  134631                          .WORD   -43147
001146  134631                          .WORD   -43147
001150  043146                          .WORD   43146
001152  043146                          .WORD   43146
001154  043146                          .WORD   43146
001156  043146                          .WORD   43146
001160  043146                          .WORD   43146
001162  043146                          .WORD   43146
001164  000001                  PAT21:  .WORD   1
001166  000000                          .WORD   0
001170  G00000G                 DPA.TBL:.WORD   RDM.CNT
001172  000220'                         .WORD   PAT02
001174  000224'                         .WORD   PAT03
001176  000230'                         .WORD   PAT04
001200  000234'                         .WORD   PAT05
001202  000240'                         .WORD   PAT06
001204  000244'                         .WORD   PAT07
001206  000306'                         .WORD   PAT08
001210  000350'                         .WORD   PAT09
001212  000412'                         .WORD   PAT10
001214  000416'                         .WORD   PAT11
001216  000460'                         .WORD   PAT12
001220  000532'                         .WORD   PAT13
001222  000536'                         .WORD   PAT14
001224  000600'                         .WORD   PAT15
001226  000642'                         .WORD   PAT16
001230  000704'                         .WORD   PAT17
001232  000760'                         .WORD   PAT18
001234  001034'                         .WORD   PAT19
001236  001110'                         .WORD   PAT20
001240  001164'                         .WORD   PAT21
001242  000000                  BST.CNT:.WORD   0
001244  000000                  BST.DEV:.WORD   0
001246                          CURRENT.VECTOR:
                                        .BLKW   1
001250                          DUOFF:  .BLKW   1
001252                          DRS.START:
                                        .BLKW   1
001254                          APT.MODE:
001254      000                         .BYTE   0
                                        .EVEN
001256                          MAIL.BOX.TESTNUM:
                                        .BLKW   1
001260                          MAIL.BOX.SUBTST:
                                        .BLKW   1
001262                          COMPARE.DATA:
```

```
                                .BLKB   1
                                .EVEN
001264                  DRS.FLAGS:
                                .BLKW   1
001266                  RD.MAX.SEQ.CNT:
                                .BLKW   1
001270                  RX.MAX.SEQ.CNT:
                                .BLKW   1


                        .GLOBL  CST, CST.ADDR, DCT, DCT.ADDR, RDRX.ADDR
                        .GLOBL  IRDRX.ADDR, BST, TALLY, T.ADDR
                        .GLOBL  DUPPKT, TRK.SGN, RDM.CNT, RANDOM
                        .GLOBL  C.ERR.TBL, MSCP.PKT, IPKT.ADDR
                        .GLOBL  PKT.USE, RETPKT, RP.USE, RP.INDX
                        .GLOBL  RP.ADDR, ELOG.PKT, BUFF.ADDR, BUFF.OWN
                        .GLOBL  IODQ, IODQ.IN, IODQ.OUT, ENTRY.REASON
                        .GLOBL  EOP.FLAG, DUP.FLAGS, CCTLR, CDISK
                        .GLOBL  CUOFF, CTLR.CNT, DUR, QIO, FREE.MEM.ADDR
                        .GLOBL  BYTS.PER.QIO, ST.CODE, SB.CODE
                        .GLOBL  STEP, OF.RC, SA.REG, CMD.TIME
                        .GLOBL  NEX, CRN.LOW, CRN.HIGH, TEMP1
                        .GLOBL  TEMP2, CREDIT.BAL, NEXT.PKT.USE
                        .GLOBL  HOURS, MINUTES, CLK.TICKS, CLK.PRESENT
                        .GLOBL  HOE.FLAG, FORCED.ERROR, FERO.LBN
                        .GLOBL  FER1.LBN, FER.BC, INIT.OCCURED
                        .GLOBL  ADDR.VECT.OK, S.PATTERN, S.DUPPKT
                        .GLOBL  P.INDEX, RD.COUNT, BRLEVEL, D.FAIL
                        .GLOBL  DBM12, DBM18, DBM19, DBM20, DBM21
                        .GLOBL  DBM22, DBM23, DBM25, DBM26, DBM27
                        .GLOBL  DBM29, DBM108, DBM109, DBM111
                        .GLOBL  DBM112, DBM120, DBM121, EH.0, EH.1
                        .GLOBL  EH.2, EH.3, EH.4, EH.5, EH.6, EH.7
                        .GLOBL  EH.8, EH.9, EH.10, EH.12, EH.13
                        .GLOBL  MSG.02, MSG.03, EGS.02, EGD.10
                        .GLOBL  EGD.11, EGD.12, EGD.13, EGD.14
                        .GLOBL  EGD.15, EGD.16, EGD.17, EGD.18
                        .GLOBL  EGD.19, EGD.20, EGD.21, EGD.22
                        .GLOBL  EGD.23, EGD.24, EGH.30, DF.MSG
                        .GLOBL  HRD.MSG, SFT.MSG, HRD.SUB, CRLF
                        .GLOBL  SWP.ERROR, SWP.XFER, SWP.FLAGS
                        .GLOBL  DUPROUND, SWP.RAT, SWP.DPAT, SWP.UCNT
                        .GLOBL  SWP.TIME, SWP.UDPAT, L$LUN, L$UNIT
                        .GLOBL  NEX.TRAP, TIME, SET.CPAR, SET.UPAR
                        .GLOBL  OUT.IODQ, IN.IODQ, GET.PKT, PUT.PKT
                        .GLOBL  GET.RETPKT, PUT.RETPKT, GET.IO.BUFF
                        .GLOBL  PUT.IO.BUFF, PUTA.BUFF, SEND, WAIT
                        .GLOBL  MODULAS, DROP.CTLR, DRV.CTLERR
                        .GLOBL  EMS.RP1, EMS.EL, EMS.CMP, EMS.ERR
                        .GLOBL  EMS.10, EMS.12, EMS.13, EMS.14
                        .GLOBL  EMS.18, EMS.21, EMS.22, EMS.24
                        .GLOBL  EMS.30
```

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

```
        020000                          IER=-           20000
        040000                          LOE=-           40000
        100000                          HOE=-           -100000


                                        .SBTTL  $T1 TEST SECTION
000000                                  .PSECT  $CODE$,  RO

000000  004137  000000G        $T1:    JSR     R1,$SAVE3               ;                       2150
000004  112737  000001  000000G       MOVB    #1,EOP.FLAG             ;                       2169
000012  112737  000001  001262'       MOVB    #1,COMPARE.DATA         ;                       2170
000020  042737  000002  000000G       BIC     #2,DUP.FLAGS            ;                       2171
000026  105037  000000G               CLRB    HOE.FLAG                ;                       2172
000032  105037  000000G               CLRB    FORCED.ERROR            ;                       2173
000036  005002                        CLR     R2                      ; I                     2176
000040  010246                1$:     MOV     R2,-(SP)                ; I,*                   2180
000042  012746  000043                MOV     #43,-(SP)
000046  004737  000000G               JSR     PC,BL$MUL
000052  005001                        CLR     R1                      ; J                     2179
000054  010003                2$:     MOV     R0,R3                   ;                       2180
000056  060103                        ADD     R1,R3                   ; J,*
000060  006303                        ASL     R3
000062  005063  000000G               CLR     MSCP.PKT(R3)
000066  005201                        INC     R1                      ; J                     2179
000070  020127  000042                CMP     R1,#42                  ; J,*
000074  003767                        BLE     2$
000076  010216                        MOV     R2,(SP)                 ; I,*                   2182
000100  012746  000106                MOV     #106,-(SP)
000104  004737  000000G               JSR     PC,BL$MUL
000110  105060  000005G               CLRB    MSCP.PKT+5(R0)
000114  062706  000006                ADD     #6,SP                   ;                       2177
000120  005202                        INC     R2                      ; I                     2176
000122  020227  000013                CMP     R2,#13                  ; I,*
000126  003744                        BLE     1$
000130  005002                        CLR     R2                      ; I                     2185
000132  005001                3$:     CLR     R1                      ; J                     2186
000134  010200                4$:     MOV     R2,R0                   ; I,*                   2187
000136  060100                        ADD     R1,R0                   ; J,*
000140  006300                        ASL     R0
000142  005060  000000G               CLR     RETPKT(R0)
000146  005201                        INC     R1                      ; J                     2186
000150  020127  000025                CMP     R1,#25                  ; J,*
000154  003767                        BLE     4$
000156  062702  000026                ADD     #26,R2                  ; *,I                   2185
000162  020227  000232                CMP     R2,#232                 ; I,*
000166  003761                        BLE     3$
000170  005002                        CLR     R2                      ; I                     2189
000172  010246                5$:     MOV     R2,-(SP)                ; I,*                   2193
000174  012746  000041                MOV     #41,-(SP)
000200  004737  000000G               JSR     PC,BL$MUL
000204  005001                        CLR     R1                      ; J                     2192
000206  010003                6$:     MOV     R0,R3                   ;                       2193
000210  060103                        ADD     R1,R3                   ; J,*
```

```
000212  006303                          ASL     R3
000214  005063  000000G                 CLR     ELOG.PKT(R3)
000220  005201                          INC     R1                      ; J
000222  020127  000040                  CMP     R1,#40                  ; J,*                   2192
000226  003767                          BLE     6$
000230  010216                          MOV     R2,(SP)                 ; I,*                   2195
000232  012746  000102                  MOV     #102,-(SP)
000236  004737  000000G                 JSR     PC,BL$MUL
000242  105060  000001G                 CLRB    ELOG.PKT+1(R0)
000246  062706  000006                  ADD     #6,SP                   ;                       2190
000252  005202                          INC     R2                      ; I                     2189
000254  020227  000014                  CMP     R2,#14                  ; I,*
000260  003744                          BLE     5$
000262  032737  000020  000000G         BIT     #20,SWP.FLAGS           ;                       219*
000270  001403                          BEQ     7$
000272  042737  000040  000000G         BIC     #40,SWP.FLAGS           ;                       2200
000300  032737  000002  000000G  7$:     BIT     #2,SWP.FLAGS            ;                       2202
000306  001403                          BEQ     8$
000310  042737  001000  000000G         BIC     #1000,SWP.FLAGS         ;                       2204
000316  132737  000001  000000G  8$:     BITB    #1,INIT.OCCURED         ;                       2206
000324  001143                          BNE     13$
000326  017700  000000G                 MOV     @FREE.MEM.ADDR,R0       ;                       2209
000332  006300                          ASL     R0
000334  063700  000000G                 ADD     FREE.MEM.ADDR,R0
000340  010037  001252'                 MOV     R0,DRS.START
000344  062737  000002  001252'         ADD     #2,DRS.START
000352  032737  000001  000000G         BIT     #1,SWP.FLAGS            ;                       2215
000360  001417                          BEQ     9$
000362  112737  000001  001254'         MOVB    #1,APT.MODE             ;                       2218
000370  013737  001252'  001256'         MOV     DRS.START,MAIL.BOX.TESTNUM ;                    2219
000376  062737  000070  001256'         ADD     #70,MAIL.BOX.TESTNUM
000404  013737  001252'  001260'         MOV     DRS.START,MAIL.BOX.SUBTST ;                     2220
000412  062737  000066  001260'         ADD     #66,MAIL.BOX.SUBTST
000420  005037  000000G          9$:    CLR     NEX                     ;                       2224
000424  105037  000000G                 CLRB    CLK.PRESENT             ;                       2225
000430  012746  000340                  MOV     #340,-(SP)              ;                       2226
000434  012746  000000G                 MOV     #NEX.TRAP,-(SP)
000440  012746  000004                  MOV     #4,-(SP)
000444  012746  000003                  MOV     #3,-(SP)
000450  104437                          TRAP    37
000452  012700  000004                  MOV     #4,R0                   ;                       2229
000456  104436                          TRAP    36
000460  032737  000001  000000G         BIT     #1,NEX                  ;                       2232
000466  001060                          BNE     12$
000470  112737  000001  000000G         MOVB    #1,CLK.PRESENT          ;                       2235
000476  005037  000000G                 CLR     CLK.TICKS               ;                       2236
000502  013716  000000G                 MOV     SWP.TIME,(SP)           ;                       2237
000506  012746  000144                  MOV     #144,-(SP)
000512  004737  000000G                 JSR     PC,BL$DIV
000516  110037  000000G                 MOVB    R0,HOURS
000522  013716  000000G                 MOV     SWP.TIME,(SP)           ;                       2238
000526  012746  000144                  MOV     #144,-(SP)
000532  004737  000000G                 JSR     PC,BL$MOD
```

```
000536  010001                          MOV     R0,R1
000540  005201                          INC     R1
000542  110137  000000G                 MOVB    R1,MINUTES
000546  123727  000000G 000074  10$:    CMPB    MINUTES,#74             ;                       2240
000554  103412                          BLO     11$
000556  005000                          CLR     R0                      ;                       2242
000560  153700  000000G                 BISB    MINUTES,R0
000564  162700  000074                  SUB     #74,R0
000570  110037  000000G                 MOVB    R0,MINUTES
000574  105237  000000G                 INCB    HOURS                   ;                       2243
000600  000762                          BR      10$                     ;                       2240
000602  005016                  11$:    CLR     (SP)                    ;                       2246
000604  113716  000000G                 MOVB    HOURS,(SP)
000610  012746  000030                  MOV     #30,-(SP)
000614  004737  000000G                 JSR     PC,$BL$MOD
000620  110037  000000G                 MOVB    R0,HOURS
000624  062706  000006                  ADD     #6,SP                   ;                       2234
000630  062706  000010          12$:    ADD     #10,SP                  ;                       2208
000634  132737  000001  000000G 13$:    BITB    #1,CLK.PRESENT          ;                       2252
000642  001416                          BEQ     14$
000644  012746  000300                  MOV     #300,-(SP)              ;                       2255
000650  012746  000000G                 MOV     #TIME,-(SP)
000654  012746  000100                  MOV     #100,-(SP)
000660  012746  000003                  MOV     #3,-(SP)
000664  104437                          TRAP    37
000666  012737  000100  177546          MOV     #100,@#177546           ;                       2256
000674  062706  000010                  ADD     #10,SP                  ;                       2254
000700  104421                  14$:    TRAP    21                      ;                       2259
000702  010037  001264'                 MOV     R0,DRS.FLAGS
000706  042700  077777                  BIC     #77777,R0               ;                       2261
000712  020027  100000                  CMP     R0,#-100000
000716  001003                          BNE     15$
000720  012700  000001                  MOV     #1,R0
000724  000401                          BR      16$
000726  005000                  15$:    CLR     R0
000730  020027  100000          16$:    CMP     R0,#-100000
000734  001003                          BNE     17$
000736  112737  000001  000000G         MOVB    #1,MDE.FLAG             ;                       2263
000744  004737  000000V         17$:    JSR     PC,INIT.TEST            ;                       2266
000750  005002                          CLR     R2                      ; CTLR                  2268
000752  010246                  18$:    MOV     R2,-(SP)                ; CTLR,*                2270
000754  012746  000126                  MOV     #126,-(SP)
000760  004737  000000G                 JSR     PC,$BL$MUL
000764  022626                          CMP     (SP)+,(SP)+
000766  005760  000002G                 TST     CST+2(R0)
000772  100040                          BPL     22$
000774  010246                          MOV     R2,-(SP)                ; CTLR,*                2271
000776  012746  000022                  MOV     #22,-(SP)
001002  004737  000000G                 JSR     PC,$BL$MUL
001006  022626                          CMP     (SP)+,(SP)+
001010  005760  000000G                 TST     DCT(R0)
001014  100027                          BPL     22$
001016  010246                          MOV     R2,-(SP)                ; CTLR,*                2276
```

```
001020  012746  000053                          MOV     #53,-(SP)
001024  004737  000000G                         JSR     PC,BL$MUL
001030  012701  000003                          MOV     #3,R1           ; *,OFFSET                      2274
001034  010003                          19$:    MOV     R0,R3           ;                               2276
001036  060103                                  ADD     R1,R3           ; OFFSET,*
001040  006303                                  ASL     R3
001042  032763  020000  000000G                 BIT     #20000,CST(R3)
001050  001403                                  BEQ     20$
001052  105037  000000G                         CLRB    EOP.FLAG        ;                               2279
001056  000405                                  BR      21$             ;                               2278
001060  062701  000012                  20$:    ADD     #12,R1          ; *,OFFSET                      2274
001064  020127  000042                          CMP     R1,#42          ; OFFSET,*
001070  003761                                  BLE     19$
001072  022626                          21$:    CMP     (SP)+,(SP)+
001074  005202                          22$:    INC     R2              ; CTLR                          2268
001076  000243                                  .WORD CLV!CLC

001100  003724                                  BLE     18$
001102  132737  000001  000000G                 BITB    #1,EOP.FLAG     ;                               2283
001110  001002                                  BNE     23$
001112  004737  000000V                         JSR     PC,MULTI.DRIVE  ;                               2285
001116  000207                          23$:    RTS     PC              ;                               2150

; Routine Size:  296 words,     Routine Base:  $CODE$ + 0000
; Maximum stack depth per invocation:  12 words




                                                .SBTTL  T1 TEST SECTION
000000  004737  000000'                  T1::
000000                                  1$:     JSR     PC,$T1          ;                               2285
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  $CODE$ + 1120
; Maximum stack depth per invocation:  2 words
```

```
;    2287  1     %sbttl 'INITIALIZATION TEST ROUTINES'
;    2288  1
;    2289  1     GLOBAL routine INIT_TEST : novalue =
;    2290  1
;    2291  1     !+
;    2292  1     !       THE INITIALIZATION TEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
;    2293  1     !       DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
;    2294  1     !       BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI-DRIVE TEST
;    2295  1     !       OR THE DM EXERCISER.
;    2296  1     !
;    2297  1     !       BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
;    2298  1     !       INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
;    2299  1     !       MARKED OFFLINE IN THEIR DCT AND CST. FOR THOSE DEVICES WHICH SURVIVE
;    2300  1     !       THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
;    2301  1     !       EACH DISK VIA ROUTINE "ACCESS". THE INITIALIZATION TEST IS DEEMED A
;    2302  1     !       SUCCESS IF A BLOCK ON THE INNER TRACK OF EACH DISK CAN BE ACCESSED.
;    2303  1     !-
;    2304  1
;    2305  2         begin
;    2306  2         DRIVER_INIT ();                         ! INIT DRIVER DATA AND DEVICES
;    2307  2
;    2308  2         incr CTLR from 0 to (MAX_CTLR - 1) do   ! FOR EACH CONTROLLER
;    2309  3             begin
;    2310  3             SET_CPAR (.CTLR);                   ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    2311  3
;    2312  3             if .CST_ADDR [STATE] eql ONLINE     ! IF CONTROLLER IS STILL ALIVE
;    2313  3             then                                ! FOR EACH DISK
;    2314  3
;    2315  3                 incr OFFSET from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do
;    2316  3
;    2317  3                 if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    2318  3                    (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    2319  4                    (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2320  3                 then
;    2321  4                     begin
;    2322  4                     SET_UPAR (.OFFSET);         ! SET UP UNIT-RELATED DATA ITEMS
;    2323  5                     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)          !ZZZ
;    2324  4                        THEN ACCESS ();                                 !ZZZ
;    2325  4                                            !SKIP IF DUP CAUSED INIT ZZZ
;    2326  4
;    2327  3                     end;                        ! IF UNIT IS PRESENT AND ONLINE
;    2328  3
;    2329  2             end;                                ! CONTROLLER LOOP
;    2330  2
;    2331  1         end;                                    ! ROUTINE INIT_TEST


                                .SBTTL  INIT.TEST INITIALIZATION TEST ROUTINES
000000  004137  000000G     INIT.TEST::
                            JSR     R1,$SAVE2                       ;                       2289
000004  004737  000000V     JSR     PC,DRIVER.INIT                  ;                       2306
000010  005002              CLR     R2                              ; CTLR                  2308
000012  010246              1$:     MOV     R2,-(SP)                ; CTLR,*                2310
```

```
000014  004737  000000G                        JSR     PC,SET.CPAR
000020  013700  000000G                        MOV     CST.ADDR,R0                                     ;                                       2312
000024  005760  000002                         TST     2(R0)
000030  100035                                 BPL     4$
000032  012701  000003                         MOV     #3,R1                                           ; *,OFFSET                              2315
000036  010100                         2$:     MOV     R1,R0                                           ; OFFSET,*                              2317
000040  006300                                 ASL     R0
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  032710  040000                         BIT     #40000,(R0)
000052  001417                                 BEQ     3$
000054  032710  020000                         BIT     #20000,(R0)                                     ;                                       2318
000060  001414                                 BEQ     3$
000062  032710  010000                         BIT     #10000,(R0)                                     ;                                       2319
000066  001011                                 BNE     3$
000070  010116                                 MOV     R1,(SP)                                         ; OFFSET,*                              2322
000072  004737  000000G                        JSR     PC,SET.UPAR
000076  032737  000002  000000G                BIT     #2,DUP.FLAGS                                    ;                                       2323
000104  001002                                 BNE     3$
000106  004737  000000V                        JSR     PC,ACCESS                                       ;                                       2324
000112  062701  000012                 3$:     ADD     #12,R1                                          ; *,OFFSET                              2315
000116  020127  000041                         CMP     R1,#41                                          ; OFFSET,*
000122  003745                                 BLE     2$
000124  005726                         4$:     TST     (SP)+                                           ;                                       2309
000126  005202                                 INC     R2                                              ; CTLR                                  2308
000130  000243                                 .WORD CLV!CLC

000132  003727                                 BLE     1$
000134    0207                                 RTS     PC                                              ;                                       2289

; Routine Size:  47 words,      Routine Base:  $CODE$ + 1134
; Maximum stack depth per invocation:  5 words
```

```
;    2332  1     GLOBAL routine DRIVER_INIT : novalue =
;    2333  1
;    2334  1     !+
;    2335  1     !       THIS ROUTINE IS EQUIVALENT IN FUNCTION TO THE INITIALIZATION ENTRY
;    2336  1     !       POINT OF A STANDARD DEVICE DRIVER. ITS RESPONSIBILITY IS TO INITIALIZE
;    2337  1     !       DRIVER DATA, AND TO BRING EACH RDRX CONTROLLER AND UNIT (DISK)
;    2338  1     !       ONLINE.
;    2339  1     !-
;    2340  1
;    2341  2         begin
;    2342  2
;    2343  2         local
;    2344  2             PKT_ADDR;
;    2345  2
;    2346  2         PKT_ADDR = MSCP_PKT + 10;                                   ! ADDR (TEXT + 0) OF 1ST MSCP PKT
;    2347  2         NEXT_PKT_USE = 0;                                           ! NEXT PACKET TO ALLOCATE
;    2348  2
;    2349  2         incr COUNT from 0 to (PKT_CNT - 1) do                       ! FOR EACH MSCP PACKET
;    2350  3             begin
;    2351  3             PKT_USE [.COUNT] = -1;                                  ! MARK PACKET FREE
;    2352  3             MSCP_PKT [.COUNT, PKT_LO] = .PKT_ADDR;                  ! LOAD ADDR INTO BUFFER DESCRIPTOR
;    2353  3             MSCP_PKT [.COUNT, PKT_HI] = 0;
;    2354  3             MSCP_PKT [.COUNT, CONNID] = CID_DISK;                   ! SET CONNECTION ID TO MSCP ID
;    2355  3             PKT_ADDR = .PKT_ADDR + (PKT_LEN * 2);                   ! ADVANCE ADDR TO NEXT PACKET
;    2356  2             end;
;    2357  2
;    2358  2         incr CTLR from 0 to (MAX_CTLR - 1) do                       ! FOR EACH CONTROLLER
;    2359  2
;    2360  2             if .CST [.CTLR, IP_ADDR] neqa 0                         ! IF CONTROLLER IS PRESENT
;    2361  2             then
;    2362  3                 begin
;    2363  3                 SET_CPAR (.CTLR);                                   ! CURRENT CONTROLLER PARAMETERS
;    2364  3                 CURRENT_VECTOR = .CST_ADDR [VEC_ADDR];              ! CURRENT CONTROLLER'S VECTOR
;    2365  3                 BRLEVEL = .CST_ADDR [BR_LEV] ^ 5;                   ! SET CURRENT CONTROLLER'S BR LEVEL
;    2366  3                 CTLR_INIT ();                                       ! INIT DEVICE AND CTLR DATA
;    2367  3
;    2368  3                 if .DCT_ADDR [STAT] eql ONLINE                      ! IF CONTROLLER IS STILL ALIVE
;    2369  3                 then                                                ! FOR EACH DIAK UNIT
;    2370  3
;    2371  3                     incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    2372  3
;    2373  3                         if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and  ! IF UNIT EXISTS
;    2374  4                            (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2375  3                         then
;    2376  4                             begin
;    2377  4                             CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_0] = %o'40';      ! BLANK DEVICE NAME
;    2378  4                             CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_1] = %o'40';      !
;    2379  4                             CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_2] = %o'40';      !
;    2380  4                             CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_3] = %o'40';      !
;    2381  4                             SET_UPAR (.OFFSET);                                     ! SET UP UNIT-RELATED DATA ITEMS
;    2382  4                             UNIT_INIT ();                                           ! BRING UNIT ONLINE
;    2383  3                             end;                                                    ! IF UNIT EXISTS
;    2384  3
```

```
;  2385  2                    end;
;  2386  2
;  2387  1          end;

                                                        ! IF CONTROLLER IS PRESENT
                                                        ! ROUTINE DRIVER INIT

                                        .SBTTL  DRIVER.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G       DRIVER.INIT::
                                        JSR     R1,$SAVE3                       ;                       2332
000004  012702  000012G               MOV     #MSCP.PKT+12,R2                 ; *,PKT.ADDR            2346
000010  105037  000000G               CLRB    NEXT.PKT.USE                    ;                       2347
000014  005001                        CLR     R1                              ; COUNT                 2349
000016  112761  000377  000000G  1$:  MOVB    #377,PKT.USE(R1)                ; *,*(COUNT)            2351
000024  010146                        MOV     R1,-(SP)                        ; COUNT,*               2352
000026  012746  000106               MOV     #106,-(SP)
000032  004737  000000G               JSR     PC,BL$MUL
000036  010260  000000G               MOV     R2,MSCP.PKT(R0)                 ; PKT.ADDR,*
000042  005060  000002G               CLR     MSCP.PKT+2(R0)                  ;                       2353
000046  105060  000011G               CLRB    MSCP.PKT+11(R0)                 ;                       2354
000052  062702  000106               ADD     #106,R2                         ; *,PKT.ADDR            2355
000056  022626                        CMP     (SP)+,(SP)+                     ;                       2350
000060  005201                        INC     R1                              ; COUNT                 2349
000062  020127  000013               CMP     R1,#13                          ; COUNT,*
000066  003753                        BLE     1$
000070  005003                        CLR     R3                              ; CTLR                  2358
000072  010346                  2$:   MOV     R3,-(SP)                        ; CTLR,*                2360
000074  012746  000126               MOV     #126,-(SP)
000100  004737  000000G               JSR     PC,BL$MUL
000104  022626                        CMP     (SP)+,(SP)+
000106  005760  000000G               TST     CST(R0)
000112  001503                        BEQ     6$
000114  010346                        MOV     R3,-(SP)                        ; CTLR,*                2363
000116  004737  000000G               JSR     PC,SET.CPAR
000122  013700  000000G               MOV     CST.ADDR,R0                     ;                       2364
000126  016037  000002  001246'       MOV     2(R0),CURRENT.VECTOR
000134  042737  177000  001246       BIC     #177000,CURRENT.VECTOR
000142  005016                        CLR     (SP)                            ;                       2365
000144  116016  000004               MOVB    4(R0),(SP)
000150  012746  000005               MOV     #5,-(SP)
000154  004737  000000G               JSR     PC,BL$SHF
000160  010037  000000G               MOV     R0,BRLEVEL
000164  004737  000000V               JSR     PC,CTLR.INIT
000170  005777  000000G               TST     @DCT.ADDR                       ;                       2366
000174  100051                        BPL     5$                              ;                       2368
000176  012701  000003               MOV     #3,R1                           ; *,OFFSET              2371
000202  013702  000000G         3$:   MOV     CST.ADDR,R2                     ;                       2373
000206  010100                        MOV     R1,R0                           ; OFFSET,*
000210  006300                        ASL     R0
000212  060200                        ADD     R2,R0
000214  032710  040000               BIT     #40000,(R0)
000220  001432                        BEQ     4$
000222  032710  010000               BIT     #10000,(R0)                     ;                       2374
000226  001027                        BNE     4$
000230  010100                        MOV     R1,R0                           ; OFFSET,*              2377
```

L6


```
000232  006300                          ASL     R0
000234  060200                          ADD     R2,R0
000236  112760  000040  000012          MOVB    #40,12(R0)
000244  112760  000040  000013          MOVB    #40,13(R0)                              ;                       2378
000252  010100                          MOV     R1,R0                                   ; OFFSET,*              2379
000254  006300                          ASL     R0
000256  060200                          ADD     R2,R0
000260  112760  000040  000014          MOVB    #40,14(R0)
000266  112760  000040  000015          MOVB    #40,15(R0)
000274  010116                          MOV     R1,(SP)                                 ;                       2380
000276  004737  0G0000G                 JSR     PC,SET.UPAR                             ; OFFSET,*              2381
000302  004737  000000V                 JSR     PC,UNIT.INIT
000306  062701  000012          4$:     ADD     #12,R1                                  ;                       2382
000312  020127  000041                  CMP     R1,#41                                  ; *,OFFSET              2371
000316  003731                          BLE     3$                                      ; OFFSET,*
000320  022626                  5$:     CMP     (SP)+,(SP)+
000322  005203                  6$:     INC     R3                                      ;                       2362
000324  000243                          .WORD CLV!CLC                                   ; CTLR                  2358

000326  003661                          BLE     2$
000330  000207                          RTS     PC                                      ;                       2332

; Routine Size:  109 words,     Routine Base:  $CODE$ + 1272
; Maximum stack depth per invocation:  7 words
```

```
;    2388  1     GLOBAL routine CTLR_INIT : novalue =
;    2389  1
;    2390  1     !+
;    2391  1     !       THIS "DRIVER" ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONTROLLER
;    2392  1     !       CONFIGURED FOR TESTING. ITS GENERAL PURPOSE IS TO BRING THE RDRX ONLINE
;    2393  1     !       TO THE HOST. SPECIFICALLY, IT IS WRITTEN TO:
;    2394  1     !
;    2395  1     !       1. INITIALIZE DRIVER CONTROLLER DATA, INCLUDING THE DCT,
;    2396  1     !       2. SET UP THE DEVICE'S INTERRUPT VECTOR ADDRESS,
;    2397  1     !       3. PERFORM A REGISTER EXISTENCE TEST TO VERIFY THE DEVICE'S PRESENCE,
;    2398  1     !       4. PERFORM A VECTOR AND BR LEVEL TEST TO VERIFY THE DEVICE'S VECTOR
;    2399  1     !               ADDRESS AND INTERRUPT REQUEST LEVEL,
;    2400  1     !       5. DO A HARD INITIALIZATION (FOUR STEPS) ON THE DEVICE.
;    2401  1     !
;    2402  1     !       IF ANY OF THESE INITIAL TESTS FAIL, THEN ALL UNITS ASSOCIATED WITH THE
;    2403  1     !       DEVICE ARE DROPPED.
;    2404  1     !-
;    2405  1
;    2406  2         begin
;    2407  2
;    2408  2         local
;    2409  2             RESULT : byte;
;    2410  2
;    2411  2         INI_CTLR_DAT ();                                        ! INITIALIZE CONTROLLER DATA
;    2412  2     !ZZZ    SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], PRIO4);    ! SET DEVICE'S ASSUMED VECTOR ADDRESS
;    2413  2         SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], .BRLEVEL);     ! SET DEVICE'S ASSUMED VECTOR ADDRESS    ZZZ
;    2414  2         DCT_ADDR [IG_INT] = TRUE;                               ! SET "IGNORE INTERRUPT" BIT
;    2415  2         L$LUN = .CST_ADDR [OF_UN + OF_DATA, D_UNIT];            ! GET FIRST UNIT NUMBER OF CONTROLLER
;    2416  2                                                                 ! (USED BY DRS FOR DEVICE FATAL CTLR ERRORS)
;    2417  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
;    2418  2     THEN                                    !CAUSED INIT, SKIP THIS CODE   ZZZ
;    2419  2
;    2420  2         if REG_EXIST () eql FAILURE                             ! REGISTER EXISTENCE TEST
;    2421  2         then
;    2422  3             begin
;    2423  3             DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
;    2424  3             return;
;    2425  2             end;
;    2426  2
;    2427  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
;    2428  2     THEN                                    !CAUSED INIT, SKIP THIS CODE   ZZZ
;    2429  2
;    2430  2         if VEC_BR_TEST () eql FAILURE                           ! VECTOR ADDR AND BR LEVEL TEST
;    2431  2         then
;    2432  3             begin
;    2433  3             DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
;    2434  3             return;
;    2435  2             end;
;    2436  2
;    2437  2         RESULT = HARD_INIT ();                                  ! ATTEMPT HARD DEVICE INIT
;    2438  2         DCT_ADDR [IG_INT] = FALSE;                              ! CLAER "IGNORE INTERRUPT" BIT
;    2439  2
;    2440  2         if .RESULT eql SUCCESS                                  ! IF HARD INIT WAS SUCCESSFUL
```

```
;    2441  2          then
;    2442  3              begin
;    2443  3              ADDR_VECT_OK = TRUE;                    ! ADDRESS/VECTOR TEST PASSED
;    2444  3              INI_RRING ();                           ! INITIALIZE RESPONSE RING
;    2445  3              WRT_RDRX (RCSA, RC_ALL, SA_GO);         ! SET "GO" BIT (START CTLR POLLING)
;    2446  3
;    2447  3              if SET_CTLR_CHAR () eql SUCCESS         ! SET CONTROLLER CHARACTERISTICS
;    2448  3              then
;    2449  4                  begin
;    2450  4                  DCT_ADDR [STAT] = ONLINE;           ! MARK CONTROLLER ONLINE IN "DRIVER"
;    2451  4                  CST_ADDR [STATE] = ONLINE;          ! MARK CONTROLLER ONLINE IN "PROGRAM"
;    2452  3                  end;
;    2453  3              end
;    2454  3
;    2455  2          else                                        ! HARD INIT FAILED
;    2456  3              begin
;    2457  3              DROP_CTLR (.CCTLR, DU_INIT);            ! DROP ALL CONTROLLER'S UNITS
;    2458  2              end;
;    2459  2
;    2460  1          end;                                        ! ROUTINE CTLR_INIT


                                        .SBTTL  CTLR.INIT INITIALIZATION TEST ROUTINES
000000  010146                  CTLR.INIT::
                                        MOV     R1,-(SP)                        ;                       2388
000002  004737  000000V                 JSR     PC,INI.CTLR.DAT                 ;                       2411
000006  013746  000000G                 MOV     BRLEVEL,-(SP)                   ;                       2413
000012  013700  000000G                 MOV     CCTLR,R0
000016  006300                          ASL     R0
000020  016046  000102'                 MOV     INT.ADDR(R0),-(SP)
000024  013746  001246'                 MOV     CURRENT.VECTOR,-(SP)
000030  012746  000003                  MOV     #3,-(SP)
000034  104437                          TRAP    37
000036  052777  040000  000000G        BIS     #40000,@DCT.ADDR                ;                       2414
000044  013700  000000G                 MOV     CST.ADDR,R0                     ;                       2415
000050  016001  000006                  MOV     6(R0),R1
000054  000301                          SWAB    R1
000056  042701  177760                  BIC     #177760,R1
000062  010137  000000G                 MOV     R1,L$LUN
000066  032737  000002  000000G        BIT     #2,DUP.FLAGS                    ;                       2417
000074  001025                          BNE     2$
000076  004737  000000V                 JSR     PC,REG.EXIST                    ;                       2420
000102  005700                          TST     R0
000104  001410                          BEQ     1$                              ;                       2423
000106  032737  000002  000000G        BIT     #2,DUP.FLAGS                    ;                       2427
000114  001015                          BNE     2$
000116  004737  000000V                 JSR     PC,VEC.BR.TEST                  ;                       2430
000122  005700                          TST     R0
000124  001011                          BNE     2$
000126  013716  000000G          1$:     MOV     CCTLR,(SP)                      ;                       2433
000132  012746  000002                  MOV     #2,-(SP)
000136  004737  000000G                 JSR     PC,DROP.CTLR
000142  062706  000012                  ADD     #12,SP                          ;                       2434
```

```
000146  000453                          BR      5$
000150  004737  000000V                2$:     JSR     PC,HARD.INIT                    ;                               2432
000154  110001                          MOVB    R0,R1                           ;                               2437
000156  042777  040000  000000G        BIC     #40000,@DCT.ADDR                ; *,RESULT
000164  120127  000001                  CMPB    R1,#1                           ;                               2438
000170  001031                          BNE     3$                              ; RESULT,*                      2440
000172  112737  000001  000000G        MOVB    #1,ADDR.VECT.OK                 ;                               2443
000200  004737  000000V                JSR     PC,INI.RRING                    ;                               2444
000204  012701  000001                  MOV     #1,R1                           ; *,RC.REG                      2445
000210  013700  000000G                MOV     RDRX.ADDR,R0
000214  010160  000002                  MOV     R1,2(R0)                        ; RC.REG,*
000220  004737  000000V                JSR     PC,SET.CTLR.CHAR                ;                               2447
000224  020027  000001                  CMP     R0,#1
000230  001020                          BNE     4$
000232  052777  100000  000000G        BIS     #100000,@DCT.ADDR               ;                               2450
000240  013700  000000G                MOV     CST.ADDR,R0                     ;                               2451
000244  052760  100000  000002        BIS     #100000,2(R0)
000252  000407                          BR      4$                              ;                               2440
000254  013716  000000G                3$:     MOV     CCTLR,(SP)              ;                               2457
000260  012746  000002                  MOV     #2,-(SP)
000264  004737  000000G                JSR     PC,DROP.CTLR
000270  005726                          TST     (SP)+                           ;                               2456
000272  062706  000010                  4$:     ADD     #10,SP                  ;                               2406
000276  012601                          5$:     MOV     (SP)+,R1                ;                               2388
000300  000207                          RTS     PC

; Routine Size:  97 words,    Routine Base:  $CODE$ + 1624
; Maximum stack depth per invocation:  7 words
```

```
;    2461  1     GLOBAL routine INI CTLR DAT : novalue =
;    2462  1
;    2463  1     !+
;    2464  1     !       THIS ROUTINE IS RESPONSIBLE FOR INITIALIZING ALL CONTROLLER RELATED
;    2465  1     !       DATA IN THE "DRIVER" PORTION OF THE EXERCISER. THIS INCLUDES THE
;    2466  1     !       CONTROLLER'S DCT AND OUTSTANDING COMMAND LIST.
;    2467  1     !
;    2468  1     !       IMPLICIT INPUTS:
;    2469  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2470  1     !               DCT_ADDR - ADDRESS OF CURENT CONTROLLER'S DCT
;    2471  1     !-
;    2472  1
;    2473  2         begin
;    2474  2         DCT_ADDR [WORD0] = 0;                                          ! CLEAR FIRST DCT WORD
;    2475  2         DCT_ADDR [RR_BEG] = COMM_AREA + 8 + (.CCTLR * COMM_LEN * 2);   ! START OF RESPONSE RING
;    2476  2         DCT_ADDR [RR_END] = .DCT_ADDR [RR_BEG] + ((RRING_LEN - 1) * 4); ! LAST SLOT IN RESPONSE RING
;    2477  2         DCT_ADDR [CR_BEG] = .DCT_ADDR [RR_END] + 4;                    ! START OF COMMAND RING
;    2478  2         DCT_ADDR [CR_END] = .DCT_ADDR [CR_BEG] + ((CRING_LEN - 1) * 4); ! LAST SLOT IN COMMAND RING
;    2479  2         DCT_ADDR [RR_POLL] = .DCT_ADDR [RR_BEG];                       ! FIRST RRING SLOT TO POLL
;    2480  2         DCT_ADDR [CR_POLL] = DCT_ADDR [CR_NEXT] = .DCT_ADDR [CR_BEG];  ! CRING POLL AND NEXT COMMAND POINTERS
;    2481  1         end;


                                        .SBTTL  INI.CTLR.DAT INITIALIZATION TEST ROUTINES
000000  004137  000000G          INI.CTLR.DAT::
                                        JSR     R1,$SAVE2                       ;                       2461
000004  013701  000000G                 MOV     DCT.ADDR,R1                     ;                       2474
000010  005011                          CLR     (R1)
000012  012702  000004                  MOV     #4,R2                           ;                       2475
000016  060102                          ADD     R1,R2
000020  013746  000000G                 MOV     CCTLR,-(SP)
000024  012746  000050                  MOV     #50,-(SP)
000030  004737  000000G                 JSR     PC,BL$MUL
000034  062700  000010'                 ADD     #COMM.AREA+10,R0
000040  010012                          MOV     R0,(R2)
000042  010061  000006                  MOV     R0,6(R1)
000046  062761  000014  000006         ADD     #14,6(R1)                       ;                       2476
000054  012700  000010                  MOV     #10,R0                          ;                       2477
000060  060100                          ADD     R1,R0
000062  016110  000006                  MOV     6(R1),(R0)
000066  062710  000004                  ADD     #4,(R0)
000072  011061  000012                  MOV     (R0),12(R1)                     ;                       2478
000076  062761  000014  000012         ADD     #14,12(R1)
000104  011261  000014                  MOV     (R2),14(R1)                     ;                       2479
000110  011061  000020                  MOV     (R0),20(R1)                     ;                       2480
000114  011061  000016                  MOV     (R0),16(R1)
000120  022626                          CMP     (SP)+,(SP)+                     ;                       2473
000122  000207                          RTS     PC                              ;                       2461

; Routine Size:  42 words,      Routine Base:  $CODE$ + 2126
; Maximum stack depth per invocation:  6 words
```

```
;  2482  1     GLOBAL routine REG_EXIST =
;  2483  1     !+
;  2484  1     !       THIS IS THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VEPIFY
;  2485  1     !       THE PRESENCE OF AN RDRX DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
;  2486  1     !       SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
;  2487  1     !       ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE'S SA AND IP
;  2488  1     !       REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
;  2489  1     !       THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
;  2490  1     !       DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
;  2491  1     !-
;  2492  2         begin
;  2493  2
;  2494  2         local
;  2495  2             DUMMY_0 : word,                         ! TEMP FOR READING SA AND IP
;  2496  2             DUMMY_1 : word;                         !
;  2497  2
;  2498  2         if .ENTRY_REASON eql NEW_PASS
;  2499  2         then
;  2500  2             return SUCCESS;                         ! SKIP TEST FOR NEXT PASS
;  2501  2
;  2502  2         OF_RC = 2;                                  ! SET UP TO READ SA FIRST
;  2503  2
;  2504  2         do
;  2505  3             begin
;  2506  3             NEX = FALSE;                            ! SET TO "TRAP NOT RECEIVED"
;  2507  3             SETVEC (4, NEX_TRAP, PRI07);            ! SET LOCATION 4 TRAP VECTOR ADDRESS
;  2508  3             DUMMY_0 = .(.RDRX_ADDR + .OF_RC);       ! READ REGISTER (THEN TRAP OR CONTINUE)
;  2509  3             DUMMY_1 = 0;                            ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
;  2510  3                                                     !       (TRAP RETURNS TO NEXT INSTRUCTION)
;  2511  3             CLRVEC (4);                             ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS
;  2512  3
;  2513  3             if .NEX                                 ! IF NEX TRAP OCCURRED
;  2514  3             then
;  2515  4                 begin
;  2516  4                 C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  2517  4
;  2518  4                 if .APT_MODE
;  2519  4                 then
;  2520  5                     begin
;  2521  5                     .MAIL_BOX_TESTNUM = 1;
;  2522  5                     .MAIL_BOX_SUBTST = 0;
;  2523  4                     end;
;  2524  4
;  2525  4                 ERRDF (10, EGD_10, EMS_10);         ! REGISTER EXISTENCE TEST FAILED
;  2526  4                 SETPRI (PRI00);                     ! LOWER PRIORITY
;  2527  4                 return FAILURE;
;  2528  4                 end
;  2529  3             else
;  2530  3                 OF_RC = .OF_RC - 2;                 ! SET UP FOR IP REG OR QUIT
;  2531  3
;  2532  3             end
;  2533  2         until .OF_RC lss 0;
;  2534  2
```

```
;   2535  2          return SUCCESS;
;   2536  1          end;


                                                .SBTTL  REG.EXIST INITIALIZATION TEST ROUTINES
000000  004137  000000G                  REG.EXIST::
                                                JSR     R1,$SAVE2                       ;                       2482
000004  123727  000000G 000005                  CMPB    ENTRY.REASON,#5                 ;                       2498
000012  001472                                  BEQ     4$                              ;                       2500
000014  012737  000002  000000G                MOV     #2,OF.RC                        ;                       2502
000022  005037  000000G                1$:     CLR     NEX                             ;                       2506
000026  012746  000340                          MOV     #340,-(SP)                      ;                       2507
000032  012746  000000G                          MOV     #NEX.TRAP,-(SP)
000036  012746  000004                          MOV     #4,-(SP)
000042  012746  000003                          MOV     #3,-(SP)
000046  104437                                  TRAP    37
000050  013700  000000G                          MOV     RDRX.ADDR,R0                    ;                       2508
000054  063700  000000G                          ADD     OF.RC,R0
000060  011001                                  MOV     (R0),R1                         ; *,DUMMY.0
000062  005002                                  CLR     R2                              ; DUMMY.1
000064  012700  000004                          MOV     #4,R0                           ;                       2509
000070  104436                                  TRAP    36                              ;                       2511
000072  032737  000001  000000G                BIT     #1,NEX                          ;                       2513
000100  001427                                  BEQ     3$
000102  013700  000000G                          MOV     CCTLR,R0                        ;                       2516
000106  006300                                  ASL     R0
000110  105260  000000G                          INCB    C.ERR.TBL(R0)
000114  032737  000001  001254'                BIT     #1,APT.MODE                     ;                       2518
000122  001405                                  BEQ     2$
000124  012777  000001  001256'                MOV     #1,@MAIL.BOX.TESTNUM            ;                       2521
000132  005077  001260'                        CLR     @MAIL.BOX.SUBTST                ;                       2522
000136  104455                        2$:     TRAP    55                              ;                       2525
000140  000012                                  .WORD   12
000142  000000G                                  .WORD   EGD.10
000144  000000G                                  .WORD   EMS.10
000146  005000                                  CLR     R0                              ;                       2526
000150  104441                                  TRAP    41
000152  062706  000010                          ADD     #10,SP                          ;                       2527
000156  000413                                  BR      5$                              ;                       2515
000160  162737  000002  000000G        3$:     SUB     #2,OF.RC                        ;                       2530
000166  062706  000010                          ADD     #10,SP                          ;                       2505
000172  005737  000000G                          TST     OF.RC                           ;                       2533
000176  002311                                  BGE     1$
000200  012700  000001                4$:     MOV     #1,R0                           ;                       2492
000204  000207                                  RTS     PC
000206  005000                        5$:     CLR     R0                              ;                       2482
000210  000207                                  RTS     PC

; Routine Size:  69 words,      Routine Base:  $CODE$ + 2252
; Maximum stack depth per invocation:  9 words
```

```
;    2537  1     GLOBAL routine VEC_BR_TEST =
;    2538  1
;    2539  1     !+
;    2540  1     !       THIS ROUTINE ATTEMPTS TO VERIFY (A) THAT THE RDRX VECTOR ADDRESS GIVEN
;    2541  1     !       BY THE USER DURING THE HW DIALOG IS VALID, AND (B) THAT THE
;    2542  1     !       USER-SPECIFIED BUS REQUEST LEVEL FOR THE DEVICE IS CORRECT. THE FIRST
;    2543  1     !       OBJECTIVE IS ACCOMPLISHED BY SETTING THE CPU PRIORITY TO 0 AND FORCING
;    2544  1     !       AN RDRX INTERRUPT. IF THE USER SPECIFIED AN INCORRECT VECTOR ADDRESS,
;    2545  1     !       THEN THE RESULT MAY BE UNPREDICTABLE. FOR THIS REASON, THE MESSAGE
;    2546  1     !       "FUNCTIONAL TEST STARTED" IS PRINTED BEFORE THE TEST, AND
;    2547  1     !       "EXERCISER STARTED" IS PRINTED AT ITS SUCCESSFUL CONCLUSION. IF
;    2548  1     !       EITHER "FUNCTIONAL TEST ..." OR "EXERCISER ..." DOES NOT APPEAR, THEN
;    2549  1     !       PROGRAM CONTROL IS ASSUMED LOST AND A FATAL TRAP IS LIKELY TO OCCUR. AT
;    2550  1     !       THIS POINT, THE EXERCISER MUST BE STARTED AGAIN.
;    2551  1     !
;    2552  1     !       IF THIS TEST SUCCEEDS, THEN THE BR LEVEL TEST IS RUN BY SETTING THE
;    2553  1     !       PROCESSOR PRIORITY TO THE ASSUMED INTERRUPT PRIORITY GIVEN BY THE
;    2554  1     !       USER. A FORCED INTERRUPT SHOULD NOT OCCURR. THEN, BY LOWERING THE
;    2555  1     !       PRIORITY BY ONE, THE DELAYED INTERRUPT SHOULD OCCUR.
;    2556  1     !-
;    2557  1
;    2558  2         begin
;    2559  2
;    2560  2         if .ENTRY_REASON eql NEW_PASS
;    2561  2         then
;    2562  3             begin
;    2563  3             SETPRI (PRIOO);                         ! LOWER PRIORITY
;    2564  3             return SUCCESS;                         ! SKIP TEST IF NEXT PASS
;    2565  2             end;
;    2566  2
;    2567  2         PRINTF (MSG_02);                            ! "FUNCTIONAL TEST STARTED"
;    2568  2
;    2569  2         if INT_GEN () eql FALSE                     ! FORCE AN INTERRUPT
;    2570  2         then
;    2571  3             begin                                   ! IF INTERRUPT DID NOT OCCUR
;    2572  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2573  3
;    2574  3             if .APT_MODE
;    2575  3             then
;    2576  4                 begin
;    2577  4                 .MAIL_BOX_TESTNUM = 1;
;    2578  4                 .MAIL_BOX_SUBTST = 0;
;    2579  3                 end;
;    2580  3
;    2581  3             ERRDF (11, EGD_11, 0);                  ! VECTOR TEST FAILED
;    2582  3             return FAILURE;
;    2583  3             end
;    2584  2         else
;    2585  3             begin                                   ! INTERRUPT DID OCCUR
;    2586  3             PRINTF (MSG 03);                        ! "EXERCISER STARTED"
;    2587  3             SETPRI (.BRLEVEL);                      ! SET PRIORITY TO ASSUMED BR LEVEL
;    2588  3
;    2589  3             if INT_GEN () eql FALSE                 ! FORCE AN INTERRUPT (SHOULD NOT OCCUR)
```

```
;    2590  3              then
;    2591  4                  begin                           ! IF INTERRUPT DID NOT OCCUR
;    2592  4                  SETPRI (.BRLEVEL    %o'40');    ! LOWER PRIORITY BY 1
;    2593  4                  DELAY (1);                      ! WAIT
;    2594  4
;    2595  4                  if .DCT_ADDR [SA SAVE] neq 0    ! IF INTERRUPT DID OCCUR (SA_SAVE WOULD BE NON ZERO)
;    2596  4                  then
;    2597  5                      begin
;    2598  5                      SETPRI (PRI00);             ! RESTORE PROCESSOR PRIORITY TO 0
;    2599  5                      return SUCCESS;             ! ONLY SUCCESSFUL EXIT POINT
;    2600  4                      end;
;    2601  4
;    2602  3                  end;
;    2603  3
;    2604  2              end;
;    2605  2
;    2606  2          SETPRI (PRI00);                         ! COME HERE ONLY FOR BR TEST FAILURE
;    2607  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2608  2
;    2609  2          if .APT_MODE
;    2610  2          then
;    2611  3              begin
;    2612  3              .MAIL_BOX_TESTNUM = 1;
;    2613  3              .MAIL_BOX_SUBTST = 0;
;    2614  2              end;
;    2615  2
;    2616  2          ERRDF (12, EGD_12, EMS_12);
;    2617  2          return FAILURE;
;    2618  1          end;


                                .GLOBL  L$DLY


                                .SBTTL  VEC.BR.TEST INITIALIZATION TEST ROUTINES
000000  010146            VEC.BR.TEST::
                                MOV     R1,-(SP)                        ;                       2537
000002  005746                  TST     -(SP)
00000   123727  000000G 000005      CMPB    ENTRY.REASON,#5             ;                       2560
000012  001003                  BNE     1$
000014  005000                  CLR     R0                              ;                       2563
000016  104441                  TRAP    41
000020  000504                  BR      8$                              ;                       2562
000022  012746  000000G     1$: MOV     #MSG.02,-(SP)                   ;                       2567
000026  012746  000001          MOV     #1,-(SP)
000032  010600                  MOV     SP,R0                           ; SP,*
000034  104417                  TRAP    17
000036  004737  000000V         JSR     PC,INT.GEN                      ;                       2569
000042  005700                  TST     R0
000044  001023                  BNE     3$
000046  013700  000000G         MOV     CCTLR,R0                        ;                       2572
000052  006300                  ASL     R0
000054  105260  000000G         INCB    C.ERR.TBL(R0)
```

```
000060  032737  000001  001254'                BIT     #1,APT.MODE                     ;                               2574
000066  001405                                  BEQ     2$
000070  012777  000001  001256'                MOV     #1,@MAIL.BOX.TESTNUM            ;                               2577
000076  005077  001260'                        CLR     @MAIL.BOX.SUBTST                ;                               2578
000102  104455                          2$:     TRAP    55                              ;                               2581
000104  000013                                  .WORD   13
000106  000000G                                 .WORD   EGD.11
000110  000000                                  .WORD   0
000112  000477                                  BR      11$
000114  012716  000000G                 3$:     MOV     #MSG.03,(SP)                    ;                               2582
000120  012746  000001                          MOV     #1,-(SP)                        ;                               2586
000124  010600                                  MOV     SP,R0                           ; SP,*
000126  104417                                  TRAP    17
000130  013700  000000G                         MOV     BRLEVEL,R0                      ;                               2587
000134  104441                                  TRAP    41
000136  004737  000000V                         JSR     PC,INT.GEN                      ;                               2589
000142  005700                                  TST     R0
000144  001035                                  BNE     9$
000146  013700  000000G                         MOV     BRLEVEL,R0                      ;                               2592
000152  162700  000040                          SUB     #40,R0
000156  104441                                  TRAP    41
000160  012701  000001                          MOV     #1,R1                           ; *,$$TMP2                      2593
000164  001411                          4$:     BEQ     7$
000166  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000172  001404                                  BEQ     6$
000174  005066  000006                  5$:     CLR     6(SP)                           ; $$TMP
000200  005300                                  DEC     R0                              ; $$TMP1
000202  001374                                  BNE     5$
000204  005301                          6$:     DEC     R1                              ; $$TMP2
000206  000766                                  BR      4$
000210  013700  000000G                 7$:     MOV     DCT.ADDR,R0                     ;                               2595
000214  005760  000002                          TST     2(R0)
000220  001407                                  BEQ     9$
000222  005000                                  CLR     R0                              ;                               2598
000224  104441                                  TRAP    41
000226  062706  000006                          ADD     #6,SP                           ;                               2599
000232  012700  000001                  8$:     MOV     #1,R0                           ;                               2597
000236  000427                                  BR      12$
000240  005726                          9$:     TST     (SP)+                           ;                               2585
000242  005000                                  CLR     R0                              ;                               2606
000244  104441                                  TRAP    41
000246  013700  000000G                         MOV     CCTLR,R0                        ;                               2607
000252  006300                                  ASL     R0
000254  105260  000000G                         INCB    C.ERR.TBL(R0)
000260  032737  000001  001254'                BIT     #1,APT.MODE                     ;                               2609
000266  001405                                  BEQ     10$
000270  012777  000001  001256'                MOV     #1,@MAIL.BOX.TESTNUM            ;                               2612
000276  005077  001260'                        CLR     @MAIL.BOX.SUBTST                ;                               2613
000302  104455                          10$:    TRAP    55                              ;                               2616
000304  000014                                  .WORD   14
000306  000000G                                 .WORD   EGD.12
000310  000000G                                 .WORD   EMS.12
000312  022626                          11$:    CMP     (SP)+,(SP)+                     ;                               2617
```

```
000314  005000                          CLR     R0                                                      !                                       2537
000316  005726                  12$:    TST     (SP)+
000320  012601                          MOV     (SP)+,R1
000322  000207                          RTS     PC

; Routine Size:  106 words,     Routine Base:  $CODE$ + 2464
; Maximum stack depth per invocation:  7 words
```

```
;    2619  1     GLOBAL routine INT GEN =
;    2620  1
;    2621  1     !+
;    2622  1     !       THIS ROUTINE BEGINS AN RDRX INITIALIZATION SEQUENCE, BUT ONLY
;    2623  1     !       COMPLETES THROUGH THE STEP 1 WRITE. ITS PURPOSE IS TO CREATE AN RDRX
;    2624  1     !       INTERRUPT (AT THE COMPLETEION OF STEP 1) IN ORDER TO HELP VERIFY THE
;    2625  1     !       THE USER-SPECIFIED VECTOR ADDRESS AND BUS REQUEST INTERRUPT LEVEL.
;    2626  1     !       A VALUE OF "TRUE" IS RETURNED TO THE CALLER IF AN INTERRUPT OCCURS,
;    2627  1     !       AND "FALSE" OTHERWISE. THE INTERRUPT IS VERIFIED BY A NON-ZERO VALUE
;    2628  1     !       IN THE "SA SAVE" WORD IN THE DEVICE'S DCT.
;    2629  1     !-
;    2630  1
;    2631  2         begin
;    2632  2
;    2633  2         local
;    2634  2             SA : word;                                  ! STORAGE FOR STEP 1 READ AND WRITE
;    2635  2
;    2636  2         DCT_ADDR [SA_SAVE] = 0;                         ! ZERO OUT SA SAVE WORD IN DCT
;    2637  2         WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
;    2638  2         DELAY (2);                                      ! WAIT
;    2639  2         INCR COUNT FROM 1 TO 500 DO                     !MAKE SURE WE GET INTO STEP 1  ZZZ
;    2640  3             BEGIN                                       !BEFORE STEP 1 WRITE           ZZZ
;    2641  3             SA = .RDRX_ADDR [RCSA, RC_ALL];             ! STEP 1 READ
;    2642  3               IF (.SA AND S1_MASK) EQL SA_S1            !DID WE GET THE S1 BIT?        ZZZ
;    2643  3               THEN                                      !                              ZZZ
;    2644  3               EXITLOOP;                                 !EXIT IF SO                    ZZZ
;    2645  3             DELAY (1);                                  !                              ZZZ
;    2646  2             END;                                        !                              ZZZ
;    2647  2
;    2648  2        SA = (WR_RING ^ 8) or (.CURRENT_VECTOR ^ -2) or SA_INT;  ! STEP 1 WRITE VALUE
;    2649  2         WRT_RDRX (RCSA, RC_ALL, .SA);                   ! STEP 1 WRITE
;    2650  2
;    2651  2         incr COUNT from 1 to 8000 do
;    2652  3             begin
;    2653  3             DELAY (1);                                  ! TOTAL DELAY COUNT OF 8,000
;    2654  3
;    2655  3             if .DCT_ADDR [SA_SAVE] neq 0                ! IF SA WAS CHANGED
;    2656  3             then
;    2657  3                 return TRUE;                            ! INTERRUPT OCCURED
;    2658  3
;    2659  3             BREAK;
;    2660  2             end;
;    2661  2
;    2662  2         return FALSE;                                   ! IF INTERRUPT DID NOT OCCUR
;    2663  1         end;


                                        .SBTTL  INT.GEN INITIALIZATION TEST ROUTINES
000000  004137  000000G          INT.GEN::
                                        JSR     R1,$SAVE4                       ;                       2619
000004  024646                          CMP     -(SP),-(SP)
000006  013700  000000G                 MOV     DCT.ADDR,R0                     ;                       2636
000012  005060  000002                  CLR     2(R0)
```

```
000016  012700  177777                    MOV     #-1,R0                  ; *,RC.REG                      2637
000022  010077  000000G                   MOV     R0 @RDRX.ADDR           ; RC.REG,*
000026  012701  000002                    MOV     #2,R1                   ; *,$$TMP2                      2638
000032  001411                    1$:     BEQ     4$
000034  013700  000000G                   MOV     L$DLY,R0                ; *,$$TMP1
000040  001404                            BEQ     3$
000042  005066  000002            2$:     CLR     2(SP)                   ; $$TMP
000046  005300                            DEC     R0                      ; $$TMP1
000050  001374                            BNE     2$
000052  005301                    3$:     DEC     R1                      ; $$TMP2
000054  000766                            BR      1$
000056  013702  000000G           4$:     MOV     RDRX.ADDR,R2            ;                               2641
000062  012703  000764                    MOV     #764,R3                 ; *,COUNT                       2639
000066  016216  000002           5$:     MOV     2(R2),(SP)              ; *,RC.REG                      2641
000072  011604                            MOV     (SP),R4                 ; RC.REG,SA
000074  010400                            MOV     R4,R0                   ; SA,*                          2642
000076  042700  001777                    BIC     #1777,R0
000102  020027  004000                    CMP     R0,#4000
000106  001416                            BEQ     10$                     ;                               2644
000110  012701  000001                    MOV     #1,R1                   ; *,$$TMP2                      2645
000114  001411                    6$:     BEQ     9$
000116  013700  000000G                   MOV     L$DLY,R0                ; *,$$TMP1
000122  001404                            BEQ     8$
000124  005066  000002            7$:     CLR     2(SP)                   ; $$TMP
000130  005300                            DEC     R0                      ; $$TMP1
000132  001374                            BNE     7$
000134  005301                    8$:     DEC     R1                      ; $$TMP2
000136  000766                            BR      6$
000140  005303                    9$:     DEC     R3                      ; COUNT                         2639
000142  001351                            BNE     5$
000144  013700  001246'           10$:    MOV     CURRENT.VECTOR,R0       ;                               2648
000150  006200                            ASR     R0
000152  006200                            ASR     R0
000154  010004                            MOV     R0,R4                   ; *,SA
000156  052704  111200                    BIS     #111200,R4              ; *,SA
000162  010401                            MOV     R4,R1                   ; SA,RC.REG                     2649
000164  010162  000002                    MOV     R1,2(R2)                ; RC.REG,*
000170  012702  017500                    MOV     #17500,R2               ; *,COUNT                       2651
000174  012701  000001            11$:    MOV     #1,R1                   ; *,$$TMP2                      2653
000200  001411                    12$:    BEQ     15$
000202  013700  000000G                   MOV     L$DLY,R0                ; *,$$TMP1
000206  001404                            BEQ     14$
000210  005066  000002            13$:    CLR     2(SP)                   ; $$TMP
000214  005300                            DEC     R0                      ; $$TMP1
000216  001374                            BNE     13$
000220  005301                    14$:    DEC     R1                      ; $$TMP2
000222  000766                            BR      12$
000224  013700  000000G           15$:    MOV     DCT.ADDR,R0             ;                               2655
000230  005760  000002                    TST     2(R0)
000234  001403                            BEQ     16$
000236  012700  000001                    MOV     #1,R0                   ;                               2657
000242  000404                            BR      17$
000244  104422                    16$:    TRAP    22
```

L 1


```
000246  005302                          DEC     R2                      ; COUNT                 2651
000250  001351                          BNE     11$
000252  005000                          CLR     R0                      ;                       2631
000254  022626                  17$:    CMP     (SP)+,(SP)+             ;                       2619
000256  000207                          RTS     PC

; Routine Size:  88 words,      Routine Base:  $CODE$ + 3010
; Maximum stack depth per invocation   9 words
```

```
;  2664  1     GLOBAL routine HARD_INIT =
;  2665  1
;  2666  1     !+
;  2667  1     !       THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
;  2668  1     !       INITIALIZE AN RDRX DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
;  2669  1     !       THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
;  2670  1     !       OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
;  2671  1     !       IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
;  2672  1     !
;  2673  1
;  2674  2         begin
;  2675  2
;  2676  2         local
;  2677  2             IE_VEC : word;                                  ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
;  2678  2                                                             !   (USED IN STEP 1 WRITE AND STEP 3 READ)
;  2679  2
;  2680  2         IE_VEC = .CURRENT_VECTOR + 2;                       ! GET VECTOR ADDR/4 (IE = 0)
;  2681  2
;  2682  2         incr ATTEMPTS from 1 to INI_ATT do
;  2683  3             begin
;  2684  3
;  2685  3             label
;  2686  3                 STEP_1_READ,
;  2687  3                 STEP_2_READ,
;  2688  3                 STEP_3_READ,
;  2689  3                 STEP_4_READ;
;  2690  3
;  2691  3             WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
;  2692  3     !
;  2693  3     !       STEP 1 READ
;  2694  3     !
;  2695  3             STEP = 1;
;  2696  3             STEP_1_READ:
;  2697  4                 begin
;  2698  4
;  2699  4                 incr COUNT from 1 to 500 do
;  2700  5                     begin
;  2701  5                     DELAY (1);                              ! TOTAL DELAY COUNT OF 500 FOR STEP 1
;  2702  5                     SA_REG = .RDRX_ADDR [RCSA, RC_ALL];     ! READ SA
;  2703  5
;  2704  5                     if (.SA_REG and S1_MASK) eql SA_S1      ! IF STEP 1 READ IS O.K.
;  2705  5                     then
;  2706  5                         leave STEP_1_READ;
;  2707  5
;  2708  5                     BREAK;
;  2709  4                     end;
;  2710  4
;  2711  4                 exitloop;
;  2712  3                 end;
;  2713  3
;  2714  3     !
;  2715  3     !       STEP 1 WRITE
;  2716  3     !
```

```
;    2717  3              SA_REG = (WR_RING ! 8) or .IE_VEC;                  ! STEP 1 WRITE VALUE
;    2718  3              WRT_RDRX (RCSA, RC_ALL, .SA_REG);                   ! STEP 1 WRITE
;    2719  3      !
;    2720  3      !       STEP 2 READ
;    2721  3      !
;    2722  3              STEP = .STEP + 1;
;    2723  3              STEP_2_READ:
;    2724  4                  begin
;    2725  4
;    2726  4                  incr COUNT from 1 to 10000 do
;    2727  5                      begin
;    2728  5                      DELAY (1);                                  ! TOTAL DELAY COUNT OF 10,000 FOR STEP 2
;    2729  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];         ! READ SA
;    2730  5
;    2731  6                      if (.SA_REG and S2_MASK) eql (SA_S2 or WR_RING) ! IF STEP 2 READ IS O.K.
;    2732  5                      then
;    2733  5                          leave STEP_2_READ;
;    2734  5
;    2735  5                      BREAK;
;    2736  4                      end;
;    2737  4
;    2738  4                  exitloop;
;    2739  3                  end;
;    2740  3
;    2741  3      !
;    2742  3      !       STEP 2 WRITE
;    2743  3      !
;    2744  3              WRT_RDRX (RCSA, RC_ALL, .DCT_ADDR [RR_BEG]);        ! RINGBASE-LO, PI = 0
;    2745  3      !
;    2746  3      !       STEP 3 READ
;    2747  3      !
;    2748  3              STEP = .STEP + 1;
;    2749  3              STEP_3_READ:
;    2750  4                  begin
;    2751  4
;    2752  4                  incr COUNT from 1 to 10000 do
;    2753  5                      begin
;    2754  5                      DELAY (1);                                  ! TOTAL DELAY COUNT OF 10,000 FOR STEP 3 READ
;    2755  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];         ! READ SA
;    2756  5
;    2757  6                      if (.SA_REG and S3_MASK) eql (SA_S3 or .IE_VEC) ! IF STEP 3 READ IS O.K.
;    2758  5                      then
;    2759  5                          leave STEP_3_READ;
;    2760  5
;    2761  5                      BREAK;
;    2762  4                      end;
;    2763  4
;    2764  4                  exitloop;
;    2765  3                  end;
;    2766  3
;    2767  3      !
;    2768  3      !       STEP 3 WRITE
;    2769  3      !
```

```
;    2770  3              WRT_RDRX (RCSA, RC_ALL, 0);                          ! PP, RINGBASE-HI = 0
;    2771  3      !
;    2772  3      !       STEP 4 READ
;    2773  3      !
;    2774  3              STEP = .STEP + 1;
;    2775  3              STEP_4_READ:
;    2776  4                  begin
;    2777  4
;    2778  4                  incr COUNT from 1 to 10000 do
;    2779  5                      begin
;    2780  5                      DELAY (1);                                   ! TOTAL DELAY COUNT OF 10,000 FOR STEP 4 READ
;    2781  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];          ! READ SA
;    2782  5
;    2783  5                      if (.SA_REG and S4_MASK) eql SA_S4           ! IF STEP 4 READ IS O.K.
;    2784  5                      then
;    2785  5                          leave STEP_4_READ;
;    2786  5
;    2787  5                      BREAK;
;    2788  4                      end;
;    2789  4
;    2790  4                  exitloop;
;    2791  3                  end;
;    2792  3      !
;    2793  3      !       STEP 4 WRITE
;    2794  3      !
;    2795  3              CREDIT_BAL = 1;                                      ! START WITH A CREDIT BALANCE = 1
;    2796  3              WRT_RDRX (RCSA, RC_ALL, 0);                          ! BURST, LF, GO = 0
;    2797  3              return SUCCESS;                                      ! SUCCESS EXIT POINT
;    2798  3
;    2799  2              end;                                                 ! TRY AGAIN OR GIVE UP
;    2800  2
;    2801  2          CREDIT_BAL = 0;                                          ! NO CREDIT BALANCE
;    2802  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2803  2
;    2804  2          if .APT_MODE
;    2805  2          then
;    2806  3              begin
;    2807  3              .MAIL_BOX_TESTNUM = 1;
;    2808  3              .MAIL_BOX_SUBTST = 0;
;    2809  2              end;
;    2810  2
;    2811  2          ERRDF (13, EGD_13, EMS_13);                              ! INIT SEQUENCE FAILED
;    2812  2          return FAILURE;
;    2813  1          end;                                                     ! ROUTINE HARD_INIT


                                        .SBTTL  HARD.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G          HARD.INIT::
                                        JSR     R1,$SAVE5                    ;                                  2664
000004  162706  000012                  SUB     #12,SP
000010  013704  001246'                 MOV     CURRENT.VECTOR,R4            ; *,IE.VEC                         2680
000014  006204                          ASR     R4                           ; IE VEC
000016  006204                          ASR     R4                           ; IE.VEC
```

```
000020  012705  000002                        MOV     #2,R5                   ; *,ATTEMPTS                    2682
000024  012700  177777                        MOV     #-1,R0                  ; *,RC.REG                      2691
000030  010077  000000G                       MOV     R0,@RDRX.ADDR           ; RC.REG,*
000034  012737  000001  000000G               MOV     #1,STEP                 ;                               2695
000042  012702  000764                        MOV     #764,R2                 ; *,COUNT                       2699
000046  012701  000001                1$:     MOV     #1,R1                   ; *,$$TMP2                      2701
000052  001411                        2$:     BEQ     5$
000054  013700  000000G                       MOV     L$DLY,R0                ; *,$$TMP1
000060  001404                                BEQ     4$
000062  005066  000010                3$:     CLR     10(SP)                  ; $$TMP
000066  005300                                DEC     R0                      ; $$TMP1
000070  001374                                BNE     3$
000072  005301                        4$:     DEC     R1                      ; $$TMP2
000074  000766                                BR      2$
000076  013700  000000G                5$:     MOV     RDRX.ADDR,R0            ;                               2702
000102  016016  000002                        MOV     2(R0),(SP)              ; *,RC.REG
000106  011637  000000G                       MOV     (SP),SA.REG             ; RC.REG,*
000112  011600                                MOV     (SP),R0                 ; SA.REG,*                      2704
000114  042700  001777                        BIC     #1777,R0
000120  020027  004000                        CMP     R0,#4000
000124  001404                                BEQ     6$                      ;                               2706
000126  104422                                TRAP    22
000130  005302                                DEC     R2                      ; COUNT                         2699
000132  001345                                BNE     1$
000134  000532                                BR      18$                     ;                               2683
000136  010437  000000G                6$:     MOV     R4,SA.REG               ; IE.VEC,*                      2717
000142  052737  111000  000000G               BIS     #111000,SA.REG
000150  013701  000000G                       MOV     SA.REG,R1               ; *,RC.REG                      2718
000154  013700  000000G                       MOV     RDRX.ADDR,R0
000160  010160  000002                        MOV     R1,2(R0)                ; RC.REG,*
000164  005237  000000G                       INC     STEP                    ;                               2722
000170  012702  023420                        MOV     #23420,R2               ; *,COUNT                       2726
000174  012701  000001                7$:     MOV     #1,R1                   ; *,$$TMP2                      2728
000200  001411                        8$:     BEQ     11$
000202  013700  000000G                       MOV     L$DLY,R0                ; *,$$TMP1
000206  001404                                BEQ     10$
000210  005066  000010                9$:     CLR     10(SP)                  ; $$TMP
000214  005300                                DEC     R0                      ; $$TMP1
000216  001374                                BNE     9$
000220  005301                        10$:    DEC     R1                      ; $$TMP2
000222  000766                                BR      8$
000224  013700  000000G                11$:    MOV     RDRX.ADDR,R0            ;                               2729
000230  016066  000002  000002                MOV     2(R0),2(SP)             ; *,RC.REG
000236  016637  000002  000000G               MOV     2(SP),SA.REG            ; RC.REG,*
000244  016600  000002                        MOV     2(SP),R0                ; SA.REG,*                      2731
000250  042700  003400                        BIC     #3400,R0
000254  020027  010222                        CMP     R0,#10222
000260  001404                                BEQ     12$                     ;                               2733
000262  104422                                TRAP    22
000264  005302                                DEC     R2                      ; COUNT                         2726
000266  001342                                BNE     7$
000270  000537                                BR      26$                     ;                               2683
000272  013700  000000G                12$:    MOV     DCT.ADDR,R0             ;                               2744
```

```
000276  016001  000004                        MOV     4(R0),R1                ; *,RC.REG
000302  013700  000000G                       MOV     RDRX.ADDR,R0
000306  010160  000002                        MOV     R1,2(R0)                ; RC.REG,*
000312  005237  000000G                       INC     STEP                    ;                               2748
000316  010403                                MOV     R4,R3                   ; IE.VEC,*                      2757
000320  052703  020000                        BIS     #20000,R3
000324  012702  023420                        MOV     #23420,R2               ; *,COUNT                       2752
000330  012701  000001              13$:      MOV     #1,R1                   ; *,$$TMP2                      2754
000334  001411                      14$:      BEQ     17$
000336  013700  000000G                       MOV     L$DLY,R0                ; *,$$TMP1
000342  001404                                BEQ     16$
000344  005066  000010              15$:      CLR     10(SP)                  ; $$TMP
000350  005300                                DEC     R0                      ; $$TMP1
000352  001374                                BNE     15$
000354  005301                      16$:      DEC     R1                      ; $$TMP2
000356  000766                                BR      14$
000360  013700  000000G             17$:      MOV     RDRX.ADDR,R0            ;                               2755
000364  016066  000002  000004               MOV     2(R0),4(SP)             ; *,RC.REG
000372  016637  000004  000000G               MOV     4(SP),SA.REG            ; RC.REG,*
000400  016600  000004                        MOV     4(SP),R0                ; SA.REG,*                      2757
000404  042700  003400                        BIC     #3400,R0
000410  020003                                CMP     R0,R3
000412  001404                                BEQ     19$                     ;                               2759
000414  104422                                TRAP    22
000416  005302                                DEC     R2                      ; COUNT                         2752
000420  001343                                BNE     13$
000422  000462                      18$:      BR      26$                     ;                               2683
000424  013700  000000G             19$:      MOV     RDRX.ADDR,R0            ;                               2770
000430  005060  000002                        CLR     2(R0)
000434  005237  000000G                       INC     STEP                    ;                               2774
000440  012703  023420                        MOV     #23420,R3               ; *,COUNT                       2778
000444  012701  000001              20$:      MOV     #1,R1                   ; *,$$TMP2                      2780
000450  001411                      21$:      BEQ     24$
000452  013700  000000G                       MOV     L$DLY,R0                ; *,$$TMP1
000456  001404                                BEQ     23$
000460  005066  000010              22$:      CLR     10(SP)                  ; $$TMP
000464  005300                                DEC     R0                      ; $$TMP1
000466  001374                                BNE     22$
000470  005301                      23$:      DEC     R1                      ; $$TMP2
000472  000766                                BR      21$
000474  013700  000000G             24$:      MOV     RDRX.ADDR,R0            ;                               2781
000500  016066  000002  000006               MOV     2(R0),6(SP)             ; *,RC.REG
000506  016637  000006  000000G               MOV     6(SP),SA.REG            ; RC.REG,*
000514  016600  000006                        MOV     6(SP),R0                ; SA.REG,*                      2783
000520  042700  003777                        BIC     #3777,R0
000524  020027  040000                        CMP     R0,#40000
000530  001404                                BEQ     25$                     ;                               2785
000532  104422                                TRAP    22
000534  005303                                DEC     R3                      ; COUNT                         2778
000536  001342                                BNE     20$
000540  000413                                BR      26$                     ;                               2683
000542  012737  000001  000000G     25$:      MOV     #1,CREDIT.BAL           ;                               2795
000550  005001                                CLR     R1                      ; RC.REG                        2796
```

```
ZRQAM3          RD/RX EXERCISER                              4-Apr 1985 13:23:31    VAX-11 Bliss-16 V4.1-582          SEQ 0302
V02.2           INITIALIZATION TEST ROUTINES                 2-Apr-1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;19  Page  47
                                                                                                                       (10)

000552  013700  000000G                          MOV     RDRX.ADDR,R0
000556  005060  000002                          CLR     2(R0)
000562  012700  000001                          MOV     #1,R0                                                   ;
000566  000425                                  BR      28$                                                     ;                2683
000570  005037  000000G                 26$:     CLR     CREDIT.BAL                                              ;
000574  013700  0C0000G                          MOV     CCTLR,R0                                                ;                2801
000600  006300                                  ASL     R0                                                      ;                2802
000602  105260  000000G                          INCB    C.ERR.TBL(R0)
000606  032737  000001  001254'                 BIT     #1,APT.MODE                                             ;
000614  001405                                  BEQ     27$                                                     ;                2804
000616  012777  000001  001256                  MOV     #1,@MAIL.BOX.TESTNUM                                    ;
000624  005077  001260'                         CLR     @MAIL.BOX.SUBTST                                        ;                2807
000630  104455                          27$:     TRAP    55                                                      ;                2808
000632  000015                                  .WORD   15                                                      ;                2811
000634  000000G                                 .WORD   EGD.13
000636  000000G                                 .WORD   EMS.13
000640  005000                                  CLR     R0                                                      ;
000642  062706  000012                  28$:     ADD     #12,SP                                                  ;                2674
000646  000207                                  RTS     PC                                                      ;                2664

; Routine Size:  212 words,     Routine Base:  $CODE$ + 3270
; Maximum stack depth per invocation:  13 words
```

```
;    2814  1      GLOBAL routine INI_RRING : novalue =
;    2815  1
;    2816  1      !+
;    2817  1      !       THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP PACKETS TO
;    2818  1      !       FILL AN RDRX RESPONSE RING. THE BUFFER DESCRIPTOR OF EACH PACKET
;    2819  1      !       (LOCATED IN FRONT OF THE PACKET ITSELF) IS LOADED INTO SUCCESSIVE
;    2820  1      !       RRING SLOTS. NOTE THAT THE BUFFER DESCRIPTORS HAVE BEEN INITIALIZED
;    2821  1      !       WITH THE FLAG AND OWNERSHIP BITS SET TO "1", MAKING EACH SLOT
;    2822  1      !       CONTROLLER-OWNED.
;    2823  1      !
;    2824  1      !       IMPLICIT INPUTS:
;    2825  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    2826  1      !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;    2827  1      !-
;    2828  1
;    2829  2          begin
;    2830  2
;    2831  2          local
;    2832  2              index : word,
;    2833  2              RRING_ADDR;
;    2834  2
;    2835  2          RRING_ADDR = .DCT_ADDR [RR_BEG];                    ! FIRST RESPONSE RING SLOT
;    2836  2
;    2837  2          incr COUNT from 1 to RRING_LEN do
;    2838  3              begin
;    2839  3              index = GET_PKT (.CCTLR);                       ! GET AN MSCP PACKET
;    2840  3              .RRING_ADDR = .MSCP_PKT [.index, PKT_LO];       ! LOAD LO-ORDER BUFF DESC INTO SLOT
;    2841  3              RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO SECOND WORD
;    2842  3              .RRING_ADDR = .MSCP_PKT [.index, PKT_HI];       ! LOAD HI-ORDER BUFF DESC INTO SLOT
;    2843  3              PKT_USE [.index] = .CCTLR;                      ! PACKET IN USE
;    2844  3              .RRING_ADDR = ..RRING_ADDR or ED_OWN or ED_FLAG;! GIVE OWNERSHIP TO CONTRLLER
;    2845  3              RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO NEXT SLOT
;    2846  2              end;
;    2847  2
;    2848  1          end;


                                        .SBTTL  INI.RRING INITIALIZATION TEST ROUTINES
000000  004137  000000G          INI.RRING::
                                        JSR     R1,$SAVE4                       ;                               2814
000004  013700  000000G                 MOV     DCT.ADDR,R0                     ;                               2835
000010  016001  000004                  MOV     4(R0),R1                        ; *,RRING.ADDR
000014  013703  000000G                 MOV     CCTLR,R3                        ;                               2839
000020  012704  000004                  MOV     #4,R4                           ; *,COUNT                       2837
000024  010346                  1$:     MOV     R3,-(SP)                        ;                               2839
000026  004737  000000G                 JSR     PC,GET.PKT
000032  010002                          MOV     R0,R2                           ; *,INDEX
000034  010216                          MOV     R2,(SP)                         ; INDEX,*                       2840
000036  012746  000106                  MOV     #106,-(SP)
000042  004737  000000G                 JSR     PC,BL$MUL
000046  016021  000000G                 MOV     MSCP.PKT(R0),(R1)+              ; *,RRING.ADDR
000052  016011  000002G                 MOV     MSCP.PKT+2(R0),(R1)             ; *,RRING.ADDR                  2842
000056  013703  000000G                 MOV     CCTLR,R3                        ;                               2843
```

```
000062  110362  000000G                        MOVB    R3,PKT.USE(R2)          ; *,*(INDEX)
000066  052721  140000                         BIS     #140000,(R1)+           ; *,RRING.ADDR                  2844
000072  022626                                 CMP     (SP)+,(SP)+             ;                               2838
000074  005304                                 DEC     R4                      ; COUNT                         2837
000076  001352                                 BNE     1$
000100  000207                                 RTS     PC                      ;                               2814

; Routine Size:  33 words,      Routine Base:  $CODE$ + 4140
; Maximum stack depth per invocation:  8 words
```

```
;    2849  1     GLOBAL routine SET_CTLR_CHAR =
;    2850  1
;    2851  1     !+
;    2852  1     !       THIS ROUTINE IS CALLED BY CTLR_INIT AFTER THE RDRX HAS BEEN HARD-
;    2853  1     !       INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
;    2854  1     !       CHARACTERISTICS" COMMAND, AND TO VALIDATE THE RESPONSE (END MESSAGE).
;    2855  1     !
;    2856  1     !       IMPLICIT INPUTS:
;    2857  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2858  1     !-
;    2859  1
;    2860  2         begin
;    2861  2
;    2862  2         local
;    2863  2             P_INDEX : word;
;    2864  2
;    2865  2
;    2866  2     ! MISCELLANEOUS INITIALIZATION                                          !ZZZ
;    2867  2                                                                            !ZZZ
;    2868  2     QIO [.CCTLR] = 0;                   !INIT NO OF OUTSTANDING QIOS       !ZZZ
;    2869  2     CST [.CCTLR, U_CNT] = 0;            !CLEAR UNITS IN CST TABLE          !ZZZ
;    2870  2     INCR COUNT FROM 0 TO (RP_CNT - 1) DO    !INIT RETURN PACKET POOL       !ZZZ
;    2871  2          RP_USE [.COUNT] = -1;                                             !ZZZ
;    2872  2                                                                            !ZZZ
;    2873  2     IODQ_IN = IODQ_OUT = 0;             !INIT I/O DONE QUEUE POINTERS      !ZZZ
;    2874  2
;    2875  2
;    2876  2         P_INDEX = GET_PKT (.CCTLR);                                 ! GET AN MSCP PACKET
;    2877  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_SCC;                       ! PACKET SIZE
;    2878  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_SCC;                       ! OPCODE = SET CTLR CHAR
;    2879  2         MSCP_PKT [.P_INDEX, C_FLAGS] = CF_MASK;                     ! CONTROLLER FLAGS
;    2880  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = IMM_CMD;                    ! IMMEDIATE COMMAND
;    2881  2
;    2882  2         if SEND (.P_INDEX) eql FAILURE                              ! ATTEMPT SEND
;    2883  2         then
;    2884  3             begin                                                   ! IF SEND WAS UNSUCCESSFUL
;    2885  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2886  3
;    2887  3             if .APT_MODE
;    2888  3             then
;    2889  4                 begin
;    2890  4                 .MAIL_BOX_TESTNUM = 1;
;    2891  4                 .MAIL_BOX_SUBTST = 0;
;    2892  3                 end;
;    2893  3
;    2894  3             ERRDF (20, EGD_20, 0);                                  ! FATAL ERROR
;    2895  3             PUT_PKT (.P_INDEX);                                     ! RETURN PACKET TO POOL
;    2896  3             DROP_CTLR (.CCTLR, DU_CFATAL);                          ! DROP CONTROLLER
;    2897  3             return FAILURE;
;    2898  3             end
;    2899  2         else
;    2900  3             begin                                                   ! IF SEND WAS SUCCESSFUL
;    2901  3
```

```
;    2902  3              do
;    2903  4                  begin
;    2904  4                  WAIT ();                                          ! WAIT FOR RETPKT RESPONSE
;    2905  4                  RP_INDX = OUT_IODQ ();                            ! GET INDEX OF RETPKT
;    2906  4                  RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);       ! CALCULATE RETPKT ADDRESS
;    2907  4
;    2908  4                  if .RP_ADDR [MESTYP] neq MT_SEQ                   ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    2909  4                  then
;    2910  4                      PUT_RETPKT (.RP_INDX);
;    2911  4
;    2912  4                  end
;    2913  3              until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    2914  4                    ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    2915  3                     ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    2916  3
;    2917  3              if .RP_ADDR [CONID] eql CID_DRIVER                    ! IF RETPKT IS FROM "DRIVER"
;    2918  3              then
;    2919  4                  begin
;    2920  4                  PRINTF (DBM23);                                   ! "ERROR IN SET_CTLR_CHAR"
;    2921  4                  PUT_RETPKT (.RP_INDX);                            ! RELEASE RETURN PACKET
;    2922  4                  DR_ERR ();                                        ! DROP CONTROLLER
;    2923  4                  return FAILURE;
;    2924  4                  end
;    2925  3              else
;    2926  4                  begin                                             ! ELSE - RETPKT IS FROM DISK MSCP
;    2927  4
;    2928  4                  if (.RP_ADDR [ENDCOD] neq (OP_SCC or OP_END)) or  ! IF WRONG ENDCODE
;    2929  5                     ((.RP_ADDR [C_FLGS] and CF_MASK) neq CF_MASK)  ! OR FLAGS IN ERROR
;    2930  4                  then
;    2931  5                      begin
;    2932  5                      C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2933  5
;    2934  5                      if .APT_MODE
;    2935  5                      then
;    2936  6                          begin
;    2937  6                          .MAIL_BOX_TESTNUM = 1;
;    2938  6                          .MAIL_BOX_SUBTST = 0;
;    2939  5                          end;
;    2940  5
;    2941  5                      ERRDF (21, EGD_21, EMS_21);                   ! FATAL ERROR
;    2942  5                      DROP_CTLR (.CCTLR, DU_CFATAL);                ! DROP CONTROLLER
;    2943  5                      PUT_RETPKT (.RP_INDX);                        ! RELEASE RETURN PACKET
;    2944  5                      return FAILURE;
;    2945  5                      end
;    2946  4                  else
;    2947  5                      begin                                         ! RETPKT HAS CORRECT ENDCODE
;    2948  5                      CMD_TIME = .RP_ADDR [C_TIME] * 2;
;    2949  5
;    2950  5      !ZZZ            if BIT_TST (SWP_FLAGS, SWF_TRC)
;    2951  5      !ZZZ            then
;    2952  5      !ZZZ                PRINTF (DBM25, .RP_ADDR [C_TIME]);
;    2953  5
;    2954  4                      end;                                          ! RETPKT HAS CORRECT ENDCODE
```

```
;    2955  4
;    2956  3                end;                                        ! IF RETPKT WAS SENT BY DISK MSCP
;    2957  3
;    2958  3            PUT_RETPKT (.RP_INDX);
;    2959  3            return SUCCESS;                                 ! IF SEND WAS SUCCESSFUL
;    2960  2            end;
;    2961  2
;    2962  1        end;                                                ! ROUTINE SET_CTLR_CHAR


                                        .SBTTL  SET.CTLR.CHAR INITIALIZATION TEST ROUTINES
000000  010146                    SET.CTLR.CHAR::
                                        MOV     R1,-(SP)                ;                                       2849
000002  013701  000000G                 MOV     CCTLR,R1                ;                                       2868
000006  105061  000000G                 CLRB    QIO(R1)
000012  010146                          MOV     R1,-(SP)                ;                                       2869
000014  012746  000126                  MOV     #126,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  105060  000005G                 CLRB    CST+5(R0)
000030  005000                          CLR     R0                      ; COUNT                                 2870
000032  112760  000377  000000G  1$:    MOVB    #377,RP.USE(R0)         ; *,*(COUNT)                            2871
000040  005200                          INC     R0                      ; COUNT                                 2870
000042  020027  000007                  CMP     R0,#7                   ; COUNT,*
000046  003771                          BLE     1$
000050  005037  000000G                 CLR     IOQ.OUT                 ;                                       2873
000054  005037  000000G                 CLR     IOQ.IN
000060  010116                          MOV     R1,(SP)                 ;                                       2876
000062  004737  000000G                 JSR     PC,GET.PKT
000066  010001                          MOV     R0,R1                   ; *,P.INDEX
000070  010116                          MOV     R1,(SP)                 ; P.INDEX,*                             2877
000072  012746  000106                  MOV     #106,-(SP)
000076  004737  000000G                 JSR     PC,BL$MUL
000102  012760  000040  000006G         MOV     #40,MSCP.PKT+6(R0)
000110  112760  000004  000022G         MOVB    #4,MSCP.PKT+22(R0)      ;                                       2878
000116  012760  000120  000030G         MOV     #120,MSCP.PKT+30(R0)    ;                                       2879
000124  105060  000004G                 CLRb    MSCP.PKT+4(R0)          ;                                       2880
000130  010116                          MOV     R1,(SP)                 ; P.INDEX,*                             2882
000132  004737  000000G                 JSR     PC,SEND
000136  005700                          TST     R0
000140  001036                          BNE     3$
000142  013700  000000G                 MOV     CCTLR,R0                ;                                       2885
000146  006300                          ASL     R0
000150  105260  000000G                 INCB    C.ERR.TBL(R0)
000154  032737  000001  001254'         BIT     #1,APT.MODE             ;                                       2887
000162  001405                          BEQ     2$
000164  012777  000001  001256'         MOV     #1,@MAIL.BOX.TESTNUM    ;                                       2890
000172  005077  001260'                 CLR     @MAIL.BOX.SUBTST        ;                                       2891
000176  104455                  2$:     TRAP    55                      ;                                       2894
000200  000024                          .WORD   24
000202  000000G                         .WORD   EGD.20
000204  000000                          .WORD   0
000206  010116                          MOV     R1,(SP)                 ; P.INDEX,*                             2895
000210  004737  000000G                 JSR     PC,PUT.PKT
```

```
000214  013716  000000G                        MOV     CCTLR,(SP)                      ;                               2836
000220  012746  000006                         MOV     #6,-(SP)
000224  004737  000000G                        JSR     PC,DROP.CTLR
000230  005726                                 TST     (SP)+                           ;                               2884
000232  005000                                 CLR     R0                              ;                               2900
000234  000554                                 BR      12$
000236  004737  000000G               3$:     JSR     PC,WAIT                         ;                               2904
000242  004737  000000G                        JSR     PC,OUT.IODQ                     ;                               2905
000246  010037  000000G                        MOV     R0,RP.INDX
000252  010016                                 MOV     R0,(SP)                         ; RP.INDX,●                     2906
000254  012746  000054                         MOV     #54,-(SP)
000260  004737  000000G                        JSR     PC,BL$MUL
000264  062700  000000G                        ADD     #RETPKT,R0
000270  010037  000000G                        MOV     R0,RP.ADDR
000274  132760  000360  000002                 BITB    #360,2(R0)                      ;                               2908
000302  001404                                 BEQ     4$
000304  013716  000000G                        MOV     RP.INDX,(SP)                    ;                               2910
000310  004737  000000G                        JSR     PC,PUT.RETPKT
000314  005726                       4$:       TST     (SP)+                           ;                               2903
000316  013701  000000G                        MOV     RP.ADDR,R1                      ;                               2913
000322  005000                                 CLR     R0
000324  126127  000003  000003                 CMPB    3(R1),#3
000332  001002                                 BNE     5$
000334  005200                                 INC     R0
000336  000407                                 BR      6$
000340  132761  000360  000002       5$:       BITB    #360,2(R1)                      ;                               2914
000346  001333                                 BNE     3$
000350  105761  000014                         TSTB    14(R1)                          ;                               2915
000354  100330                                 BPL     3$
000356  006000                       6$:       ROR     R0                              ;                               2917
000360  103015                                 BCC     7$
000362  012716  000000G                        MOV     #DBM23,(SP)                     ;                               2920
000366  012746  000001                         MOV     #1, (SP)
000372  010600                                 MOV     SP,R0                           ; SP,●
000374  104417                                 TRAP    17
000376  013716  000000G                        MOV     RP.INDX,(SP)                    ;                               2921
000402  004737  000000G                        JSR     PC,PUT.RETPKT
000406  004737  000000V                        JSR     PC,DR.ERR                       ;                               2922
000412  000447                                 BR      10$                             ;                               2923
000414  126127  000014  000204       7$:       CMPB    14(R1),#204                     ;                               2928
000422  001007                                 BNE     8$
000424  016100  000022                         MOV     22(R1),R0                       ;                               2929
000430  042700  177657                         BIC     #177657,R0
000434  020027  000120                         CMP     R0,#120
000440  001437                                 BEQ     11$
000442  013700  000000G               8$:       MOV     CCTLR,R0                        ;                               2932
000446  006300                                 ASL     R0
000450  105260  000000G                        INCB    C.ERR.TBL(R0)
000454  032737  000001  001254'                BIT     #1,APT.MODE                     ;                               2934
000462  001405                                 BEQ     9$
000464  012777  000001  001256                 MOV     #1,@MAIL.BOX.TESTNUM            ;                               2937
000472  005077  001260'                        CLR     @MAIL.BOX.SUBTST                ;                               2938
000476  104455                       9$:       TRAP    55                              ;                               2941
```

```
000500  000025                                  .WORD   25
000502  000000G                                 .WORD   EGD.21
000504  000000G                                 .WORD   EMS.21
000506  013716  000000G                         MOV     CCTLR,(SP)              ;                       2942
000512  012746  000006                          MOV     #6,-(SP)
000516  004737  000000G                         JSR     PC,DROP.CTLR
000522  013716  000000G                         MOV     RP.INDX,(SP)            ;                       2943
000526  004737  000000G                         JSR     PC,PUT.RETPKT
000532  062706  000010                  10$:    ADD     #10,SP                  ;                       2944
000536  000416                                  BR      13$                     ;                       2931
000540  016137  000024  000000G         11$:    MOV     24(R1),CMD.TIME         ;                       2948
000546  006337  000000G                         ASL     CMD.TIME
000552  013716  000000G                         MOV     RP.INDX,(SP)            ;                       2958
000556  004737  000000G                         JSR     PC,PUT.RETPKT
000562  012700  000001                          MOV     #1,R0                   ;                       2900
000566  062706  000006                  12$:    ADD     #6,SP                   ;                       2882
000572  000401                                  BR      14$                     ;                       2860
000574  005000                          13$:    CLR     R0                      ;                       2849
000576  012601                          14$:    MOV     (SP)+,R1
000600  000207                                  RTS     PC

; Routine Size:  193 words,     Routine Base:  $CODE$ + 4242
; Maximum stack depth per invocation:  7 words
```

```
;  2963  1     routine UNIT_INIT : novalue =
;  2964  1
;  2965  1     !+
;  2966  1     !       THIS ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONFIGURED UNIT
;  2967  1     !       (DISK) WHICH IS ATTACHED TO A CONTROLLER THAT SURVIVED
;  2968  1     !       INITIALIZATION. ITS PURPOSE IS TO FORMAT AND SEND AN "ONLINE"
;  2969  1     !       MESSAGE, AND TO VERIFY THE RESPONSE.
;  2970  1     !
;  2971  1     !       IMPLICIT INPUTS:
;  2972  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  2973  1     !               CDISK   CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  2974  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;  2975  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  2976  1     !
;  2977  1
;  2978  2         begin
;  2979  2         local
;  2980  2             MAX0_LBNS : WORD UNSIGNED,              ! UNIT'S MAXIMUM LO WORD LBN
;  2981  2             MAX1_LBNS : WORD UNSIGNED;              ! UNIT'S MAXIMUM HI WORD LBN
;  2982  2
;  2983  2
;  2984  2         P_INDEX = GET_PKT (.CCTLR);                 ! GET AN MSCP PACKET
;  2985  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_ONL;       ! PACKET SIZE
;  2986  2         MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;       ! SET DISK ADDRESS (RD/RX DISK NUMBER)
;  2987  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_ONL;       ! OPCODE FOR "ONLINE"
;  2988  2     !ZZZ MSCP_PKT [.P_INDEX, DDPAR] = BIT00;        ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
;  2989  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = SEQ_CMD;    ! SEQUENTIAL COMMAND
;  2990  2
;  2991  2         if SEND (.P_INDEX) eql FAILURE              ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
;  2992  2         then
;  2993  3             begin
;  2994  3             T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  2995  3
;  2996  3             if .APT_MODE                            !ZZZ
;  2997  3             then
;  2998  4                 begin
;  2999  4                 .MAIL_BOX_TESTNUM = 1;
;  3000  4                 .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  3001  3                 end;
;  3002  3
;  3003  3             CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
;  3004  3             ERRDF (22, EGD_22, 0);                  !
;  3005  3             DUR [.L$LUN] = DU_ONLINE;               ! SETUP REASON TO DROP UNIT
;  3006  3             DODU (.L$LUN);                          ! DROP UNIT
;  3007  3             PUT_PKT (.P_INDEX);                     ! RETURN PACKET TO POOL
;  3008  3             end
;  3009  2         else
;  3010  3             begin                                   ! OTHERWISE (SEND WAS SUCCESSFUL)
;  3011  3
;  3012  3             do
;  3013  4                 begin
;  3014  4                 WAIT ();                            ! WAIT FOR RETPKT RESPONSE
;  3015  4                 RP_INDX = OUT_IODQ ();              ! GET INDEX OF RETPKT
```

```
;    3016  4                RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;    3017  4
;    3018  4                if .RP_ADDR [MESTYP] neq MT_SEQ     ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    3019  4                then
;    3020  4                    PUT_RETPKT (.RP_INDX);
;    3021  4
;    3022  4                end
;    3023  3            until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    3024  4                  ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    3025  3                   ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    3026  3
;    3027  3            if .RP_ADDR [CONID] eql CID_DRIVER      ! IF RETPKT IS FROM "DRIVER"
;    3028  3            then
;    3029  4                begin
;    3030  4                PRINTF (DBM26);                     ! "ERROR IN UNIT_INIT"
;    3031  4                DR_ERR ();                          ! DROP CONTROLLER
;    3032  4                end
;    3033  3            else
;    3034  3
;    3035  4                if .RP_ADDR [ENDCOD] neq (OP_ONL or OP_END) ! IF RETPKT IS FROM DISK MSCP
;    3036  3                then
;    3037  4                    begin
;    3038  4                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    3039  4
;    3040  4                    if .APT_MODE                    !ZZZ
;    3041  4                    then
;    3042  5                        begin
;    3043  5                        .MAIL_BOX_TESTNUM = 1;
;    3044  5                        .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    3045  4                        end;
;    3046  4
;    3047  4                    CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    3048  4                    ERRDF (23, EGD_23, EMS_21);     ! FATAL ERROR
;    3049  4                    DUR [.L$LUN] = DU_ONLINE;       ! SETUP REASON TO DROP UNIT
;    3050  4                    DODU (.L$LUN);                  ! DROP UNIT
;    3051  4                    end
;    3052  3                else
;    3053  4                    begin                           ! RETPKT HAS GOOD ENDCODE
;    3054  4                    ST_CODE = .RP_ADDR [STSCOD];    ! GET STATUS CODE
;    3055  4                    SB_CODE = .RP_ADDR [SUBCOD];    ! GET SUB-CODE
;    3056  4
;    3057  4                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_0] = .RP_ADDR [NAME_0] + %o'100';                  ! UNIT NAME
ZZZ
;    3058  4                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .RP_ADDR [NAME_1_HI] * 16;                    !ZZZ
;    3059  4                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] +    !ZZZ
;    3060  4                                                              .RP_ADDR [NAME_1_LO] + %o'100';               !ZZZ
;    3061  4                    CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_2] = .RP_ADDR [NAME_NUM] / 10 + %o'60';            !ZZZ
;    3062  4                    CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_3] = (.RP_ADDR [NAME_NUM] mod 10) + %o'60';        !ZZZ
;    3063  4
;    3064  4
;    3065  4
;    3066  4                    IF .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] EQL %O'104'         !IF NAME IS _D          !ZZZ
;    3067  4                    THEN                                                                                    !ZZZ
;    3068  4                    CST ADDR [.CUOFF, D TYPE] = FIXED                               !ITS FIXED.             !ZZZ
```

```
;    3069  4                    ELSE
;    3070  4                    CST_ADDR [.CUOFF, D_TYPE] = REMOVABLE;                          !OTHERWISE REMOVABLE    !ZZZ
;    3071  4                                                                                                            !ZZZ
;    3072  4
;    3073  4
;    3074  4                    if .ST_CODE neq ST_SUC          ! IF STATUS CODE IS NOT SUCCESSFUL
;    3075  4                    then
;    3076  5                        begin
;    3077  5                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    3078  5
;    3079  5                        if .APT_MODE                !ZZZ
;    3080  5                        then
;    3081  6                            begin
;    3082  6                            .MAIL_BOX_TESTNUM = 1;
;    3083  6                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    3084  5                            end;
;    3085  5
;    3086  5                        CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    3087  5                        ERRDF (15, EGD_15, EMS_30); ! ONLINE FAILED
;    3088  5                        DUR [.L$LUN] = DU_ONLINE;   ! SET UP REASON FOR DROPPING UNIT
;    3089  5                        DODU (.L$LUN);              ! DROP UNIT
;    3090  5                        end
;    3091  4                    else
;    3092  5                        begin                       ! SUCCESFUL OPERATION
;    3093  5
;    3094  5                        MAXO_LBNS = .RP_ADDR [SIZE0];                               ! LOAD LOWER WORD OF UNIT SIZE
;    3095  5                        MAX1_LBNS = .RP_ADDR [SIZE1];                               ! LOAD UPPER WORD OF UNIT SIZE
;    3096  5
;    3097  6                        if (.MAXO_LBNS eql 0)                                       ! THIS SUBTRACTS ONE FROM THE TOTAL
;    3098  5                        then                                                        ! BECAUSE EVERYTHING STARTS AT 0
;    3099  6                            begin                                                   ! THROUGH (MAXIMUM - 1)
;    3100  6                            MAXO_LBNS = %O'177777';
;    3101  6                            MAX1_LBNS = .MAX1_LBNS - 1;
;    3102  6                            end
;    3103  5                        else
;    3104  5                            MAXO_LBNS = .MAXO_LBNS - 1;
;    3105  5
;    3106  5                        if  (.CST_ADDR [.CUOFF + 2, D_BEG1] gtru .MAX1_LBNS) or             ! THIS SECTION CHECKS TO SEE
 IF LBNs LISTED
;    3107  5
;    3108  6                            ((.CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .MAX1_LBNS) and           ! IN SOFTWARE QUESTIONS WERE
 TO LARGE FOR
;    3109  6                            (.CST_ADDR [.CUOFF + 1, D_BEG0] gtru (.MAXO_LBNS - 1)))         ! DEVICE SPECIFIED
;    3110  6                                                                                    ! note 1 less then max. or diagnosti
c  will error
;    3111  5                        then                                                        ! operator error
;    3112  6                            begin
;    3113  6                            CST_ADDR [.CUOFF + 2, D_BEG1] = 0;
;    3114  6                            CST_ADDR [.CUOFF + 1, D_BEG0] = 0;                      ! change beginning lbn to 0
;    3115  5                            end;
;    3116  5
;    3117  5                         if
;    3118  5                            (.CST_ADDR [.CUOFF + 4, D_END1] gtru .MAX1_LBNS) or
;    3119  5
;    3120  6                            ((.CST_ADDR [.CUOFF + 4, D_END1] eqlu .MAX1_LBNS) and
;    3121  6                            (.CST_ADDR [.CUOFF + 3, D_END0] gtru .MAXO_LBNS))
```

```
;  3122  5                        then
;  3123  6                            begin
;  3124  6                            CST_ADDR [.CUOFF + 4, D_END1] = .MAX1_LBNS;
;  3125  6                            CST_ADDR [.CUOFF + 3, D_END0] = .MAX0_LBNS;          ! end ending lbn to max_lbn
;  3126  5                            end;
;  3127  5
;  3128  6                        if  (.CST_ADDR [.CUOFF + OF_BEG1, D_BEG1] gtru           !MAKE SURE START ADDRESS
      !ZZZ
;  3129  5                            .CST_ADDR [.CUOFF + OF_END1, D_END1]) or             !IS NO LARGER THAN END ADDRESS
      !ZZZ
;  3130  5                                                                                 !
      !ZZZ
;  3131  7                            ((.CST_ADDR [.CUOFF + OF_BEG1, D_BEG1] eqlu          !
      !ZZZ
;  3132  6                              .CST_ADDR [.CUOFF + OF_END1, D_END1]) and          !
      !ZZZ
;  3133  7                            (.CST_ADDR [.CUOFF + OF_BEG, D_BEG0] gtru            !
      !ZZZ
;  3134  6                             .CST_ADDR [.CUOFF + OF_END, D_END0] ))              !
      !ZZZ
;  3135  6                                                                                 !
      !ZZZ
;  3136  5                        then                                                     !
      !ZZZ
;  3137  6                            begin                                                !
      ZZZ
;  3138  6                            CST_ADDR [.CUOFF + OF_BEG1, D_BEG1] = 0;             !IF IT IS, THEN
      ZZZ
;  3139  6                            CST_ADDR [.CUOFF + OF_BEG, D_BEG0] = 0;              ! change beginning lbn to 0
      ZZZ
;  3140  5                            end;                                                 !
      ZZZ
;  3141  5
       !ZZZ
;  3142  5
;  3143  7                        if (((.ENTRY_REASON eql RESTART) or                      ! if restart or
;  3144  6                            (.ENTRY_REASON eql START)) and                       ! if continue
;  3145  6
;  3146  6                            (.CRN_LOW leq 8) and                                 ! and
;  3147  6                            (.CRN_HIGH eql 0))                                   ! first initialization
;  3148  6
;  3149  5                        THEN                                                     ! intialize block numbers
;  3150  6                            begin
;  3151  6                            BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0];      ! LOAD sequential LBN table
;  3152  6                            BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1];      !
;  3153  6                            TRK_SGN [.L$LUN] = 1;                                       ! POSITIVE TRACKING DIRECTIO
N
;  3154  5                            end;
;  3155  5
;  3156  5
;  3157  5                                                                                 ! THIS SECTION LOADS TYPE INTO  CST TABLE
;  3158  5     !ZZZ                        selectoneu .RP_ADDR [R_MODEL] of                    ! MODEL BYTE TELLS WHAT TYPE OF UNIT
 IN
;  3159  5     !ZZZ                                                                            ! INDENTIFICATION BLOCK
;  3160  5     !ZZZ                            set
;  3161  5     !ZZZ
;  3162  5     !ZZZ                            [%O'6'] :   CST_ADDR [.CUOFF, D_TYPE] = RD_51;      ! RD 51
;  3163  5     !ZZZ                            [%O'7'] :   CST_ADDR [.CUOFF, D_TYPE] = RX_50;      ! RX 50
;  3164  5     !ZZZ                            [%O'10'] :  CST_ADDR [.CUOFF, D_TYPE] = RD_52;      ! RD 52
;  3165  5     !ZZZ
;  3166  5     !ZZZ                            [otherwise] : BEGIN
;  3167  5     !ZZZ                                          ERRDF (25 ,EGD_24 ,EMS_30);           ! ERROR UNKNOWN DEVICE
;  3168  5     !ZZZ                                          END;
;  3169  5     !ZZZ                            tes;
;  3170  5
```

```
;  3173  5                    if ((.RP_ADDR [U_FLGS] and OF_WPH) eq OF_WPH) and          ! STATUS CODE IS O.K.
;  3174  6                       (.CST_ADDR [.CUOFF, D_PROT] eq UNPROTECTED)

;  3175  5                    then
;  3176  6                        begin
;  3177  6                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  3178  6
;  3179  6                        if .APT_MODE                        !ZZZ
;  3180  6                        then
;  3181  7                            begin
;  3182  7                            .MAIL_BOX_TESTNUM = 1;
;  3183  7                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  3184  6                            end;
;  3185  6
;  3186  6                        CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;  3187  6                        ERRDF (16, EGD_16, EMS_30);                                 ! WRITE-PROTECT CONFLICT
;  3188  6                        DUR [.L$LUN] = DU_PROTECT;                                  ! SET REASON TO DROP UNIT
;  3189  6                        DODU (.L$LUN);                                              ! DROP UNIT
;  3190  6                        end
;  3191  5                    else
;  3192  6                        begin                                                       ! WRITE PROTECT SWITCH IS O.K.
;  3193  6                        CST_ADDR [.CUOFF, D_STAT] = ONLINE;                         ! SET ONLINE FLAG
;  3194  6                        CST [.CCTLR, U_CNT] = .CST [.CCTLR, U_CNT] + 1;             ! ADD UNIT TO CTRL TABLE
;  3195  5                        end;
;  3196  4                    end;
;  3197  3                end;                                    ! IF RETPKT HAS CORRECT ENDCODE
;  3198  3
;  3199  3            PUT_RETPKT (.RP_INDX);
;  3200  2            end;                                        ! IF SEND WAS SUCCESSFUL
;  3201  2
;  3202  1        end;                                            ! ROUTINE UNIT-INIT


                                        .SBTTL  UNIT.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G           UNIT.INIT:
                                        JSR     R1,$SAVES                       ;                                       2963
000004  005746                          TST     -(SP)
000006  013746  000000G                 MOV     CCTLR,-(SP)                     ;                                       2984
000012  004737  000000G                 JSR     PC,GET.PKT
000016  010037  000000G                 MOV     R0,P.INDEX
000022  010016                          MOV     R0,(SP)                         ; P.INDEX,*                             2985
000024  012746  000106                  MOV     #106,-(SP)
000030  004737  000000G                 JSR     PC,BL$MUL
000034  012760  000044  000006G         MOV     #44,MSCP.PKT+6(R0)
000042  013760  000000G  000016G         MOV     CDISK,MSCP.PKT+16(R0)           ;                                       2986
000050  112760  000011  000022G         MOVB    #11,MSCP.PKT+22(R0)             ;                                       2987
000056  112760  000001  000004G         MOVB    #1,MSCP.PKT+4(R0)               ;                                       2989
000064  013716  000000G                 MOV     P.INDEX,(SP)                    ;                                       2991
000070  004737  000000G                 JSR     PC,SEND
000074  005700                          TST     R0
000076  001054                          BNE     2$
000100  013700  000000G                 MOV     T.ADDR,R0                       ;                                       2994
000104  105260  000051                  INCB    51(R0)
000110  032737  000001  001254'         BIT     #1,APT.MODE                     ;                                       2996
000116  001415                          BEQ     1$
000120  012777  000001  001256'         MOV     #1,@MAIL.BOX.TESTNUM            ;                                       2999
```

```
000126  013700  000000G                        MOV     CUOFF,R0                ;                       3000
000132  006300                                  ASL     R0
000134  063700  000000G                        ADD     CST.ADDR,R0
000140  111077  001260'                        MOVB    (R0),@MAIL.BOX.SUBTST
000144  042777  177760  001260'                BIC     #177760,@MAIL.BOX.SUBTST
000152  013700  000000G              1$:       MOV     CUOFF,R0                ;                       3003
000156  006300                                  ASL     R0
000160  063700  000000G                        ADD     CST.ADDR,R0
000164  052710  010000                        BIS     #10000,(R0)
000170  104455                                TRAP    55                      ;                       3004
000172  000026                                .WORD   26
000174  000000G                               .WORD   EGD.22
000176  000000                                .WORD   0
000200  013700  000000G                        MOV     L$LUN,R0                ;                       3005
000204  112760  000007  000000G               MOVB    #7,DUR(R0)
000212  104451                                TRAP    51                      ;                       3006
000214  013716  000000G                        MOV     P.INDEX,(SP)            ;                       3007
000220  004737  000000G                        JSR     PC,PUT.PKT
000224  000137  007000'                        JMP     28$                     ;                       2991
000230  004737  000000G              2$:       JSR     PC,WAIT                 ;                       3014
000234  004737  000000G                        JSR     PC,OUT.IODQ             ;                       3015
000240  010037  000000G                        MOV     R0,RP.INDX
000244  010016                                MOV     R0,(SP)                 ; RP.INDX,*             3016
000246  012746  000054                        MOV     #54,-(SP)
000252  004737  000000G                        JSR     PC,BL$MUL
000256  062700  000000G                        ADD     #RETPKT,R0
000262  010037  000000G                        MOV     R0,RP.ADDR
000266  132760  000360  000002               BITB    #360,2(R0)              ;                       3018
000274  001404                                BEQ     3$
000276  013716  000000G                        MOV     RP.INDX,(SP)            ;                       3020
000302  004737  000000G                        JSR     PC,PUT.RETPKT
000306  005726                      3$:       TST     (SP)+                   ;                       3013
000310  013702  000000G                        MOV     RP.ADDR,R2              ;                       3023
000314  005000                                CLR     R0
000316  126227  000003  000003               CMPB    3(R2),#3
000324  001002                                BNE     4$
000326  005200                                INC     R0
000330  000407                                BR      5$
000332  132762  000360  000002       4$:      BITB    #360,2(R2)              ;                       3024
000340  001333                                BNE     2$
000342  105762  000014                        TSTB    14(R2)                  ;                       3025
000346  100330                                BPL     2$
000350  006000                      5$:       ROR     R0                      ;                       3027
000352  103012                                BCC     6$
000354  012716  000000G                        MOV     #DBM26,(SP)             ;                       3030
000360  012746  000001                        MOV     #1,-(SP)
000364  010600                                MOV     SP,R0                   ; SP,*
000366  104417                                TRAP    17
000370  004737  000000V                        JSR     PC,DR.ERR               ;                       3031
000374  005726                                TST     (SP)+                   ;                       3029
000376  000456                                BR      8$                      ;                       3027
000400  013766  000000G 000004       6$:      MOV     CUOFF,4(SP)             ;                       3047
000406  006366  000004                        ASL     4(SP)
```

```
000412  063766  000000G 000004                 ADD     CST.ADDR,4(SP)
000420  126227  000014  000211                 CMPB    14(R2),#211             ;                       3035
000426  001444                                 BEQ     9$
000430  013700  000000G                        MOV     T.ADDR,R0               ;                       3038
000434  105260  000050                         INCB    50(R0)
000440  032737  000001  001254'               BIT     #1,APT.MODE             ;                       3040
000446  001415                                 BEQ     7$
000450  012777  000001  001256'               MOV     #1,@MAIL.BOX.TESTNUM    ;                       3043
000456  013700  000000G                        MOV     CUOFF,R0                ;                       3044
000462  006300                                 ASL     R0
000464  063700  000000G                        ADD     CST.ADDR,R0
000470  111077  001260'                       MOVB    (R0),@MAIL.BOX.SUBTST
000474  042777  177760  001260'               BIC     #177760,@MAIL.BOX.SUBTST
000502  052776  010000  000004        7$:     BIS     #10000,@4(SP)           ;                       3047
000510  104455                                 TRAP    55                      ;                       3048
000512  000027                                 .WORD   27
000514  000000G                                .WORD   EGD.23
000516  000000G                                .WORD   EMS.21
000520  013700  000000G                        MOV     L$LUN,R0                ;                       3049
000524  112760  000007  000000G               MOVB    #7,DUR(R0)
000532  104451                                 TRAP    51                      ;                       3050
000534  000137  006770'                8$:     JMP     27$                     ;                       3035
000540  116237  000016  000000G        9$:     MOVB    16(R2),ST.CODE          ;                       3054
000546  042737  177740  000000G               BIC     #177740,ST.CODE
000554  016200  000016                         MOV     16(R2),R0               ;                       3055
000560  006200                                 ASR     R0
000562  006200                                 ASR     R0
000564  006200                                 ASR     R0
000566  006200                                 ASR     R0
000570  006200                                 ASR     R0
000572  042700  174000                         BIC     #174000,R0
000576  010037  000000G                        MOV     R0,SB.CODE
000602  013701  000000G                        MOV     CUOFF,R1                ;                       3057
000606  006301                                 ASL     R1
000610  063701  000000G                        ADD     CST.ADDR,R1
000614  012703  000012                         MOV     #12,R3
000620  060103                                 ADD     R1,R3
000622  116200  000036                         MOVB    36(R2),R0
000626  006200                                 ASR     R0
000630  042700  177740                         BIC     #177740,R0
000634  062700  000100                         ADD     #100,R0
000640  110013                                 MOVB    R0,(R3)
000642  116200  000036                         MOVB    36(R2),R0               ;                       3058
000646  042700  177776                         BIC     #177776,R0
000652  006300                                 ASL     R0
000654  006300                                 ASL     R0
000656  006300                                 ASL     R0
000660  006300                                 ASL     R0
000662  110063  000001                         MOVB    R0,1(R3)
000666  005000                                 CLR     R0                      ;                       3059
000670  156300  000001                         BISB    1(R3),R0
000674  016201  000034                         MOV     34(R2),R1
000700  006201                                 ASR     R1
```

```
000702  006201                          ASR     R1
000704  006201                          ASR     R1
000706  006201                          ASR     R1
000710  000301                          SWAB    R1
000712  042701  177760                  BIC     #177760,R1
000716  060100                          ADD     R1,R0
000720  010001                          MOV     R0,R1                   ;                       3060
000722  062701  000100                  ADD     #100,R1
000726  110163  000001                  MOVB    R1,1(R3)
000732  013701  000000G                 MOV     CUOFF,R1                ;                       3061
000736  006301                          ASL     R1
000740  063701  000000G                 ADD     CST.ADDR,R1
000744  116216  000034                  MOVB    34(R2),(SP)
000750  042716  177700                  BIC     #177700,(SP)
000754  012746  000012                  MOV     #12,-(SP)
000760  004737  000000G                 JSR     PC,BL$DIV
000764  010004                          MOV     R0,R4
000766  062704  000060                  ADD     #60,R4
000772  110461  000014                  MOVB    R4,14(R1)
000776  116216  000034                  MOVB    34(R2),(SP)             ;                       3062
001002  042716  177700                  BIC     #177700,(SP)
001006  012746  000012                  MOV     #12,-(SP)
001012  004737  000000G                 JSR     PC,BL$MOD
001016  010004                          MOV     R0,R4
001020  062704  000060                  ADD     #60,R4
001024  110461  000015                  MOVB    R4,15(R1)
001030  126327  000001  000104          CMPB    1(R3),#104              ;                       3066
001036  001004                          BNE     10$
001040  152776  000020  000010          BISB    #20,@10(SP)             ;                       3068
001046  000403                          BR      11$                     ;                       3066
001050  142776  000020  000010  10$:    BICB    #20,@10(SP)             ;                       3070
001056  005737  000000G         11$:    TST     ST.CODE                 ;                       3074
001062  001440                          BEQ     13$
001064  013700  000000G                 MOV     T.ADDR,R0               ;                       3077
001070  105260  000050                  INCB    50(R0)
001074  032737  000001  001254'         BIT     #1,APT.MODE             ;                       3079
001102  001411                          BEQ     12$
001104  012777  000001  001256'         MOV     #1,@MAIL.BOX.TESTNUM    ;                       3082
001112  117677  000010  001260'         MOVB    @10(SP),@MAIL.BOX.SUBTST ;                      3083
001120  042777  177760  001260'         BIC     #177760,@MAIL.BOX.SUBTST
001126  052776  010000  000010  12$:    BIS     #10000,@10(SP)          ;                       3086
001134  104455                          TRAP    55                      ;                       3087
001136  000017                          .WORD   17
001140  000000G                         .WORD   EGD.15
001142  000000G                         .WORD   EMS.30
001144  013700  000000G                 MOV     L$LUN,R0                ;                       3088
001150  112760  000007  000000G         MOVB    #7,DUR(R0)
001156  104451                          TRAP    51                      ;                       3089
001160  000137  006766'                 JMP     26$                     ;                       3074
001164  016203  000044          13$:    MOV     44(R2),R3               ; *,MAX0.LBNS           3094
001170  016204  000046                  MOV     46(R2),R4               ; *,MAX1.LBNS           3095
001174  005703                          TST     R3                      ; MAX0.LBNS             3097
001176  001004                          BNE     14$
```

```
001200  012703  177777                    MOV     #-1,R3                  ; *,MAXO.LBNS                   3100
001204  005304                            DEC     R4                      ; MAX1.LBNS                     3101
001206  000401                            BR      15$                     ;                               3097
001210  005303                    14$:    DEC     R3                      ; MAXO.LBNS                     3104
001212  013701  000000G           15$:    MOV     CUOFF,R1                ;                               3106
001216  006301                            ASL     R1
001220  063701  000000G                    ADD     CST.ADDR,R1
001224  012705  000004                    MOV     #4,R5
001230  060105                            ADD     R1,R5
001232  021504                            CMP     (R5),R4                 ; *,MAX1.LBNS
001234  101013                            BHI     16$
001236  001022                            BNE     17$                     ;                               3108
001240  013701  000000G                    MOV     CUOFF,R1                ;                               3109
001244  006301                            ASL     R1
001246  063701  000000G                    ADD     CST.ADDR,R1
001252  010300                            MOV     R3,R0                   ; MAXO.LBNS,*
001254  005300                            DEC     R0
001256  026100  000002                    CMP     2(R1),R0
001262  101410                            BLOS    17$
001264  005015                    16$:    CLR     (R5)                    ;                               3113
001266  013701  000000G                    MOV     CUOFF,R1                ;                               3114
001272  006301                            ASL     R1
001274  063701  000000G                    ADD     CST.ADDR,R1
001300  005061  000002                    CLR     2(R1)
001304  013701  000000G           17$:    MOV     CUOFF,R1                ;                               3118
001310  006301                            ASL     R1
001312  063701  000000G                    ADD     CST.ADDR,R1
001316  012700  000010                    MOV     #10,R0
001322  060100                            ADD     R1,R0
001324  021004                            CMP     (R0),R4                 ; *,MAX1.LBNS
001326  101011                            BHI     18$
001330  001020                            BNE     19$                     ;                               3120
001332  013701  000000G                    MOV     CUOFF,R1                ;                               3121
001336  006301                            ASL     R1
001340  063701  000000G                    ADD     CST.ADDR,R1
001344  026103  000006                    CMP     6(R1),R3                ; *,MAXO.LBNS
001350  101410                            BLOS    19$
001352  010410                    18$:    MOV     R4,(R0)                 ; MAX1.LBNS,*                   3124
001354  013701  000000G                    MOV     CUOFF,R1                ;                               3125
001360  006301                            ASL     R1
001362  063701  000000G                    ADD     CST.ADDR,R1
001366  010361  000006                    MOV     R3,6(R1)                ; MAXO.LBNS,*
001372  021510                    19$:    CMP     (R5),(R0)               ;                               3128
001374  101017                            BHI     20$
001376  001026                            BNE     21$                     ;                               3131
001400  013700  000000G                    MOV     CUOFF,R0                ;                               3133
001404  006300                            ASL     R0
001406  063700  000000G                    ADD     CST.ADDR,R0
001412  013701  000000G                    MOV     CUOFF,R1                ;                               3134
001416  006301                            ASL     R1
001420  063701  000000G                    ADD     CST.ADDR,R1
001424  026061  000002  000006            CMP     2(R0),6(R1)             ;                               3133
001432  101410                            BLOS    21$
```

```
001434  005015                          20$:    CLR     (R5)
001436  013701  000000G                         MOV     CUOFF,R1        ;                       3138
001442  006301                                  ASL     R1              ;                       3139
001444  063701  000000G                         ADD     CST.ADDR,R1
001450  005061  000002                          CLR     2(R1)
001454  123727  000000G 000002        21$:    CMPB    ENTRY.REASON,#2 ;                       3143
001462  001404                                  BEQ     22$
001464  123727  000000G 000001                  CMPB    ENTRY.REASON,#1 ;                       3144
001472  001031                                  BNE     23$
001474  023727  000000G 000010        22$:    CMP     CRN.LOW,#10     ;                       3146
001502  003025                                  BGT     23$
001504  005737  000000G                         TST     CRN.HIGH        ;                       3147
001510  001022                                  BNE     23$
001512  013700  000000G                         MOV     L$LUN,R0        ;                       3151
001516  010004                                  MOV     R0,R4
001520  006304                                  ASL     R4
001522  006304                                  ASL     R4
001524  013701  000000G                         MOV     CUOFF,R1
001530  006301                                  ASL     R1
001532  063701  000000G                         ADD     CST.ADDR,R1
001536  016164  000002  000000G                MOV     2(R1),BST(R4)
001544  011564  000002G                         MOV     (R5),BST+2(R4)
001550  112760  000001  000000G                MOVB    #1,TRK.SGN(R0)  ;                       3152
001556  032762  020000  000022        23$:    BIT     #20000,22(R2)   ;                       3153
001564  001442                                  BEQ     25$             ;                       3173
001566  005776  000010                          TST     @10(SP)
001572  100037                                  BPL     25$             ;                       3174
001574  013700  000000G                         MOV     T.ADDR,R0       ;                       3177
001600  105260  000050                          INCB    50(R0)
001604  032737  000001  001254'                BIT     #1,APT.MODE     ;                       3179
001612  001411                                  BEQ     24$
001614  012777  000001  001256'                MOV     #1,@MAIL.BOX.TESTNUM ;                  3182
001622  117677  000010  001260'                MOVB    @10(SP),@MAIL.BOX.SUBTST ;              3183
001630  042777  177760  001260'                BIC     #177760,@MAIL.BOX.SUBTST
001636  052776  010000  000010        24$:    BIS     #10000,@10(SP)  ;                       3186
001644  104455                                  TRAP    55              ;                       3187
001646  000020                                  .WORD   20
001650  000000G                                 .WORD   EGD.16
001652  000000G                                 .WORD   EMS.30
001654  013700  000000G                         MOV     L$LUN,R0        ;                       3188
001660  112760  000011  000000G                MOVB    #11,DUR(R0)
001666  104451                                  TRAP    51              ;                       3189
001670  000414                                  BR      26$             ;                       3173
001672  052776  020000  000010        25$:    BIS     #20000,@10(SP)  ;                       3193
001700  013716  000000G                         MOV     CCTLR,(SP)      ;                       3194
001704  012746  000126                          MOV     #126,-(SP)
001710  004737  000000G                         JSR     PC,BL$MUL
001714  105260  000005G                         INCB    CST+5(R0)
001720  005726                                  TST     (SP)+           ;                       3192
001722  022626                          26$:    CMP     (SP)+,(SP)+     ;                       3053
001724  013716  000000G                 27$:    MOV     RP.INDX,(SP)    ;                       3199
001730  004737  000000G                         JSR     PC,PUT.RETPKT
001734  062706  000006                  28$:    ADD     #6,SP           ;                       2963
```

001740 000207 RTS PC

; Routine Size: 497 words, Routine Base: $CODE$ + 5044
; Maximum stack depth per invocation: 13 words

```
;    3203  1      GLOBAL routine DR_ERR : novalue =
;    3204  1
;    3205  1      !+
;    3206  1      !       THIS ROUTINE IS DESIGNED TO PROCESS RETURN PACKETS THAT ORIGINATE AT
;    3207  1      !       THE "DRIVER" RATHER THAN THE DEVICE. DRIVER-ORIGINATED PACKETS INDICATE
;    3208  1      !       EITHER A FATAL DEVICE ERROR OR A COMMAND TIMEOUT. SINCE THIS ROUTINE IS
;    3209  1      !       ONLY CALLED DURING THE INITIALIZATION TEST, IT TREATS A COMMAND TIMEOUT
;    3210  1      !       AS AN INITIALIZATION ERROR.
;    3211  1      !
;    3212  1      !       IMPLICIT INPUTS:
;    3213  1      !               RP_ADDR   ADDRESS OF A RETPKT THAT ORIGINATED AT THE "DRIVER"
;    3214  1      !                       (I.E., CONNECTION ID = CID_DRIVER)
;    3215  1      !-
;    3216  1
;    3217  2          begin
;    3218  2
;    3219  2          local
;    3220  2              REASON : word initial (DU_TIME);        ! ASSUME COMMAND TIMEOUT
;    3221  2
;    3222  2          if .RP_ADDR [MESTYP] eql MT_FATAL           ! IF FATAL DEVICE ERROR
;    3223  2          then
;    3224  2
;    3225  2          DROP_CTLR (.CCTLR, .REASON);                ! DROP ALL UNITS ON CONTROLLER
;    3226  1          end;


                                        .SBTTL  DR.ERR INITIALIZATION TEST ROUTINES
000000  010146                 DR.ERR::MOV     R1,-(SP)                        ;                               3203
000002  012701  000012                  MOV     #12,R1                          ; #,REASON                      3217
000006  013700  000000G                 MOV     RP.ADDR,R0                      ;                               3222
000012  116000  000002                  MOVB    2(R0),R0
000016  042700  177417                  BIC     #177417,R0
000022  020027  000060                  CMP     R0,#60
000026  001006                         BNE     1$
000030  013746  000000G                 MOV     CCTLR,-(SP)                     ;
000034  010146                         MOV     R1,-(SP)                        ; REASON,#                      3225
000036  004737  000000G                 JSR     PC,DROP.CTLR
000042  022626                         CMP     (SP)+,(SP)+
000044  012601                 1$:     MOV     (SP)+,R1                        ;                               3203
000046  000207                         RTS     PC

; Routine Size:  20 words,      Routine Base:  $CODE$ + 7006
; Maximum stack depth per invocation:  4 words
```

```
;    3227  1     routine ACCESS : novalue =
;    3228  1
;    3229  1     !+
;    3230  1     !       THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT DISK
;    3231  1     !       CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
;    3232  1     !       SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE DISK, AND CHECKING
;    3233  1     !       THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
;    3234  1     !
;    3235  1     !       IMPLICIT INPUTS:
;    3236  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3237  1     !               CDISK   CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    3238  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    3239  1     !
;    3240  1
;    3241  2         begin
;    3242  2
;    3243  2     local
;    3244  2             RESULT : word initial (FAILURE),        ! GUILTY UNTIL PROVEN INNOCENT
;    3245  2             LBN : word,
;    3246  2             PASS : word initial (1);                ! LOOP PASS COUNT
;    3247  2
;    3248  2         ST_CODE = SB_CODE = 0;                      ! STATUS CODE AND SUB CODE
;    3249  2         LBN = (((.MAX_LBN [.L$LUN] - 1) ^  1) and %o'77777') - 1;
;    3250  2                                                     ! START WITH LAST LBN ON TOP SURFACE: [(X-1)/2]  1
;    3251  2
;    3252  2         do
;    3253  3             begin                                   ! LOOP STARTS HERE
;    3254  3             P_INDEX = GET_PKT (.CCTLR);             ! GET AN MSCP PACKET
;    3255  3             MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;   ! SET DISK ADDR (RD/RX DISK NUMBER)
;    3256  3             MSCP_PKT [.P_INDEX, OPCODE] = OP_ACC;   ! ACCESS OPCODE
;    3257  3             MSCP_PKT [.P_INDEX, BC_LO] = 512;       ! BYTE COUNT (1 BLOCK)
;    3258  3             MSCP_PKT [.P_INDEX, LBN_L] = .LBN;      ! LOGICAL BLOCK NUMBER
;    3259  3             MSCP_PKT [.P_INDEX, CMD_TYPE] = NON_SEQ_CMD,    ! NON-SEQUENTIAL COMMAND
;    3260  3
;    3261  3             if SEND (.P_INDEX) eql FAILURE          ! ATTEMPT TO SEND; IF CTLR NOT ONLINE
;    3262  3             then
;    3263  4                 begin
;    3264  4                 PUT_PKT (.P_INDEX);                 ! RETURN PACKET TO POOL
;    3265  4                 PASS = 2;                           ! NO MORE TRIES
;    3266  4                 end
;    3267  3             else
;    3268  4                 begin                               ! IF SEND WAS SUCCESSFUL
;    3269  4
;    3270  4                 do
;    3271  5                     begin
;    3272  5                     WAIT ();                        ! WAIT FOR RESPONSE
;    3273  5                     RP_INDX = OUT_IODQ ();          ! GET RETPKT (RESPONSE) INDEX
;    3274  5                     RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
;    3275  5
;    3276  5                     if .RP_ADDR [MESTYP] neq MT_SEQ ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    3277  5                     then
;    3278  5                         PUT_RETPKT (.RP_INDX);
;    3279  5
```

```
;    3280  5                    end
;    3281  4                until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    3282  5                      ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    3283  4                       (.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    3284  4
;    3285  4                if .RP_ADDR [CONID] eql CID_DRIVER  ! IF RETPKT CAME FROM "DRIVER"
;    3286  4                then
;    3287  4                    PASS = 2                        ! NO MORE TRIES
;    3288  4                else
;    3289  4
;    3290  5                    if .RP_ADDR [ENDCOD] neq (OP_ACC or OP_END)
;    3291  4                    then
;    3292  5                        begin
;    3293  5                        PRINTF (D8M29);             ! "RETPKT HAS BAD ENDCODE"
;    3294  5    !ZZZ                EMSCMD ();
;    3295  5                        end
;    3296  4                    else
;    3297  5                        begin                       ! RETPKT HAS CORRECT ENDCODE
;    3298  5                        ST_CODE = .RP_ADDR [STSCOD];! GET STATUS CODE FROM PACKET
;    3299  5                        SB_CODE = .RP_ADDR [SUBCOD];! GET SUB-CODE FROM PACKET
;    3300  5
;    3301  5                        if .ST_CODE eql ST_SUC      ! IF STATUS CODE INDICATES SUCCESS
;    3302  5                        then
;    3303  6                            begin
;    3304  6                            RESULT = SUCCESS;
;    3305  6                            PASS = 2;               ! NO NEED TO TRY AGAIN
;    3306  5                            end;
;    3307  5
;    3308  4                        end;                        ! IF RETPKT HAS CORRECT ENDCODE
;    3309  4
;    3310  4                    PUT_RETPKT (.RP_INDX);
;    3311  3                    end;                            ! IF SEND WAS SUCCESSFUL
;    3312  3
;    3313  3                LBN = .LBN + 1;                     ! ADVANCE TO FIRST LBN OF BOTTOM SURFACE
;    3314  3                PASS = .PASS + 1;                   ! SECOND PASS
;    3315  3                end                                 ! END OF PASS LOOP
;    3316  2            until .PASS gequ 3;
;    3317  2
;    3318  2            if .RESULT eql FAILURE
;    3319  2            then
;    3320  3                begin
;    3321  3                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    3322  3                CST_ADDR [.CUOFF, D_FATAL] = TRUE;  ! FATAL ERROR
;    3323  3                ERRDF (17, EGD_17, EMS_30);                 ! ACCESS FAILED
;    3324  3                DUR [.L$LUN] = DU_ACCESS;           ! SET REASON TO DROP UNIT
;    3325  3                DODU (.L$LUN);                      ! DROP UNIT
;    3326  2                end;                                ! IF ACCESS FAILED
;    3327  2
;    3328  1            end;                                    ! ROUTINE ACCESS


                                    .SBTTL  ACCESS INITIALIZATION TEST ROUTINES
000000  004137  000000G       ACCESS: JSR     R1,$SAVE4                               ;                                   3227
```

```
000004  005003                          CLR     R3                      ; RESULT                        3241
000006  012702  000001                  MOV     #1,R2                   ; *,PASS
000012  005037  000000G                 CLR     SB.CODE                 ;                               3248
000016  005037  000000G                 CLR     ST.CODE
000022  013700  000000G                 MOV     L$LUN,R0                ;                               3249
000026  006300                          ASL     R0
000030  016000  000054'                 MOV     MAX.LBN(R0),R0
000034  060200                          ADD     R2,R0
000036  006200                          ASR     R0
000040  010004                          MOV     R0,R4                   ; *,LBN
000042  042704  100000                  BIC     #100000,R4              ; *,LBN
000046  005304                          DEC     R4                      ; LBN
000050  013746  000000G         1$:     MOV     CCTLR,-(SP)             ;                               3254
000054  004737  000000G                 JSR     PC,GET.PKT
000060  010037  000000G                 MOV     R0,P.INDEX
000064  010016                          MOV     R0,(SP)                 ; P.INDEX,*                     3255
000066  012746  000106                  MOV     #106,-(SP)
000072  004737  000000G                 JSR     PC,BL$MUL
000076  013760  000000G  000016G        MOV     CDISK,MSCP.PKT+16(R0)
000104  112760  000020  000022G        MOVB    #20,MSCP.PKT+22(R0)     ;                               3256
000112  012760  001000  000026G        MOV     #1000,MSCP.PKT+26(R0)   ;                               3257
000120  010460  000046G                 MOV     R4,MSCP.PKT+46(R0)      ; LBN,*                         3258
000124  112760  000002  000004G        MOVB    #2,MSCP.PKT+4(R0)       ;                               3259
000132  013716  000000G                 MOV     P.INDEX,(SP)            ;                               3261
000136  004737  000000G                 JSR     PC,SEND
000142  005700                          TST     R0
000144  001007                          BNE     2$
000146  013716  000000G                 MOV     P.INDEX,(SP)            ;                               3264
000152  004737  000000G                 JSR     PC,PUT.PKT
000156  012702  000002                  MOV     #2,R2                   ; *,PASS                        3265
000162  000522                          BR      9$                      ;                               3261
000164  004737  000000G         2$:     JSR     PC,WAIT                 ;                               3272
000170  004737  000000G                 JSR     PC,OUT.IODQ             ;                               3273
000174  010037  000000G                 MOV     R0,RP.INDX
000200  010016                          MOV     R0,(SP)                 ; RP.INDX,*                     3274
000202  012746  000054                  MOV     #54,-(SP)
000206  004737  000000G                 JSR     PC,BL$MUL
000212  062700  000000G                 ADD     #RETPKT,R0
000216  010037  000000G                 MOV     R0,RP.ADDR
000222  132760  000360  000002         BITB    #360,2(R0)              ;                               3276
000230  001404                          BEQ     3$
000232  013716  000000G                 MOV     RP.INDX,(SP)            ;                               3278
000236  004737  000000G                 JSR     PC,PUT.RETPKT
000242  005726                  3$:     TST     (SP)+                   ;                               3271
000244  013701  000000G                 MOV     RP.ADDR,R1              ;                               3281
000250  005000                          CLR     R0
000252  126127  000003  000003         CMPB    3(R1),#3
000260  001002                          BNE     4$
000262  005200                          INC     R0
000264  000407                          BR      5$
000266  132761  000360  000002  4$:     BITB    #360,2(R1)              ;                               3282
000274  001333                          BNE     2$
000276  105761  000014                  TSTB    14(R1)                  ;                               3283
```

```
000302  100330                          BPL     2$
000304  006000                  5$:     ROR     R0                              ;                       3285
000306  103442                          BLO     7$                              ;                       3287
000310  126127  000014  000220          CMPB    14(R1),#220                     ;                       3290
000316  001410                          BEQ     6$
000320  012716  000000G                 MOV     #DBM29,(SP)                     ;                       3293
000324  012746  000001                  MOV     #1,-(SP)
000330  010600                          MOV     SP,R0                           ; SP,*
000332  104417                          TRAP    17
000334  005726                          TST     (SP)+                           ;                       3292
000336  000430                          BR      8$                              ;                       3290
000340  116137  000016  000000G  6$:     MOVB    16(R1),ST.CODE                  ;                       3298
000346  042737  177740  000000G          BIC     #177740,ST.CODE
000354  016100  000016                  MOV     16(R1),R0                       ;                       3299
000360  006200                          ASR     R0
000362  006200                          ASR     R0
000364  006200                          ASR     R0
000366  006200                          ASR     R0
000370  006200                          ASR     R0
000372  042700  174000                  BIC     #174000,R0
000376  010037  000000G                 MOV     R0,SB.CODE
000402  005737  000000G                 TST     ST.CODE                         ;                       3301
000406  001004                          BNE     8$
000410  012703  000001                  MOV     #1,R3                           ; *,RESULT              3304
000414  012702  000002          7$:     MOV     #2,R2                           ; *,PASS                3305
000420  013716  000000G          8$:     MOV     RF.INDX,(SP)                    ;                       3310
000424  004737  000000G                  JSR     PC,PUT.RETPKT
000430  005204                  9$:     INC     R4                              ; LBN                   3313
000432  005202                          INC     R2                              ; PASS                  3314
000434  022626                          CMP     (SP)+,(SP)+                     ;                       3253
000436  020227  000003                  CMP     R2,#3                           ; PASS.*                3316
000442  103602                          BLO     1$
000444  005703                          TST     R3                              ; RESULT                3318
000446  001025                          BNE     10$
000450  013700  000000G                 MOV     T.ADDR,R0                       ;                       3321
000454  105260  000050                  INCB    50(R0)
000460  013700  000000G                 MOV     CUOFF,R0                        ;                       3322
000464  006300                          ASL     R0
000466  063700  000000G                 ADD     CST.ADDR,R0
000472  052710  010000                  BIS     #10000,(R0)
000476  104455                          TRAP    55                              ;                       3323
000500  000021                          .WORD   21
000502  000000G                         .WORD   EGD.17
000504  000000G                         .WORD   EMS.30
000506  013700  000000G                 MOV     L$LUN,R0                        ;                       3324
000512  112760  000010  000000G         MOVB    #10,DUR(R0)
000520  104451                          TRAP    51                              ;                       3325
000522  000207                  10$:    RTS     PC                              ;                       3227

; Routine Size:  170 words,    Routine Base:  $CODE$ + 7056
; Maximum stack depth per invocation:  10 words
```

```
;    3329  1      %sbttl 'MULTI-DRIVE TEST ROUTINES'
;    3330  1
;    3331  1
;    3332  1      GLOBAL routine MULTI_DRIVE : novalue =
;    3333  1
;    3334  1
;    3335  1      !+
;    3336  1      !       THIS SUBTEST IS THE MOST SIGNIFICANT PART OF THE ENTIRE PROGRAM. THE
;    3337  1      !       MULTI-DRIVE TEST IS A HOST-CONTROLLED EXERCISER DESIGNED TO GIVE THE
;    3338  1      !       USER AN INDICATION OF HOW ONE OR SEVERAL RDRX DRIVES WOULD PERFORM IN
;    3339  1      !       AN OPERATING SYSTEM ENVIRONMENT.
;    3340  1      !
;    3341  1      !       THIS ROUTINE ACTS AS AN "EXECUTIVE" TO THE WHOLE PROCESS. AFTER
;    3342  1      !       INVOKING MD_INIT TO INITIALIZE MULTI-DRIVE TEST DATA, THIS ROUTINE
;    3343  1      !       ENTERS A LOOP WHICH ISSUES QIOs TO ALL ACTIVE CONTROLLERS AND PROCESSES
;    3344  1      !       ANY RESPONSES. IN ADDITION, ALL OUTSTANDING COMMANDS ARE TIMED IN
;    3345  1      !       DRV_TIMCHK WHICH IS INVOKED EVERY SECOND. NORMAL TERMINATION OF THIS
;    3346  1      !       LOOP OCCURS WHEN QIOs ARE NO LONGER BEING ISSUED, AND ALL OUTSTANDING
;    3347  1      !       QIOS HAVE COMPLETED.
;    3348  1      !-
;    3349  1
;    3350  1
;    3351  2          begin
;    3352  2
;    3353  2          local
;    3354  2              CUR_PRIORITY : word;
;    3355  2
;    3356  2          label
;    3357  2              SEND_COMMANDS;
;    3358  2
;    3359  2          MD_INIT ();                                          ! INIT MULTI-DRIVE TEST DATA
;    3360  2          INIT_OCCURED = TRUE;                                 !
;    3361  2
;    3362  2
;    3363  3          do  begin                                            ! START OF EXECUTIVE LOOP
;    3364  3
;    3365  3              incr CTLR from 0 to (MAX_CTLR - 1) do            ! FOR EACH CONTROLLER
;    3366  4                  begin
;    3367  4                  SET_CPAR (.CTLR);                            ! SET UP CURRENT CONTROLLER PARAMETERS
;    3368  4                  GETPRI (CUR_PRIORITY);                       !
;    3369  4      !ZZZ        SETPRI (PRI04);                              ! NO INTERRUPTS WHEN EXAMINING SA
;    3370  4                  SETPRI (.BRLEVEL);                           ! NO INTERRUPTS WHEN EXAMINING SA          ZZ
Z
;    3371  4                  ICTLR = .CCTLR;                              ! FAKE INTERRUPTING CONTROLLER'S NUMBER
;    3372  4                  ICST_ADDR = .CST_ADDR;                       ! FAKE INTERRUPTING CONTROLLER'S CST ADDR
;    3373  4                  IDCT_ADDR = .DCT_ADDR;                       ! FAKE INTERRUPTING CONTROLLER'S DCT ADDR
;    3374  4                  IRDRX_ADDR = .ICST_ADDR [IP_ADDR];           ! FAKE INTERRUPTING CONTROLLER'S ADDRESS
;    3375  4                  IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];  ! CONTENTS OF THE SA REGISTER
;    3376  4
;    3377  5                  if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)     ! IF SA SHOWS AN ERROR
;    3378  4                  then
;    3379  5                      begin
;    3380  5                      FATAL_ERROR ();                          ! DECLARE FATAL ERROR
;    3381  5                      SETPRI (.CUR_PRIORITY);                  ! LOWER PRIORITY
```

```
;    3382  5                    exitloop;                                  ! QUIT
;    3383  5                    end
;    3384  5
;    3385  4                else
;    3386  4                    SETPRI (.CUR_PRIORITY);                    ! IF NO ERROR, CONTINUE
;    3387  4
;    3388  4                if QIO_OK ()                                   ! IF O.K. TO ISSUE QIO(S) TO CONTROLLER
;    3389  4                then
;    3390  4                    SEND_COMMANDS:
;    3391  5                        begin
;    3392  5                        QIO_GEN ();                            ! GENERATE 1 OR 2 QIOs
;    3393  5
;    3394  5                        if (.MX1 geq 0) and                    ! IF SUCCESS ON FIRST QIO
;    3395  6                           (not .EOP_FLAG)
;    3396  5                        then
;    3397  5
;    3398  5                            if SEND (.MX1) eql SUCCESS         ! ATTEMPT TO SEND IT. IF SUCCESS
;    3399  5                            then
;    3400  6                            BEGIN                              !ZZZ
;    3401  6                                QIO [.CTLR] = .QIO [.CTLR] + 1;! INCR OUTSTANDING QIO COUNT ZZZ
;    3402  6                                RW_BALANCE = .RW_BALANCE + 1;  !INCR RD/WR RATIO COUNT
;    3403  6                            END                                !ZZZ
;    3404  6
;    3405  5                            else
;    3406  6                                begin
;    3407  6                                PUT_PKT (.MX1);                ! RETURN PACKET TO POOL
;    3408  6                                leave SEND_COMMANDS;
;    3409  5                                end;
;    3410  5
;    3411  5
;    3412  5                        if (.MX2 geq 0) and                    ! IF SUCCESS ON SECOND QIO
;    3413  6                           (not .EOP_FLAG)
;    3414  5                        then
;    3415  6                            begin
;    3416  6
;    3417  6                            do
;    3418  6                                BREAK                          ! WAIT TILL 1 MORE SLOT AVAILABLE IN CRING
;    3419  6                            until (.DCT_ADDR [CRING_CNT] lssu CRING_LEN);  !
;    3420  6
;    3421  6                            if SEND (.MX2) eql SUCCESS         ! ATTEMPT TO SEND IT.
;    3422  6                            then
;    3423  7                            BEGIN                              !ZZZ
;    3424  7                                QIO [.CTLR] = .QIO [.CTLR] + 1;! IF SUCCESS, INCR OUTSTANDING QIO COUNT
;    3425  7                                RW_BALANCE = 0;                !INDICATE RD/WT PAIR WAS LAST ISSUED  ZZZ
;    3426  7                            END                                !ZZZ
;    3427  7
;    3428  6                            else
;    3429  7                                begin
;    3430  7                                PRINTF (D8M121, .CRN_HIGH, .CRN_LOW);
;    3431  7                                COMPARE_DATA = FALSE;          ! NO SENSE IN COMPARING WRITE DATA
;    3432  7                                PUT_PKT (.MX2);                ! RETURN PACKET TO POOL
;    3433  6                                end;
;    3434  6
```

```
;    3435  5                          end;
;    3436  5
;    3437  4                      end;
;    3438  3                  end;                                    ! O.K. TO ISSUE QIO(S)
;    3439  3                                                          ! CONTROLLER LOOP
;    3440  3
;    3441  3              BREAK;
;    3442  3              PROC_RETPKT ();                             ! LET SUPERVISOR CATCH USER REQUESTS
;    3443  3                                                          ! PROCESS ANY RETURN PACKETS
;    3444  3              end
;    3445  3          until ((not QIO_OUT ()) or                      ! EXECUTIVE PROCESSING LOOP
;    3446  4                 ((.DCT_ADDR [CRING_CNT] eql 0) and
;    3447  2                  (.EOP_FLAG)));
;    3448  2
;    3449  2
;    3450  2          DCT_ADDR [IG_INT] = TRUE;                       ! NO FURTHER INTERRUPTS ON THIS CONTROLLER
;    3451  2
;    3452  2
;    3453  1          end;                                            ! EXERCISER


                                .SBTTL  MULTI.DRIVE MULTI-DRIVE TEST ROUTINES
C00000  004137  000000G         MULTI.DRIVE::
                        JSR     R1,$SAVE3                       ;                               3332
000004  005746                 TST     -(SP)
000006  004737  000000V         JSR     PC,MD.INIT              ;                               3359
000012  112737  000001  000000G MOVB    #1,INIT.OCCURED         ;                               3360
000020  005001                 1$:     CLR     R1              ; CTLR                          3365
000022  010146                 2$:     MOV     R1,-(SP)        ; CTLR,*                        3367
000024  004737  000000G         JSR     PC,SET.CPAR
000030  104440                 TRAP    40                      ;                               3368
000032  010003                 MOV     R0,R3                   ; *,CUR.PRIORITY
000034  013700  000000G         MOV     BRLEVEL,R0              ;                               3370
000040  104441                 TRAP    41
000042  013737  000000G 000104' MOV     CCTLR,ICTLR             ;                               3371
000050  013737  000000G 000076' MOV     CST.ADDR,ICST.ADDR      ;                               3372
000056  013737  000000G 000100' MOV     DCT.ADDR,IDCT.ADDR      ;                               3373
000064  017737  000076' 000000G MOV     @ICST.ADDR,IRDRX.ADDR   ;                               3374
000072  013700  000100'         MOV     IDCT.ADDR,R0            ;                               3375
000076  013702  000000G         MOV     IRDRX.ADDR,R2
000102  016266  000002  000002  MOV     2(R2),2(SP)             ; *,RC.REG
000110  016660  000002  000002  MOV     2(SP),2(R0)             ; RC.REG,*
000116  016600  000002          MOV     2(SP),R0                ;                               3377
000122  042700  077777          BIC     #77777,R0
000126  020027  100000          CMP     R0,#-10000^
000132  001006                  BNE     3$
000134  004737  000000V         JSR     PC,FATAL.ERROR          ;                               3380
000140  010300                  MOV     R3,R0                   ; CUR.PRIORITY,*                3381
000142  104441                  TRAP    41
000144  005726                  TST     (SP)+                   ;                               3379
000146  000515                  BR      9$
000150  010300                  3$:     MOV     R3,R0           ; CUR.PRIORITY,*                3386
000152  104441                  TRAP    41
```

```
000154  004737  000000V                        JSR     PC,QIO.OK                       ;                               3388
000160  006000                                  ROR     R0
000162  103103                                  BCC     8$
000164  004737  000000V                        JSR     PC,QIO.GEN                      ;                               3392
000170  013700  000110'                        MOV     MX1,R0                          ;                               3394
000174  002424                                  BLT     5$
000176  132737  000001  000000G                BITB    #1,EOP.FLAG                     ;                               3395
000204  001020                                  BNE     5$
000206  010016                                  MOV     R0,(SP)                         ;                               3398
000210  004737  000000G                        JSR     PC,SEND
000214  020027  000001                        CMP     R0,#1
000220  001005                                  BNE     4$
000222  105261  000000G                        INCB    QIO(R1)                         ; *(CTLR)                       3401
000226  005237  000106'                        INC     RW.BALANCE                      ;                               3402
000232  000405                                  BR      5$                              ;                               3398
000234  013716  000110'              4$:       MOV     MX1,(SP)                        ;                               3407
000240  004737  000000G                        JSR     PC,PUT.PKT
000244  000452                                  BR      8$                              ;                               3406
000246  005737  000112'              5$:       TST     MX2                             ;                               3412
000252  002447                                  BLT     8$
000254  132737  000001  000000G                BITB    #1,EOP.FLAG                     ;                               3413
000262  001043                                  BNE     8$
000264  104422                      6$:         TRAP    22                              ;                               3417
000266  127727  000000G 000004                CMPB    @DCT.ADDR,#4                    ;                               3419
000274  103373                                  BHIS    6$
000276  013716  000112'                        MOV     MX2,(SP)                        ;                               3421
000302  004737  000000G                        JSR     PC,SEND
000306  020027  000001                        CMP     R0,#1
000312  001005                                  BNE     7$
000314  105261  000000G                        INCB    QIO(R1)                         ; *(CTLR)                       3424
000320  005037  000106'                        CLR     RW.BALANCE                      ;                               3425
000324  000422                                  BR      8$                              ;                               3421
000326  013716  000000G              7$:       MOV     CRN.LOW,(SP)                    ;                               3430
000332  013746  000000G                        MOV     CRN.HIGH, (SP)
000336  012746  000000G                        MOV     #DBM121,-(SP)
000342  012746  000003                        MOV     #3,-(SP)
000346  010600                                  MOV     SP,R0                           ; SP,*
000350  104417                                  TRAP    17
000352  105037  001262'                        CLRB    COMPARE.DATA                    ;                               3431
000356  013716  000112'                        MOV     MX2,(SP)                        ;                               3432
000362  004737  000000G                        JSR     PC,PUT.PKT
000366  062706  000006                        ADD     #6,SP                           ;                               3429
000372  005726                      8$:         TST     (SP)+                           ;                               3366
000374  005201                                  INC     R1                              ; CTLR                          3365
000376  000243                                  .WORD CLV!CLC

000400  003610                                  BLE     2$
000402  104422                      9$:         TRAP    22                              ;                               3438
000404  004737  000000V                        JSR     PC,PROC.RETPKT                  ;                               3442
000410  004737  000000V                        JSR     PC,QIO.OUT                      ;                               3445
000414  006000                                  ROR     R0
000416  103011                                  BCC     12$
000420  105777  000000G                        TSTB    @DCT.ADDR                       ;                               3446
```

```
000424  001402                                  BEQ     11$
000426  000137  007622                  10$:     JMP     1$
000432  132737  000001  000000G         11$:     BITB    #1,EOP.FLAG             ;                       3447
000440  001772                                  BEQ     10$
000442  052777  040000  000000G         12$:     BIS     #40000,@DCT.ADDR        ;                       3450
000450  005726                                  TST     (SP)+                   ;                       3332
000452  000207                                  RTS     PC

; Routine Size:  150 words,    Routine Base:  $CODE$ + 7602
; Maximum stack depth per invocation:  11 words


;   3454  1
```

```
;    3455  1     GLOBAL routine MD_INIT : novalue =
;    3456  1
;    3457  1     !+
;    3458  1     !       THIS ROUTINE IS CALLED BY ROUTINE MULTI_DRIVE TO INITIALIZE DATA ITEMS
;    3459  1     !       USED BY THE MULTI-DRIVE TEST.
;    3460  1     !-
;    3461  1
;    3462  2         begin
;    3463  2
;    3464  2     !!ZZZ    local
;    3465  2     !!ZZZ    AVG_XFER_SIZE : word,                                  ! SIZE (BYTES) OF AN AVERAGE I/O XFER
;    3466  2     !!ZZZ    QUICK_PASS_CNT : word;                                 ! AVG NO. OF I/O OPERATIONS IN A QUICK PASS
;    3467  2
;    3468  2         if not .INIT_OCCURED                                       ! IF THIS IS A START
;    3469  2         then
;    3470  2             INIT_IO_BUFF ();                                       ! PARTITION FREE MEMORY INTO I/O BUFFERS
;    3471  2
;    3472  2         if (.ENTRY_REASON neq CONT) and                            ! IF START, RESTART, OR PWR FAIL
;    3473  3            (.ENTRY_REASON neq NEW_PASS)
;    3474  2         then
;    3475  2
;    3476  2             incr CTLR from 0 to (MAX_CTLR - 1) do
;    3477  3                 begin
;    3478  3                 SET_CPAR (.CTLR);
;    3479  3
;    3480  4                 INCR DISK FROM (0 * OF_UN) TO (3 * UNIT_SIZE                   !ZZZ
;    3481  3                      * OF_UN) BY UNIT_SIZE DO                          !ZZZ
;    3482  4                      BEGIN                                             !ZZZ
;    3483  4                      SET_UPAR (.DISK);                                 !ZZZ
;    3484  4                      DPST [.L$LUN] = DP_CNT;        !INIT DATA PTRN SEQ TABLEZZZ
;    3485  3                      END;                                              !ZZZ
;    3486  3
;    3487  2                 END;                                                   !ZZZ
;    3488  2             INCR COUNT FROM 0 TO (QIO_PER_CTLR * MAX_CTLR - 1) DO   !INIT   !ZZZ
;    3489  2                 BUFF_OWN [.COUNT] = -1;                 !I/O BUFF ALLOC TABLE   !ZZZ
;    3490  1             END;                                        !END MD_INIT            !ZZZ


                                        .SBTTL  MD.INIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                 MD.INIT::
                                        JSR     R1,$SAVE2                       ;                       3455
000004  132737  000001  000000G                BITB    #1,INIT.OCCURED         ;                       3468
000012  001002                          BNE     1$
000014  004737  000000V                 JSR     PC,INIT.IO.BUFF                 ;                       3470
000020  123727  000000G 000003  1$:     CMPB    ENTRY.REASON,#3                 ;                       3472
000026  001433                          BEQ     4$
000030  123727  000000G 000005          CMPB    ENTRY.REASON,#5                 ;                       3473
000036  001427                          BEQ     4$
000040  005002                          CLR     R2                              ; CTLR                  3476
000042  010246                  2$:     MOV     R2,-(SP)                        ; CTLR,*                3478
000044  004737  000000G                 JSR     PC,SET.CPAR
000050  012701  000003                  MOV     #3,R1                           ; *,DISK                3480
000054  010116                  3$:     MOV     R1,(SP)                         ; DISK,*                3483
```

```
ZRQAM3          RD/RX EXERCISER                          4-Apr 1985 13:23:31    VAX-11 Bliss-16 V4.1-582          SEQ 0332
V02.2           MULTI DRIVE TEST ROUTINES                2 Apr 1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;19  Page  77
                                                                                                                    (17)

000056  004737  000000G                          JSR     PC,SET.UPAR
000062  013700  000000G                          MOV     L$LUN,R0                                                3484
000066  112760  000025  000050'                  MOVB    #25,DPST(R0)
000074  062701  000012                           ADD     #12,R1                  ; *,DISK                        3480
000100  020127  000041                           CMP     R1,#41                  ; DISK,*
000104  003763                                   BLE     3$                                                      3477
000106  005726                                   TST     (SP)+                   ;                               3476
000110  005202                                   INC     R2                      ; CTLR
000112  000243                                   .WORD CLV!CLC

000114  003752                                   BLE     2$
000116  005000                           4$:     CLR     R0                      ; COUNT                         3488
000120  112760  000377  000000G         5$:     MOVB    #377,BUFF.OWN(R0)       ; *,*(COUNT)                    3489
000126  005200                                   INC     R0                      ; COUNT                         3488
000130  020027  000007                           CMP     R0,#7                   ; COUNT,*
000134  003771                                   BLE     5$
000136  000207                                   RTS     PC                      ;                               3455

; Routine Size:  48 words,      Routine Base:  $CODE$ + 10256
; Maximum stack depth per invocation:  5 words


;   3491  1
```

```
;    3492  1     GLOBAL routine INIT_IO_BUFF : novalue =
;    3493  1
;    3494  1     !+
;    3495  1     !       THIS ROUTINE IS CALLED BY MD_INIT WHEN THE MULTI-DRIVE TEST IS FIRST
;    3496  1     !       STARTED. IT IS RESPONSIBLE FOR PARTITIONING FREE MEMORY INTO A
;    3497  1     !       COLLECTION OF I/O BUFFERS. THE SIZE OF EACH I/O BUFFER IS DETERMINED
;    3498  1     !       BY A NUMBER OF FACTORS, INCLUDING THE NUMBER OF UNITS, THE NUMBER OF
;    3499  1     !       CONTROLLERS, AND THE SIZE OF FREE MEMORY.
;    3500  1     !
;    3501  1     !       ONCE THE BUFFER SIZE IS DETERMINED, THE NUMBER OF I/O BUFFERS IS
;    3502  1     !       CALCULATED. FINALLY, THE BUFFER ADDRESS (BUFF_ADDR) TABLE IS LOADED
;    3503  1     !       WITH FIXED BUFFER DESCRIPTORS THAT ARE USED IN THE ALLOCATION AND
;    3504  1     !       DEALLOCATION PROCESS.
;    3505  1     !
;    3506  1     !       IMPLICIT INPUTS:
;    3507  1     !               CTLR_CNT    THE NUMBER OF CONTROLLERS CONFIGURED
;    3508  1     !               L$UNIT - THE NUMBER OF UNITS AVAILABLE FOR TESTING
;    3509  1     !               FREE_MEM_ADDR - START OF FREE MEMORY
;    3510  1     !-
;    3511  1
;    3512  2         begin
;    3513  2         BUFF_ADDR [0] = (.FREE_MEM_ADDR + 2 + 1) and %o'177776';                    ! START OF READ/WRITE BUFFERS
;    3514  2
;    3515  2         while (.BUFF_ADDR [0] and %o'37') neq 0 do                                  ! FORCE FIRST I/O BUFFER TO START
;    3516  2             BUFF_ADDR [0] = .BUFF_ADDR [0] + 2;                                     ! ON EVEN BOUNDARY
;    3517  2
;    3518  2         BYTS_PER_QIO = ((.DRS_START - .BUFF_ADDR [0]) / (QIO_PER_CTLR * MAX_CTLR)) and %o'177740';
;    3519  2                                                                                     ! MAX TRANSFER SIZE
;    3520  2
;    3521  2         if .BYTS_PER_QIO gtru MAX_XFER_SIZE
;    3522  2         then
;    3523  2             BYTS_PER_QIO = MAX_XFER_SIZE;                                           ! ADJUST TRANSFER SIZE LOWER
;    3524  2
;    3525  2         if .BYTS_PER_QIO lssu 32
;    3526  2         then
;    3527  3             begin
;    3528  3             ERRSF (2, EGS_02, 0);                                                   ! ERROR IF NOT ENOUGH MEMORY
;    3529  3             DOCLN;
;    3530  2             end;
;    3531  2
;    3532  2         if (QIO_PER_CTLR * MAX_CTLR) gtru 1
;    3533  2         then
;    3534  2
;    3535  2             incr index from 1 to (QIO_PER_CTLR * MAX_CTLR - 1) do                   ! INIT REMAINING TABLE ENTRIES
;    3536  2                 BUFF_ADDR [.index] = .BUFF_ADDR [.index - 1] + .BYTS_PER_QIO;       ! FIXED BUFFER ADDRESS
;    3537  2
;    3538  1         end;                                                                        ! ROUTINE INIT_IO_BUFF


                                .SBTTL  INIT.IO.BUFF MULTI DRIVE TEST ROUTINES
000000  004137  000000G  INIT.IO.BUFF::
                        JSR     R1,$SAVE3                       ;                           3492
000004  013700  000000G  MOV     FREE.MEM.ADDR,R0                ;                           3513
```

```
000010  062700  000003                        ADD     #3,R0
000014  010037  000000G                       MOV     R0,BUFF.ADDR
000020  042737  000001  000000G               BIC     #1,BUFF.ADDR
000026  032737  000037  000000G       1$:     BIT     #37,BUFF.ADDR           ;                       3515
000034  001404                               BEQ     2$
000036  062737  000002  000000G               ADD     #2,BUFF.ADDR            ;                       3516
000044  000770                               BR      1$                      ;                       3515
000046  013746  001252'              2$:     MOV     DRS.START,-(SP)         ;                       3518
000052  163716  000000G                       SUB     BUFF.ADDR,(SP)
000056  012746  000010                        MOV     #10,-(SP)
000062  004737  000000G                       JSR     PC,BL$DIV
000066  010037  000000G                       MOV     R0,BYTS.PER.QIO
000072  042737  000037  000000G               BIC     #37,BYTS.PER.QIO
000100  023727  000000G 001400               CMP     BYTS.PER.QIO,#1400      ;                       3521
000106  101403                               BLOS    3$
000110  012737  001400  000000G               MOV     #1400,BYTS.PER.QIO      ;                       3523
000116  023727  000000G 000040       3$:     CMP     BYTS.PER.QIO,#40        ;                       3525
000124  103005                               BHIS    4$
000126  104454                               TRAP    54                      ;                       3528
000130  000002                               .WORD   2
000132  000000G                              .WORD   EGS.02
000134  000000                               .WORD   0
000136  104444                               TRAP    44
000140  012702  000001              4$:     MOV     #1,R2                   ; *,INDEX               3532
000144  010200                      5$:     MOV     R2,R0                   ; INDFX,*               3536
000146  006300                              ASL     R0
000150  010201                              MOV     R2,R1                   ; INDEX,*
000152  006301                              ASL     R1
000154  016103  177776G                      MOV     BUFF.ADDR-2(R1),R3
000160  063703  000000G                       ADD     BYTS.PER.QIO,R3
000164  010360  000000G                       MOV     R3,BUFF.ADDR(R0)
000170  005202                              INC     R2                      ; INDEX                 3532
000172  020227  000007                      CMP     R2,#7                   ; INDEX,*
000176  003762                              BLE     5$
000200  022626                              CMP     (SP)+,(SP)+             ;                       3512
000202  000207                              RTS     PC                      ;                       3492

; Routine Size:  66 words,      Routine Base:  $CODE$ + 10416
; Maximum stack depth per invocation:  8 words
```

```
;  3539  1     GLOBAL routine QIO_OK =
;  3540  1
;  3541  1
;  3542  1     !+
;  3543  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE "EXECUTIVE" IN ORDER TO
;  3544  1     !       DETERMINE WHETHER OR NOT A QIO REQUEST (OR QIO PAIR) SHOULD BE
;  3545  1     !       GENERATED TO THE CURRENT CONTROLLER. A VALUE OF "TRUE" IS RETURNED IF
;  3546  1     !       THE CONTROLLER MEETS 3 REQUIREMENTS:
;  3547  1     !
;  3548  1     !               A. THE CONTROLLER IS ONLINE;
;  3549  1     !               B. THE NUMBER OF OUTSTANDING QIOs IS AT LEAST 2 LESS THAN THE
;  3550  1     !                       MAXIMUM ALLOWED FOR ANY ONE CONTROLLER;
;  3551  1     !               C. THERE IS AT LEAST ONE DISK ONLINE TO THE CONTROLLER.
;  3552  1     !
;  3553  1     !       IF ANY OF THESE TEST FAIL, THEN A VALUE OF "FALSE" IS RETURNED.
;  3554  1     !
;  3555  1     !       IMPLICIT INPUTS:
;  3556  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  3557  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  3558  1     !-
;  3559  1
;  3560  1
;  3561  1         if (.CST_ADDR [STATE] eql ONLINE) and                   ! IF CONTROLLER IS ONLINE
;  3562  1            (not .EOP_FLAG) and
;  3563  1            ((.QIO [.CCTLR] + 2) lequ QIO_PER_CTLR) and          ! IF OUTSTANDING QIO COUNT IS O.K.
;  3564  2            (.CST_ADDR [U_CNT] neq 0)                            ! IF THERE IS VALID UNIT
;  3565  2
;  3566  1         then
;  3567  1             return TRUE                                         ! "TRUE" EXIT POINT
;  3568  1
;  3569  1         else
;  3570  1             return FALSE;                                       ! "FALSE" EXIT POINT


                                        .SBTTL  QIO.OK MULTI-DRIVE TEST ROUTINES
000000  013700  000000G                QIO.OK::MOV     CST.ADDR,R0                     ;3561
000004  005760  000002                        TST     2(R0)
000010  100027                               BPL     1$
000012  132737  000001  000000G               BITB    #1,EOP.FLAG                     ;3562
000020  001023                               BNE     1$
000022  013700  000000G                       MOV     CCTLR,R0                        ;3563
000026  116000  000000G                       MOVB    QIO(R0),R0
000032  042700  177400                       BIC     #177400,R0
000036  062700  000002                       ADD     #2,R0
000042  020027  000010                       CMP     R0,#10
000046  101010                               BHI     1$
000050  013700  000000G                       MOV     CST.ADDR,R0                     ;3564
000054  105760  000005                       TSTB    5(R0)
000060  001403                               BEQ     1$
000062  012700  000001                       MOV     #1,R0                           ;3570
000066  000207                               RTS     PC
000070  005000                        1$:    CLR     R0
000072  000207                               RTS     PC                              ;3539
```

```
; Routine Size:  30 words,    Routine Base:  $CODE$ + 10622
; Maximum stack depth per invocation:  0 words

;   3571  1
```

```
;    3572  1      GLOBAL routine QIO_OUT =
;    3573  1
;    3574  1      !+
;    3575  1      !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR DETERMINING THE
;    3576  1      !       END OF THE MULTI-DRIVE TEST. ITS PURPOSE IS TO EXAMINE THE QIO VECTOR
;    3577  1      !       FOR ANY OUTSTANDING QIOs ON ANY CONTROLLER. A VALUE OF "TRUE" IS
;    3578  1      !       RETURNED IF THERE IS AT LEAST ONE QIO OUTSTANDING ON ANY CONTROLLER.
;    3579  1      !       OTHERWISE, "FALSE" IS RETURNED INDICATING NO OUTSTANDING QIOs.
;    3580  1      !-
;    3581  1
;    3582  2          begin
;    3583  2
;    3584  2          incr CTLR from 0 to (MAX_CTLR - 1) do
;    3585  3              begin
;    3586  3              SET_CPAR (.CTLR);                       ! SET UP CURRENT CONTROLLER PARAMETERS
;    3587  3
;    3588  3              if .CST_ADDR [STATE] eql ONLINE         ! IF CONTROLLER IS ONLINE
;    3589  3              then
;    3590  3                  return TRUE;
;    3591  3
;    3592  2              end;
;    3593  2
;    3594  2          return FALSE;                               ! EXIT - NO CONTROLLERS ONLINE
;    3595  1          end;


                                        .SBTTL  QIO.OUT MULTI-DRIVE TEST ROUTINES
000000  010146                    QIO.OUT::
                                        MOV     R1,-(SP)                        ;                       3572
000002  005001                          CLR     R1                              ; CTLR                  3584
000004  010146                  1$:     MOV     R1,-(SP)                        ; CTLR,*                3586
000006  004737  000000G                 JSR     PC,SET.CPAR
000012  013700  000000G                 MOV     CST.ADDR,R0                     ;                       3588
000016  005760  000002                  TST     2(R0)
000022  100004                          BPL     2$
000024  005726                          TST     (SP)+                           ;                       3590
000026  012700  000001                  MOV     #1,R0
000032  000405                          BR      3$
000034  005726                  2$:     TST     (SP)+                           ;                       3585
000036  005201                          INC     R1                              ; CTLR                  3584
000040  000243                          .WORD CLV!CLC

000042  003760                          BLE     1$
000044  005000                          CLR     R0                              ;                       3582
000046  012601                  3$:     MOV     (SP)+,R1                        ;                       3572
000050  000207                          RTS     PC

; Routine Size:  21 words,      Routine Base:  $CODE$ + 10716
; Maximum stack depth per invocation:  3 words
```

```
;    3596  1      GLOBAL routine QIO_GEN : novalue =
;    3597  1
;    3598  1      !+
;    3599  1      !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR AN ONLINE
;    3600  1      !       CONTROLLER ELIGIBLE TO RECEIVE I/O TRANSFER REQUESTS. IT IS
;    3601  1      !       RESPONSIBLE FOR SECURING ONE OR TWO MSCP PACKETS AND LOADING THEM
;    3602  1      !       WITH VARIOUS PARAMETERS COMPRISING THE I/O REQUEST. THE I/O REQUEST
;    3603  1      !       GENERATED HERE IS DESTINED TO A PARTICULAR UNIT SELECTED AT RANDOM FROM
;    3604  1      !       THOSE CONFIGURED UNDER THE CURRENT CONTROLLER.
;    3605  1      !
;    3606  1      !       EACH FIELD OF THE PACKET(S) IS LOADED WITHIN INDIVIDUAL ROUTINES
;    3607  1      !       (QIO_FUNC, QIO_LBN, QIO_SIZE, ETC.). MOST OF THE VALUES SELECTED FOR
;    3608  1      !       EACH FIELD ARE BASED ON A SET OF RANDOM NUMBER GENERATED AT THE START.
;    3609  1      !
;    3610  1      !       UNDER NORMAL CIRCUMSTANCES, ONLY ONE I/O REQUEST IS GENERATED. HOWEVER,
;    3611  1      !       IF THIS I/O REQUEST IS A "WRITE", AND IF THE OPERATOR SELECTED THE
;    3612  1      !       OPTION FOR HOST WRITE-COMPARES, THEN A SECOND "READ" REQUEST WILL BE
;    3613  1      !       GENERATED WITH THE SAME LBN AND BYTE COUNT.
;    3614  1      !
;    3615  1      !       AFTER THE PACKET(S) HAVE BEEN LOADED, THIS ROUTINE REGAINS CONTROL
;    3616  1      !       AND ATTEMPTS TO GET ONE OR TWO I/O BUFFERS FOR THE ACTUAL DATA
;    3617  1      !       TRANSFERS. THE SUCCESS / FAIL STATUS OF THIS ENTIRE OPERATION IS
;    3618  1      !       PASSED BACK TO THE CALLER THROUGH THE GLOBALS "MX1" AND "MX2"; THEY
;    3619  1      !       CONTAIN VALID MSCP PACKET INDECES, OR -1.
;    3620  1      !
;    3621  1      !       IMPLICIT INPUTS:
;    3622  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    3623  1      !-
;    3624  1
;    3625  2          begin
;    3626  2          MX2 = -1;                                   ! ASSUME FAILURE IN SECURING 2ND PACKET
;    3627  2
;    3628  2          if (MX1 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 1ST PACKET. IF FAILURE
;    3629  2          then
;    3630  2              return;                                 ! NO POINT IN CONTINUING
;    3631  2
;    3632  2          if (MX2 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 2ND PACKET. IF FAILURE
;    3633  2          then
;    3634  3              begin
;    3635  3              PUT_PKT (.MX1);                         ! RETURN 1ST PACKET TO POOL
;    3636  3              MX1 = -1;                               ! INDICATE FAILURE
;    3637  3              return;                                 ! DONE
;    3638  2              end;
;    3639  2
;    3640  2          MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2);     ! CALCULATE STARTING ADDRESSES
;    3641  2          MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2);     !     OF BOTH PACKETS
;    3642  2          GET_RANDOM ();                              ! GENERATE A SET OF RANDOM NUMBERS
;    3643  2          QIO_UNIT ();                                ! LOAD RANDOM UNIT NUMBER INTO PACKETS
;    3644  2
;    3645  2          if .EOP_FLAG                                ! RETURN IF NO UNIT ONLINE
;    3646  2          then
;    3647  2              return;
;    3648  2
```

```
;    3649  2          QIO_FUNC ();                        ! LOAD RANDOM FUNCTION CODE (OPCODE)
;    3650  2          QIO_LBN ();                         ! LOAD LBN (RANDOM OR SEQUENTIAL)
;    3651  2          QIO_SIZE ();                        ! LOAD RANDOM BYTE COUNT
;    3652  2          GET_IO_BUFF (MAD1 [BUF 0]);         ! TRY TO GET AN I/O BUFFER
;    3653  2
;    3654  2          if .MX2 geq 0                       ! IF TWO QIOs ARE TO BE ISSUED
;    3655  2          then
;    3656  3              begin
;    3657  3              GET_IO_BUFF (MAD2 [BUF 0]);     ! TRY TO GET 2ND I/O BUFFER
;    3658  3
;    3659  3              if .MAD2 [BUF_0] eqla 0         ! IF 2ND BUFFER ALLOCATION FAILED
;    3660  3              then
;    3661  4                  begin
;    3662  4
;    3663  4                  if .MAD1 [BUF 0] neqa 0     ! IF 1ST I/O BUFFER WAS ALLOCATED
;    3664  4                  then
;    3665  5                      begin
;    3666  5                      PUT_IO_BUFF (MAD1 [BUF_0]);     ! RETURN 1ST I/O BUFFER TO POOL
;    3667  5                      MAD1 [BUF_0] = 0;               ! MARK IT AS FAILED
;    3668  4                      end;
;    3669  4
;    3670  4                  PUT_PKT (.MX2);             ! RETURN 2ND PACKET TO POOL
;    3671  4                  MX2 = -1;                   ! INDICATE FAILURE
;    3672  3                  end;                        ! IF 2ND I/O BUFFER ALLOCATION FAILED
;    3673  3
;    3674  2              end;                            ! IF TWO QIOs ARE TO BE ISSUED
;    3675  2
;    3676  2          if .MAD1 [BUF_0] eqla 0             ! IF 1ST I/O BUFFER ALLOCATION FAILED
;    3677  2          then
;    3678  3              begin
;    3679  3              PUT_PKT (.MX1);                 ! RETURN 1ST PACKET TO POOL
;    3680  3              MX1 = -1;                       ! INDICATE FAILURE
;    3681  3              end
;    3682  2          else
;    3683  2
;    3684  2              if .MAD1 [OPCODE] eql OP_WRT    ! OTHERWISE, IF 1ST OPCODE IS A WRITE  (ALL IS O.K.)
;    3685  2              then
;    3686  2                  FILL_BUFF ();               ! FILL 1ST I/O BUFFER WITH APPROPRIATE DATA PATTERN
;    3687  2
;    3688  1          end;                                ! ROUTINE QIO_GEN


                                        .SBTTL  QIO.GEN MULTI-DRIVE TEST ROUTINES
000000  012737  177777  000112'  QIO.GEN::
                                        MOV     #-1,MX2                         ;                       3626
000006  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3628
000012  004737  000000G                 JSR     PC,GET.PKT
000016  010037  000110'                 MOV     R0,MX1
000022  005726                          TST     (SP)+
000024  005700                          TST     R0                              ; MX1
000026  002563                          BLT     6$                              ;                       3630
000030  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3632
000034  004737  000000G                 JSR     PC,GET.PKT
```

```
000040  010037  000112'                        MOV     R0,MX2
000044  005726                                 TST     (SP)+
000046  005700                                 TST     R0                      ; MX2
000050  002011                                 BGE     1$
000052  013746  000110'                        MOV     MX1,-(SP)               ;                       3635
000056  004737  000000G                        JSR     PC,PUT.PKT
000062  012737  177777  000110'                MOV     #-1,MX1                 ;                       3636
000070  005726                                 TST     (SP)+                   ;                       3637
000072  000207                                 RTS     PC                      ;                       3634
000074  013746  000110'             1$:        MOV     MX1,-(SP)               ;                       3640
000100  012746  000106                         MOV     #106,-(SP)
000104  004737  000000G                        JSR     PC,BL$MUL
000110  062700  000000G                        ADD     #MSCP.PKT,R0
000114  010037  000114'                        MOV     R0,MAD1
000120  013716  000112'                        MOV     MX2,(SP)                ;                       3641
000124  012746  000106                         MOV     #106,-(SP)
000130  004737  000000G                        JSR     PC,BL$MUL
000134  062700  000000G                        ADD     #MSCP.PKT,R0
000140  010037  000116'                        MOV     R0,MAD2
000144  004737  000000V                        JSR     PC,GET.RANDOM           ;                       3642
000150  004737  000000V                        JSR     PC,QIO.UNIT             ;                       3643
000154  132737  000001  000000G               BITB    #1,EOP.FLAG             ;                       3645
000162  001103                                 BNE     5$                      ;                       3596
000164  004737  000000V                        JSR     PC,QIO.FUNC             ;                       3649
000170  004737  000000V                        JSR     PC,QIO.LBN              ;                       3650
000174  004737  000000V                        JSR     PC,QIO.SIZE             ;                       3651
000200  013716  000114'                        MOV     MAD1,(SP)               ;                       3652
000204  062716  000032                         ADD     #32,(SP)
000210  004737  000000G                        JSR     PC,GET.IO.BUFF
000214  005737  000112'                        TST     MX2                     ;                       3654
000220  002437                                 BLT     3$
000222  013716  000116'                        MOV     MAD2,(SP)               ;                       3657
000226  062716  000032                         ADD     #32,(SP)
000232  004737  000000G                        JSR     PC,GET.IO.BUFF
000236  013700  000116'                        MOV     MAD2,R0                 ;                       3659
000242  005760  000032                         TST     32(R0)
000246  001024                                 BNE     3$
000250  013700  000114'                        MOV     MAD1,R0                 ;                       3663
000254  062700  000032                         ADD     #32,R0
000260  005710                                 TST     (R0)
000262  001407                                 BEQ     2$
000264  010016                                 MOV     R0,(SP)                 ;                       3666
000266  004737  000000G                        JSR     PC,PUT.IO.BUFF
000272  013700  000114'                        MOV     MAD1,R0                 ;                       3667
000276  005060  000032                         CLR     32(R0)
000302  013716  000112'             2$:        MOV     MX2,(SP)                ;                       3670
000306  004737  000000G                        JSR     PC,PUT.PKT
000312  012737  177777  000112'                MOV     #-1,MX2                 ;                       3671
000320  013700  000114'             3$:        MOV     MAD1,R0                 ;                       3676
000324  005760  000032                         TST     32(R0)
000330  001010                                 BNE     4$
000332  013716  000110'                        MOV     MX1,(SP)                ;                       3679
000336  004737  000000G                         SR     PC,PUT.PKT
```

```
000342  012737  177777  000110'                MOV     #-1,MX1                 ;                       3680
000350  000410                                 BR      5$                      ;                       3676
000352  013700  000114'                4$:     MOV     MAD1,R0                 ;                       3684
000356  126027  000022  000042                CMPB    22(R0),#42
000364  001002                                 BNE     5$
000366  004737  000000V                        JSR     PC,FILL.BUFF            ;                       3686
000372  062706  000006                5$:     ADD     #6,SP                   ;                       3625
000376  000207                        6$:     RTS     PC                      ;                       3596

; Routine Size:  128 words,     Routine Base:  $CODE$ + 10770
; Maximum stack depth per invocation:  4 words
```

```
;    3689  1      GLOBAL routine GET_RANDOM : novalue =
;    3690  1
;    3691  1      !+
;    3692  1      !       THIS ROUTINE IS CALLED BY QIO_GEN TO GENERATE A SET OF RANDOM NUMBERS,
;    3693  1      !       AND TO STORE THEM INTO THE RANDOM NUMBER TABLE (RANDOM). THE RANDOM
;    3694  1      !       NUMBERS ARE USED TO SELECT I/O REQUEST PARAMETERS FOR THE CURRENT QIO
;    3695  1      !       OR QIO FAIR. IN ADDITION, IF DATA PATTERN #1 IS BEING USED, THESE
;    3696  1      !       RANDOM NUMBERS WILL BE USED IN THE WRITE OPERATION.
;    3697  1      !-
;    3698  1
;    3699  2          begin
;    3700  2
;    3701  2          own
;    3702  2              SEED : word initial (173),
;    3703  2              NEXT_RANDNUM : word initial (245);
;    3704  2
;    3705  2          incr COUNT from 0 to (RDM_LEN - 1) do
;    3706  3              begin
;    3707  3              SEED = (.SEED + .NEXT_RANDNUM + 1) * 4;
;    3708  3              NEXT_RANDNUM = (.NEXT_RANDNUM / 4) + .SEED;
;    3709  3              RANDOM [.COUNT] = .NEXT_RANDNUM;
;    3710  2              end;
;    3711  2
;    3712  1          end;


001272                                  .PSECT  $GGG$,  RO
001272  000255                SEED:     .WORD   255
001274  000365                NEXT.RANDNUM:
                                        .WORD   365


                                        .SBTTL  GET.RANDOM MULTI-DRIVE TEST ROUTINES
011370                                  .PSECT  $CODE$,  RO

000000  004137  000000G       GET.RANDOM::
                                        JSR     R1,$SAVE3                       ;                       3689
000004  013703  001272'                 MOV     SEED,R3                         ;                       3707
000010  013702  001274'                 MOV     NEXT.RANDNUM,R2
000014  005001                          CLR     R1                              ; COUNT                 3705
000016  010200                1$:       MOV     R2,R0                           ;                       3707
000020  060300                          ADD     R3,R0
000022  006300                          ASL     R0
000024  006300                          ASL     R0
000026  010037  001272'                 MOV     R0,SEED
000032  062737  000004  001272'         ADD     #4,SEED
000040  010246                          MOV     R2,-(SP)                        ;                       3708
000042  012746  000004                  MOV     #4,-(SP)
000046  004737  000000G                 JSR     PC,BL$DIV
000052  013703  001272'                 MOV     SEED,R3
000056  060300                          ADD     R3,R0
000060  010037  001274'                 MOV     R0,NEXT.RANDNUM
```

```
000064  010002                          MOV     R0,R2                           ; NEXT.RANDNUM,*                        3709
000066  010261  000000G                  MOV     R2,RANDOM(R1)                   ; *,*(COUNT)
000072  022626                          CMP     (SP)+,(SP)+                     ;                                       3706
000074  062701  000002                  ADD     #2,R1                           ; *,COUNT                               3705
000100  020127  000036                  CMP     R1,#36                          ; COUNT,*
000104  003744                          BLE     1$
000106  000207                          RTS     PC                              ;                                       3689

; Routine Size:  36 words,      Routine Base:  $CODE$ + 11370
; Maximum stack depth per invocation:  7 words
```

```
;    3713  1
;    3714  1      GLOBAL routine RANDY : novalue =
;    3715  1
;    3716  1      !+
;    3717  1      !       THIS ROUTINE GENERATES A 32-BIT RANDOM NUMBER. THE LOW 16 BITS
;    3718  1      !       ARE OUTPUT IN "RNDY0". THE HIGH 16 BITS ARE OUTPUT IN "RNDY1".
;    3719  1      !
;    3720  1      !       THE LOW 3 BITS OF CLK_TICKS SELECTS A WORD FROM 'RNDYIN'. THIS
;    3721  1      !       IS 'R_STRING'. FRAME_CNT (0-9) SELECTS A 7-BIT FRAME OF THIS
;    3722  1      !       WORD. BITS OF THIS FRAME ARE USED AS FOLLOWS:
;    3723  1      !
;    3724  1      !            BITS 0-2 ... SELECT A PATTERN FOR LOW WORD.
;    3725  1      !            BITS 1-3 ... SELECT A PATTERN FOR HIGH WORD.
;    3726  1      !            BIT   4  ... IF 1, SHIFT PATTERN LEFT.
;    3727  1      !            BITS 4-6 ... SELECTS MASKS FOR FINAL OUTPUT.
;    3728  1      !
;    3729  1      !-
;    3730  1
;    3731  2          begin
;    3732  2
;    3733  2          local
;    3734  2            PAT_LO: WORD,                                   !LO WORD OF PATTERN
;    3735  2            PAT_HI: WORD,                                   !HI WORD OF PATTERN
;    3736  2            SHIFT : WORD,                                   !LEFT-SHIFT BIT
;    3737  2            MSKNO : WORD;                                   !WHICH MASK TO USE
;    3738  2
;    3739  2
;    3740  2         IF .FRAME_CNT EQLU 0                               !IF IT'S TIME TO SAMPLE CLOCK AGAIN
;    3741  2         THEN
;    3742  3            BEGIN
;    3743  3            R_STRING = .RNDYIN [.CLK_TICKS AND 7]           !CLOCK BITS SELECT 16 BIT STRING
;    3744  2            END;
;    3745  2
;    3746  2
;    3747  2          PAT_LO = .RNDYIN [(.R_STRING ^ -.FRAME_CNT) AND 7];       !BITS 0-2 OF FRAME SELECT LO WD OF PATTERN
;    3748  2          PAT_HI = .RNDYIN [(.R_STRING ^(-1 -.FRAME_CNT)) AND 7];   !BITS 1-3 OF FRAME SELECT HI WD OF PATTERN
;    3749  2
;    3750  2
;    3751  2          SHIFT =  (.R_STRING ^ (-4 -.FRAME_CNT)) AND 1;            !BIT 4 OF FRAME IS SHIFTER.
;    3752  2          PAT_LO =  .PAT_LO ^ .SHIFT;                               !SHIFT PATTERN IF SHIFTER = 1
;    3753  2          PAT_HI = (.PAT_HI ^ .SHIFT) + .SHIFT;                     !SHIFT PATTERN AND ADD 1 IF SHIFTER = 1
;    3754  2
;    3755  2
;    3756  2          MSKNO = (.R_STRING ^ (-4 -.FRAME_CNT)) AND 7;             !GET MASK INDEX
;    3757  2          RNDY0 = .PAT_LO AND (.RNDMS0 [.MSKNO]);                   !MASK LO WORD
;    3758  2          RNDY1 = .PAT_HI AND (.RNDMS1 [.MSKNO]);                   !MASK HI WORD
;    3759  2
;    3760  2
;    3761  2          FRAME_CNT = .FRAME_CNT + 1;                               !SHIFT FRAME LEFT ONE BIT
;    3762  2          IF .FRAME_CNT GTRU 9                                      !IF DONE TEN RANDOM 32-BIT NUMBERS
;    3763  2          THEN
;    3764  2             FRAME_CNT = 0;                                         !ZERO IT, SO WE'LL READ CLOCK NEXT TIME
;    3765  2
```

```
;   3766  1     END;


                                        .SBTTL  RANDY MULTI-DRIVE TEST ROUTINES
000000  004137  000000G           RANDY:: JSR     R1,$SAVE4                       ;                       3714
000004  013702  000132'                   MOV     FRAME.CNT,R2                    ;                       3740
000010  001010                           BNE     1$
000012  013700  000000G                   MOV     CLK.TICKS,R0                    ;                       3743
000016  042700  177770                   BIC     #177770,R0
000022  006300                           ASL     R0
000024  016037  000136' 000134           MOV     RNDYIN(R0),R.STRING             ;                       3742
000032  013701  000134'           1$:     MOV     R.STRING,R1                     ;                       3747
000036  010146                           MOV     R1,-(SP)
000040  010246                           MOV     R2,-(SP)
000042  005416                           NEG     (SP)
000044  004737  000000G                   JSR     PC,BL$SHF
000050  042700  177770                   BIC     #177770,R0
000054  006300                           ASL     R0
000056  016004  000136'                   MOV     RNDYIN(R0),R4                   ; *,PAT.LO
000062  010116                           MOV     R1,(SP)                         ;                       3748
000064  012746  177777                   MOV     #-1,-(SP)
000070  160216                           SUB     R2,(SP)
000072  004737  000000G                   JSR     PC,BL$SHF
000076  042700  177770                   BIC     #177770,R0
000102  006300                           ASL     R0
000104  016003  000136'                   MOV     RNDYIN(R0),R3                   ; *,PAT.HI
000110  010116                           MOV     R1,(SP)                         ;                       3751
000112  012746  177774                   MOV     #-4,-(SP)
000116  160216                           SUB     R2,(SP)
000120  004737  000000G                   JSR     PC,BL$SHF
000124  010001                           MOV     R0,R1
000126  010102                           MOV     R1,R2                           ; *,SHIFT
000130  042702  177776                   BIC     #177776,R2                      ; *,SHIFT
000134  010416                           MOV     R4,(SP)                         ; PAT.LO,*              3752
000136  010246                           MOV     R2,-(SP)                        ; SHIFT,*
000140  004737  000000G                   JSR     PC,BL$SHF
000144  010004                           MOV     R0,R4                           ; *,PAT.LO
000146  010316                           MOV     R3,(SP)                         ; PAT.HI,*              3753
000150  010246                           MOV     R2,-(SP)                        ; SHIFT,*
000152  004737  000000G                   JSR     PC,BL$SHF
000156  060200                           ADD     R2,R0                           ; SHIFT,*
000160  010003                           MOV     R0,R3                           ; *,PAT.HI
000162  010102                           MOV     R1,R2                           ; *,MSKNO               3756
000164  042702  177770                   BIC     #177770,R2                      ; *,MSKNO
000170  010200                           MOV     R2,R0                           ; MSKNO,*               3757
000172  006300                           ASL     R0
000174  016037  000160' 000126'         MOV     RNDMS0(R0),RNDY0
000202  005104                           COM     R4
000204  040437  000126'                   BIC     R4,RNDY0
000210  010200                           MOV     R2,R0                           ; MSKNO,*               3758
000212  006300                           ASL     R0
000214  016037  000200' 000130'         MOV     RNDMS1(R0),RNDY1
000222  005103                           COM     R3
```

```
000224  040337  000130'                        BIC     R3,RNDY1
000230  005237  000132'                        INC     FRAME.CNT                       ;       3761
000234  023727  000132' 000011                 CMP     FRAME.CNT,#11                   ;       3762
000242  101402                                 BLOS    2$
000244  005037  000132                         CLR     FRAME.CNT                       ;       3764
000250  062706  000014                 2$:     ADD     #14,SP                          ;       3731
000254  000207                                 RTS     PC                              ;       3714

; Routine Size:  87 words,      Routine Base:  $CODE$ + 11500
; Maximum stack depth per invocation:  12 words
```

```
;    3767  1      GLOBAL routine QIO_UNIT : novalue =
;    3768  1
;    3769  1      !+
;    3770  1      !       THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
;    3771  1      !       CONFIGURED UNDER THE CURRENT CONTROLLER (CCTLR) TO BE USED FOR THE
;    3772  1      !       CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
;    3773  1      !       UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
;    3774  1      !       RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
;    3775  1      !
;    3776  1      !       IMPLICIT INPUTS:
;    3777  1      !               CST_ADDR   ADDRESS OF CURRENT CONTROLLER'S CST
;    3778  1      !
;    3779  1      !       IMPLICIT OUTPUTS:
;    3780  1      !               THE RD/RX DISK NUMBER (DISK ADDRESS) IS LOADED INTO THE
;    3781  1      !                       APPROPRIATE FIELD OF BOTH MSCP PACKETS.
;    3782  1      !-
;    3783  1
;    3784  2          begin
;    3785  2
;    3786  2          local
;    3787  2              MOD_COUNT : byte,
;    3788  2              TBL_COUNT : byte,
;    3789  2              SELECT_RD : byte initial (byte (TRUE)),
;    3790  2      !ZZZ    RD_COUNT : word initial (0),
;    3791  2              RX_COUNT : word initial (0);
;    3792  2
;    3793  2      !
;    3794  2      ! THE UNITS WILL BE SELECTED ON AN ADJUSTABLE RATIO, RD51/52 TO RX50,
;    3795  2      ! SELECTED VIA THE SOFTWARE PARAMETERS
;    3796  2      !
;    3797  2      ! THIS MODE IS FOR SELECTING DEVICES ON THE FOLLOWING SCHEME:
;    3798  2      ! CHOOSE A DEVICE AND KEEP IT SELECTED FOR A CONSTANT TIME, THEN
;    3799  2      ! MOVE TO THE NEXT. THIS IS NON-RANDOM, FIXED SEQUENTIAL OPERATIONAL
;    3800  2      ! MODE
;    3801  2      !
;    3802  2
;    3803  2                  RD_COUNT = 0;                                          !ZZZ
;    3804  2                  RX_COUNT = 0;                                          !ZZZ
;    3805  2
;    3806  2                  incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT SIZE do
;    3807  2
;    3808  2                      if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3809  2                         (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3810  3                         (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3811  2                      then
;    3812  2
;    3813  3                          if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;    3814  2                          then
;    3815  2                              RD_COUNT = .RD_COUNT + 1        ! NUMBER OF RD5 /52s UNDER TEST
;    3816  2                          else
;    3817  2                              RX_COUNT = .RX_COUNT + 1;       ! NUMBER OF RX50s UNDER TEST
;    3818  2
;    3819  2
```

```
;  3820  2
;  3821  2          if (not BIT_TST (SWP_FLAGS, SWF_RDM)) and          ! NOT RANDOM MODE
;  3822  3             (not BIT_TST (SWP_FLAGS, SWF_SEQ))              ! NOT RANDOM SEQUEUNTIAL MODE
;  3823  2          then
;  3824  2
;  3825  2              if (.BST_CNT neq 0) and
;  3826  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;  3827  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;  3828  3                 (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;  3829  2              then
;  3830  3                  begin                                      ! ALREADY WITHIN DEVICE
;  3831  3                  BST_CNT = .BST_CNT - 1;
;  3832  3                  SET_UPAR (.BST_DEV);
;  3833  3                  MAD1 [DK_NUM] = .CDISK;
;  3834  3                  MAD2 [DK_NUM] = .CDISK;
;  3835  3                  return;
;  3836  3                  end
;  3837  2              else
;  3838  3                  begin                                      ! GET NEW DEVICE
;  3839  3
;  3840  3      !ZZZ        incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  3841  3      !ZZZ
;  3842  3      !ZZZ            if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  3843  3      !ZZZ               (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  3844  3      !ZZZ               (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;  3845  3      !ZZZ            then
;  3846  3      !ZZZ
;  3847  3      !ZZZ                if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;  3848  3      !ZZZ                then
;  3849  3      !ZZZ                    RD_COUNT = .RD_COUNT + 1       ! NUMBER OF RD51/52s UNDER TEST
;  3850  3      !ZZZ                else
;  3851  3      !ZZZ                    RX_COUNT = .RX_COUNT + 1;      ! NUMBER OF RX50s UNDER TEST
;  3852  3
;  3853  3                  incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR   1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  3854  4                      begin
;  3855  4
;  3856  4                      if (.BST_DEV eql 0) or
;  3857  5                         (.BST_DEV eql ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN))
;  3858  4                      then
;  3859  4                          BST_DEV = OF_UN
;  3860  4                      else
;  3861  4                          BST_DEV = .BST_DEV + UNIT_SIZE;
;  3862  4
;  3863  4                      if (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;  3864  4                         (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;  3865  5                         (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;  3866  4                      then
;  3867  5                          begin
;  3868  5
;  3869  5                          if .CST_ADDR [.BST_DEV + OF_DATA, D_TYPE] eql REMOVABLE
;  3870  5                          then
;  3871  5                              BST_CNT = .RX_MAX_SEQ_CNT / .RX_COUNT
;  3872  5                          else
```

```
;  3873  5                        BST_CNT = .RD_MAX_SEQ_CNT / .RD_COUNT;
;  3874  5
;  3875  5                    if .BST_CNT eql 0
;  3876  5                    then
;  3877  5                        BST_CNT = 1;
;  3878  5
;  3879  5                    SET_UPAR (.BST_DEV);
;  3880  5                    MAD1 [DK_NUM] = .CDISK;
;  3881  5                    MAD2 [DK_NUM] = .CDISK;
;  3882  5                    return;
;  3883  4                    end;
;  3884  4
;  3885  3                end;
;  3886  3
;  3887  2            end;
;  3888  2
;  3889  2     !
;  3890  2     ! RANDOM SELECTION OF DRIVES
;  3891  2     !
;  3892  2     ! DETERMINE IF RD51/52s ARE TO BE SELECTED
;  3893  2     !
;  3894  2
;  3895  2         if ((.RANDOM [RDM_LEN - 1] and %o'077777') mod 100) gequ .SWP_RAT
;  3896  2         then
;  3897  2             SELECT_RD = FALSE;
;  3898  2
;  3899  2     !
;  3900  2     ! IF RD51/52s SELECTED
;  3901  2     !
;  3902  2     ! COUNT NUMBER OF RD51/52s AVAILABLE
;  3903  2     !
;  3904  2
;  3905  2         if .SELECT_RD
;  3906  2         then
;  3907  3             begin
;  3908  3             MOD_COUNT = 0;                          ! COUNT THE NUMBER OF RDs UNDER TEST
;  3909  3
;  3910  3             incr OFFSET from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do
;  3911  3
;  3912  3                 if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  3913  3                    (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  3914  3                    (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED) and
;  3915  4                    (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;  3916  3                 then
;  3917  4                     begin
;  3918  4                     STORAGE [.MOD_COUNT] = .OFFSET;
;  3919  4                     MOD_COUNT = .MOD_COUNT + 1;
;  3920  3                     end;
;  3921  3
;  3922  3     !
;  3923  3     ! SELECT ON OF THE RD51/52s
;  3924  3     !
;  3925  3
```

```
;    3926  3
;    3927  3              if .MOD_COUNT neq 0                                ! IF AT LEAST ONE RD51/52 PRESENT
;    3928  3              then
;    3929  4                  begin
;    3930  4                  TBL_COUNT = 0;
;    3931  4
;    3932  4                  do
;    3933  5                      begin
;    3934  5                      SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3935  5                      TBL_COUNT = .TBL_COUNT + 1;
;    3936  5                      end
;    3937  5                  until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3938  5                         (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3939  4                         (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3940  4                        (.TBL_COUNT eql RDM_LEN);
;    3941  4
;    3942  4                  MAD1 [DK_NUM] = .CDISK;
;    3943  4                  MAD2 [DK_NUM] = .CDISK;
;    3944  4                  return;
;    3945  3                  end;
;    3946  3
;    3947  2              end;
;    3948  2
;    3949  2          !
;    3950  2          ! IF NO RD51/52 SELECTED, SELECT AN RX50
;    3951  2          !
;    3952  2          ! COUNT THE NUMBER OF RX50s
;    3953  2          !
;    3954  2
;    3955  2              MOD_COUNT = 0;
;    3956  2
;    3957  2              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3958  2
;    3959  2                  if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3960  2                     (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3961  2                     (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql REMOVABLE) and
;    3962  3                     (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3963  2                  then
;    3964  3                      begin
;    3965  3                      STORAGE [.MOD_COUNT] = .OFFSET;
;    3966  3                      MOD_COUNT = .MOD_COUNT + 1;
;    3967  2                      end;
;    3968  2
;    3969  2          !
;    3970  2          ! AND CHOOSE ONE!
;    3971  2          !
;    3972  2
;    3973  2              if .MOD_COUNT neq 0
;    3974  2              then
;    3975  3                  begin
;    3976  3                  TBL_COUNT = 0;
;    3977  3
;    3978  3                  a
```

```
;    3979  4                begin
;    3980  4                SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3981  4                TBL_COUNT = .TBL_COUNT + 1;
;    3982  4                end
;    3983  4            until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3984  4                   (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3985  3                   (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3986  3                  (.TBL_COUNT eql RDM_LEN);
;    3987  3
;    3988  3            MAD1 [DK_NUM] = .CDISK;
;    3989  3            MAD2 [DK_NUM] = .CDISK;
;    3990  3            return;
;    3991  2            end;
;    3992  2
;    3993  2        !
;    3994  2        ! IF NO UNIT SELECTED SO FAR BY ABOVE METHOD, SELECT ANY ONE AT RANDOM
;    3995  2        !
;    3996  2        ! COUNT ALL UNITS AVAILABLE
;    3997  2        !
;    3998  2
;    3999  2            MOD_COUNT = 0;
;    4000  2
;    4001  2            incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    4002  2
;    4003  2                if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    4004  2                   (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    4005  3                   (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    4006  2                then
;    4007  3                    begin
;    4008  3                    STORAGE [.MOD_COUNT] = .OFFSET;
;    4009  3                    MOD_COUNT = .MOD_COUNT + 1;
;    4010  2                    end;
;    4011  2
;    4012  2        !
;    4013  2        ! SELECT ANY ONE ONE UNIT AT RANDOM
;    4014  2        !
;    4015  2            if .MOD_COUNT neq 0
;    4016  2            then
;    4017  3                begin
;    4018  3                TBL_COUNT = 0;
;    4019  3
;    4020  3                do
;    4021  4                    begin
;    4022  4                    SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    4023  4                    TBL_COUNT = .TBL_COUNT + 1;
;    4024  4                    end
;    4025  4                until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    4026  4                       (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    4027  3                       (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    4028  3                      (.TBL_COUNT eql RDM_LEN);
;    4029  3
;    4030  3                MAD1 [DK_NUM] = .CDISK;
;    4031  3                MAD2 [DK_NUM] = .CDISK;
```

```
;    4032  3              return
;    4033  3              end
;    4034  3
;    4035  3      !
;    4036  3      ! DECLARE END-OF-PASS IF NO UNIT ONLINE
;    4037  3      !
;    4038  3
;    4039  2          else
;    4040  2              EOP_FLAG = TRUE;
;    4041  2
;    4042  1          end;                                        ! ROUTINE QIO_UNIT


                                .SBTTL  QIO.UNIT MULTI DRIVE TEST ROUTINES
000000  004137  000000G   QIO.UNIT::
                            JSR     R1,$SAVE4                   ;                       3767
000004  112704  000001             MOVB    #1,R4               ; *,SELECT.RD           3784
000010  005003                     CLR     R3                  ; RX.COUNT
000012  005037  000000G            CLR     RD.COUNT            ;                       3803
000016  013702  000000G            MOV     CST.ADDR,R2         ;                       3808
000022  012701  000006             MOV     #6,R1               ; *,OFFSET              3806
000026  010100              1$:     MOV     R1,R0               ; OFFSET,*              3808
000030  060200                     ADD     R2,R0
000032  032710  040000             BIT     #40000,(R0)
000036  001415                     BEQ     3$
000040  032710  020000             BIT     #20000,(R0)         ;                       3809
000044  001412                     BEQ     3$
000046  032710  010000             BIT     #10000,(R0)         ;                       3810
000052  001007                     BNE     3$
000054  132710  000020             BITB    #20,(R0)            ;                       3813
000060  001403                     BEQ     2$
000062  005237  000000G            INC     RD.COUNT            ;                       3815
000066  000401                     BR      3$                  ;                       3813
000070  005203              2$:     INC     R3                  ; RX.COUNT              3817
000072  062701  000024      3$:     ADD     #24,R1              ; *,OFFSET              3806
000076  020127  000102             CMP     R1,#102             ; OFFSET,*
000102  003751                     BLE     1$
000104  032737  000002  000000G    BIT     #2,SWP.FLAGS        ;                       3821
000112  001163                     BNE     13$
000114  032737  001000  000000G    BIT     #1000,SWP.FLAGS     ;                       3822
000122  001157                     BNE     13$
000124  005737  001242'           TST     BST.CNT             ;                       3825
000130  001447                     BEQ     4$
000132  013700  001244'           MOV     BST.DEV,R0          ;                       3826
000136  006300                     ASL     R0
000140  060200                     ADD     R2,R0
000142  032710  040000             BIT     #40000,(R0)
000146  001440                     BEQ     4$
000150  013700  001244'           MOV     BST.DEV,R0          ;                       3827
000154  006300                     ASL     R0
000156  060200                     ADD     R2,R0
000160  032710  020000             BIT     #20000,(R0)
000164  001431                     BEQ     4$
```

```
000166  013700  001244'                        MOV     BST.DEV,R0                      ;                       3828
000172  006300                                  ASL     R0
000174  060200                                  ADD     R2,R0
000176  032710  010000                          BIT     #10000,(R0)
000202  001022                                  BNE     4$
000204  005337  001242'                        DEC     BST.CNT                         ;                       3831
000210  013746  001244'                        MOV     BST.DEV,-(SP)                   ;                       3832
000214  004737  000000G                       JSR     PC,SET.UPAR
000220  013700  000114'                        MOV     MAD1,R0                         ;                       3833
000224  013760  000000G 000016                MOV     CDISK,16(R0)
000232  013700  000116'                        MOV     MAD2,R0                         ;                       3834
000236  013760  000000G 000016                MOV     CDISK,16(R0)
000244  022726                                TST     (SP)+                           ;                       3835
000246  000207                                RTS     PC                              ;                       3830
000250  012702  000003              4$:       MOV     #3,R2                           ; *,OFFSET              3853
000254  013700  001244'              5$:       MOV     BST.DEV,R0                      ;                       3856
000260  001403                                BEQ     6$
000262  020027  000041                        CMP     R0,#41                          ;                       3857
000266  001004                                BNE     7$
000270  012737  000003  001244'      6$:       MOV     #3,BST.DEV                      ;                       3859
000276  000403                                BR      8$                              ;                       3856
000300  062737  000012  001244'      7$:       ADD     #12,BST.DEV                     ;                       3861
000306  013700  001244'              8$:       MOV     BST.DEV,R0                      ;                       3863
000312  006300                                ASL     R0
000314  063700  000000G                       ADD     CST.ADDR,R0
000320  032710  040000                        BIT     #40000,(R0)
000324  001451                                BEQ     12$
000326  032710  020000                        BIT     #20000,(R0)                     ;                       3864
000332  001446                                BEQ     12$
000334  032710  010000                        BIT     #10000,(R0)                     ;                       3865
000340  001043                                BNE     12$
000342  132710  000020                        BITB    #20,(R0)                        ;                       3869
000346  001004                                BNE     9$
000350  013746  001270'                        MOV     RX.MAX.SEQ.CNT,-(SP)            ;                       3871
000354  010346                                MOV     R3,-(SP)                        ; RX.COUNT,*
000356  000404                                BR      10$
000360  013746  001266'              9$:       MOV     RD.MAX.SEQ.CNT,-(SP)            ;                       3873
000364  013746  000000G                       MOV     RD.COUNT,-(SP)
000370  004737  000000G              10$:      JSR     PC,BL$DIV
000374  010037  001242'                        MOV     R0,BST.CNT
000400  001003                                BNE     11$
000402  012737  000001  001242'              MOV     #1,BST.CNT                      ;                       3875
000410  013716  001244'              11$:      MOV     BST.DEV,(SP)                    ;                       3877
000414  004737  000000G                       JSR     PC,SET.UPAR                     ;                       3879
000420  013700  000114'                        MOV     MAD1,R0
000424  013760  000000G 000016                MOV     CDISK,16(R0)                    ;                       3880
000432  013700  000116'                        MOV     MAD2,R0
000436  013760  000000G 000016                MOV     CDISK,16(R0)                    ;                       3881
000444  022626                                CMP     (SP)+,(SP)+
000446  000207                                RTS     PC                              ;                       3882
000450  062702  000012              12$:      ADD     #12,R2                          ;                       3867
000454  020227  000041                        CMP     R2,#41                          ; *,OFFSET              3853
000460  003675                                BLE     5$                              ; OFFSET,*
```

```
000462  013746  000036G                13$:    MOV     RANDOM+36,-(SP)         ;                               3896
000466  042716  100000                        BIC     #100000,(SP)
000472  012746  000144                        MOV     #144,-(SP)
000476  004737  000000G                        JSR     PC,BL$MOD
000502  022626                                CMP     (SP)+,(SP)+
000504  020037  000000G                        CMP     R0,SWP.RAT
000510  103401                                BLO     14$
000512  105004                                CLRB    R4                      ; SELECT.RD                     3898
000514  006004                        14$:    ROR     R4                      ; SELECT.RD                     3906
000516  103105                                BCC     19$
000520  105003                                CLRB    R3                      ; MOD.COUNT                     3909
000522  012701  000003                        MOV     #3,R1                   ; *,OFFSET                      3911
000526  010100                        15$:    MOV     R1,R0                   ; OFFSET,*                      3913
000530  006300                                ASL     R0
000532  063700  000000G                        ADD     CST.ADDR,R0
000536  032710  040000                        BIT     #40000,(R0)
000542  001417                                BEQ     16$
000544  032710  020000                        BIT     #20000,(R0)             ;                               3914
000550  001414                                BEQ     16$
000552  132710  000020                        BITB    #20,(R0)                ;                               3915
000556  001411                                BEQ     16$
000560  032710  010000                        BIT     #10000,(R0)             ;                               3916
000564  001006                                BNE     16$
000566  005000                                CLR     R0                      ;                               3919
000570  150300                                BISB    R3,R0                   ; MOD.COUNT,*
000572  006300                                ASL     R0
000574  010160  000064'                       MOV     R1,STORAGE(R0)          ; OFFSET,*
000600  105203                                INCB    R3                      ; MOD.COUNT                     3920
000602  062701  000C12                16$:    ADD     #12,R1                  ; *,OFFSET                      3911
000606  020127  000041                        CMP     R1,#41                  ; OFFSET,*
000612  003745                                BLE     15$
000614  105703                                TSTB    R3                      ; MOD.COUNT                     3927
000616  001445                                BEQ     19$
000620  105002                                CLRB    R2                      ; TBL.COUNT                     3930
000622  005000                        17$:    CLR     R0                      ;                               3934
000624  150200                                BISB    R2,R0                   ; TBL.COUNT,*
000626  006300                                ASL     R0
000630  016046  000000G                       MOV     RANDOM(R0),-(SP)
000634  042716  10C000                        BIC     #100000,(SP)
000640  005046                                CLR     -(SP)
000642  110316                                MOVB    R3,(SP)                 ; MOD.COUNT,*
000644  004737  000000G                        JSR     PC,BL$MOD
000650  006300                                ASL     R0
000652  016016  000064'                       MOV     STORAGE(R0),(SP)
000656  004737  000000G                        JSR     PC,SET.UPAR
000662  105202                                INCB    R2                      ; TBL.COUNT                     3935
000664  022626                                CMP     (SP)+,(SP)+             ;                               3933
000666  013700  000000G                        MOV     CUOFF,R0                ;                               3937
000672  006300                                ASL     R0
000674  063700  000000G                        ADD     CST.ADDR,R0
000700  032710  040000                        BIT     #40000,(R0)
000704  001406                                BEQ     18$
000706  032710  020000                        BIT     #20000,(R0)             ;                               3938
```

```
000712  001403                          BEQ     18$
000714  032710  010000                  BIT     #10000,(R0)             ;                               3939
000720  001510                          BEQ     24$
000722  120227  000020          18$:    CMPB    R2,#20                  ; TBL.COUNT,*                   3940
000726  001335                          BNE     17$
000730  000504                          BR      24$                     ;                               3942
000732  105003          19$:    CLRB    R3                      ; MOD.COUNT                     3955
000734  012701  000C03                  MOV     #3,R1                   ; *,OFFSET                      3957
000740  010100          20$:    MOV     R1,R0                   ; OFFSET,*                      3959
000742  006300                          ASL     R0
000744  063700  000000G                 ADD     CST.ADDR,R0
000750  032710  040000                  BIT     #40000.(R0)
000754  001417                          BEQ     21$
000756  032710  020000                  BIT     #20000,(R0)             ;                               3960
000762  001414                          BEQ     21$
000764  132710  000020                  BITB    #20,(R0)                ;                               3961
000770  001011                          BNE     21$
000772  032710  010000                  BIT     #10000,(R0)             ;                               3962
000776  001006                          BNE     21$
001000  C05000                          CLR     R0                      ;                               3965
001002  150300                          BISB    R3,R0                   ; MOD.COUNT,*
001004  006300                          ASL     R0
001006  010160  000064'                 MOV     R1,STORAGE(R0)          ; OFFSET,*
001012  105203                          INCB    R3                      ; MOD.COUNT                     3966
001014  062701  000012          21$:    ADD     #12,R1                  ; *,OFFSET                      3957
001020  020127  000041                  CMP     R1,#41                  ; OFFSET,*
001024  003745                          BLE     20$
001026  105703                          TSTB    R3                      ; MOD.COUNT                     3973
001030  001445                          BEQ     25$
001032  105002                          CLRB    R2                      ; TBL.COUNT                     3976
001034  005000          22$:    CLR     R0                      ;                               3980
001036  150200                          BISB    R2,R0                   ; TBL.COUNT,*
001040  006300                          ASL     R0
001042  016046  000000G                 MOV     RANDOM(R0),-(SP)
001046  042716  100000                  BIC     #100000,(SP)
001052  005046                          CLR     -(SP)
001054  110316                          MOVB    R3,(SP)                 ; MOD.COUNT,*
001056  004737  000000G                 JSR     PC,BL$MOD
001062  006300                          ASL     R0
001064  016016  000064'                 MOV     STORAGE(R0),(SP)
001070  004737  000000G                 JSR     PC,SET.UPAR
001074  105202                          INCB    R2                      ; TBL.COUNT                     3981
001076  022626                          CMP     (SP)+,(SP)+             ;                               3979
001100  013700  000000G                 MOV     CUOFF,R0                ;                               3983
001104  006300                          ASL     R0
001106  063700  000000G                 ADD     CST.ADDR,R0
001112  032710  040000                  BIT     #40000,(R0)
001116  001406                          BEQ     23$
001120  032710  020000                  BIT     #20000,(R0)             ;                               3984
001124  001403                          BEQ     23$
001126  032710  010000                  BIT     #10000,(R0)             ;                               3985
001132  001505                          BEQ     30$
001134  120227  000020          23$:    CMPB    R2,#20                  ; TBL.COUNT,*                   3986
```

```
001140  001335                          BNE     22$
001142  000501                  24$:    BR      30$                     ;                               3988
001144  105003                  25$:    CLRB    R3                      ; MOD.COUNT                     3999
001146  012701  000003                  MOV     #3,R1                   ; *,OFFSET                      4001
001152  010100                  26$:    MOV     R1,R0                   ; OFFSET,*                      4003
001154  006300                          ASL     R0
001156  063700  000000G                 ADD     CST.ADDR,R0
001162  032710  040000                  BIT     #40000,(R0)
001166  001414                          BEQ     27$
001170  032710  020000                  BIT     #20000,(R0)             ;                               4004
001174  001411                          BEQ     27$
001176  032710  010000                  BIT     #10000,(R0)             ;                               4005
001202  001006                          BNE     27$
001204  005000                          CLR     R0                      ;                               4008
001206  150300                          BISB    R3,R0                   ; MOD.COUNT,*
001210  006300                          ASL     R0
001212  010160  000064'                 MOV     R1,STORAGE(R0)          ; OFFSET,*
001216  105203                          INCB    R3                      ; MOD.COUNT                     4009
001220  062701  000012          27$:    ADD     #12,R1                  ; *,OFFSET                      4001
001224  020127  000041                  CMP     R1,#41                  ; OFFSET,*
001230  003750                          BLE     26$
001232  105703                          TSTB    R3                      ; MOD.COUNT                     4015
001234  001457                          BEQ     31$
001236  105002                          CLRB    R2                      ; TBL.COUNT                     4018
001240  005000                  28$:    CLR     R0                      ;                               4022
001242  150200                          BISB    R2,R0                   ; TBL.COUNT,*
001244  006300                          ASL     R0
001246  016046  000000G                 MOV     RANDOM(R0),-(SP)
001252  042716  100000                  BIC     #100000,(SP)
001256  005046                          CLR     -(SP)
001260  110316                          MOVB    R3,(SP)                 ; MOD.COUNT,*
001262  004737  000000G                 JSR     PC,BL$MOD
001266  006300                          ASL     R0
001270  016016  000064'                 MOV     STORAGE(R0),(SP)
001274  004737  000000G                 JSR     PC,SET.UPAR
001300  105202                          INCB    R2                      ; TBL.COUNT                     4023
001302  022626                          CMP     (SP)+,(SP)+             ;                               4021
001304  013700  000000G                 MOV     CUOFF,R0                ;                               4025
001310  006300                          ASL     R0
001312  063700  000000G                 ADD     CST.ADDR,R0
001316  032710  040000                  BIT     #40000,(R0)
001322  001406                          BEQ     29$
001324  032710  020000                  BIT     #20000,(R0)             ;                               4026
001330  001403                          BEQ     29$
001332  032710  010000                  BIT     #10000,(R0)             ;                               4027
001336  001403                          BEQ     30$
001340  120227  000020          29$:    CMPB    R2,#20                  ; TBL.COUNT,*                   4028
001344  001335                          BNE     28$
001346  013700  000114'          30$:    MOV     MAD1,R0                 ;                               4030
001352  013760  000000G 000016          MOV     CDISK,16(R0)
001360  013700  000116'                 MOV     MAD2,R0                 ;                               4031
001364  013760  000000G 000016          MOV     CDISK,16(R0)
001372  000207                          RTS     PC                      ;                               4017
```

```
ZRQAM3          RD/RX EXERCISER                                 4-Apr-1985 13:23:31    VAX 11 Bl;ee-16 V4.1 582                SEQ 0357
V02.2           MULTI-DRIVE TEST ROUTINES                       2-Apr-1965 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAG0.BL2;19    Page 102
                                                                                                                               (24)

001374  112737  000001  000000G         31$:    MOVB    #1,EOP.FLAG                    ;                                        4040
001402  000207                                  RTS     PC                             ;                                        3767

; Routine Size:  386 words,     Routine Base:  $CODE$ + 11756
; Maximum stack depth per invocation:  8 words
```

```
;  4043  1     GLOBAL routine QIO_FUNC : novalue =
;  4044  1
;  4045  1     !+
;  4046  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
;  4047  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
;  4048  1     !       BY THE FOLLWING ALGORITHM:
;  4049  1     !
;  4050  1     !               IF THE CHOSEN UNIT IS PROTECTED
;  4051  1     !               THEN
;  4052  1     !                       FUNCTION = READ
;  4053  1     !               ELSE (UNPROTECTED)
;  4054  1     !                       FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
;  4055  1     !                       NUMBER
;  4056  1     !
;  4057  1     !       IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE-
;  4058  1     !       COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
;  4059  1     !       THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
;  4060  1     !       PACKET. OTHERWISE, THE SECOND MSCP PACKET IS RETURNED TO THE POOL.
;  4061  1     !
;  4062  1     !
;  4063  1     !       PERIODIACLLY, THIS ROUTINE WILL CALL THE DUP ROUTINE BEFORE IT   ZZZ
;  4064  1     !       BEGINS ITS OWN TASK. IF THE OPERATOR HAS SELECTED, "ALSO RUN     ZZZ
;  4065  1     !       DUP EXERCISER," THEN DUP TESTING OF DBNS WILL BE INTERLEAVED     ZZZ
;  4066  1     !       WITH THE REGULAR MSCP TESTING OF THE LBNS.                       ZZZ
;  4067  1     !
;  4068  1     !       TO AVOID LONG, CUMULATIVE INIT TIMES, THE DUP CODE IS ONLY       ZZZ
;  4069  1     !       EXECUTED AFTER (25 TIMES 'DUPROUND') MSCP I/O'S HAVE BEEN DONE.  ZZZ
;  4070  1     !       THE NUMBER OF DUP I/O'S IS 'DUPROUND'. THIS GIVES US A 25 TO 1   ZZZ
;  4071  1     !       INTERLEAVE.                                                      ZZZ
;  4072  1     !
;  4073  1     !       THE DUP TESTING IS DONE BY EXECUTING CONTROLLER LOCAL PROGRAMS   ZZZ
;  4074  1     !       TO READ OR WRITE/READ DBNS. AFTER THE DUP TESTING, THE CON-      ZZZ
;  4075  1     !       TROLLER IS REINITIALIZED, AND QIO_FUNC ROUTINE CONTINUES FROM    ZZZ
;  4076  1     !       WHERE IT LEFT OFF.                                               ZZZ
;  4077  1     !
;  4078  1     !
;  4079  1     !       IMPLICIT INPUTS:
;  4080  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  4081  1     !               CUOFF - CURRENT UNIT CST OFFSET
;  4082  1     !
;  4083  1     !       IMPLICIT OUTPUTS:
;  4084  1     !               THE OPCODE FIELD OF ONE OR BOTH MSCP PACKETS IS LOADED.
;  4085  1     !-
;  4086  1
;  4087  2         begin
;  4088  2
;  4089  2         local
;  4090  2             FUNC : word;                                                 ! OPCODE (READ OR WRITE)
;  4091  2
;  4092  2     DUOFF = .CUOFF;                                  !SAVE IN CASE OTHER CMDS ZZZ
;  4093  2                                                      !LEFT IN QUEUE           ZZZ
;  4094  3     IF ((.CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] LEQ 0) AND !MSCP CNT=0       ZZZ
;  4095  3         (.CST_ADDR [.DUOFF, D_TYPE] NEQ RX_50) AND          !FIXED DISK      ZZZ
```

```
;  4096  3          (.CST_ADDR [.DUOFF + OF_DUPFLAGS, NODUPMEDIA] NEQ 1)) !MEDIA IN      ZZZ
;  4097  3                                                                               !ZZZ
;  4098  2      THEN                                                                     !ZZZ
;  4099  3          BEGIN                                                                !ZZZ
;  4100  3          PUT_PKT (.MX2);                          !RETURN 2ND ENVLOPE         !ZZZ
;  4101  3          MX2 = -1;                                !INDICATE FAILURE           ZZZ
;  4102  3          DUP ();                                  !DO DUP TEST                ZZZ
;  4103  3          CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] =  !REINIT MSCP FUN-           ZZZ
;  4104  3                  (25 * .DUPROUND);                !CTION COUNTER              ZZZ
;  4105  3                                                                               !ZZZ
;  4106  3      !       THE FOLLOWING REINITS 2 ENVELOPES, SO THAT THE MSCP EXERCISER    ZZZ
;  4107  3      !       CAN PROCEED AS BEFORE THE DUP EXERCISER WAS CALLED.              ZZZ
;  4108  3      !                                                                        ZZZ
;  4109  3          DUP_FLAGS = .DUP_FLAGS OR SWP_DINT;      !SET DUP INIT FLAG          ZZZ
;  4110  3          INIT_TEST ();                            !REINIT CONTROLLER          ZZZ
;  4111  3          DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT); !CLR DUP INIT DLAG        ZZZ
;  4112  3                                                                               !ZZZ
;  4113  3          MX2 = -1;                                !ASSUME NO 2ND ENVELOPE     ZZZ
;  4114  3          IF (MX1 = GET_PKT (.CCTLR)) LSS 0        !TRY FOR 1ST ENVELOPE       ZZZ
;  4115  4              OR (.EOP_FLAG)                       !IF FAILURE                 ZZZ
;  4116  3          THEN     RETURN;                         !NO POINT TO GO ON          ZZZ
;  4117  3          IF (MX2 = GET_PKT (.CCTLR)) LSS 0        !TRY FOR 2ND ENVELOPE       ZZZ
;  4118  4              OR (.EOP_FLAG)                       !IF FAILURE                 ZZZ
;  4119  4          THEN     BEGIN                                                       !ZZZ
;  4120  4                   PUT_PKT (.MX1);                 !PUT 1ST BACK IN POOL       ZZZ
;  4121  4                   MX1 = -1;                       !INDICATE FAILURE           ZZZ
;  4122  4                   RETURN;                         !DONE                       ZZZ
;  4123  3                   END;                                                        !ZZZ
;  4124  3                                                                               !ZZZ
;  4125  3                   MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2); !CALC START ADDR    ZZZ
;  4126  3                   MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2); !OF BOTH ENVELOPES  ZZZ
;  4127  3                   GET_RANDOM ();                  !GET SET OF RANDOM NOS      ZZZ
;  4128  3                   QIO_UNIT ();                    !PUT RAND UNIT NO IN        ZZZ
;  4129  2                   END;                            !ENVELOPES                  ZZZ
;  4130  2                                                                              !ZZZ
;  4131  2      !       MSCP CODE STARTS HERE                                           ZZZ
;  4132  2      !                                                                       ZZZ
;  4133  2
;  4134  2      CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] =      !                          ZZZ
;  4135  2          .CST_ADDR [.CUOFF + OF_COUNT, D_COUNT)  1;  !DECR MSCP FUNCTION CNTR ZZZ
;  4136  2
;  4137  2          MAD2 [OPCODE] = 0;                                                  ! ASSUME 2ND PACKET NOT NEEDED
;  4138  2
;  4139  2          if (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED) and       ! IF "FORCED ERROR" SET IN LAST READ,
;  4140  2             (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and             !       REWRITE SAME BLOCK
;  4141  3             (.FORCED_ERROR)                                                  !
;  4142  2          then
;  4143  2              FUNC = OP_WRT
;  4144  2          else
;  4145  2
;  4146  2              if .CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql PROTECTED           ! IF UNIT IS PROTECTED
;  4147  2              then
;  4148  2                  FUNC = OP_RD                                                ! SET FUNCTION TO READ
```

```
;  4149  2              else
;  4150  2
;  4151  3                  if (.RANDOM [1] and 1)                          ! USE 2ND RANDOM NUMBER TO SELECT
;  4152  2                  then
;  4153  2                      FUNC = OP_RD                                !     READ
;  4154  2                  else
;  4155  2                      FUNC = OP_WRT;                              !     WRITE
;  4156  2
;  4157  2
;  4158  2
;  4159  3          IF (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED)   !I/O'S ARE CANCELLED WHEN CMD ZZZ
;  4160  2          THEN                                                        !RING IS FULL. DON'T LET THIS ZZZ
;  4161  2              IF .RW_BALANCE GEQU TOO_MANY_READS                      !UPSET THE BALANCE BETWEEN    ZZZ
;  4162  2              THEN                                                    !THE NUMBER OF READS AND      ZZZ
;  4163  2                  FUNC = OP_WRT;                                      !WRITES.                      ZZZ
;  4164  2
;  4165  2
;  4166  2          if (MAD1 [OPCODE] = .FUNC) eql OP_WRT                       ! LOAD CHOSEN OPCODE. IF WRITE
;  4167  2          then
;  4168  3              begin
;  4169  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                          ! NON-SEQUENTIAL COMMAND
;  4170  3
;  4171  4              if BIT_TST (SWP_FLAGS, SWF_CWC)                         ! IF CONTROLLER DOES WRITE-COMPARES
;  4172  3              then
;  4173  3                  MAD1 [MODIFY] = MD_CMP                              ! ADD COMPARE MODIFIER
;  4174  3              else
;  4175  3
;  4176  4                  if BIT_TST (SWP_FLAGS, SWF_HWC)                     ! IF HOST DOES WRITE-COMPARES
;  4177  3                  then
;  4178  4                      begin
;  4179  4                      MAD1 [MODIFY] = MD_EXP;                         ! SET WRITE AS AN EXPRESS REQUEST
;  4180  4                      MAD2 [OPCODE] = OP_RD;                          ! SET READ OPCODE INTO 2ND MSCP PACKET
;  4181  4                      MAD2 [MODIFY] = MD_EXP;                         ! SET READ AS AN EXPRESS REQUEST TOO
;  4182  4                      MAD2 [CMD_TYPE] = NON_SEQ_CMD;                  ! NON-SEQUENTIAL COMMAND
;  4183  3                      end;
;  4184  3              end
;  4185  2          else
;  4186  3              begin
;  4187  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                          ! NON-SEQUENTIAL COMMAND
;  4188  3
;  4189  4              if BIT_TST (SWP_FLAGS, SWF_CRC)                         ! IF READ-COMPARES   FUNCTION IS READ
;  4190  3              then
;  4191  3                  MAD1 [MODIFY] = MD_CMP;                             ! ADD COMPARE MODIFIER
;  4192  3
;  4193  2              end;
;  4194  2
;  4195  2          if .MAD2 [OPCODE] eql 0                                     ! IF NO OPCODE IN 2ND PACKET
;  4196  2          then
;  4197  3              begin
;  4198  3              PUT_PKT (.MX2);                                         ! RETURN 2ND PACKET TO POOL
;  4199  3              MX2 = 1;                                                ! MARK IT UNUSED
;  4200  2              end;
;  4201  2
```

```
;   4202  1          end;                                                    ! ROUTINE QIO FUNC


                                        .SBTTL  QIO.FUNC MULTI DRIVE TEST ROUTINES
000000  004137  000000G          QIO.FUNC::
                                        JSR     R1,$SAVE4                       ;                       4043
000004  013737  000000G 001250'         MOV     CUOFF,DUOFF                     ;                       4092
000012  013702  000000G                 MOV     CST.ADDR,R2                     ;                       4094
000016  013701  001250'                 MOV     DUOFF,R1
000022  010100                          MOV     R1,R0
000024  006300                          ASL     R0
000026  060200                          ADD     R2,R0
000030  005760  000022                  TST     22(R0)
000034  003146                          BGT     4$
000036  010100                          MOV     R1,R0                           ;                       4095
000040  006300                          ASL     R0
000042  060200                          ADD     R2,R0
000044  132710  000020                  BITB    #20,(R0)
000050  001540                          BEQ     4$
000052  010100                          MOV     R1,R0                           ;                       4096
000054  006300                          ASL     R0
000056  060200                          ADD     R2,R0
000060  005760  000020                  TST     20(R0)
000064  100532                          BMI     4$
000066  013746  000112'                 MOV     MX2,-(SP)                       ;                       4100
000072  004737  000000G                 JSR     PC,PUT.PKT
000076  012737  177777  000112         MOV     #-1,MX2                         ;                       4101
000104  004737  000000V                 JSR     PC,DUP                          ;                       4102
000110  013701  001250'                 MOV     DUOFF,R1                        ;                       4103
000114  006301                          ASL     R1
000116  063701  000000G                 ADD     CST.ADDR,R1
000122  013716  000000G                 MOV     DUPROUND,(SP)                   ;                       4104
000126  012746  000031                  MOV     #31,-(SP)
000132  004737  000000G                 JSR     PC,BL$MUL
000136  010061  000022                  MOV     R0,22(R1)
000142  052737  000002  000000G         BIS     #2,DUP.FLAGS                    ;                       4109
000150  004737  001134'                 JSR     PC,INIT.TEST                    ;                       4110
000154  042737  000002  000000G         BIC     #2,DUP.FLAGS                    ;                       4111
000162  012737  177777  000112'         MOV     #-1,MX2                         ;                       4113
000170  013716  000000G                 MOV     CCTLR,(SP)                      ;                       4114
000174  004737  000000G                 JSR     PC,GET.PKT
000200  010037  000110'                 MOV     R0,MX1
000204  002426                          BLT     2$
000206  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                       4115
000214  001022                          BNE     2$                              ;                       4043
000216  013716  000000G                 MOV     CCTLR,(SP)                      ;                       4117
000222  004737  000000G                 JSR     PC,GET.PKT
000226  010037  000112'                 MOV     R0,MX2
000232  002404                          BLT     1$
000234  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                       4118
000242  001411                          BEQ     3$
000244  013716  000110'          1$:     MOV     MX1,(SP)                        ;                       4120
000250  004737  000000G                 JSR     PC,PUT.PKT
```

M12


```
000254  012737  177777  000110'                MOV     #-1,MX1                                 ;                       4121
000262  022626                          2$:     CMP     (SP)+,(SP)+                             ;                       4122
000264  000207                                  RTS     PC                                      ;                       4119
000266  013716  000110'                  3$:     MOV     MX1,(SP)                                ;                       4125
000272  012746  000106                          MOV     #106.,-(SP)
000276  004737  000000G                         JSR     PC,BL$MUL
000302  062700  000000G                         ADD     #MSCP.PKT,R0
000306  010037  000114'                         MOV     R0,MAD1
000312  013716  000112'                         MOV     MX2,(SP)                                ;                       4126
000316  012746  000106                          MOV     #106.,-(SP)
000322  004737  000000G                         JSR     PC,BL$MUL
000326  062700  000000G                         ADD     #MSCP.PKT,R0
000332  010037  000116'                         MOV     R0,MAD2
000336  004737  011370'                         JSR     PC,GET.RANDOM                           ;                       4127
000342  004737  011756'                         JSR     PC,QIO.UNIT                             ;                       4128
000346  062706  000010                          ADD     #10,SP                                  ;                       4099
000352  013700  000000G                 4$:     MOV     CUOFF,R0                                ;                       4134
000356  006300                                  ASL     R0
000360  063700  000000G                         ADD     CST.ADDR,R0
000364  005360  000022                          DEC     22(R0)                                  ;                       4135
000370  013701  000116'                         MOV     MAD2,R1                                 ;                       4137
000374  012704  000022                          MOV     #22,R4
000400  060104                                  ADD     R1,R4
000402  105014                                  CLRB    (R4)
000404  013700  000000G                         MOV     CUOFF,R0                                ;                       4139
000410  006300                                  ASL     R0
000412  063700  000000G                         ADD     CST.ADDR,R0
000416  005003                                  CLR     R3
000420  005710                                  TST     (R0)
000422  100010                                  BPL     5$
000424  005203                                  INC     R3
000426  132710  000020                          BITB    #20,(R0)                                ;                       4140
000432  001404                                  BEQ     5$
000434  132737  000001  000000G                 BITB    #1,FORCED.ERROR                         ;                       4141
000442  001012                                  BNE     7$                                      ;                       4143
000444  032710  100000                  5$:     BIT     #100000,(R0)                            ;                       4146
000450  001404                                  BEQ     6$                                      ;                       4148
000452  032737  000001  000002G                 BIT     #1,RANDOM+2                             ;                       4151
000460  001403                                  BEQ     7$
000462  012702  000041                  6$:     MOV     #41,R2                                  ; *,FUNC                4153
000466  000402                                  BR      8$                                      ;                       4151
000470  012702  000042                  7$:     MOV     #42,R2                                  ; *,FUNC                4155
000474  006003                          8$:     ROR     R3                                      ;                       4159
000476  103006                                  BCC     9$
000500  023727  000106'  000002                 CMP     RW.BALANCE,#2                           ;                       4161
000506  103402                                  BLO     9$
000510  012702  000042                          MOV     #42,R2                                  ; *,FUNC                4163
000514  013700  000114'                  9$:     MOV     MAD1,R0                                 ;                       4166
000520  013703  000000G                         MOV     SWP.FLAGS,R3                            ;                       4171
000524  110260  000022                          MOVB    R2,22(R0)                               ; FUNC,*                4166
000530  020227  000042                          CMP     R2,#42                                  ; FUNC,*
000534  001025                                  BNE     10$
000536  112760  000002  000004                 MOVB    #2,4(R0)                                ;                       4169
```

N12


```
000544  032703  000020                          BIT     #20,R3                  ;                       4171
000550  001025                                  BNE     11$                     ;                       4173
000552  032703  000040                          BIT     #40,R3                  ;                       4176
000556  001425                                  BEQ     12$
000560  012760  100000  000024                  MOV     # 100000,24(R0)         ;                       4179
000566  112714  000041                          MOVB    #41,(R4)                ;                       4180
000572  012761  100000  000024                  MOV     #-100000,24(R1)         ;                       4181
000600  112761  000002  000004                  MOVB    #2,4(R1)                ;                       4182
000606  000411                                  BR      12$                     ;                       4166
000610  112760  000002  000004          10$:    MOVB    #2,4(R0)                ;                       4187
000616  032703  000004                          BIT     #4,R3                   ;                       4189
000622  001403                                  BEQ     12$
000624  012760  040000  000024          11$:    MOV     #40000,24(R0)           ;                       4191
000632  105714                          12$:    TSTB    (R4)                    ;                       4195
000634  001010                                  BNE     13$
000636  013746  000112'                          MOV     MX2,-(SP)               ;                       4198
000642  004737  000000G                          JSR     PC,PUT.PKT
000646  012737  177777  000112'                  MOV     #-1,MX2                 ;                       4199
000654  005726                                  TST     (SP)+                   ;                       4197
000656  000207                          13$:    RTS     PC                      ;                       4043

; Routine Size:  216 words,     Routine Base:  $CODE$ + 13362
; Maximum stack depth per invocation:  10 words
```

```
;  4203  1
;  4204  1     GLOBAL ROUTINE DUP : NOVALUE =                                          !ZZZ
;  4205  1     !+                                                                      !ZZZ
;  4206  1     !       THIS ROUTINE IS CALLED BY QIO_FUNC AFTER 25 * 'DUPROUND' RD/WTS.  ZZZ
;  4207  1     !       THIS EXERCISER WAS PLACED IN THE MIDDLE OF THE MSCP EXERCISER,    ZZZ
;  4208  1     !       SO COMMON INIT AND OTHER ROUTINES COULD BE USED.                  ZZZ
;  4209  1     !                                                                         ZZZ
;  4210  1     !       THE DUP EXERCISER WILL PERFORM EITHER READ-ONLY, OR WRITE-READ-   ZZZ
;  4211  1     !       COMPARE OPERATIONS ON THE DIAGNOSTIC BLOCKS (DBNS). IT WILL       ZZZ
;  4212  1     !       RECORD THE STATISTICS IN THE TALLY TABLES.                        ZZZ
;  4213  1     !                                                                         ZZZ
;  4214  1     !       THE PROGRAM USES CONTROLLER LOCAL PROGRAMS TO WRITE AND READ      ZZZ
;  4215  1     !       DBNS. WHEN WRITING TO THE DBNS, A ONE WORD PATTERN WILL BE        ZZZ
;  4216  1     !       SELECTED, AND REPLICATED THROUGH A 256 WORD BLOCK FOR DATA.       ZZZ
;  4217  1     !       THE ROUTINE WILL WRITE 'DUPROUND' NUMBER OF SEQUENTIAL DBN        ZZZ
;  4218  1     !       BLOCKS. IF THE CONTROLLER LOCAL PROGRAMS EXIST, AND THE OPERATOR  ZZZ
;  4219  1     !       SELECTS 'WRITE TO DIAGNOSTIC AREA', WRITE-READ-COMPARES WILL BE   ZZZ
;  4220  1     !       PERFORMED ON THE DBNS. OTHERWISE, READS WITH NO COMPARES WILL BE  ZZZ
;  4221  1     !       DONE. BAD BLOCKS FOUND IN THE COMPARISON TESTS WILL NOT BE LIST-  ZZZ
;  4222  1     !       ED IN THE RCT TABLES.                                             ZZZ
;  4223  1     !                                                                         ZZZ
;  4224  1     !       AFTER 'DUPROUND' NUMBER OF DBNS HAVE BEEN TESTED, THE ENVELOPES   ZZZ
;  4225  1     !       WILL BE REINITIATED, SO THAT THE MSCP EXERCISER CAN CONTINUE      ZZZ
;  4226  1     !       AS BEFORE.                                                        ZZZ
;  4227  1     !                                                                         ZZZ
;  4228  1     !       IMPLICIT INPUTS:                                                  ZZZ
;  4229  1     !                       CCTLR - CURRENT CONTROLLER NUMBER                 ZZZ
;  4230  1     !                       CST_ADDRS - CONTAINS THE CURRENT CONTROLLER       ZZZ
;  4231  1     !                                   STATUS TABLE                          ZZZ
;  4232  1     !                       CUOFF - CURRENT OFFSET IN CST TABLE FOR           ZZZ
;  4233  1     !                               PARTICULAR DRIVE                          ZZZ
;  4234  1     !                                                                         ZZZ
;  4235  1     !       IMPLICIT OUTPUTS:                                                 ZZZ
;  4236  1     !                       S_PATTERN - PATTERN BEING WRITTEN TO DBNS         ZZZ
;  4237  1     !                                                                         ZZZ
;  4238  1     !                                                                        !ZZZ
;  4239  1     !-                                                                       !ZZZ
;  4240  1
```

```
;    4241  1
;    4242  1
;    4243  2     BEGIN
;    4244  2     OWN
;    4245  2             TEMP : WORD;
;    4246  2
;    4247  2     !PRINTX (DBM110);
;    4248  2     !PRINTX (DER10);
;    4249  2
;    4250  2     until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;    4251  2           (.EOP_FLAG eql true) do                   ! Make sure all MSCP commands are completed
;    4252  3         begin
;    4253  3         BREAK;                                      ! BREAK FOR ACT
;    4254  3         PROC_RETPKT();                              ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;    4255  3         RP_INDX = .RP_INDX + 1;                     ! INCREMENT RP_INDX
;    4256  3         if .RP_INDX geq RP_CNT then (RP_INDX = 0;); ! MAKE SURE THE COUNTER DOES NOT GET TO BIG
;    4257  3         RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;    4258  2         end;
;    4259  2
;    4260  2
;    4261  2     S_PATTERN = .RANDOM [1];                                  !OTHER UNIT VARIABLES
;    4262  2
;    4263  2     IF (.CST_ADDR [.DUOFF + OF_DBN, D_DBN] + .dupround) GEQ 144     ! TEST TO SEE IF NEXT DBN'S TO LARGE
;    4264  2     THEN (CST_ADDR [.DUOFF + OF_DBN, D_DBN] = 0);             ! CIRCLE ARROUND IF DBN TO LARGE
;    4265  2
;    4266  2     DUPIDLE ();                                               ! DO A GET DUST STATUS TO FIND IF LOCAL DUP MEDIA
;    4267  2     IF .CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] EQL 1 THEN RETURN; ! IF DUP LOCAL MEDIA NOT THERE THEN RETURN
;    4268  2
;    4269  2     TEMP = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];
;    4270  2     INCR DBNCNT FROM (.TEMP + 1) TO (.TEMP + .duproundDO                  ! INCREMENT FROM RELATIVE DBN TO DBN + dupro
und
;    4271  3             BEGIN
;    4272  3             IF .CST_ADDR [.DUOFF + OF_DBN, DUPWRITE]                          ! IF WRITE FLAG SET IN CST TABLE THE
N WRITE DBN'S
;    4273  3             THEN
;    4274  4                     BEGIN
;    4275  4                     DUPIDLE ();                                   ! MAKE SURE THE CONTROLLER IS IN AN IDLE STA
TE
;    4276  4                     DUPWRTDBN ();                                 ! CALL ROUTINE TO HANDLE WRITTING ROUTINES
;    4277  3                     END;
;    4278  3
;    4279  3             DUPIDLE ();                                           ! MAKE SURE CONTROLLER IN IDLE STATE
;    4280  3             DUPREDDBN ();                                         ! CALL ROUTINE TO HANDLE READING DBN'S
;    4281  3
;    4282  3             CST_ADDR [.DUOFF + OF_DBN, D_DBN] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN] + 1; ! INCREMENT RELATIVE DBN COUNTER
;    4283  3             IF .CST_ADDR [.DUOFF + OF_DBN, D_DBN]  GTRU  MAX_DBN                        !BUT NOT MORE THAN MAX NUMBER
   ZZZ
;    4284  3             THEN                                                                        !IF BIGGER THAN MAX
   ZZZ
;    4285  3               CST_ADDR [.DUOFF + OF_DBN, D_DBN]  =  0;                                  !MAKE IT ZERO
   ZZZ
;    4286  3
;    4287  3
;    4288  3
;    4289  3             IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1                ! ERROR IN DUP REINITIALIZE
;    4290  3             THEN RETURN;                                                  ! AND RETURN
;    4291  2             END;
;    4292  1     END;
```

```
001276                                          .PSECT  $GGG$,  RO
001276                                  TEMP:   .BLKW   1


                                                .SBTTL  DUP MULTI-DRIVE TEST ROUTINES
014242                                          .PSECT  $CODE$,  RO

000000  004137  000000G                 DUP::   JSR     R1,$SAVE3                       ;                       4204
000004  013700  000000G                 1$:     MOV     RP.ADDR,RO                      ;                       4250
000010  023760  000000G 000004                 CMP     CRN.LOW,4(RO)
000016  001433                                 BEQ     3$
000020  123727  000000G 000001                 CMPB    EOP.FLAG,#1                     ;                       4251
000026  001427                                 BEQ     3$
000030  104422                                 TRAP    22                              ;                       4252
000032  004737  000000V                       JSR     PC,PROC.RETPKT                  ;                       4254
000036  005237  000000G                       INC     RP.INDX                         ;                       4255
000042  023727  000000G 000010                 CMP     RP.INDX,#10                     ;                       4256
000050  002402                                 BLT     2$
000052  005037  000000G                       CLR     RP.INDX
000056  013746  000000G                 2$:     MOV     RP.INDX,-(SP)                   ;                       4257
000062  012746  000054                       MOV     #54,-(SP)
000066  004737  000000G                       JSR     PC,BL$MUL
000072  062700  000000G                       ADD     #RETPKT,RO
000076  010037  000000G                       MOV     RO,RP.ADDR
000102  022626                               CMP     (SP)+,(SP)+
000104  000737                               BR      1$                              ;                       4252
000106  013737  000002G 000000G         3$:     MOV     RANDOM+2,S.PATTERN              ;                       4250
000114  013700  001250'                       MOV     DUOFF,RO                        ;                       4261
000120  006300                               ASL     RO                              ;                       4263
000122  063700  000000G                       ADD     CST.ADDR,RO
000126  005001                               CLR     R1
000130  156001  000020                       BISB    20(RO),R1
000134  063701  000000G                       ADD     DUPROUND,R1
000140  020127  000220                       CMP     R1,#220
000144  002402                               BLT     4$
000146  105060  000020                       CLRB    20(RO)
000152  004737  000000V                 4$:     JSR     PC,DUPIDLE                      ;                       4264
000156  013700  001250'                       MOV     DUOFF,RO                        ;                       4266
000162  006300                               ASL     RO                              ;                       4267
000164  063700  000000G                       ADD     CST.ADDR,RO
000170  005760  000020                       TST     20(RO)
000174  100462                               BMI     9$
000176  116037  000020  001276'               MOVB    20(RO),TEMP                     ;                       4269
000204  105037  001277'                       CLRB    TEMP+1
000210  013703  001276'                       MOV     TEMP,R3                         ;                       4270
000214  063703  000000G                       ADD     DUPROUND,R3
000220  013700  001250'                       MOV     DUOFF,RO                        ;                       4272
000224  006300                               ASL     RO
000226  063700  000000G                       ADD     CST.ADDR,RO
000232  010001                               MOV     RO,R1
000234  062701  000020                       ADD     #20,R1
```

```
000240  013702  001276'                    MOV     TEMP,R2                 ; *,DBNCNT                      4270
000244  000433                             BR      8$
000246  032711  010000          5$:       BIT     #10000,(R1)             ;                               4272
000252  001404                             BEQ     6$
000254  004737  000000V                    JSR     PC,DUPIDLE              ;                               4275
000260  004737  000000V                    JSR     PC,DUPWRTDBN            ;                               4276
000264  004737  000000V          6$:       JSR     PC,DUPIDLE              ;                               4279
000270  004737  000000V                    JSR     PC,DUPREDDBN            ;                               4280
000274  013700  001250'                    MOV     DUOFF,R0                ;                               4282
000300  006300                             ASL     R0
000302  063700  000000G                    ADD     CST.ADDR,R0
000306  010001                             MOV     R0,R1
000310  062701  000020                     ADD     #20,R1
000314  105211                             INCB    (R1)
000316  121127  000077                     CMPB    (R1),#77                ;                               4283
000322  101401                             BLOS    7$
000324  105011                             CLRB    (R1)                    ;                               4285
000326  032711  040000          7$:       BIT     #40000,(R1)             ;                               4289
000332  001003                             BNE     9$                      ;                               4290
000334  005202                 8$:         INC     R2                      ; DBNCNT                        4270
000336  020203                             CMP     R2,R3                   ; DBNCNT,*
000340  003742                             BLE     5$
000342  000207                 9$:         RTS     PC                      ;                               4204

; Routine Size:  114 words,    Routine Base:  $CODE$ + 14242
; Maximum stack depth per invocation:  7 words


;   4293  1
```

```
;  4294  1     GLOBAL ROUTINE DUPWRTDBN : NOVALUE =
;  4295  1
;  4296  1     !+
;  4297  1     ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;  4298  1     ! "WRTDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;  4299  1     ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM WRITES TO A DIAGONOSTIC BLOCK (DBN)
;  4300  1     ! THE WORD IN "S_PATTERN" IS WRITTEN TO THE 256 WORDS IN THE DBN.  IF AN ERROR OCCURS
;  4301  1     ! WHILE RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;  4302  1     ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;  4303  1     !
;  4304  1     !       IMPLICIT INPUTS:
;  4305  1     !               CST_ADDRS    CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;  4306  1     !               DUOFF        - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;  4307  1     !               S_PATTERN    - CONTAINS PATTERN WORD!-
;  4308  2     BEGIN
;  4309  2     LOCAL                                                   !                                                ZZZ
;  4310  2           TRYNUM : WORD,                                    !                                                ZZZ
;  4311  2           MAX_TRY_COUNT : word initial (9);                 ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR         ZZZ
;  4312  2                                                             !                                                ZZZ
;  4313  2     LABEL                                                   !                                                ZZZ
;  4314  2          DUP_WLOOP;                                         !START OF DUP WRITE RETRY LOOP                   ZZZ
;  4315  2
;  4316  2     !PRINTX (DER11);
;  4317  2     T_ADDR [T_DBN_WT] = .T_ADDR [T_DBN_WT] + 1;             ! INCREMENT # OF WRITES GIVEN
;  4318  2
;  4319  2     TRYNUM = 0;                                             !ZERO TRY COUNTER                                ZZZ
;  4320  2     DUP_WLOOP:                                              !LABEL FOR LOOP EXCAPE ON GOOD WRITE             ZZZ
;  4321  3     BEGIN                                                   !BEGIN DUP_WLOOP                                 ZZZ
;  4322  3     INCR TRIES FROM 1 TO 10 DO                              !START TRYING DUP WRITES                         ZZZ
;  4323  4     BEGIN                                                   !BEGIN LARGE DO LOOP                             ZZZ
;  4324  4                                                             !                                                ZZZ
;  4325  4
;  4326  4     MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                       ! PACKET SIZE                                                EXECUTE LOCAL PROGRAM WRT DB
N
;  4327  4     MSCP_PKT [.MX1, OPCODE] = OP_ELP;                       ! OPCODE = EXECUTE LOCAL PROGRAM
;  4328  4     MSCP_PKT [.MX1, L1] = %asci 'W';                        ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;  4329  4     MSCP_PKT [.MX1, L2] = %asci 'R';
;  4330  4     MSCP_PKT [.MX1, L3] = %asci 'T';
;  4331  4     MSCP_PKT [.MX1, L4] = %asci 'D';
;  4332  4     MSCP_PKT [.MX1, L5] = %asci 'B';
;  4333  4     MSCP_PKT [.MX1, L6] = %asci 'N';
;  4334  4     MSCP_PKT [.MX1, MODIFY] = 1;                            ! STANDALONE MODIFIER
;  4335  4     !ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                 ! CALL IT IMMEDIATE
;  4336  4     MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                       ! CALL ALL DUP CMDS SEQUENTIAL.                                  ZZZ
;  4337  4     DUPCOMMAND ();                                          ! SENDS AND RECEIVES THE COMMAND
;  4338  4
;  4339  4     IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  !status error
;  4340  4     THEN RETURN;                                            ! AND RETURN
;  4341  4
;  4342  5     DO (MX1 = GET_PKT (.CCTLR))
;  4343  4     UNTIL (.MX1 GEQ 0);                        ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4344  4
;  4345  4             MSCP_PKT [.MX1, MSGLEN] = SZ_REC;               ! PACKET SIZE                                                RECIEVE DATA
;  4346  4             MSCP_PKT [.MX1, OPCODE] = OP_RCD;               ! OPCODE = RECEIVE DATA
```

```
.   4347  4                MSCP_PKT [.MX1, BC_LO] = 80;                    ! BYTE COUNT TO BE TRANSFERED EQUALS 2 ***see pg 26 of DUP s
pec
;   4348  4                MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;   4349  4                MSCP_PKT [.MX1, MODIFY] = 0;                    !
;   4350  4    !ZZZ        MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;   4351  4                MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;               ! CALL ALL DUP CMDS SEQUENTIAL.                         ZZZ
;   4352  4                DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;   4353  4
;   4354  4                IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;   4355  4                   (.DUPPKT [DUPTYPE] NEQU 1) OR                !dup type error
;   4356  5                   (.DUPPKT [DUPMESG] NEQU 6)
;   4357  4                THEN
;   4358  5                        (D_FAIL = 1;                            !TELL HARD_ERROR IT WAS A DUP PROBLEM           ZZZ
;   4359  5                         HARD_ERROR ();                         !                                               ZZZ
;   4360  5                         D_FAIL = 0;                            !                                               ZZZ
;   4361  5                         CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;   4362  4                         RETURN;);                              ! NO POINT IN CONTINUING
;   4363  4
;   4364  5                DO (MX1 = GET_PKT (.CCTLR))
;   4365  4                UNTIL (.MX1 GEQ 0);                             ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;   4366  4
;   4367  4                MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;               ! PACKET SIZE                                   SEND DATA
;   4368  4                MSCP_PKT [.MX1, OPCODE] = OP_SDD;               ! OPCODE = SEND DATA
;   4369  4                MSCP_PKT [.MX1, BC_LO] = 6;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 6
;   4370  4                MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;   4371  4                DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; !LOAD UNIT NUMBER (RDRX)
;   4372  4                DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;   4373  4                DUPPKT [DUPBF2] = .S_PATTERN;                   ! LOAD PATTERN
;   4374  4                MSCP_PKT [.MX1, MODIFY] = 0;                    !
;   4375  4    !ZZZ        MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;   4376  4                MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;               ! CALL ALL DUP CMDS SEQUENTIAL.                         ZZZ
;   4377  4                DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;   4378  4
;   4379  4                IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  ! status error
;   4380  4                THEN RETURN;
;   4381  4
;   4382  5                DO (MX1 = GET_PKT (.CCTLR))
;   4383  4                UNTIL (.MX1 GEQ 0);                     ! TRY TO GET AN ENVELCPE. IF FAILURE LOOP PRG ERROR
;   4384  4
;   4385  4                MSCP_PKT [.MX1, MSGLEN] = SZ_REC;               ! PACKET SIZE                                   RECEIVE DATA
;   4386  4                MSCP_PKT [.MX1, OPCODE] = OP_RCD;               ! OPCODE = RECEIVE DATA
;   4387  4                MSCP_PKT [.MX1, BC_LO] = 4;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;   4388  4                MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;   4389  4                MSCP_PKT [.MX1, MODIFY] = 0;                    !
;   4390  4    !ZZZ        MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;   4391  4                MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;               ! CALL ALL DUP CMDS SEQUENTIAL.                         ZZZ
;   4392  4                DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;   4393  4
;   4394  4
;   4395  4                IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND                       ZZZ
;   4396  4                   (.DUPPKT [DUPTYPE] EQL 3) AND                !NO dup type error                              ZZZ
;   4397  4                   (.DUPPKT [DUPMESG] EQL 3) AND                !                                               ZZZ
;   4398  5                   (.DUPPKT [DUPBF1]  EQL 0)                    !AND A successful write code                    ZZZ
;   4399  5                                                                !                                               ZZZ
```

```
;    4400  4              THEN                                        !THEN                                                    ZZZ
;    4401  4                 LEAVE DUP_WLOOP                          !I/O OK. EXIT RETRY LOOP.                                ZZZ
;    4402  4              ELSE                                        !                                                        ZZZ
;    4403  5                 BEGIN                                    !                                                        ZZZ
;    4404  5                 TRYNUM = .TRYNUM + 1;                    !INCR ATTEMPT COUNT                                      ZZZ
;    4405  5                 IF .TRYNUM  EQL .MAX_TRY_COUNT           !IF IT FAILED ALL RETRIES, THEN                          ZZZ
;    4406  5                 THEN                                     !REPORT THE ERROR.                                       ZZZ
;    4407  6                         (D_FAIL = 1;                     !TELL HARD_ERROR IT WAS A DUP PROBLEM                    ZZZ
;    4408  6                          HARD_ERROR ();                  !                                                        ZZZ
;    4409  6                          D_FAIL = 0;                     !                                                        ZZZ
;    4410  6                          CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;   ! SET FLAG                                   ZZZ
;    4411  5                          RETURN;);                       ! NO POINT IN CONTINUING                                 ZZZ
;    4412  4                 END;                                     !                                                        ZZZ
;    4413  3          END;                                            !END LARGE DO LOOP                                       ZZZ
;    4414  3                                                          !                                                        ZZZ
;    4415  2          END;                                            !END DUP_WLOOP                                           ZZZ
;    4416  2
;    4417  2
;    4418  3          DO (MX1 = GET_PKT (.CCTLR))
;    4419  2          UNTIL (.MX1 GEQ 0);                             ! TRY TO GET AN ENVELOPE.
;    4420  2
;    4421  2          T_ADDR [T_BLK_WT] = .T_ADDR [T_BLK_WT] + 1;     !INCREMENT COUNTER IF A SUCCESS
;    4422  2
;    4423  1          END;


                                        .SBTTL  DUPWRTDBN MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          DUPWRTDBN::
                                        JSR     R1,$SAVE4                       ;                                        4294
000004  012704  000011                  MOV     #11,R4                          ; *,MAX.TRY.COUNT                        4308
000010  013700  000000G                 MOV     T.ADDR,R0                       ;                                        4317
000014  005260  000060                  INC     60(R0)
000020  005002                          CLR     R2                              ; TRYNUM                                 4319
000022  012703  000012                  MOV     #12,R3                          ; *,TRIES                                4322
000026  013746  000110'         1$:     MOV     MX1,-(SP)                       ;                                        4326
000032  012746  000106                  MOV     #106,-(SP)
000036  004737  000000G                 JSR     PC,BL$MUL
000042  012760  000022  000006G         MOV     #22,MSCP.PKT+6(R0)
000050  112760  000003  000022G         MOVB    #3,MSCP.PKT+22(R0)              ;                                        4327
000056  112760  000127  000026G         MOVB    #127,MSCP.PKT+26(R0)            ;                                        4328
000064  112760  000122  000027G         MOVB    #122,MSCP.PKT+27(R0)            ;                                        4329
000072  112760  000124  000030G         MOVB    #124,MSCP.PKT+30(R0)            ;                                        4330
000100  112760  000104  000031G         MOVB    #104,MSCP.PKT+31(R0)            ;                                        4331
000106  112760  000102  000032G         MOVB    #102,MSCP.PKT+32(R0)            ;                                        4332
000114  112760  000116  000033G         MOVB    #116,MSCP.PKT+33(R0)            ;                                        4333
000122  012760  000001  000024G         MOV     #1,MSCP.PKT+24(R0)              ;                                        4334
000130  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)            ;                                        4336
000136  004737  000000V                 JSR     PC,DUPCOMMAND                   ;                                        4337
000142  013700  001250'                 MOV     DUOFF,R0                        ;                                        4339
000146  006300                          ASL     R0
000150  063700  000000G                 ADD     CST.ADDR,R0
000154  032760  040000  000020         BIT     #40000,20(R0)
000162  001402                          BEQ     2$
```

```
000164  022626                          CMP     (SP)+,(SP)+                     ;
000166  000207                          RTS     PC                              ;                       4294
000170  013716  000000G         2$:     MOV     CCTLR,(SP)                      ;                       4340
000174  004737  000000G                 JSR     PC,GET.PKT                      ;                       4342
000200  010037  000110'                 MOV     R0,MX1
000204  002771                          BLT     2$
000206  010016                          MOV     R0,(SP)                         ;                       4343
000210  012746  000106                  MOV     #106,-(SP)                      ;                       4345
000214  004737  000000G                 JSR     PC,BL$MUL
000220  012760  000034  000006G         MOV     #34,MSCP.PKT+6(R0)
000226  112760  000005  000022G         MOVB    #5,MSCP.PKT+22(R0)              ;                       4346
000234  012760  000120  000026G         MOV     #120,MSCP.PKT+26(R0)            ;                       4347
000242  012760  000000G 000032G         MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       4348
000250  005060  000024G                 CLR     MSCP.PKT+24(R0)                 ;                       4349
000254  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)            ;                       4351
000262  004737  000000V                 JSR     PC,DUPCOMMAND                   ;                       4352
000266  013700  001250'                 MOV     DUOFF,R0                        ;                       4354
000272  006300                          ASL     R0
000274  063700  000000G                 ADD     CST.ADDR,R0
000300  032760  040000  000020         BIT     #40000,20(R0)
000306  001004                          BNE     3$
000310  023727  000000G 010006         CMP     DUPPKT,#10006                   ;                       4355
000316  001422                          BEQ     4$
000320  112737  000001  000000G 3$:     MOVB    #1,D.FAIL                       ;                       4358
000326  004737  000000V                 JSR     PC,HARD.ERROR                   ;                       4359
000332  105037  000000G                 CLRB    D.FAIL                          ;                       4360
000336  013700  001250'                 MOV     DUOFF,R0                        ;                       4361
000342  006300                          ASL     R0
000344  063700  000000G                 ADD     CST.ADDR,R0
000350  052760  040000  000020         BIS     #40000,20(R0)
000356  062706  000006                  ADD     #6,SP                           ;                       4362
000362  000207                          RTS     PC                              ;                       4358
000364  013716  000000G         4$:     MOV     CCTLR,(SP)                      ;                       4364
000370  004737  000000G                 JSR     PC,GET.PKT
000374  010037  000110'                 MOV     R0,MX1
000400  002771                          BLT     4$
000402  010016                          MOV     R0,(SP)                         ;                       4365
000404  012746  000106                  MOV     #106,-(SP)                      ;                       4367
000410  004737  000000G                 JSR     PC,BL$MUL
000414  012760  000034  000006G         MOV     #34,MSCP.PKT+6(R0)
000422  112760  000004  000022G         MOVB    #4,MSCP.PKT+22(R0)              ;                       4368
000430  012760  000006  000026G         MOV     #6,MSCP.PKT+26(R0)              ;                       4369
000436  012760  000000G 000032G         MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       4370
000444  013701  001250'                 MOV     DUOFF,R1                        ;                       4371
000450  006301                          ASL     R1
000452  063701  000000G                 ADD     CST.ADDR,R1
000456  111137  000000G                 MOVB    (R1),DUPPKT
000462  042737  177760  000000G         BIC     #177760,DUPPKT
000470  013701  001250'                 MOV     DUOFF,R1                        ;                       4372
000474  006301                          ASL     R1
000476  063701  000000G                 ADD     CST.ADDR,R1
000502  116137  000020  000002G         MOVB    20(R1),DUPPKT+2
000510  105037  000003G                 CLRB    DUPPKT+3
```

```
000514  013737  000000G 000004G          MOV     S.PATTERN,DUPPKT+4              ;                       4373
000522  005060  000024G                  CLR     MSCP.PKT+24(R0)                 ;                       4374
000526  142760  000360  000010G          BICB    #360,MSCP.PKT+10(R0)            ;                       4376
000534  004737  000000V                  JSR     PC,DUPCOMMAND                   ;                       4377
000540  013700  001250'                  MOV     DUOFF,R0                        ;                       4379
000544  006300                           ASL     R0
000546  063700  000000G                  ADD     CST.ADDR,R0
000552  032760  040000  000020          BIT     #40000,20(R0)
000560  001403                           BEQ     5$
000562  062706  000010                   ADD     #10,SP                          ;                       4294
000566  000207                           RTS     PC                              ;                       4380
000570  013716  000000G           5$:     MOV     CCTLR,(SP)                      ;                       4382
000574  004737  000000G                  JSR     PC,GET.PKT
000600  010037  000110'                  MOV     R0,MX1
000604  002771                           BLT     5$                              ;                       4383
000606  010016                           MOV     R0,(SP)                         ;                       4385
000610  012746  000106                   MOV     #106,-(SP)
000614  004737  000000G                  JSR     PC,BL$MUL
000620  012760  000034  000006G          MOV     #34,MSCP.PKT+6(R0)
000626  112760  000005  000022G          MOVB    #5,MSCP.PKT+22(R0)              ;                       4386
000634  012760  000004  000026G          MOV     #4,MSCP.PKT+26(R0)              ;                       4387
000642  012760  000000G 000032G          MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       4388
000650  005060  000024G                  CLR     MSCP.PKT+24(R0)                 ;                       4389
000654  142760  000360  000010G          BICB    #360,MSCP.PKT+10(R0)            ;                       4391
000662  004737  000000V                  JSR     PC,DUPCOMMAND                   ;                       4392
000666  013700  001250'                  MOV     DUOFF,R0                        ;                       4395
000672  006300                           ASL     R0
000674  063700  000000G                  ADD     CST.ADDR,R0
000700  032760  040000  000020          BIT     #40000,20(R0)
000706  001012                           BNE     6$
000710  023727  000000G 030003          CMP     DUPPKT,#30003                   ;                       4396
000716  001006                           BNE     6$
000720  005737  000002G                  TST     DUPPKT+2                        ;                       4398
000724  001003                           BNE     6$
000726  062706  000012                   ADD     #12,SP                          ;                       4401
000732  000433                           BR      8$
000734  005202                   6$:     INC     R2                              ; TRYNUM                4404
000736  020204                           CMP     R2,R4                           ; TRYNUM,MAX.TRY.COUNT  4405
000740  001022                           BNE     7$
000742  112737  000001  000000G          MOVB    #1,D.FAIL                       ;                       4407
000750  004737  000000V                  JSR     PC,HARD.ERROR                   ;                       4408
000754  105037  000000G                  CLRB    D.FAIL                          ;                       4409
000760  013700  001250'                  MOV     DUOFF,R0                        ;                       4410
000764  006300                           ASL     R0
000766  063700  000000G                  ADD     CST.ADDR,R0
000772  052760  040000  000020          BIS     #40000,20(R0)
001000  062706  000012                   ADD     #12,SP                          ;                       4411
001004  000207                           RTS     PC                              ;                       4407
001006  062706  000012           7$:     ADD     #12,SP                          ;                       4323
001012  005303                           DEC     R3                              ; TRIES                 4322
001014  001402                           BEQ     8$
001016  000137  014634'                  JMP     1$
001022  013746  000000G           8$:     MOV     CCTLR, (SP)                     ;                       4418
```

```
001026  004737  000000G                 JSR     PC,GET.PKT
001032  010037  000110'                 MOV     R0,MX1
001036  005726                          TST     (SP)+
001040  005700                          TST     R0                      ; MX1                   4419
001042  002767                          BLT     8$                                              4421
001044  013700  000000G                 MOV     T.ADDR,R0               ;                       4294
001050  005260  000056                  INC     56(R0)
001054  000207                          RTS     PC                      ;

; Routine Size:  279 words,     Routine Base:  $CODE$ + 14606
; Maximum stack depth per invocation:  11 words
```

```
;  4424  1     GLOBAL ROUTINE DUPREDDBN : NOVALUE =
;  4425  1
;  4426  1     !+
;  4427  1     ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;  4428  1     ! "REDDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;  4429  1     ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM READS A DIAGONOSTIC BLOCK (DBN)
;  4430  1     ! AND PLACES IT IN THE DUP BUFFER CALLED "DUPPKT".  IF AN ERROR OCCURS WHILE
;  4431  1     ! RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;  4432  1     ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;  4433  1     !
;  4434  1     !
;  4435  1     !       IMPLICIT INPUTS:
;  4436  1     !               CST_ADDR    - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;  4437  1     !               DUOFF       - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;  4438  1     !
;  4439  2     BEGIN
;  4440  2     LOCAL                                                  !                                        ZZZ
;  4441  2           TRYNUM : WORD,                                   !                                        ZZZ
;  4442  2           MAX_TRY_COUNT : word initial (9);                ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR ZZZ
;  4443  2                                                            !                                        ZZZ
;  4444  2     LABEL                                                  !                                        ZZZ
;  4445  2          DUP_RLOOP;                                        !START OR DUP READ RETRY LOOP            ZZZ
;  4446  2
;  4447  2
;  4448  2     !PRINTX (DER12);
;  4449  2     T_ADDR [T_DBN_RD] = .T_ADDR [T_DBN_RD] + 1;            ! INCREMENT # OF READS GIVEN
;  4450  2
;  4451  2     TRYNUM = 0;                                            !ZERO TRY COUNTER                        ZZZ
;  4452  2     DUP_RLOOP:                                             !LABEL FOR LOOP EXCAPE ON GOOD READ      ZZZ
;  4453  3     BEGIN                                                  !BEGIN DUP_RLOOP                         ZZZ
;  4454  3     INCR TRIES FROM 1 TO 10 DO                             !START TRYING DUP READS                  ZZZ
;  4455  4     BEGIN                                                  !BEGIN LARGE DO LOOP                     ZZZ
;  4456  4                                                            !                                        ZZZ
;  4457  4
;  4458  4          MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                 ! PACKET SIZE                                 EXECUTE REDDBN PROGRAM
;  4459  4          MSCP_PKT [.MX1, OPCODE] = OP_ELP;                 ! OPCODE = EXECUTE LOCAL PROGRAM
;  4460  4          MSCP_PKT [.MX1, L1] = %asci 'R';                  ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;  4461  4          MSCP_PKT [.MX1, L2] = %asci 'E';
;  4462  4          MSCP_PKT [.MX1, L3] = %asci 'D';
;  4463  4          MSCP_PKT [.MX1, L4] = %asci 'D';
;  4464  4          MSCP_PKT [.MX1, L5] = %asci 'B';
;  4465  4          MSCP_PKT [.MX1, L6] = %asci 'N';
;  4466  4          MSCP_PKT [.MX1, MODIFY] = 1;                      ! STANDALONE MODIFIER
;  4467  4          !ZZZ  MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                  ! CALL IT IMMEDIATE
;  4468  4          MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                 ! CALL ALL DUP CMDS SEQUENTIAL.                    ZZZ
;  4469  4          DUPCOMMAND ();                                    ! SENDS AND RECEIVES THE COMMAND
;  4470  4
;  4471  4          IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1            !status error
;  4472  4          THEN RETURN;
;  4473  4
;  4474  5          DO (MX1 = GET_PKT (.CCTLR))
;  4475  4          UNTIL (.MX1 GEQ 0);                    ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4476  4
```

```
;  4477  4            MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                ! PACKET SIZE                                   RECIEVE DATA
;  4478  4            MSCP_PKT [.MX1, OPCODE] = OP_RCD;                ! OPCODE = RECEIVE DATA
;  4479  4            MSCP_PKT [.MX1, BC_LO] = 80;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 2 ******see pg 26 DUP op
ec
;  4480  4            MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4481  4            MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4482  4     !ZZZ   MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4483  4            MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.                      ZZZ
;  4484  4            DUPCOMMAND ();                                   ! SENDS AND RECEIVES  HE COMMAND
;  4485  4
;  4486  4            IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;  4487  4               (.DUPPKT [DUPTYPE] NEQU 1) OR                 !dup type error
;  4488  5               (.DUPPKT [DUPMESG] NEQU 5)
;  4489  4            THEN
;  4490  5                    (D_FAIL = 1;                             !TELL HARD_ERROR IT WAS A DUP PROBLEM           ZZZ
;  4491  5                     HARD_ERROR ();
.  4492  5                     D_FAIL = 0;                             !                                               ZZZ
;  4493  5                     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;  4494  4                     RETURN;);                               ! NO POINT IN CONTINUING
;  4495  4
;  4496  5            DO (MX1 = GET_PKT (.CCTLR))
;  4497  4            UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4498  4
;  4499  4            MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;                ! PACKET SIZE                                   SEND DATA
;  4500  4            MSCP_PKT [.MX1, OPCODE] = OP_SDD;                ! OPCODE = SEND DATA
;  4501  4            MSCP_PKT [.MX1, BC_LO] = 4;                      ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;  4502  4            MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4503  4            DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; ! LOAD UNIT NUMBER (RDRX)
;  4504  4            DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;  4505  4            MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4506  4     !ZZZ   MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4507  4            MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.                      ZZZ
;  4508  4            DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4509  4
;  4510  4             IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1          !status error
;  4511  4             THEN RETURN;
;  4512  4
;  4513  5            DO (MX1 = GET_PKT (.CCTLR))
;  4514  4            UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4515  4
;  4516  4            MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                ! PACKET SIZE                                   RECEIVE DATA
;  4517  4            MSCP_PKT [.MX1, OPCODE] = OP_RCD;                ! OPCODE = GET DUST STATUS
;  4518  4            MSCP_PKT [.MX1, BC_LO] = 514;                    ! BYTE COUNT TO BE TRANSFERED EQUALS 512
;  4519  4            MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4520  4            MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4521  4     !ZZZ   MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4522  4            MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.                      ZZZ
;  4523  4            DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4524  4
;  4525  4            IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND     !IF status OK AND                   ZZZ
;  4526  4               (.DUPPKT [DUPTYPE] EQL 6) AND                 !NO dup type error                      ZZZ
;  4527  5               (.DUPPKT [DUPMESG] EQL 2)                     !                                       ZZZ
;  4528  4            THEN                                             !THEN                                   ZZZ
;  4529  4               LEAVE DUP_RLOOP                               !I/O OK. EXIT RETRY LOOP.               ZZZ
```

```
;    4530  4          ELSE                                                    !                                                       ZZZ
;    4531  5             BEGIN                                                !                                                       ZZZ
;    4532  5             TRYNUM = .TRYNUM + 1;                                !INCR ATTEMPT COUNT                                     ZZZ
;    4533  5             IF .TRYNUM  EQL .MAX_TRY_COUNT                       !IF IT FAILED ALL RETRIES, THEN                         ZZZ
;    4534  5             THEN                                                 !REPORT THE ERROR.                                      ZZZ
;    4535  6                     (D_FAIL = 1;                                 !TELL HARD_ERROR IT WAS A DUP PROBLEM                   ZZZ
;    4536  6                      HARD_ERROR ();                              !                                                       ZZZ
;    4537  6                      D_FAIL = 0;                                 !                                                       ZZZ
;    4538  6                      CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;   ! SET FLAG                                              ZZZ
;    4539  5                      RETURN;);                                   ! NO POINT IN CONTINUING                                ZZZ
;    4540  4             END;                                                 !                                                       ZZZ
;    4541  3          END;                                                    !END LARGE DO LOOP                                      ZZZ
;    4542  3                                                                  !                                                       ZZZ
;    4543  2       END;                                                       !END DUP_RLOOP                                          ZZZ
;    4544  2
;    4545  3       DO (MX1 = GET_PKT (.CCTLR))
;    4546  2       UNTIL (.MX1 GEQ 0);                    ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4547  2
;    4548  2       T_ADDR [T_BLK_RD] = .T_ADDR [T_BLK_RD] + 1;              !IF DUP NO ERROR THEN INCREMENT COUNTER
;    4549  2
;    4550  1       END;


                                        .SBTTL  DUPREDDBN MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          DUPREDDBN::
                                        JSR     R1,$SAVE4                       ;                                               4424
000004  012704  000011                  MOV     #11,R4                          ; *,MAX.TRY.COUNT                               4439
000010  013700  000000G                 MOV     T.ADDR,R0                       ;                                               4449
000014  005260  000064                  INC     64(R0)
000020  005002                          CLR     R2                              ; TRYNUM                                        4451
000022  012703  000012                  MOV     #12,R3                          ; *,TRIES                                       4454
000026  013746  000110'         1$:     MOV     MX1,-(SP)                       ;                                               4458
000032  012746  000106                  MOV     #106,-(SP)
000036  004737  000000G                 JSR     PC,BL$MUL
000042  012760  000022  000006G         MOV     #22,MSCP.PKT+6(R0)
000050  112760  000003  000022G         MOVB    #3,MSCP.PKT+22(R0)              ;                                               4459
000056  112760  000122  000026G         MOVB    #122,MSCP.PKT+26(R0)            ;                                               4460
000064  112760  000105  000027G         MOVB    #105,MSCP.PKT+27(R0)            ;                                               4461
000072  112760  000104  000030G         MOVB    #104,MSCP.PKT+30(R0)            ;                                               4462
000100  112760  000104  000031G         MOVB    #104,MSCP.PKT+31(R0)            ;                                               4463
000106  112760  000102  000032G         MOVB    #102,MSCP.PKT+32(R0)            ;                                               4464
000114  112760  000116  000033G         MOVB    #116,MSCP.PKT+33(R0)            ;                                               4465
000122  012760  000001  000024G         MOV     #1,MSCP.PKT+24(R0)              ;                                               4466
000130  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)            ;                                               4468
000136  004737  000000V                 JSR     PC,DUPCOMMAND                   ;                                               4469
000142  013700  001250'                 MOV     DUOFF,R0                        ;                                               4471
000146  006300                          ASL     R0
000150  063700  000000G                 ADD     CST.ADDR,R0
000154  032760  040000  000020         BIT     #40000,20(R0)
000162  001402                          BEQ     2$
000164  022626                          CMP     (SP)+,(SP)+                     ;                                               4424
000166  000207                          RTS     PC                              ;                                               4472
000170  013716  000000G         2$:     MOV     CCTLR,(SP)                      ;                                               4474
```

```
000174  004737  000000G                        JSR     PC,GET.PKT
000200  010037  000110'                        MOV     R0,MX1
000204  002771                                 BLT     2$
000206  010016                                 MOV     R0,(SP)                 ;4475
000210  012746  000106                         MOV     #106,-(SP)              ;4477
000214  004737  000000G                        JSR     PC,BL$MUL
000220  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000226  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)      ;4478
000234  012760  000120  000026G                MOV     #120,MSCP.PKT+26(R0)    ;4479
000242  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0) ;4480
000250  005060  000024G                        CLR     MSCP.PKT+24(R0)         ;4481
000254  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)    ;4483
000262  004737  000000V                        JSR     PC,DUPCOMMAND           ;4484
000266  013700  001250'                        MOV     DUOFF,R0                ;4486
000272  006300                                 ASL     R0
000274  063700  000000G                        ADD     CST.ADDR,R0
000300  032760  040000  000020                BIT     #40000,20(R0)
000306  001004                                 BNE     3$
000310  023727  000000G 010005                CMP     DUPPKT,#10005           ;4487
000316  001422                                 BEQ     4$
000320  112737  000001  000000G       3$:     MOVB    #1,D.FAIL               ;4490
000326  004737  000000V                        JSR     PC,HARD.ERROR           ;4491
000332  105037  000000G                        CLRB    D.FAIL                  ;4492
000336  013700  001250'                        MOV     DUOFF,R0                ;4493
000342  006300                                 ASL     R0
000344  063700  000000G                        ADD     CST.ADDR,R0
000350  052760  040000  000020                BIS     #40000,20(R0)
000356  062706  000006                         ADD     #6,SP                   ;4494
000362  000207                                 RTS     PC                      ;4490
000364  013716  000000G       4$:     MOV     CCTLR,(SP)              ;4496
000370  004737  000000G                        JSR     PC,GET.PKT
000374  010037  000110'                        MOV     R0,MX1
000400  002771                                 BLT     4$
000402  010016                                 MOV     R0,(SP)                 ;4497
000404  012746  000106                         MOV     #106,-(SP)              ;4499
000410  004737  000000G                        JSR     PC,BL$MUL
000414  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000422  112760  000004  000022G                MOVB    #4,MSCP.PKT+22(R0)      ;4500
000430  012760  000004  000026G                MOV     #4,MSCP.PKT+26(R0)      ;4501
000436  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0) ;4502
000444  013701  001250'                        MOV     DUOFF,R1                ;4503
000450  006301                                 ASL     R1
000452  063701  000000G                        ADD     CST.ADDR,R1
000456  111137  000000G                        MOVB    (R1),DUPPKT
000462  042737  177760  000000G               BIC     #177760,DUPPKT
000470  013701  001250'                        MOV     DUOFF,R1                ;4504
000474  006301                                 ASL     R1
000476  063701  000000G                        ADD     CST.ADDR,R1
000502  116137  000020  000002G                MOVB    20(R1),DUPPKT+2
000510  105037  000003G                        CLRB    DUPPKT+3
000514  005060  000024G                        CLR     MSCP.PKT+24(R0)         ;4505
000520  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)    ;4507
000526  004737  000000V                        JSR     PC,DUPCOMMAND           ;4508
```

```
000532  013700  001250'                        MOV     DUOFF,R0
000536  006300                                  ASL     R0                                      ;                       4510
000540  063700  000000G                        ADD     CST.ADDR,R0
000544  032760  040000  000020                 BIT     #40000,20(R0)
000552  001403                                  BEQ     5$
000554  062706  000010                         ADD     #10,SP
000560  000207                                  RTS     PC                                      ;                       4424
000562  013716  000000G                5$:     MOV     CCTLR,(SP)                              ;                       4511
000566  004737  000000G                        JSR     PC,GET.PKT                              ;                       4513
000572  010037  000110'                        MOV     R0,MX1
000576  002771                                  BLT     5$
000600  010016                                  MOV     R0,(SP)                                 ;                       4514
000602  012746  000106                         MOV     #106,-(SP)                              ;                       4516
000606  004737  000000G                        JSR     PC,BL$MUL
000612  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000620  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)
000626  012760  001002  000026G                MOV     #1002,MSCP.PKT+26(R0)                   ;                       4517
000634  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0)                 ;                       4518
000642  005060  000024G                        CLR     MSCP.PKT+24(R0)                         ;                       4519
000646  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)                    ;                       4520
000654  004737  000000V                        JSR     PC,DUPCOMMAND                           ;                       4522
000660  013700  001250'                        MOV     DUOFF,R0                                ;                       4523
000664  006300                                  ASL     R0                                      ;                       4525
000666  063700  000000G                        ADD     CST.ADDR,R0
000672  032760  040000  000020                 BIT     #40000,20(R0)
000700  001007                                  BNE     6$
000702  023727  000000G 060002                 CMP     DUPPKT,#60002                           ;                       4526
000710  001003                                  BNE     6$
000712  062706  000012                         ADD     #12,SP                                  ;                       4529
000716  000433                                  BR      8$
000720  005202                6$:     INC     R2                                      ; TRYNUM                        4532
000722  020204                                  CMP     R2,R4                                   ; TRYNUM,MAX.TRY.COUNT  4533
000724  001022                                  BNE     7$
000726  112737  000001  000000G                MOVB    #1,D.FAIL                               ;                       4535
000734  004737  000000V                        JSR     PC,HARD.ERROR                           ;                       4536
000740  105037  000000G                        CLRB    D.FAIL                                  ;                       4537
000744  013700  001250'                        MOV     DUOFF,R0                                ;                       4538
000750  006300                                  ASL     R0
000752  063700  000000G                        ADD     CST.ADDR,R0
000756  052760  040000  000020                 BIS     #40000,20(R0)
000764  062706  000012                         ADD     #12,SP                                  ;                       4539
000770  000207                                  RTS     PC                                      ;                       4535
000772  062706  000012                7$:     ADD     #12,SP                                  ;                       4455
000776  005303                                  DEC     R3                                      ; TRIES                 4454
001000  001402                                  BEQ     8$
001002  000137  015712'                        JMP     1$
001006  013746  000000G                8$:     MOV     CCTLR, (SP)                             ;                       4545
001012  004737  000000G                        JSR     PC,GET.PKT
001016  010037  000110'                        MOV     R0,MX1
001022  005726                                  TST     (SP)+
001024  005700                                  TST     R0                                      ; MX1                   4546
001026  002767                                  BLT     8$
001030  013700  000000G                        MOV     T.ADDR,R0                               ;                       4548
```

```
ZRQAM3          RD/RX EXERCISER                                 4-Apr-1985 13:23:31    VAX-11 Bliss-16 V4.1-582              SEQ 0379
V02.2           MULTI-DRIVE TEST ROUTINES                       2-Apr-1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;19  Page 124
                                                                                                                              (29)

001034  005260  000062                          INC     62(R0)
001040  00020/                                  RTS     PC                                                                    4424

; Routine Size:  273 words,     Routine Base:  $CODE$ + 15664
; Maximum stack depth per invocation:  11 words


;   4551  1
```

```
;    4552  1
;    4553  1      GLOBAL ROUTINE DUPCOMMAND : NOVALUE =
;    4554  1
;    4555  1      !+
;    4556  1      ! THIS ROUTINE IS CALLED BY DUP TO PROCESS COMMANDS.
;    4557  1      ! THE COMMAND ENVOLOPES ARE FILLED IN DUP ROUTINES IN THE "MX1" INDEX.
;    4558  1      ! WITH THE INDEX THIS ROUTINE SENDS THE COMMAND, WAITS FOR A
;    4559  1      ! RESPONSES AND THEN PROCESSES THE RETURN PACKET.
;    4560  1      !-
;    4561  2      BEGIN
;    4562  2      !PRINTX (DER13);
;    4563  2
;    4564  2      MSCP_PKT [.MX1, CREDITS] = 0;                   ! DUP DOES NOT USE THE CREDIT SYSTEM
;    4565  2      MSCP_PKT [.MX1, CONNID] = CID_DUP;              ! MAKE PACKAGE EQUAL A DUP COMMAND
;    4566  2      MSCP_PKT [.MX1, DK_NUM] = 0;                    ! DISK NUMBER (NOT APPLICABLE)
;    4567  2
;    4568  2      IF SEND (.MX1) EQLU FAILURE                     ! ATTEMPT SEND; IF CTLR IS OFFLINE
;    4569  2      THEN
;    4570  3              BEGIN
;    4571  3              PUT_PKT (.MX1);
;    4572  3              MX1 = -1;                               ! RETURN ENVELOPE TO POOL
;    4573  3              CST_ADDR [.DUOFF + OF DBN, DUPERROR] = 1;
;    4574  3      !       PRINTF (DBM112);                        ! "DUP: PKT NOT AVAILABLE"  ZZZ
;    4575  3              END
;    4576  3
;    4577  2      ELSE
;    4578  2              do
;    4579  3                      begin
;    4580  3                      BREAK;                                  ! BREAK FOR ACT
;    4581  3                      PROC_RETPKT ();                                 ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;    4582  3                      end
;    4583  2              until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;    4584  2              (.EOP_FLAG eql true);                           ! or end of pass caused by error
;    4585  1      END;


                                        .SBTTL  DUPCOMMAND MULTI-DRIVE TEST ROUTINES
000000  013746  000110'            DUPCOMMAND::
                                        MOV     MX1,-(SP)                       ;                               4564
000004  012746  000106                  MOV     #106,-(SP)
000010  004737  000000G                 JSR     PC,BL$MUL
000014  142760  000017  000010G         BICB    #17,MSCP.PKT+10(R0)
000022  112760  000002  000011G         MOVB    #2,MSCP.PKT+11(R0)              ;                               4565
000030  005060  000016G                 CLR     MSCP.PKT+16(R0)                 ;                               4566
000034  013716  000110'                 MOV     MX1,(SP)                        ;                               4568
000040  004737  000000G                 JSR     PC,SEND
000044  005700                          TST     R0
000046  001020                          BNE     1$
000050  013716  000110'                 MOV     MX1,(SP)                        ;                               4571
000054  004737  000000G                 JSR     PC,PUT.PKT
000060  012737  177777  000110'         MOV     #-1,MX1                         ;                               4572
000066  013700  001250'                 MOV     DUOFF,R0                        ;                               4573
000072  006300                          ASL     R0
```

```
ZRQAM3          RD/RX EXERCISER                              4-Apr-1985 13:23:31    VAX-11 Bliss-16 V4.1-582              SEQ 0381
V02.2           MULTI-DRIVE TEST ROUTINES                    2-Apr 1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAG0.BL2;19  Page 126
                                                                                                                          (30)

000074  063700  000000G                          ADD     CST.ADDR,R0
000100  052760  040000  000020                   BIS     #40000,20(R0)
000106  000415                                   BR      2$                                                          4568
000110  104422                          1$:      TRAP    22                                                          4579
000112  004737  000000V                          JSR     PC,PROC.RETPKT                                              4581
000116  013700  000000G                          MOV     RP.ADDR,R0                                                  4583
000122  023760  000000G 000004                   CMP     CRN.LOW,4(R0)
000130  001404                                   BEQ     2$
000132  123727  000000G 000001                   CMPB    EOP.FLAG,#1                                                 4584
000140  001363                                   BNE     1$
000142  022626                          2$:      CMP     (SP)+,(SP)+                                                 4561
000144  000207                                   RTS     PC                                                          4553

; Routine Size:  51 words.      Routine Base:  $CODE$ + 16726
; Maximum stack depth per invocation:  4 words
```

```
;    4586  1
;    4587  1      GLOBAL ROUTINE DUPIDLE : NOVALUE =
;    4588  1      !+
;    4589  1      ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO INSURE THAT THE CONTROLLER
;    4590  1      ! IS NOT IN A ACTIVE STATE. IF CALLED AND THE CONTROLLER IS IN AN ACTIVE
;    4591  1      ! STATE THE CONTROLLER WILL GIVE AN ABORT COMMAND WHICH SHOULD KILL THE
;    4592  1      ! CURRENT JOB OR LOCAL PROGRAM.
;    4593  1      !-
;    4594  2      BEGIN
;    4595  2      CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 0;               !CLEAR DUP ERROR FLAG;
;    4596  2
;    4597  2      MSCP_PKT [.MX1, MSGLEN] = SZ_GDS;              ! PACKET SIZE                                  GET DUST STATUS
;    4598  2      MSCP_PKT [.MX1, OPCODE] = OP_GDS;              ! OPCODE = GET DUST STATUS
;    4599  2      MSCP_PKT [.MX1, MODIFY] = 0;                   !
;    4600  2      !ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM CMD;        ! CALL IT IMMEDIATE
;    4601  2      MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;              ! CALL ALL DUP CMDS SEQUENTIAL.                    ZZZ
;    4602  2      DUPCOMMAND ();                                 ! SENDS AND RECEIVES THE COMMAND
;    4603  2                                                     ! GDS ONLY RETURNS SUCCESS or it don't return
;    4604  2
;    4605  3      DO (MX1 = GET_PKT (.CCTLR))
;    4606  2      UNTIL (.MX1 GEQ 0);                            ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;    4607  2
;    4608  2      if .CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] neq IDLE       ! if not in idle state then abort the program
;    4609  2      then
;    4610  3          begin
;    4611  3          MSCP_PKT [.MX1, MSGLEN] = SZ_ABT;                   ! PACKET SIZE                          ABORT CMD
;    4612  3          MSCP_PKT [.MX1, OPCODE] = OP_ABT;                   ! OPCODE = ABORT PROGRAM
;    4613  3          MSCP_PKT [.MX1, MODIFY] = 0;                        !
;    4614  3      !ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                 ! CALL IT IMMEDIATE
;    4615  3          MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                   ! CALL ALL DUP CMDS SEQUENTIAL.            ZZZ
;    4616  3          DUPCOMMAND ();                                      ! SENDS AND RECEIVES THE COMMAND
;    4617  3                                                              !ONLY ERROR IS already in idle state
;    4618  4          DO (MX1 = GET_PKT (.CCTLR))
;    4619  3          UNTIL (.MX1 GEQ 0);                            ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;    4620  2          end;
;    4621  1      end;


                                                .SBTTL  DUPIDLE MULTI-DRIVE TEST ROUTINES
000000  010146                            DUPIDLE::
                                                MOV     R1,-(SP)                        ;                       4587
000002  013700  001250'                         MOV     DUOFF,R0                        ;                       4595
000006  006300                                  ASL     R0
000010  063700  000000G                         ADD     CST.ADDR,R0
000014  042760  040000  000020                  BIC     #40000,20(R0)
000022  013746  000110'                         MOV     MX1,-(SP)                       ;                       4597
000026  012746  000106                          MOV     #106,-(SP)
000032  004737  000000G                         JSR     PC,BL$MUL
000036  012760  000014  000006G                 MOV     #14,MSCP.PKT+6(R0)
000044  112760  000001  000022G                 MOVB    #1,MSCP.PKT+22(R0)              ;                       4598
000052  005060  000024G                         CLR     MSCP.PKT+24(R0)                 ;                       4599
000056  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)            ;                       4601
000064  004737  016726'                         JSR     PC,DUPCOMMAND                   ;                       4602
```

```
000070  013716  000000G                1$:     MOV     CCTLR,(SP)                      ;                       4605
000074  004737  000000G                        JSR     PC,GET.PKT
000100  010037  000110'                        MOV     R0,MX1
000104  010001                                MOV     R0,R1                           ; MX1,*                 4606
000106  002770                                BLT     1$
000110  013700  001250'                        MOV     DUOFF,R0                        ;                       4608
000114  006300                                ASL     R0
000116  063700  000000G                        ADD     CST.ADDR,R0
000122  032760  020000  000020                BIT     #20000,20(R0)
000130  001432                                BEQ     3$
000132  010116                                MOV     R1,(SP)                         ;                       4611
000134  012746  000106                        MOV     #106,-(SP)
000140  004737  000000G                        JSR     PC,BL$MUL
000144  012760  000014  000006G               MOV     #14,MSCP.PKT+6(R0)
000152  112760  000006  000022G               MOVB    #6,MSCP.PKT+22(R0)              ;                       4612
000160  005060  000024G                       CLR     MSCP.PKT+24(R0)                 ;                       4613
000164  142760  000360  000010G               BICB    #360,MSCP.PKT+10(R0)            ;                       4615
000172  004737  016726'                        JSR     PC,DUPCOMMAND                   ;                       4616
000176  013716  000000G                2$:     MOV     CCTLR,(SP)                      ;                       4618
000202  004737  000000G                        JSR     PC,GET.PKT
000206  010037  000110'                        MOV     R0,MX1
000212  002771                                BLT     2$
000214  005726                                TST     (SP)+                           ;                       4619
000216  022626                        3$:     CMP     (SP)+,(SP)+                     ;                       4610
000220  012601                                MOV     (SP)+,R1                        ;                       4594
000222  000207                                RTS     PC                              ;                       4587

; Routine Size:  74 words,    Routine Base:  $CODE$ + 17074
; Maximum stack depth per invocation:  5 words
```

```
;    4622  1     GLOBAL routine QIO_LBN : novalue =
;    4623  1
;    4624  1     !+
;    4625  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE LOGICAL BLOCK NUMBER TO
;    4626  1     !       BE USED FOR THE CURRENT QIO OR QIO PAIR.
;    4627  1     !
;    4628  1     !       IF THE OPERATOR CHOSE THE RANDOM SEEK MODE OPTION, THEN THE LBN IS
;    4629  1     !       RANDOMLY CHOSEN WITHIN THE SPECIFIED LIMITS FOR THE LBN.
;    4630  1     !       OTHERWISE, THE NEXT SEQUENTIAL LBN IS DERIVED FROM THE BLOCK SEQUENCE
;    4631  1     !       TABLE (BST).
;    4632  1     !
;    4633  1     !       IMPLICIT INPUTS:
;    4634  1     !               L$LUN - CURRENT (DIAGNOSTIC SUPERVIOR) UNIT NUMBER
;    4635  1     !
;    4636  1     !       IMPLICIT OUTPUTS:
;    4637  1     !               THE LBN IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;    4638  1     !-
;    4639  1
;    4640  2         begin
;    4641  2
;    4642  2         own
;    4643  2             LBN0_SAVE : word initial (0),                   !LO LBN SELECTED IN PREVIOUS PASS
   ZZZ
;    4644  2             LBN1_SAVE : word initial (0);                   !HI LBN SELECTED IN PREVIOUS PASS
   ZZZ
;    4645  2
;    4646  2         local
;    4647  2             S0_TEMP : word,                                 ! TEMPORARY STORAGE FOR START LBN LO
   ZZZ
;    4648  2             S1_TEMP : word,                                 ! TEMPORARY STORAGE FOR START LBN HI
   ZZZ
;    4649  2             E0_TEMP : word,                                 ! TEMPORARY STORAGE FOR END LBN LO
   ZZZ
;    4650  2             E1_TEMP : word,                                 ! TEMPORARY STORAGE FOR END LBN HI
   ZZZ
;    4651  2             ADD0_LBN : word,                                ! TEMPORARY STORAGE USED FOR COMPUTING DESIRED LBN L
0  ZZZ
;    4652  2             ADD1_LBN : word,                                ! TEMPORARY STORAGE USED FOR COMPUTING DESIRED LBN H
I  ZZZ
;    4653  2             LBN0 : word,                                    ! LOGICAL BLOCK NUMBER LO
   ZZZ
;    4654  2             LBN1 : word,                                    ! LOGICAL BLOCK NUMBER HI
   ZZZ
;    4655  2             WINCHESTER : byte initial (byte (TRUE));        ! FLAG TO INDICATE WINCHESTER DISK SELECTED
;    4656  2
;    4657  2         label
;    4658  2             FIND_LBN;
;    4659  2
;    4660  2         S0_TEMP = .CST_ADDR [.CUOFF + OF_BEG, D_BEG0];      ! STARTING LBN LO
   ZZZ
;    4661  2         S1_TEMP = .CST_ADDR [.CUOFF + OF_BEG1, D_BEG1];     ! STARTING LBN HI
   ZZZ
;    4662  2         E0_TEMP = .CST_ADDR [.CUOFF + OF_END, D_END0];      ! ENDING LBN LO
   ZZZ
;    4663  2         E1_TEMP = .CST_ADDR [.CUOFF + OF_END1, D_END1];     ! ENDING LBN HI
   ZZZ
;    4664  2
;    4665  2         FIND_LBN:
;    4666  3             begin                                           !BEGIN A.
   ZZZ
;    4667  3
;    4668  3             if (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and
;    4669  3                (BIT_TST (SWP_FLAGS, SWF_FER)) and
;    4670  3                (.MAD1 [OPCODE] eql OP_WRT) and
;    4671  4                (.FORCED ERROR)
```

```
;  4674  4                 LBNO= .PERO LBN;                                ! IF "FORCED ERROR" DETECTED, REWRITE ERROR LBN L
 ZZZ
     ZZZ
;  4722  4                 LBNO = .BST [.L$LUN, LO WRD];                   ! GET LBN FROM BST (LO WORD)
 ZZZ
;  4675  4                 LBN1= .PER1 LBN;                                ! IF "FORCED ERROR" DETECTED, REWRITE ERROR LBN HI
 ZZZ
;  4676  4                 leave FIND_LBN;
;  4677  3                 end;
;  4678  3
;  4679  3             if .CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql REMOVABLE
;  4680  3             then
;  4681  3                 WINCHESTER = FALSE;
;  4682  3
;  4683  4             if BIT_TST (SWP_FLAGS, SWF_RDM)                     ! IF RANDOM SEEK MODE
;  4684  3             then
;  4685  4                 begin
;  4686  4
;  4687  4                 if (.WINCHESTER) and
;  4688  5                    (((.RANDOM [0] and %o'077777') mod (100)) lequ 49)
;  4689  4                 then
;  4690  5                     BEGIN                                                       !
     ZZZ
;  4691  5                     LBNO = .LBNO_SAVE;                                          ! REDUCE SEEKS ON RDs by 50%
     ZZZ
;  4692  5                     LBN1 = .LBN1_SAVE;                                          ! REDUCE SEEKS ON RDs by 50%
     ZZZ
;  4693  5                     END                                                         !
     ZZZ
;  4694  4                 else
;  4695  5                     begin
;  4696  5                     RANDY ();                                                   !GET A 32 BIT RANDOM NUMBER
     ZZZ
;  4697  5                     IF (.RNDY1 GTRU .E1_TEMP) OR                                !IF NUMBER GREATER THAN MAX
     ZZZ
;  4698  6                        ((.RNDY1 EQLU .E1_TEMP) AND                              !
     ZZZ
;  4699  6                        (.RNDYO GTRU .EO_TEMP))                                  !
     ZZZ
;  4700  5                     THEN                                                        !
     ZZZ
;  4701  6                        BEGIN                                                    !THEN MASK IT WITH HI LIMIT
     ZZZ
;  4702  6                        RNDY1 = .RNDY1 AND .E1_TEMP;                             !
     ZZZ
;  4703  6                        RNDYO = .RNDYO AND .EO_TEMP;                             !
     ZZZ
;  4704  5                        END;                                                     !
     ZZZ
;  4705  5                                                                                 !
     ZZZ
;  4706  5                     IF (.RNDY1 LSSU .S1_TEMP) OR                                !IF NUMBER LESS THAN MIN
     ZZZ
;  4707  6                        ((.RNDY1 EQLU .S1_TEMP) AND                              !
     ZZZ
;  4708  6                        (.RNDYO LSSU .SO_TEMP))                                  !
     ZZZ
;  4709  5                     THEN                                                        !
     ZZZ
;  4710  6                        BEGIN                                                    !THEN MASK IT WITH LO LIMIT
     ZZZ
;  4711  6                        RNDY1 = .RNDY1 AND .S1_TEMP;                             !
     ZZZ
;  4712  6                        RNDYO = .RNDYO AND .SO_TEMP;                             !
     ZZZ
;  4713  5                        END;                                                     !
     ZZZ
;  4714  5                                                                                 !
     ZZZ
;  4715  5                     LBNO =  RNDYO;                                              !LO HALF
```

```
;                ZZZ
;       4717  4            end;
;       4718  4        end
;       4719  4
;       4720  4    ELSE  LBN1 = .BST [.L$LUN, HI WRD];                    !! GET LBN FROM BST (HI WORD)
;       4721  4
;       4722  4    begin IF .TRK_SGN [.L$LUN] EQLU 1                      !ELSE - SEQUENTIAL LBN MODE (BEGIN A)
;                ZZZ                                                     !IF WE WANT SERIAL INCREMENT
;       4726  4        THEN                                             !
;                ZZZ
;       4727  5        BEGIN                                            !(BEGIN B)
;                ZZZ
```

```
;  4728  5                    IF .BST [.L$LUN, LO WRD] EQLU %o'177777'          !IF OVERFLOW FROM LO WD TO HI WD
;  4729  5                    THEN                                              !
;  4730  6                    BEGIN                                             !
;  4731  6                        BST [.L$LUN, LO WRD] = 0;                     !ZERO LO WORD
;  4732  6                        BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] + 1;          !INCREMENT HI WORD
;  4733  6                    END                                               !
;  4734  5                    ELSE                                                          !OTHERWISE JUST INCR LO WORD
;  4735  5                        BST [.L$LUN, LO WRD] = .BST [.L$LUN, LO WRD] + 1;         !
;  4736  5
;  4737  5                                                                      !NOW TAKE CARE OF OVERFLOW WHILE INCREMENTIN
G.
;  4738  6                    IF (.BST [.L$LUN, HI_WRD] GTRU .E1_TEMP)          !IF LBN1 OVER HI LIMIT
;  4739  7                      OR ((.BST [.L$LUN, HI_WRD] EQLU .E1_TEMP)       !OR LBN1 EQUALS HI LIMIT AND LBNO IS OVER LI
MIT
;  4740  6                      AND (.BST [.L$LUN, LO WRD] GTRU .E0_TEMP))      !
;  4741  5                    THEN                                              !
;  4742  6                       BEGIN                                          !
;  4743  6                       BST [.L$LUN, LO_WRD] = .E0 TEMP;               'THEN SET HI LIMITS
;  4744  6                       BST [.L$LUN, HI_WRD] = .E1 TEMP;               !INTO BST FOR NEXT TIME
;  4745  6                       TRK SGN [.L$LUN] =   1;                        !AND REVERSE DIRECTION
;  4746  5                       END;                                           !
;  4747  5                END                                                   !(END B)
;  4748  5
;  4749  5
;  4750  5
;  4751  4                ELSE                                                  !IF WE WANT SERIAL DECREMENT
;  4752  5                BEGIN                                                 !(BEGIN C)
;  4753  5                    IF .BST [.L$LUN, LO_WRD] EQLU 0                   !IF NEED TO BORROW FROM HI WD
;  4754  5                    THEN                                              !
;  4755  6                    BEGIN                                             !
;  4756  6                        BST [.L$LUN, LO_WRD] = %o'177777';            !LO WORD
;  4757  6                        BST [.L$LUN, HI WRD] = .BST [.L$LUN, HI WRD] - 1;          !DECREMENT HI WORD
;  4758  6                    END                                               !
;  4759  5                    ELSE                                                          !OTHERWISE JUST DECR LO WORD
;  4760  5                        BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD]   1;         !
;  4761  5
;  4762  5                                                                      !NOW TAKE CARE OF UNDERFLOW WHILE INCREMENTI
NG.
```

```
;  4764  7                    OR ((.BST [.L$LUN, HI WRD] EQLU .S1 TEMP)
   ZZZ
;  4765  6                    AND (.BST [.L$LUN, LO WRD] LSSU .S0 TEMP))
   ZZZ
;  4766  5                  THEN BST [.L$LUN, HI WRD] = .S1 TEMP;
   ZZZ
;  4767  4                    BEGIN SGN [.L$LUN] = -1;
   ZZZ
;  4771  5                    END;
   ZZZ
;  4772  4            END;
   ZZZ
;  4773  3         END;
   ZZZ
;  4774  3
   ZZZ
;  4775  2       END;
   ZZZ
;  4776  2
;  4777  2
;  4778  3    IF ((.S1_TEMP EQLU .E1_TEMP) AND (.S0 TEMP EQLU .E0 TEMP))
   ZZZ
;  4779  2    THEN
   ZZZ
;  4780  3       BEGIN
   ZZZ
```

```
!OR LBNI EQUALS LO LIMIT AND LBNO IS BELOW
!
!!INTO BST FOR NEXT TIME
!                !AND REVERSE DIRECTION
                 !
!
!END C.
!
!END A.

!IF START ADDR SAME AS END ADDR
!JUST USE THE START ADDRESS.
!
```

```
;    4781  3            LBN0 = .S0_TEMP;                                    !
     ZZZ
;    4782  3            LBN1 = .S1_TEMP;                                    !
     ZZZ
;    4783  2            END;                                                !
     ZZZ
;    4784  2
;    4785  2        MAD1 [LBN_L] = .LBN0;                                   ! LOAD LBN INTO 1ST PACKET
     ZZZ
;    4786  2        MAD1 [LBN_H] = .LBN1;                                   ! LOAD LBN INTO 1ST PACKET
     ZZZ
;    4787  2
;    4788  2        if .MX2 geq 0                                           ! IF 2 QIOs
     ZZZ
;    4789  2        then
;    4790  2            MAD2 [LBN_L] = .LBN0;                               ! LOAD LBN INTO 2ND PACKET
     ZZZ
;    4791  2            MAD2 [LBN_H] = .LBN1;                               ! LOAD LBN INTO 2ND PACKET
     ZZZ
;    4792  2
;    4793  2        LBN0_SAVE = .LBN0;                                      ! SAVE FOR USE NEXT CYCLE IF NEEDED
     ZZZ
;    4794  2        LBN1_SAVE = .LBN1;                                      ! SAVE FOR USE NEXT CYCLE IF NEEDED
     ZZZ
;    4795  2
;    4796  1        end;                                                    ! ROUTINE QIO LBN


001300                                  .PSECT  $GGG$,  RO
001300  000000                  LBN0.SAVE:
                                        .WORD   0
001302  000000                  LBN1.SAVE:
                                        .WORD   0


                                        .SBTTL  QIO.LBN MULTI-DRIVE TEST ROUTINES
017320                                  .PSECT  $CODE$,  RO

000000  004137  000000G         QIO.LBN::
                                JSR     R1,$SAVE5                       ;                       4622
000004  005746                  TST     -(SP)
000006  112701  000001          MOVB    #1,R1                           ; *,WINCHESTER          4640
000012  013705  000000G         MOV     CST.ADDR,R5                     ;                       4660
000016  013702  000000G         MOV     CUOFF,R2
000022  010200                  MOV     R2,R0
000024  006300                  ASL     R0
000026  060500                  ADD     R5,R0
000030  016046  000002          MOV     2(R0),-(SP)                     ; *,S0.TEMP
000034  010200                  MOV     R2,R0                           ;                       4661
000036  006300                  ASL     R0
000040  060500                  ADD     R5,R0
000042  016003  000004          MOV     4(R0),R3                        ; *,S1.TEMP
000046  010200                  MOV     R2,R0                           ;                       4662
000050  006300                  ASL     R0
000052  060500                  ADD     R5,R0
000054  016046  000006          MOV     6(R0),-(SP)                     ; *,E0.TEMP
000060  010200                  MOV     R2,R0                           ;                       4663
000062  006300                  ASL     R0
000064  060500                  ADD     R5,R0
000066  016004  000010          MOV     10(R0),R4                       ; *,E1.TEMP
000072  006302                  ASL     R2                              ;                       4668
000074  060502                  ADD     R5,R2
```

```
000076  132712  000020                          BITB    #20,(R2)
000102  001426                                  BEQ     2$
000104  032737  004000  000000G                 BIT     #4000,SWP.FLAGS         ;                               4669
000112  001417                                  BEQ     1$
000114  013700  000114'                         MOV     MAD1,R0                 ;                               4670
000120  126027  000022  000042                  CMPB    22(R0),#42
000126  001011                                  BNE     1$
000130  130137  000000G                         BITB    R1,FORCED.ERROR         ;                               4671
000134  001406                                  BEQ     1$
000136  013705  000000G                         MOV     FER0.LBN,R5             ; *,LBN0                        4674
000142  013766  000000G 000004                  MOV     FER1.LBN,4(SP)          ; *,LBN1                        4675
000150  000571                                  BR      17$                     ;                               4673
000152  132712  000020                  1$:     BITB    #20,(R2)                ;                               4679
000156  001001                                  BNE     3$
000160  105001                          2$:     CLRB    R1                      ; WINCHESTER                    4681
000162  032737  000002  000000G         3$:     BIT     #2,SWP.FLAGS            ;                               4683
000170  001474                                  BEQ     9$
000172  006001                                  ROR     R1                      ; WINCHESTER                    4687
000174  103022                                  BCC     4$
000176  013746  000000G                         MOV     RANDOM,-(SP)            ;                               4688
000202  042716  100000                          BIC     #100000,(SP)
000206  012746  000144                          MOV     #144,-(SP)
000212  004737  000000G                         JSR     PC,BL$MOD
000216  022626                                  CMP     (SP)+,(SP)+
000220  020027  000061                          CMP     R0,#61
000224  101006                                  BHI     4$
000226  013705  001300'                         MOV     LBN0.SAVE,R5            ; *,LBN0                        4691
000232  013766  001302'  000004                 MOV     LBN1.SAVE,4(SP)         ; *,LBN1                        4692
000240  000535                                  BR      17$                     ;                               4687
000242  004737  011500'                  4$:     JSR     PC,RANDY                ;                               4696
000246  023704  000130'                         CMP     RNDY1,R4                ; *,E1.TEMP                     4697
000252  101004                                  BHI     5$
000254  001013                                  BNE     6$                      ;                               4698
000256  023716  000126'                         CMP     RNDY0,(SP)              ; *,E0.TEMP                     4699
000262  101410                                  BLOS    6$
000264  010400                          5$:     MOV     R4,R0                   ; E1.TEMP,*                     4702
000266  005100                                  COM     R0
000270  040037  000130'                         BIC     R0,RNDY1
000274  011600                                  MOV     (SP),R0                 ; E0.TEMP,*                     4703
000276  005100                                  COM     R0
000300  040037  000126'                         BIC     R0,RNDY0
000304  023703  000130'                  6$:     CMP     RNDY1,R3                ; *,S1.TEMP                     4706
000310  103405                                  BLO     7$
000312  001015                                  BNE     8$                      ;                               4707
000314  023766  000126'  000002                 CMP     RNDY0,2(SP)             ; *,S0.TEMP                     4708
000322  103011                                  BHIS    8$
000324  010300                          7$:     MOV     R3,R0                   ; S1.TEMP,*                     4711
000326  005100                                  COM     R0
000330  040037  000130'                         BIC     R0,RNDY1
000334  016600  000002                          MOV     2(SP),R0                ; S0.TEMP,*                     4712
000340  005100                                  COM     R0
000342  040037  000126'                         BIC     R0,RNDY0
000346  013705  000126'                  8$:     MOV     RNDY0,R5                ; *,LBN0                        4715
```

```
000352  013766  000130' 000004            MOV     RNDY1,4(SP)             ; *,LBN1                4716
000360  000465                            BR      17$                     ;                       4683
000362  013702  000000G           9$:     MOV     L$LUN,R2                ;                       4722
000366  010201                            MOV     R2,R1
000370  006301                            ASL     R1
000372  006301                            ASL     R1
000374  012700  000000G                   MOV     #BST,R0
000400  060100                            ADD     R1,R0
000402  011005                            MOV     (R0),R5                 ; *,LBN0
000404  062701  000002G                   ADD     #BST+2,R1               ;                       4723
000410  011166  000004                    MOV     (R1),4(SP)              ; *,LBN1
000414  062702  000000G                   ADD     #TRK.SGN,R2             ;                       4725
000420  121227  000001                    CMPB    (R2),#1
000424  001021                            BNE     13$
000426  020527  177777                    CMP     R5,#-1                  ;                       4728
000432  001003                            BNE     10$
000434  005010                            CLR     (R0)                    ;                       4731
000436  005211                            INC     (R1)                    ;                       4732
000440  000401                            BR      11$                     ;                       4728
000442  005210                    10$:    INC     (R0)                    ;                       4735
000444  021104                    11$:    CMP     (R1),R4                 ; *,E1.TEMP             4738
000446  101003                            BHI     12$
000450  001031                            BNE     17$                     ;                       4739
000452  021016                            CMP     (R0),(SP)               ; *,E0.TEMP             4740
000454  101427                            BLOS    17$
000456  011610                    12$:    MOV     (SP),(R0)               ; E0.TEMP,*             4743
000460  010411                            MOV     R4,(R1)                 ; E1.TEMP,*             4744
000462  112712  000377                    MOVB    #377,(R2)               ;                       4745
000466  000422                            BR      17$                     ;                       4725
000470  005710                    13$:    TST     (R0)                    ;                       4753
000472  001004                            BNE     14$
000474  012710  177777                    MOV     #-1,(R0)                ;                       4756
000500  005311                            DEC     (R1)                    ;                       4757
000502  000401                            BR      15$                     ;                       4753
000504  005310                    14$:    DEC     (R0)                    ;                       4760
000506  021103                    15$:    CMP     (R1),R3                 ; *,S1.TEMP             4763
000510  002404                            BLT     16$
000512  001010                            BNE     17$                     ;                       4764
000514  021066  000002                    CMP     (R0),2(SP)              ; *,S0.TEMP             4765
000520  103005                            BHIS    17$
000522  016610  000002            16$:    MOV     2(SP),(R0)              ; S0.TEMP,*             4768
000526  010311                            MOV     R3,(R1)                 ; S1.TEMP,*             4769
000530  112712  000001                    MOVB    #1,(R2)                 ;                       4770
000534  020304                    17$:    CMP     R3,R4                   ; S1.TEMP,E1.TEMP       4778
000536  001007                            BNE     18$
000540  026616  000002                    CMP     2(SP),(SP)              ; S0.TEMP,E0.TEMP
000544  001004                            BNE     18$
000546  016605  000002                    MOV     2(SP),R5                ; S0.TEMP,LBN0          4781
000552  010366  000004                    MOV     R3,4(SP)                ; S1.TEMP,LBN1          4782
000556  013700  000114'            18$:    MOV     MAD1,R0                 ;                       4785
000562  010560  000046                    MOV     R5,46(R0)               ; LBN0,*
000566  016660  000004  000050            MOV     4(SP),50(R0)            ; LBN1,*                4786
000574  005737  000112'                    TST     MX2                     ;                       4788
```

```
ZRQAM3          RD/RX EXERCISER                              4-Apr 1985 13:23:31    VAX-11 Bliss-16 V4.1 582          SEQ 039C
V02.2           MULTI-DRIVE TEST ROUTINES                    2-Apr-1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;19  Page 135
                                                                                                                       (32)

000600  002404                                      BLT     19$
000602  013700  000116'                             MOV     MAD2,R0                                                    4790
000606  010560  000046                              MOV     R5,46(R0)          ; LBN0,*
000612  013700  000116'                     19$:    MOV     MAD2,R0                                                    4791
000616  016660  000004  000050                      MOV     4(SP),50(R0)       ; LBN1,*
000624  010537  001300'                             MOV     R5,LBN0.SAVE       ; LBN0,*                                4793
000630  016637  000004  001302'                     MOV     4(SP),LBN1.SAVE    ; LBN1,*                                4794
000636  062706  000006                              ADD     #6,SP                                                      4622
000642  000207                                      RTS     PC

; Routine Size:  210 words,     Routine Base:  $CODE$ + 17320
; Maximum stack depth per invocation:  12 words


;   4797  1
```

```
;  4798  1     !!ZZZ routine QIO_SIZE : novalue =
;  4799  1     GLOBAL ROUTINE QIO_SIZE : NOVALUE =
;  4800  1
;  4801  1     !+
;  4802  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O TRANSFER BYTE COUNT
;  4803  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE BYTE COUNT IS
;  4804  1     !       DETERMINED BY A RANDOM NUMBER, AND WILL ALWAYS FALL BETWEEN 1 AND THE
;  4805  1     !       I/O BUFFER SIZE (BYTS_PER_QIO). It is assumed that BYTS_PER_QIO will
;  4806  1     !       never be larger than one binary word or  65000 bytes.
;  4807  1     !
;  4808  1     !       IMPLICIT OUTPUTS:
;  4809  1     !               THE BYTE COUNT IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;  4810  1     !-
;  4811  1
;  4812  2         begin
;  4813  2
;  4814  2         local
;  4815  2             SIZE : word,                                                    ! BYTE COUNT
;  4816  2             BLOCKS_LEFT : word;                                             ! REMAINING BLOCKS LEFT
;  4817  2
;  4818  2         SIZE = ((.RANDOM [4] and %o'077777') mod (.BYTS_PER_QIO + 1)) and %o'177760';!GET BYTE COUNT FROM RANDOM NUMBER
;  4819  2
;  4820  2         if .SIZE eql 0
;  4821  2         then
;  4822  2             SIZE = 16;
;  4823  2
;  4824  2         if .CST_ADDR [.CUOFF + 4, D_END1] gtru .MAD1 [LBN_H]
;  4825  2         then BLOCKS_LEFT = %o'177777'
;  4826  2         else BLOCKS_LEFT = .CST_ADDR [.CUOFF + 3, D_END0] - .MAD1 [LBN_L] + 1;          ! find
;  4827  2                                                                                        ! REMAINING BLOCK COUNT
;  4828  2         if ((.SIZE + BYTES_PER_SECT - 1) / BYTES_PER_SECT) gtru .BLOCKS_LEFT           ! IF BLOCK COUNT NOT ENOUGH
;  4829  2         then
;  4830  2             SIZE = .BLOCKS_LEFT * BYTES_PER_SECT;                                      ! ADJUST BYTE COUNT DOWN
;  4831  2
;  4832  2         MAD1 [BC_LO] = .SIZE;                                                          ! LOAD SIZE INTO 1ST MSCP PACKET
;  4833  2
;  4834  2         if .MX2 geq 0                                                                  ! IF 2 QIOS
;  4835  2         then
;  4836  2             MAD2 [BC_LO] = .SIZE;                                                      ! LOAD SIZE INTO 2ND MSCP PACKET
;  4837  2
;  4838  1         end;                                                                           ! ROUTINE QIO_SIZE


                                        .SBTTL  QIO.SIZE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          QIO.SIZE::
                                        JSR     R1,$SAVE3                       ;                                       4799
000004  013746  000010G                 MOV     RANDOM+10,-(SP)                 ;                                       4818
000010  042716  100000                  BIC     #100000,(SP)
000014  013746  000000G                 MOV     BYTS.PER.QIO,-(SP)
000020  005216                          INC     (SP)
000022  004737  000000G                 JSR     PC,BL$MOD
000026  010003                          MOV     R0,R3                           ; *.SIZE
000030  042703  000017                  BIC     #17,R3                          ; *.SIZE
```

```
000034  001002                          BNE     1$                      ;
000036  012703  000020                  MOV     #20,R3                  ; *,SIZE                        4820
000042  013700  000000G         1$:     MOV     CUOFF,R0                ;                               4822
000046  006300                          ASL     R0                                                      4824
000050  063700  000000G                 ADD     CST.ADDR,R0
000054  013701  000114'                 MOV     MAD1,R1
000060  026061  000010  000050          CMP     10(R0),50(R1)
000066  101403                          BLOS    2$
000070  012702  177777                  MOV     #-1,R2                  ; *,BLOCKS.LEFT                 4825
000074  000413                          BR      3$                      ;                               4824
000076  013700  000000G         2$:     MOV     CUOFF,R0                ;                               4826
000102  006300                          ASL     R0
000104  063700  000000G                 ADD     CST.ADDR,R0
000110  016000  000006                  MOV     6(R0),R0
000114  166100  000046                  SUB     46(R1),R0
000120  010002                          MOV     R0,R2                   ; *,BLOCKS.LEFT
000122  005202                          INC     R2                      ; BLOCKS.LEFT
000124  010316                  3$:     MOV     R3,(SP)                 ; SIZE,*                        4828
000126  062716  000777                  ADD     #777,(SP)
000132  012746  001000                  MOV     #1000,-(SP)
000136  004737  000000G                 JSR     PC,BL$DIV
000142  005726                          TST     (SP)+
000144  020002                          CMP     R0,R2                   ; *,BLOCKS.LEFT
000146  101405                          BLOS    4$
000150  010200                          MOV     R2,R0                   ; BLOCKS.LEFT,*                 4830
000152  000300                          SWAB    R0
000154  105000                          CLRB    R0
000156  006300                          ASL     R0
000160  010003                          MOV     R0,R3                   ; *,SIZE
000162  010361  000026          4$:     MOV     R3,26(R1)               ; SIZE,*                        4832
000166  005737  000112'                 TST     MX2                     ;                               4834
000172  002404                          BLT     5$
000174  013700  000116'                 MOV     MAD2,R0                 ;                               4836
000200  010360  000026                  MOV     R3,26(R0)               ; SIZE,*
000204  022626                  5$:     CMP     (SP)+,(SP)+             ;                               4812
000206  000207                          RTS     PC                      ;                               4799

; Routine Size:  68 words,      Routine Base:  $CODE$ + 20164
; Maximum stack depth per invocation:  8 words
```

```
;    4839  1     GLOBAL routine FILL_BUFF : novalue =
;    4840  1
;    4841  1     !+
;    4842  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO LOAD THE I/O BUFFER DESCRIBED IN
;    4843  1     !       THE FIRST MSCP PACKET WITH THE APPROPRIATE DATA PATTERN.
;    4844  1     !
;    4845  1     !       THE DATA PATTERN TO BE SELECTED IS BASED ON THE FOLLOWING ALGORITHM:
;    4846  1     !
;    4847  1     !               IF THE OPERATOR DEFINED A DATA PATTERN
;    4848  1     !               THEN
;    4849  1     !                       SELECT IT
;    4850  1     !               ELSE
;    4851  1     !                       GET DATA PATTERN NUMBER FROM SW P-TABLE
;    4852  1     !                       IF DATA PATTERN NUMBER = 0
;    4853  1     !                       THEN
;    4854  1     !                               GET DATA PATTERN NUMBER FROM THE UNIT'S ENTRY
;    4855  1     !                               IN THE DATA PATTERN SEQUENCE TABLE (DPST)
;    4856  1     !
;    4857  1     !       NOTE THAT PATTERN # 1 CONSISTS OF RANDOM NUMBERS, AND PATTERNS # 17 -
;    4858  1     !       21 USE THE ACTUAL LBN OF THE WRITE REQUEST.
;    4859  1     !
;    4860  1     !       IMPLICIT INPUTS:
;    4861  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    4862  1     !-
;    4863  1
;    4864  2         begin
;    4865  2
;    4866  2         local
;    4867  2             DP_NUM : word,                          ! DATA PATTERN NUMBER SELECTED
;    4868  2             DP_ADDR,                                ! ADDR OF DATA PATTERN (LENGTH)
;    4869  2             IOB_ADDR,                               ! I/O BUFFER ADDRESS (DESTINATION)
;    4870  2             SRC_ADDR,                               ! WORKING SOURCE ADDRESS
;    4871  2             COUNT : word;                           ! NO. OF WORDS IN DATA PATTERN
;    4872  2
;    4873  3         if BIT_TST (SWP_FLAGS, SWF_UDP)             ! IF USER DEFINED A DATA PATTERN
;    4874  2         then
;    4875  2             DP_ADDR = SWP_UCNT                      ! SELECT IT
;    4876  2         else
;    4877  3             begin
;    4878  3
;    4879  3             if .SWP_DPAT neq 0                      ! IF USER SELECTED A PRE-DEFINED DATA PATTERN
;    4880  3             then
;    4881  3                 DP_NUM = .SWP_DPAT                  ! SELECT IT
;    4882  3             else
;    4883  4                 begin
;    4884  4                 DP_NUM = .DPST [.L$LUN];            ! GET PATTERN NUMBER FROM SEQUENCE TABLE
;    4885  4                 DPST [.L$LUN] = .DPST [.L$LUN] + 1; ! ADVANCE TO NEXT PATTERN NUMBER
;    4886  4
;    4887  4                 if .DPST [.L$LUN] gtru DP_CNT       ! CHECK FOR HIGH LIMIT
;    4888  4                 then
;    4889  4                     DPST [.L$LUN] = 1;
;    4890  4
;    4891  3                 end;
```

```
;    4892  3
;    4893  3              DP_ADDR = .DPA_TBL [.DP_NUM   1];              ! ADDRESS OF DATA PATTERN (COUNT)
;    4894  3
;    4895  3              if .DP_NUM gequ 17
;    4896  3              then
;    4897  3
;    4898  3                  if .DP_NUM                                 ! CHECK MACRO (IF PATTERN 17, 19, OR 21)
;    4899  3                  then
;    4900  3                      (.DP_ADDR + 2) = .MAD1 [LBN_L]         ! LOAD LBN INTO FIRST WORD OF PATTERN
;    4901  3                  else
;    4902  3                      (.DP_ADDR + 4) = .MAD1 [LBN_L];        ! LOAD LBN INTO SECOND WORD OF PATTERN
;    4903  3
;    4904  2              end;
;    4905  2
;    4906  2          IOB_ADDR = .MAD1 [BUF_0];                          ! I/O BUFFER ADDRESS
;    4907  2          COUNT = ..DP_ADDR;                                 ! NO. OF WORDS IN DATA PATTERN
;    4908  2          SRC_ADDR = .DP_ADDR + 2;                           ! START OF THE ACTUAL DATA PATTERN
;    4909  2
;    4910  2          incr N from 1 to ((.MAD1 [BC_LO] + 1) / 2) do      ! FOR EACH WORD IN THIS WRITE REQUEST
;    4911  3              begin
;    4912  3              .IOB_ADDR = ..SRC_ADDR;                        ! MOVE 1 WORD
;    4913  3              IOB_ADDR = .IOB_ADDR + 2;                      ! ADVANCE DESTINATION ADDRESS
;    4914  3              SRC_ADDR = .SRC_ADDR + 2;                      ! ADVANCE SOURCE ADDRESS
;    4915  3              COUNT = .COUNT - 1;                            ! DECREMENT COUNT
;    4916  3
;    4917  3              if .COUNT eql 0                                ! IF END OF DATA PATTERN
;    4918  3              then
;    4919  4                  begin
;    4920  4                  COUNT = ..DP_ADDR;                         ! REPEAT DATA PATTERN
;    4921  4                  SRC_ADDR = .DP_ADDR + 2;
;    4922  3                  end;
;    4923  3
;    4924  2              end;                                           ! WORD TRANSFER LOOP
;    4925  2
;    4926  1          end;                                               ! ROUTINE FILL_BUFF


                                    .SBTTL  FILL.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G       FILL.BUFF::
                                    JSR     R1,$SAVES                   ;                                   4839
000004  005746                      TST     -(SP)
000006  032737  000100  000000G      BIT     #100,SWP.FLAGS              ;                                   4873
000014  001403                      BEQ     1$
000016  012701  000000G              MOV     #SWP.UCNT,R1                ; *.DP.ADDR                         4875
000022  000443                      BR      5$                          ;                                   4873
000024  013700  000000G      1$:     MOV     SWP.DPAT,R0                 ;                                   4879
000030  001402                      BEQ     2$
000032  010002                      MOV     R0,R2                       ; *.DP.NUM                          4881
000034  000414                      BR      3$                          ;                                   4879
000036  013700  000000G      2$:     MOV     L$LUN,R0                    ;                                   4884
000042  062700  000050'              ADD     #DPST,R0
000046  005002                      CLR     R2                          ; DP.NUM
000050  151002                      BISB    (R0),R2                     ; *.DP.NUM
```

```
000052  105210                          INCB    (R0)                    ;                                       4885
000054  121027  000025                  CMPB    (R0),#25                ;                                       4887
000060  101402                          BLOS    3$
000062  112710  000001                  MOVB    #1,(R0)                 ;                                       4889
000066  010200                  3$:     MOV     R^,R0                   ; DP.NUM,*                              4893
000070  006300                          ASL     R0
000072  016001  001166'                  MOV     DPA.TBL-2(R0),R1        ; *,DP.ADDR
000076  020227  000021                  CMP     R2,#21                  ; DP.NUM,*                              4895
000102  103413                          BLO     5$
000104  013700  000114'                  MOV     MAD1,R0                 ;                                       4900
000110  006002                          ROR     R2                      ; DP.NUM                                4898
000112  103004                          BCC     4$
000114  016061  000046  000002          MOV     46(R0),2(R1)            ; *,*(DP.ADDR)                          4900
000122  000403                          BR      5$                      ;                                       4898
000124  016061  000046  000004  4$:     MOV     46(R0),4(R1)            ; *,*(DP.ADDR)                          4902
000132  013700  000114'          5$:     MOV     MAD1,R0                 ;                                       4906
000136  016004  000032                  MOV     32(R0),R4               ; *,IOB.ADDR
000142  011103                          MOV     (R1),R3                 ; DP.ADDR,COUNT                         4907
000144  012705  000002                  MOV     #2,R5                   ;                                       4908
000150  060105                          ADD     R1,R5                   ; DP.ADDR,*
000152  010502                          MOV     R5,R2                   ; *,SRC.ADDR
000154  016046  000026                  MOV     26(R0),-(SP)            ;                                       4910
000160  005216                          INC     (SP)
000162  012746  000002                  MOV     #2,-(SP)
000166  004737  000000G                 JSR     PC,BL$DIV
000172  010066  000004                  MOV     R0,4(SP)
000176  005000                          CLR     R0                      ; N
000200  000405                          BR      7$
000202  012224                  6$:     MOV     (R2)+,(R4)+             ; SRC.ADDR,IOB.ADDR                     4912
000204  005303                          DEC     R3                      ; COUNT                                 4915
000206  001002                          BNE     7$                      ;                                       4917
000210  011103                          MOV     (R1),R3                 ; DP.ADDR,COUNT                         4920
000212  010502                          MOV     R5,R2                   ; *,SRC.ADDR                            4921
000214  005200                  7$:     INC     R0                      ; N                                     4910
000216  020066  000004                  CMP     R0,4(SP)                ; N,*
000222  003767                          BLE     6$
000224  062706  000006                  ADD     #6,SP                   ;                                       4839
000230  000207                          RTS     PC

; Routine Size:  77 words,      Routine Base:  $CODE$ + 20374
; Maximum stack depth per invocation:  10 words


;   4927  1
;   4928  1
```

```
;    4929  1     GLOBAL ROUTINE PROC_RETPKT : NOVALUE =
;    4930  1
;    4931  1     !+
;    4932  1     !       THIS ROUTINE IS CALLED FROM THE MULTI_DRIVE "EXECUTIVE" AND DUP_COMMAND TO CHECK FOR
;    4933  1     !       AND PROCESS ANY RETURN PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER"
;    4934  1     !       PORTION OF THE PROGRAM. THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK
;    4935  1     !       BETWEEN THE TWO PROGRAM PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH
;    4936  1     !       REQUIRE PROCESSING.
;    4937  1     !
;    4938  1     !       UNDER THE MULTI-DRIVE TEST, RETURN PACKETS ORIGINATE FROM TWO SOURCES:
;    4939  1     !               1. MSCP         - THE MORE COMMON, DESCRIBING A COMPLETED I/O
;    4940  1     !                                 OPERATION.
;    4941  1     !               2. DUP          - THE LESS COMMON, DESCRIBING A PORTION OF I/O
;    4942  1     !                                 COMMUNICATIONS WITH THE CONTROLLER PROGRAM.
;    4943  1     !               3. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
;    4944  1     !                                 COMMAND TIMEOUT.
;    4945  1     !-
;    4946  1
;    4947  1         while .IODQ_IN neq .IODQ_OUT do                         ! DO UNTIL I/O DONE QUEUE IS EMPTY
;    4948  2             begin
;    4949  2             RP_INDX = OUT_IODQ ();                              ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
;    4950  2             RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);         ! CALCULATE RETPKT ADDRESS
;    4951  3             if NOT (.RP_ADDR [CONID] eql CID_DUP)               ! if not DUP then
;    4952  2             then (SET_CPAR (.RP_ADDR [CTLR]));                  ! SET UP CURRENT CONTROLLER PARAMETERS
;    4953  2
;    4954  2             selectoneu .RP_ADDR [CONID] of                      ! CONNECTION ID INDICATES PACKET SOURCE
;    4955  2                 set
;    4956  2
;    4957  2                 [CID_MSCP] :        IO_RETPKT ();               ! DISK MSCP (I/O TRANSFER DONE)
;    4958  2                 [CID_DUP] :         DIO_RETPKT ();              ! DUP (I/O TRANSFER DONE)
;    4959  2                 [CID_DRIVER] :      DR_RETPKT ();               ! MESSAGE FROM "DRIVER"
;    4960  2
;    4961  2                 [otherwise] :       PRINTF (DBM12, .RP_ADDR [CONID]);!"CONN ID = XXXXX RECEIVED"
;    4962  2                 tes;
;    4963  2
;    4964  1             end;                                                ! UNITL I/O DONE QUEUE IS EMPTY


                                        .SBTTL  PROC.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                  PROC.RETPKT::
                                        MOV     R1,-(SP)                    ;                                       4929
000002  023737  000000G 000000G 1$:     CMP     IODQ.IN,IODQ.OUT            ;                                       4947
000010  001467                          BEQ     7$
000012  004737  000000G                 JSR     PC,OUT.IODQ                 ;                                       4949
000016  010037  000000G                 MOV     R0,RP.INDX
000022  010046                          MOV     R0,-(SP)                    ; RP.INDX,*                             4950
000024  012746  000054                  MOV     #54,-(SP)
000030  004737  000000G                 JSR     PC,BL$MUL
000034  062700  000000G                 ADD     #RETPKT,R0
000040  010037  000000G                 MOV     R0,RP.ADDR
000044  126027  000003  000002         CMPB    3(R0),#2                    ;                                       4951
000052  001406                          BEQ     2$
000054  116016  000002                  MOVB    2(R0),(SP)                  ;                                       4952
```

```
000060  042716  177760                          BIC     #177760,(SP)
000064  004737  000000G                         JSR     PC,SET.CPAR
000070  013700  000000G               2$:       MOV     RP.ADDR,R0              ;                       4954
000074  005001                                   CLR     R1
000076  156001  000003                          BISB    3(R0),R1
000102  005701                                   TST     R1                      ;                       4957
000104  001003                                   BNE     3$
000106  004737  000000V                         JSR     PC,IO.RETPKT
000112  000424                                   BR      6$                      ;                       4954
000114  020127  000002               3$:       CMP     R1,#2                   ;                       4958
000120  001003                                   BNE     4$
000122  004737  000000V                         JSR     PC,DIO.RETPKT
000126  000416                                   BR      6$                      ;                       4954
000130  020127  000003               4$:       CMP     R1,#3                   ;                       4959
000134  001003                                   BNE     5$
000136  004737  000000V                         JSR     PC,DR.RETPKT
000142  000410                                   BR      6$                      ;                       4954
000144  010116               5$:       MOV     R1,(SP)                 ;                       4961
000146  012746  000000G                         MOV     #DBM12,-(SP)
000152  012746  000002                          MOV     #2,-(SP)
000156  010600                                   MOV     SP,R0                   ; SP,*
000160  104417                                   TRAP    17
000162  022626                                   CMP     (SP)+,(SP)+
000164  022626               6$:       CMP     (SP)+,(SP)+             ;                       4948
000166  000705                                   BR      1$                      ;                       4947
000170  012601               7$:       MOV     (SP)+,R1                ;                       4929
000172  000207                                   RTS     PC

; Routine Size:  62 words,      Routine Base:  $CODE$ + 20626
; Maximum stack depth per invocation:  7 words
```

```
;  4965  1     !*
;  4966  1     GLOBAL ROUTINE DIO_RETPKT : NOVALUE =
;  4967  1
;  4968  1     !+
;  4969  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL DUP I/O TRANSFER
;  4970  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;  4971  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS.
;  4972  1     !
;  4973  1     !       IMPLICIT INPUTS:
;  4974  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  4975  1     !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  4976  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;  4977  1     !               DUOFF - CST OFFSET FOR THE CURRENT UNIT
;  4978  1     !               L$LUN - CURRENT UNIT NUMBER
;  4979  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  4980  1     !
;  4981  1     !       IMPLICIT OUTPUTS
;  4982  1     !               CST_ADDR [.DUOFF = OF DBN, NODUPMEDIA] - IF THIS BIT SET NO DUP EXERCISER
;  4983  1     !
;  4984  1     !-
;  4985  2     BEGIN
;  4986  2
;  4987  2     LOCAL   FLAG : BYTE INITIAL(BYTE(TRUE)),
;  4988  2             SUM2 : WORD,
;  4989  2             SUM : WORD;                                    ! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
;  4990  2     !PRINTX (DER18);
;  4991  2
;  4992  2     IF .RP_ADDR [STATUS] NEQU ST_SUC                               ! IF STATUS CODE INDICATES ERROR
;  4993  2     THEN
;  4994  3             BEGIN
;  4995  3             CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;              ! SET DUP ERROR FLAG
;  4996  3             HARD_ERROR ();
;  4997  3             IF .RP_ADDR [ENDCOD] EQLU (OP_ELP + OP_END) OR  ! IF ENDCODE IS EXECUTE LOCAL PROGRAM
;  4998  4                .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END)     !             OR GET DUST STATUS
;  4999  4             THEN    BEGIN
;  5000  4                     CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;            ! TURN OFF DUP EXERCISR
;  5001  3                     END;
;  5002  3             END
;  5003  2     ELSE                                           ! ELSE - I/O WAS SUCCESSFUL
;  5004  3         BEGIN
;  5005  3
;  5006  4         IF .PP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END) ! IF ENDCODE IS GET DUST STATUS
;  5007  3         THEN
;  5008  4             BEGIN
;  5009  4             IF .RP_ADDR [9,11,1,0] EQL 1
;  5010  4                     THEN CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = ACTIVE             ! CONTROLLER IN AN ACTIVE STAE
;  5011  4                     ELSE CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = IDLE;              ! CONTROLLER IN AN IDLE STATE
;  5012  4             IF .RP_ADDR [9,9,1,0] NEQ 1 THEN       ! TEST TO SEE IF CONTROLLER LOCAL PROGRAMS(PG 18 OF DUP DOC)
;  5013  5                     BEGIN
;  5014  5                     HARD_ERROR ();
;  5015  5                     CST_ADDR [.DUOFF + OF DBN, NODUPMEDIA] = 1;            ! TURN OFF DUP EXERCISR
;  5016  4                     END;
;  5017  3             END;
```

```
;    5018  3
;    5019  3
;    5020  3     IF (.RP_ADDR [ENDCOD] EQL (OP_RCD + OP_END)) AND
;    5021  3        (.DUPPKT [DUPTYPE] EQL 6) AND
;    5022  3        (.DUPPKT [DUPMESG] EQL 2) AND                !IF IT IS A RECEIVE DBN COMMAND WITH TYPE 6 AND MESSAGE 2 THEN
;    5023  4        (.CST_ADDR [.DUOFF + OF_DBN, DUPWRITE] EQLU 1)       ! IF WRITE FLAG SET IN CST TABLE THEN COMPARE BLOCKS
;    5024  3             THEN DUP_COMPARE ();
;    5025  3
;    5026  2     END;                                            ! COMPARE THE FOLLOWING 512 BYTES
;    5027  2
;    5028  2     PUT_RETPKT (.RP_INDX);
;    5029  1     END;                                            ! ROUTINE DIO_RETPKT


                                        .SBTTL  DIO.RETPKT MULTI DRIVE TEST ROUTINES
000000  010146                    DIO.RETPKT::
                                        MOV     R1,-(SP)                        ;                                       4966
000002  112700  000001                  MOVB    #1,R0                           ; *,FLAG                                4985
000006  013701  000000G                 MOV     RP.ADDR,R1                      ;                                       4992
000012  005761  000016                  TST     16(R1)
000016  001435                          BEQ     2$
000020  013700  001250'                 MOV     DUOFF,R0                        ;                                       4995
000024  006300                          ASL     R0
000026  063700  000000G                 ADD     CST.ADDR,R0
000032  052760  040000  000020          BIS     #40000,20(R0)
000040  004737  000000V                 JSR     PC;HARD.ERROR
000044  013700  000000G                 MOV     RP.ADDR,R0                      ;                                       4996
000050  126027  000014  000203          CMPB    14(R0),#203                     ;                                       4997
000056  001404                          BEQ     1$
000060  126027  000014  000201          CMPB    14(R0),#201                     ;                                       4998
000066  001130                          BNE     6$
000070  013700  001250'          1$:     MOV     DUOFF,R0                        ;                                       5000
000074  006300                          ASL     R0
000076  063700  000000G                 ADD     CST.ADDR,R0
000102  052760  100000  000020          BIS     #100000,20(R0)
000110  000467                          BR      6$
000112  126127  000014  000201    2$:     CMPB    14(R1),#201                     ;                                       4992
000120  001036                          BNE     5$                              ;                                       5006
000122  013700  001250'                 MOV     DUOFF,R0                        ;                                       5010
000126  006300                          ASL     R0
000130  063700  000000G                 ADD     CST.ADDR,R0
000134  032761  004000  000022          BIT     #4000,22(R1)                    ;                                       5009
000142  001404                          BEQ     3$
000144  052760  020000  000020          BIS     #20000,20(R0)                   ;                                       5010
000152  000403                          BR      4$                              ;                                       5009
000154  042760  020000  000020    3$:     BIC     #20000,20(R0)                   ;                                       5011
000162  032761  001000  000022    4$:     BIT     #1000,22(R1)                    ;                                       5012
000170  001012                          BNE     5$
000172  004737  000000V                 JSR     PC,HARD.ERROR                   ;                                       5014
000176  013700  001250'                 MOV     DUOFF,R0                        ;                                       5015
000202  006300                          ASL     R0
000204  063700  000000G                 ADD     CST.ADDR,R0
000210  052760  100000  000020          BIS     #100000,20(R0)
```

```
ZRQAM3          RD/RX EXERCISER                                 4-Apr-1985 13:23:31    VAX-11 Bliss-16 V4.1 582                SEQ 0400
V02.2           MULTI-DRIVE TEST ROUTINES                       2-Apr-1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;19    Page 145
                                                                                                                               (36)

000216  013700  000000G                       5$:     MOV     RP.ADDR,R0              ;                                       5020
000222  126027  000014  000205                        CMPB    14(R0),#205
000230  001017                                      BNE     6$
000232  023727  000000G 060002                      CMP     DUPPKT,#60002           ;                                       5021
000240  001013                                      BNE     6$
000242  013700  001250'                            MOV     DUOFF,R0                ;                                       5023
000246  006300                                      ASL     R0
000250  063700  000000G                           ADD     CST.ADDR,R0
000254  032760  010000  000020                     BIT     #10000,20(R0)
000262  001402                                      BEQ     6$
000264  004737  000000V                           JSR     PC,DUP.COMPARE          ;                                       5024
000270  013746  000000G                       6$:     MOV     RP.INDX,-(SP)           ;                                       5028
000274  004737  000000G                           JSR     PC,PUT.RETPKT
000300  005726                                      TST     (SP)+                   ;                                       4985
000302  012601                                      MOV     (SP)+,R1                ;                                       4966
000304  000207                                      RTS     PC

; Routine Size:  99 words.      Routine Base:  $CODE$ + 21022
; Maximum stack depth per invocation:  3 words


;   5030  1
```

```
;    5031  1     GLOBAL ROUTINE DUP_COMPARE : NOVALUE =
;    5032  1
;    5033  1
;    5034  1     !+
;    5035  1     !       THIS ROUTINE IS CALLED BY DIO_RETPKT WHEN THE RECEIVE DATA COMMAND
;    5036  1     !       IS BEING PROCESSED. THIS COMMAND COMPARES THE WRITTEN BUFFER WITH
;    5037  1     !       THE PATERN WORD GIVEN IN SEND DATA COMMAND.  FOR EVERY WORD COMPARED
;    5038  1     !       THE ROUTINE INCREMENTS THE TALLY TABLE.  IF THE COMPARE SHOWS AN
;    5039  1     !       ERROR. THE DBN HARD ERROR COUNTER WILL BE INCREMENTED AND THE
;    5040  1     !       THE DBN NUMBER AND BYTE COUNT WILL BE PRINTED.
;    5041  1     !
;    5042  1     !       IMPLICIT INPUTS:
;    5043  1     !               S_PATTERN       ! THE SAVED PATTERN WRITTEN TO THE DBN'S
;    5044  1     !               S_DUPPKT        ! THE POINTER FOR DUP BUFFER
;    5045  1     !               T_ADDR          ! THE ADDRESS OF THE TALLY TABLE FOR THIS UNIT
;    5046  1     !               CST_ADDR        ! THE ADDRESS OF PRESENT CONTROLLER STATUS TABLE
;    5047  1     !-
;    5048  2     BEGIN
;    5049  2
;    5050  2     OWN
;    5051  2             COUNT : WORD;
;    5052  2
;    5053  2     !PRINTX (DER19);
;    5054  2     S_DUPPKT = 0;
;    5055  2     INCR COUNT FROM 1 TO 256 DO     !INDEX PIONTER FOR DATA STORED IN MSCP ENV PACKET
;    5056  3             BEGIN
;    5057  3             S_DUPPKT = .S_DUPPKT + 2;                       ! INITIALLY THIS SKIPS THE FIRST WORD OF DUPPKT
;    5058  3             IF .(DUPPKT + .S_DUPPKT) NEQ .S_PATTERN THEN    !IF THE CONTENTS OF DBN DOESN'T EQUAL PATTERN
;    5059  4                     BEGIN
;    5060  4                     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;       ! SET DUP ERROR FLAG
;    5061  4                     ERRHRD (46, EH_10, EMS_22);             !LIST ERROR
;    5062  4                     EXITLOOP;
;    5063  3                     END;
;    5064  2             END;                                    !GO THROUGH ALL DBN WORDS
;    5065  1     END;                                            !END ROUTINE DUP-COMPARE


001304                                  .PSECT  $GGG$,  RO
001304                          COUNT:  .BLKW   1


                                        .SBTTL  DUP.COMPARE MULTI-DRIVE TEST ROUTINES
021330                                  .PSECT  $CODE$,  RO

000000  010146                  DUP.COMPARE::
                                        MOV     R1,-(SP)                ;                               5031
000002  005037  000000G                 CLR     S.DUPPKT                ;                               5054
000006  012701  000400                  MOV     #400,R1                 ; *.COUNT                       5055
000012  062737  000002  000000G 1$:     ADD     #2,S.DUPPKT             ;                               5057
000020  013700  000000G                 MOV     S.DUPPKT,R0             ;                               5058
000024  026037  000000G 000000G         CMP     DUPPKT(R0),S.PATTERN
000032  001415                          BEQ     2$
```

```
000034  013700  001250'                        MOV     DUOFF,R0                        ;                       5060
000040  006300                                  ASL     R0
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  052760  040000  000020                 BIS     #40000,20(R0)
000054  104456                                  TRAP    56                              ;                       5061
000056  000056                                  .WORD   56
000060  000000G                                .WORD   EH.10
000062  000000G                                .WORD   EMS.22
000064  000402                                  BR      3$                              ;                       5059
000066  005301                          2$:     DEC     R1                              ; COUNT                 5055
000070  001350                                  BNE     1$
000072  012601                          3$:     MOV     (SP)+,R1                        ;                       5031
000074  000207                                  RTS     PC

; Routine Size:  31 words,      Routine Base:  $CODE$ + 21330
; Maximum stack depth per invocation:  3 words


;    5066  1
;    5067  1
;    5068  1
```

```
;    5069  1     GLOBAL routine IO_RETPKT : novalue =
;    5070  1
;    5071  1     !+
;    5072  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
;    5073  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;    5074  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
;    5075  1     !       PERFORMING HOST WRITE-COMPARES IF REQUIRED.
;    5076  1     !
;    5077  1     !       IMPLICIT INPUTS:
;    5078  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    5079  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5080  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    5081  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    5082  1     !               L$LUN - CURRENT UNIT NUMBER
;    5083  1     !-
;    5084  1
;    5085  2         begin
;    5086  2
;    5087  2         local
;    5088  2             FLAG : byte initial (byte (TRUE));
;    5089  2
;    5090  2         FSET_UPAR ();                           ! FIND UNIT'S ENTRY IS CST AND SET UP UNIT-RELATED D  4
;    5091  2         ST_CODE   .RP_ADDR [STSCOD];            ! GET STATUS CODE FROM RETPKT
;    5092  2         SB_CODE = .RP_ADDR [SUBCOD];            ! GET SUB-CODE, IF ANY
;    5093  2
;    5094  3         if (.ST_CODE neq ST_SUC)                ! IF STATUS CODE INDICATES ERROR
;    5095  2         then
;    5096  3             begin
;    5097  3             HARD_ERROR ();                      ! UPDATE ERROR COUNT
;    5098  3             COMPARE_DATA = FALSE;               ! NO POINT IN DOING HOST COMPARES ON ERRORS
;    5099  3
;    5100  3             if (.ST_CODE neq ST_OFL) and        ! DROP UNIT IF ERROR COUNTS EXCEEDS LIMIT
;    5101  3                (.ST_CODE neq ST_AVL) and
;    5102  4                (.T_ADDR [ERR_HARD] gequ .SWP_ERROR)
;    5103  3             then
;    5104  4                 begin
;    5105  4                 DUR [.L$LUN] = DU_HERR;         ! LOAD REASON FOR DROPPING UNIT
;    5106  4                 DODU (.L$LUN);                  ! DROP UNIT
;    5107  3                 end;
;    5108  3
;    5109  3             end
;    5110  2         else                                    ! IF I/O WAS SUCCESSFUL
;    5111  3             begin
;    5112  3             UPD_IO_TALLY ();                    ! UPDATE I/O TALLY (STATISTICS)
;    5113  3
;    5114  4             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)
;    5115  3             then
;    5116  3                 COMPARE_DATA = TRUE;            ! HOST COMPARES MAY BE ALLOWED IF NO FURTHER ERRORS
;    5117  3
;    5118  3             if (BIT_TST (SWP_FLAGS, SWF_HWC)) and   ! IF HOST IS DOING WRITE-COMPARES
;    5119  4                (.COMPARE_DATA)
;    5120  3             then
;    5121  3                 FLAG = HOST_WRT_CHK ();         ! SAVE I/O PACKET OR DO WRITE-CHECK
```

```
;    5122  3
;    5123  2              end;
;    5124  2
;    5125  2          if .FLAG                          ! IF FLAG IS STILL TRUE
;    5126  2          then
;    5127  2              SWEEP ();                     ! DEALLOCATE BUFFER(S) AND RETPKT(S)
;    5128  2
;    5129  2          QIO [.CCTLR] = .QIO [.CCTLR] - 1; ! DECREMENT NO. OF OUTSTANDING QIOs
;    5130  1          end;                              ! ROUTINE IO_RETPKT


                                        .SBTTL  IO.RETPKT MULTI DRIVE TEST ROUTINES
000000  004137  000000G          IO.RETPKT::
                                        JSR     R1,$SAVE2                       ;                       5069
000004  112701  000001                  MOVB    #1,R1                           ; *,FLAG                5085
000010  004737  000000V                  JSR     PC,FSET.UPAR                    ;                       5090
000014  013700  000000G                  MOV     RP.ADDR,R0                      ;                       5091
000020  116037  000016  000000G          MOVB    16(R0),ST.CODE
000026  042737  177740  000000G          BIC     #177740,ST.CODE
000034  016002  000016                   MOV     16(R0),R2                       ;                       5092
000040  006202                           ASR     R2
000042  006202                           ASR     R2
000044  006202                           ASR     R2
000046  006202                           ASR     R2
000050  006202                           ASR     R2
000052  042702  174000                   BIC     #174000,R2
000056  010237  000000G                  MOV     R2,SB.CODE
000062  005737  000000G                  TST     ST.CODE                         ;                       5094
000066  001431                           BEQ     1$
000070  004737  000000V                  JSR     PC,HARD.ERROR                   ;                       5097
000074  105037  001262'                  CLRB    COMPARE.DATA                    ;                       5098
000100  023727  000000G 000003          CMP     ST.CODE,#3                      ;                       5100
000106  001447                           BEQ     3$
000110  023727  000000G 000004          CMP     ST.CODE,#4                      ;                       5101
000116  001443                           BEQ     3$
000120  013700  000000G                  MOV     T.ADDR,R0                       ;                       5102
000124  026037  000014  000000G          CMP     14(R0),SWP.ERROR
000132  103435                           BLO     3$
000134  013700  000000G                  MOV     L$LUN,R0                        ;                       5105
000140  112760  000004  000000G          MOVB    #4,DUR(R0)
000146  104451                           TRAP    51                              ;                       5106
000150  000426                           BR      3$                              ;                       5094
000152  004737  000000V          1$:     JSR     PC,UPD.IO.TALLY                 ;                       5112
000156  013700  000000G                  MOV     RP.ADDR,R0                      ;                       5114
000162  126027  000014  000242          CMPB    14(R0),#242
000170  001003                           BNE     2$
000172  112737  000001  001262'          MOVB    #1,COMPARE.DATA                 ;                       5116
000200  032737  000040  000000G  2$:     BIT     #40,SWP.FLAGS                   ;                       5118
000206  001407                           BEQ     3$
000210  032737  000001  001262'          BIT     #1,COMPARE.DATA                 ;                       5119
000216  001403                           BEQ     3$
000220  004737  000000V                  JSR     PC,HOST.WRT.CHK                 ;                       5121
000224  110001                           MOVB    R0,R1                           ; *,FLAG
```

F16


```
000226  006001                          3$:     ROR     R1                      ; FLAG                  5125
000230  103002                                  BCC     4$                                              5127
000232  004737  000000V                        JSR     PC,SWEEP                ;                       5129
000236  013700  000000G                4$:     MOV     CCTLR,R0                ;                       5069
000242  105360  000000G                        DECB    QIO(R0)
000246  000207                                  RTS     PC                      ;

; Routine Size:  84 words,      Routine Base:  $CODE$ + 21426
; Maximum stack depth per invocation:  5 words
```

```
;    5131  1      GLOBAL routine FSET_UPAR : novalue =
;    5132  1
;    5133  1      !+
;    5134  1      !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
;    5135  1      !       CONTROLLER STATUS TABLE (CST) FOR THE DISK ADDRESS WHICH IS
;    5136  1      !       CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
;    5137  1      !       CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT RELATED
;    5138  1      !       DATA PARAMETERS.
;    5139  1      !
;    5140  1      !       IMPLICIT INPUTS:
;    5141  1      !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    5142  1      !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    5143  1      !-
;    5144  1
;    5145  2          begin
;    5146  2
;    5147  2          incr OFFSET from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    5148  2
;    5149  2              if .CST_ADDR [.OFFSET * OF_DATA, D_DISK_NUM] eql .RP_ADDR [DISK]        ! IF RETPKT UNIT MATCHES CST ENTR
;    5150  2              then
;    5151  3                  begin
;    5152  3                  SET_UPAR (.OFFSET);                                                 ! SET UP UNIT-RELATED DATA
;    5153  3                  return;                                                             ! DONE
;    5154  2                  end;
;    5155  2
;    5156  2          PRINTF (DBM19, .RP_ADDR [DISK], .CCTLR);                                    ! "CAN'T FIND DISK XXX IN CST X"
;    5157  1          end;                                                                        ! ROUTINE FSET_UPAR


                                        .SBTTL  FSET.UPAR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          FSET.UPAR::
                                        JSR     R1,$SAVE4                       ;                               5131
000004  012702  000003                  MOV     #3,R2                           ; *,OFFSET                      5147
000010  010201                  1$:     MOV     R2,R1                           ; OFFSET,*                      5149
000012  006301                          ASL     R1
000014  063701  000000G                 ADD     CST.ADDR,R1
000020  013700  000000G                 MOV     RP.ADDR,R0
000024  016004  000010                  MOV     10(R0),R4
000030  111103                          MOVB    (R1),R3
000032  042703  177760                  BIC     #177760,R3
000036  020304                          CMP     R3,R4
000040  001005                          BNE     2$
000042  010246                          MOV     R2,-(SP)                        ; OFFSET,*                      5152
000044  004737  000000G                 JSR     PC,SET.UPAR
000050  005726                          TST     (SP)+                           ;                               5153
000052  000207                          RTS     PC                              ;                               5151
000054  062702  000012          2$:     ADD     #12,R2                          ; *,OFFSET                      5147
000060  020227  000041                  CMP     R2,#41                          ; OFFSET,*
000064  003751                          BLE     1$
000066  013746  000000G                 MOV     CCTLR,-(SP)                     ;                               5156
000072  013700  000000G                 MOV     RP.ADDR,R0
000076  016046  000010                  MOV     10(R0),-(SP)
000102  012746  000000G                 MOV     #DBM19,-(SP)
```

```
ZRQAM3          RD/RX EXERCISER                                 4-Apr-1985 13:23:31    VAX-11 Bliss-16 V4.1 582              SEQ 0407
V02.2           MULTI-DRIVE TEST ROUTINES                       2-Apr-1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAG0.BL2;19  Page 152
                                                                                                                             (39)

000106  012746  000003                          MOV     #3, (SP)
000112  010600                                  MOV     SP,R0                           ; SP,*
000114  104417                                  TRAP    17
000116  062706  000010                          ADD     #10,SP                          ;                                      5145
000122  000207                                  RTS     PC                              ;                                      5131

; Routine Size:  42 words,      Routine Base:  $CODE$ + 21676
; Maximum stack depth per invocation:  11 words
```

```
;    5158  1      GLOBAL routine HARD_ERROR : novalue =
;    5159  1
;    5160  1      !+
;    5161  1      !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO INCREMENT THE HARD
;    5162  1      !       ERROR STATISTIC FIELD FOR THE CURRENT UNIT. IF THE HARD ERROR COUNT
;    5163  1      !       HAS EXCEEDED THE OPERATOR-SPECIFIED LIMIT, THEN THE UNIT IS DROPPED
;    5164  1      !       FROM TESTING.
;    5165  1      !
;    5166  1      !       IMPLICIT INPUTS:
;    5167  1      !               .$LUN - CURRENT UNIT NUMBER
;    5168  1      !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    5169  1      !               CUOFF - CST OFFSET FOR CURRENT UNIT
;    5170  1      !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    5171  1      !-
;    5172  1
;    5173  2          begin
;    5174  2          T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;                          ! INCREMENT UNIT'S HARD ERROR COUNT
;    5175  2      if .RP_ADDR [CONID] EQL CID_MSCP                                         !FOR MSCP ERRORS   ZZZ
;    5176  2      THEN                                                                     !                  ZZZ
;    5177  2
;    5178  2          selectoneu .ST_CODE of
;    5179  2              set
;    5180  2
;    5181  2              [ST_SUC]:       if .SB_CODE neq 0                            ! SUCCESS WITH NON ZERO SUB CODE
;    5182  2                              then
;    5183  3                                  begin
;    5184  3
;    5185  3                                  if .SB_CODE eql 4
;    5186  3                                  then
;    5187  3                                      T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1
;    5188  3                                  else
;    5189  3                                      T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5190  3
;    5191  3                                  if .APT_MODE
;    5192  3                                  then
;    5193  3                                      ERR_HRD_RTNE_APT (44)
;    5194  3                                  else
;    5195  3                                      ERR_HRD_RTNE (44);
;    5196  3
;    5197  2                                  end;
;    5198  2
;    5199  3                      [ST_CMD]:   begin                                    ! INVALID COMMAND
;    5200  3                                  T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5201  3
;    5202  3                                  if .APT_MODE
;    5203  3                                  then
;    5204  3                                      ERR_HRD_RTNE_APT (31)
;    5205  3                                  else
;    5206  3                                      ERR_HRD_RTNE (31);
;    5207  3
;    5208  2                                  end;
;    5209  2
;    5210  3                      [ST_ABO]:   begin                                    ! COMMMAND ABORTED
```

```
;  5211  3                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5212  3
;  5213  3                    if .APT_MODE
;  5214  3                    then
;  5215  3                        ERR_HRD_RTNE_APT (32)
;  5216  3                    else
;  5217  3                        ERR_HRD_RTNE (32);
;  5218  3
;  5219  2                    end;
;  5220  2
;  5221  3            [ST_OFL] :      begin                                   ! OFFLINE
;  5222  3                           T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5223  3
;  5224  3                           if .APT_MODE
;  5225  3                           then
;  5226  4                               begin
;  5227  4                               .MAIL_BOX_TESTNUM = 1;
;  5228  4                               .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  5229  3                               end;
;  5230  3
;  5231  3                           ERRDF (18, EGD_18, EMS_18);             !
;  5232  3                           DUR [.L$LUN] = DU_DFATAL;               ! DEVICE FATAL ERROR
;  5233  3                           DODU (.L$LUN);                          ! DROP UNIT
;  5234  2                           end;
;  5235  2
;  5236  3            [ST_AVL]:       begin                                   ! WENT TO AVAILABLE STATE
;  5237  3                           T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5238  3
;  5239  3                           if .APT_MODE
;  5240  3                           then
;  5241  4                               begin
;  5242  4                               .MAIL_BOX_TESTNUM = 1;
;  5243  4                               .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  5244  3                               end;
;  5245  3
;  5246  3                           ERRDF (24, EGD_18, EMS_24);             !
;  5247  3                           DUR [.L$LUN] = DU_DFATAL;               ! DEVICE FATAL ERROR
;  5248  3                           DODU (.L$LUN);                          ! DROP UNIT
;  5249  2                           end;
;  5250  2
;  5251  3            [ST_MFE]:       begin                                   ! MEDIA FORMAT ERROR
;  5252  3                           T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1;
;  5253  3
;  5254  3                           if .APT_MODE
;  5255  3                           then
;  5256  3                               ERR_HRD_RTNE_APT (35)
;  5257  3                           else
;  5258  3                               ERR_HRD_RTNE (35);
;  5259  3
;  5260  2                           end;
;  5261  2
;  5262  3            [ST_WPT]:       begin                                   ! DEVICE WRITE PROTECTED
;  5263  3
```

```
;    5264  3                          if .SB CODE eql 128
;    5265  3                          then
;    5266  3                              T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1
;    5267  3                          else
;    5268  3                              T ADDR [ERR_HRD_DRV] = .T_ADDR [ERR HRD_DRV] + 1;
;    5269  3
;    5270  3                          if .APT_MODE
;    5271  3                          then
;    5272  3                              ERR_HRD_RTNE_APT (36)
;    5273  3                          else
;    5274  3                              ERR_HRD RTNE (36);
;    5275  3
;    5276  2                          end;
;    5277  2
;    5278  3                  [ST_CMP]:       begin                                           ! COMPARE ERROR
;    5279  3                          T_ADDR [ERR HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;    5280  3
;    5281  3                          if .APT MODE
;    5282  3                          then
;    5283  3                              ERR_HRD_RTNE_APT (37)
;    5284  3                          else
;    5285  3                              ERR_HRD_RTNE (37);
;    5286  3
;    5287  2                          end;
;    5288  2
;    5289  3                  [ST_DAT]:       begin                                           ! DATA ERROR
;    5290  3
;    5291  3                          if .SB_CODE eql 2
;    5292  3                          then
;    5293  3                              T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;    5294  3                          else
;    5295  3                              T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;    5296  3
;    5297  3                          if (.SB_CODE eql 0) and
;    5298  3                             (not .FORCED_ERROR) and
;    5299  4                             (BIT_TST (SWP_FLAGS, SWF_FER))
;    5300  3                          then
;    5301  4                              begin
;    5302  4                              FORCED_ERROR = TRUE;                                ! BLOCK WITH "FORCED ERROR" FOUND
;    5303  4                              FERO_LBN = .RP_ADDR [LBN_LO];                       !
 ZZZ
;    5304  4                              FER1_LBN = .RP_ADDR [LBN_HI];                       !
 ZZZ
;    5305  4                              FER_BC = .RP_ADDR [CBCNT_LO];                       !
;    5306  3                              end;
;    5307  3
;    5308  3
;    5309  3                          if .APT_MODE
;    5310  3                          then
;    5311  3                              ERR_HRD_RTNE_APT (38)
;    5312  3                          else
;    5313  3                              ERR_HRD RTNE (38);
;    5314  3
;    5315  2                          end;
;    5316  2
```

```
;  5317  3              [ST HST]:       begin                                                   ! HOST ACCESS ERROR
;  5318  3                              T ADDR [ERR HRD_HST] = .T ADDR [ERR HRD HST] + 1;
;  5319  3
;  5320  3                              if .APT_MODE
;  5321  3                              then
;  5322  3                                  ERR_HRD_RTNE_APT (39)
;  5323  3                              else
;  5324  3                                  ERR_HRD RTNE (39);
;  5325  3
;  5326  2                              end;
;  5327  2
;  5328  3              [ST_CNT]:       begin                                                   ! CONTROLLER ERROR
;  5329  3                              T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5330  3
;  5331  3                              if .APT_MODE
;  5332  3                              then
;  5333  3                                  ERR_HRD_RTNE_APT (40)
;  5334  3                              else
;  5335  3                                  ERR_HRD_RTNE (40);
;  5336  3
;  5337  2                              end;
;  5338  2
;  5339  3              [ST DRV]:       begin                                                   ! DRIVE ERROR
;  5340  3
;  5341  3                              if .SB_CODE eql 3
;  5342  3                              then
;  5343  3                                  T_ADDR [ERR_HRD_SEK] = .T ADDR [ERR_HRD_SEK] + 1
;  5344  3                              else
;  5345  3                                  T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR HRD_DRV] + 1;
;  5346  3
;  5347  3                              if .APT_MODE
;  5348  3                              then
;  5349  3                                  ERR_HRD_RTNE_APT (41)
;  5350  3                              else
;  5351  3                                  ERR_HRD_RTNE (41);
;  5352  3
;  5353  2                              end;
;  5354  2
;  5355  3              [ST_DIA]:       begin                                                   ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  5356  3                              T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5357  3
;  5358  3                              if .APT_MODE
;  5359  3                              then
;  5360  3                                  ERR_HRD_RTNE_APT (43)
;  5361  3                              else
;  5362  3                                  ERR_HRD_RTNE (43);
;  5363  3
;  5364  2                              end;
;  5365  2
;  5366  3              [otherwise]:    begin                                                   ! PRINT STATUS CODE IF NO MATCH
;  5367  3                              C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  5368  3
;  5369  3                              if .APT MODE
```

```
;    5370  3                          then
;    5371  3                              ERR_HRD_RTNE_APT (45)
;    5372  3                          else
;    5373  3                              ERR_HRD_RTNE (45);
;    5374  3
;    5375  2                          end;
;    5376  2
;    5377  2              tes;
;    5378  2
;    5379  2         if .RP_ADDR [CONID] EQL CID_DUP                          !FOR DUP ERRORS                    ZZZ
;    5380  2         OR .D_FAIL EQL 1                                         !EVEN IF UNRECOGNIZABLE AS SUCH    ZZZ
;    5381  2         THEN                                                     !                                  ZZZ
;    5382  2
;    5383  2             selectoneu .RP_ADDR [STSCOD] of
;    5384  2                 SET
;    5385  3                 [%o'0']          : begin                       ! if status code succesful
;    5386  3                                    if .RP_ADDR [ENDCOD] EQLU (OP_GDS * OP_END) and ! IF ENDCODE IS GET DUST STATUS
;    5387  3                                       .RP_ADDR [9,9,1,0] NEQ 1                     ! TEST TO SEE IF CONTROLLER LOCAL PR
OGRAMS
;    5388  3                                    then                                            ! (PG 18 OF DUP DOC)
;    5389  4                                        BEGIN
;    5390  4                                        ERR_HRD_RTNE (60);       !UNABLE TO LOAD LOCAL CONTROLLER DUP
 MEDIA !ZZZ
;    5391  4                                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    5392  4                                        END
;    5393  3                                    else
;    5394  4                                        begin
;    5395  5                                        if (.DUPPKT [DUPTYPE] eql 5)    ! if fatal error
;    5396  4                                        then
;    5397  4     !ZZZ                                   begin                       !DON'T DROP DEVICE ON DUP ERROR
       ZZZ
;    5398  4     !ZZZ                               DUR [.L$LUN] = DU_DFATAL;       !GIVE F.E. A CHANCE TO SEE ERRORS
       ZZZ
;    5399  4     !ZZZ                               DODU (.L$LUN);                  ! FATAL DEVICE ERROR DROP UNIT);
       ZZZ
;    5400  4     !ZZZ                               end;                            ! SET REASON FOR DROPPING UNIT
       ZZZ
;    5401  4                                    selectoneu .DUPPKT [DUPMESG] of
;    5402  4                                        SET
;    5403  5                                        [%o'1'] : begin
;    5404  5                                                  ERR_HRD_RTNE (62);          ! illegal unit number     !ZZZ
;    5405  5                                                  T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5406  4                                                  end;
;    5407  5                                        [%o'2'] : begin
;    5408  5                                                  ERR_HRD_RTNE (63);          ! illegal relative or physical b
lock # !ZZZ
;    5409  5                                                  T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5410  4                                                  end;
;    5411  5                                        [%o'3'] : begin
;    5412  5                                                  ERR_HRD_RTNE (64);          ! device error    !ZZZ
;    5413  5                                                  T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    5414  4                                                  end;
;    5415  5                                        [%o'4'] : begin
;    5416  5                                                  ERR_HRD_RTNE (65);          ! zero lenght message   !ZZZ
;    5417  5                                                  T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5418  4                                                  end;
;    5419  5                                        [OTHERWISE] : begin
;    5420  5                                                  ERR_HRD_RTNE (66);      ! DUP UNKNOWN STATUS CODE   !ZZZ
;    5421  5                                                  C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HR
D] + 1;
;    5422  4                                                  end;
```

```
;    5423  4                                                  tes;
;    5424  3                                              end;
;    5425  3
;    5426  2                                          end;
;    5427  3                          [%o'1']           : begin
;    5428  3                                          ERR_HRD_RTNE (67);      ! INVALID COMMAND              !ZZZ
;    5429  3                                          T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    5430  2                                          end;
;    5431  3                          [%o'2']           : begin
;    5432  3                                          ERR_HRD_RTNE (68);      ! NO REGION AVAILABLE          !ZZZ
;    5433  3                                          T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    5434  2                                          end;
;    5435  3                          [%o'3']           : begin
;    5436  3                                          ERR_HRD_RTNE (69);              ! NO REGION SUITABLE          !ZZZ
;    5437  3                                          T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5438  2                                          end;
;    5439  3                          [%o'4']           : begin
;    5440  3                                          ERR_HRD_RTNE (70);      ! PROGRAM NOT KNOWN            !ZZZ
;    5441  3                                          T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5442  2                                          end;
;    5443  3                          [%o'5']           : begin
;    5444  3                                          ERR_HRD_RTNE (71);              ! LOAD FAILURE                !ZZZ
;    5445  3                                          T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    5446  2                                          end;
;    5447  3                          [%o'6']           : begin
;    5448  3                                          ERR_HRD_RTNE (72);              ! STANDALONE                  !ZZZ
;    5449  3                                          T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5450  2                                          end;
;    5451  3                          [OTHERWISE]   : begin
;    5452  3                                          ERR_HRD_RTNE (73);              ! DUP UNKNOWN STATUS CODE    !ZZZ
;    5453  3                                          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    5454  2                                          end;
;    5455  2                          TES;
;    5456  2
;    5457  1          end;                                                                    ! ROUTINE HARD_ERROR


                                                    .SBTTL  HARD.ERROR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                    HARD.ERROR::
                                                    JSR     R1,$SAVE4                       ;                       5158
000004  013701  000000G                             MOV     T.ADDR,R1                       ;                       5174
000010  005261  000014                              INC     14(R1)
000014  013703  000000G                             MOV     RP.ADDR,R3                      ;                       5175
000020  105763  000003                              TSTB    3(R3)
000024  001171                                      BNE     12$
000026  013702  000000G                             MOV     ST.CODE,R2                      ;                       5178
000032  001027                                      BNE     4$                              ;                       5181
000034  013704  000000G                             MOV     SB.CODE,R4
000040  001563                                      BEQ     12$
000042  012700  000050                              MOV     #50,R0                          ;                       5187
000046  060100                                      ADD     R1,R0
000050  020427  000004                              CMP     R4,#4                           ;                       5185
000054  001002                                      BNE     1$
```

```
000056  105210                          INCB    (R0)                    ;                       5187
000060  000402                          BR      2$                      ;                       5185
000062  105260  000001                  1$:     INCB    1(R0)           ;                       5189
000066  032737  000001  001254'        2$:     BIT     #1,APT.MODE     ;                       5191
000074  001403                          BEQ     3$
000076  012746  000054                  MOV     #54,-(SP)               ;                       5193
000102  000557                          BR      14$
000104  012746  000054                  3$:     MOV     #54,-(SP)       ;                       5195
000110  000557                          BR      16$
000112  020227  000001                  4$:     CMP     R2,#1           ;                       5199
000116  001014                          BNE     6$
000120  105261  000051                  INCB    51(R1)                  ;                       5200
000124  032737  000001  001254'        BIT     #1,APT.MODE             ;                       5202
000132  001403                          BEQ     5$
000134  012746  000037                  MOV     #37,-(SP)               ;                       5204
000140  000570                          BR      20$
000142  012746  000037                  5$:     MOV     #37,-(SP)       ;                       5206
000146  000571                          BR      22$
000150  020227  000002                  6$:     CMP     R2,#2           ;                       5210
000154  001014                          BNE     8$
000156  105261  000050                  INCB    50(R1)                  ;                       5211
000162  032737  000001  001254'        BIT     #1,APT.MODE             ;                       5213
000170  001403                          BEQ     7$
000172  012746  000040                  MOV     #40,-(SP)               ;                       5215
000176  000571                          BR      24$
000200  012746  000040                  7$:     MOV     #40,-(SP)       ;                       5217
000204  000571                          BR      26$
000206  020227  000003                  8$:     CMP     R2,#3           ;                       5221
000212  001036                          BNE     10$
000214  105261  000050                  INCB    50(R1)                  ;                       5222
000220  032737  000001  001254'        BIT     #1,APT.MODE             ;                       5224
000226  001415                          BEQ     9$
000230  012777  000001  001256'        MOV     #1,@MAIL.BOX.TESTNUM    ;                       5227
000236  013700  000000G                 MOV     CUOFF,R0                ;                       5228
000242  006300                          ASL     R0
000244  063700  000000G                 ADD     CST.ADDR,R0
000250  111077  001260'                 MOVB    (R0),@MAIL.BOX.SUBTST
000254  042777  177760  001260'        BIC     #177760,@MAIL.BOX.SUBTST
000262  104455                          9$:     TRAP    55              ;                       5231
000264  000022                          .WORD   22
000266  000000G                         .WORD   EGD.18
000270  000000G                         .WORD   EMS.18
000272  013700  000000G                 MOV     L$LUN,R0                ;                       5232
000276  112760  000005  000000G         MOVB    #5,DUR(R0)
000304  104451                          TRAP    51                      ;                       5233
000306  000440                          BR      12$                     ;                       5178
000310  020227  000004                  10$:    CMP     R2,#4           ;                       5236
000314  001037                          BNE     13$
000316  105261  000050                  INCB    50(R1)                  ;                       5237
000322  032737  000001  001254'        BIT     #1,APT.MODE             ;                       5239
000330  001415                          BEQ     11$
000332  012777  000001  001256'        MOV     #1,@MAIL.BOX.TESTNUM    ;                       5242
000340  013700  000000G                 MOV     CUOFF,R0                ;                       5243
```

```
000344  006300                                  ASL     R0
000346  063700  000000G                         ADD     CST.ADDR,R0
000352  111077  001260'                         MOVB    (R0),@MAIL.BOX.SUBTST
000356  042777  177760  001260'                 BIC     #177760,@MAIL.BOX.SUBTST
000364  104455                          11$:    TRAP    55                              ;5246
000366  000030                                  .WORD   30
000370  000000G                                 .WORD   EGD.18
000372  000000G                                 .WORD   EMS.24
000374  013700  000000G                         MOV     L$LUN,R0                        ;5247
000400  112760  000005  000000G                 MOVB    #5,DUR(R0)
000406  104451                                  TRAP    51                              ;5248
000410  000137  023226'                  12$:    JMP     51$                             ;5178
000414  020227  000005                  13$:    CMP     R2,#5                           ;5251
000420  001014                                  BNE     17$
000422  105261  000046                          INCB    46(R1)                          ;5252
000426  032737  000001  001254'                BIT     #1,APT.MODE                     ;5254
000434  001403                                  BEQ     15$
000436  012746  000043                          MOV     #43,-(SP)                       ;5256
000442  000564                          14$:    BR      35$
000444  012746  000043                  15$:    MOV     #43,-(SP)                       ;5258
000450  000564                          16$:    BR      37$
000452  020227  000006                  17$:    CMP     R2,#6                           ;5262
000456  001026                                  BNE     23$
000460  012700  000050                          MOV     #50,R0                          ;5266
000464  060100                                  ADD     R1,R0
000466  023727  000000G 000200                  CMP     SB.CODE,#200                    ;5264
000474  001003                                  BNE     18$
000476  105260  000001                          INCB    1(R0)                           ;5266
000502  000401                                  BR      19$                             ;5264
000504  105210                          18$:    INCB    (R0)                            ;5268
000506  032737  000001  001254'          19$:    BIT     #1,APT.MODE                     ;5270
000514  001404                                  BEQ     21$
000516  012746  000044                          MOV     #44,-(SP)                       ;5272
000522  000137  023150'                  20$:    JMP     43$
000526  012746  000044                  21$:    MOV     #44,-(SP)                       ;5274
000532  000416                          22$:    BR      26$
000534  020227  000007                  23$:    CMP     R2,#7                           ;5278
000540  001014                                  BNE     27$
000542  105261  000047                          INCB    47(R1)                          ;5279
000546  032737  000001  001254'                BIT     #1,APT.MODE                     ;5281
000554  001403                                  BEQ     25$
000556  012746  000045                          MOV     #45,-(SP)                       ;5283
000562  000561                          24$:    BR      43$
000564  012746  000045                  25$:    MOV     #45,-(SP)                       ;5285
000570  000561                          26$:    BR      45$
000572  020227  000010                  27$:    CMP     R2,#10                          ;5289
000576  001054                                  BNE     32$
000600  012700  000046                          MOV     #46,R0                          ;5293
000604  060100                                  ADD     R1,R0
000606  023727  000000G 000002                  CMP     SB.CODE,#2                      ;5291
000614  001002                                  BNE     28$
000616  105210                                  INCB    (R0)                            ;5293
000620  000402                                  BR      29$                             ;5291
```

```
000622  105260  000001                  28$:    INCB    1(R0)                           ;                       5295
000626  005737  000000G                 29$:    TST     $B.CODE                         ;                       5297
000632  001024                                  BNE     30$
000634  132737  000001  000000G                BITB    #1,FORCED.ERROR                 ;                       5298
000642  001020                                  BNE     30$
000644  032737  004000  000000G                BIT     #4000,SWP.FLAGS                 ;                       5299
000652  001414                                  BEQ     30$
000654  112737  000001  000000G                MOVB    #1,FORCED.ERROR                 ;                       5302
000662  016337  000050  000000G                MOV     50(R3),FER0.LBN                 ;                       5303
000670  016337  000052  000000G                MOV     52(R3),FER1.LBN                 ;                       5304
000676  016337  000044  000000G                MOV     44(R3),FER.BC                   ;                       5305
000704  032737  000001  001254'        30$:    BIT     #1,APT.MODE                     ;                       5309
000712  001403                                  BEQ     31$
000714  012746  000046                          MOV     #46,-(SP)                       ;                       5311
000720  000521                                  BR      47$
000722  012746  000046                  31$:    MOV     #46,-(SP)                       ;                       5313
000726  000523                                  BR      49$
000730  020227  000011                  32$:    CMP     R2,#11                          ;                       5317
000734  001014                                  BNE     34$
000736  105261  000051                          INCB    51(R1)                          ;                       5318
000742  032737  000001  001254'                BIT     #1,APT.MODE                     ;                       5320
000750  001403                                  BEQ     33$
000752  012746  000047                          MOV     #47,-(SP)                       ;                       5322
000756  000502                                  BR      47$
000760  012746  000047                  33$:    MOV     #47,-(SP)                       ;                       5324
000764  000504                                  BR      49$
000766  020227  000012                  34$:    CMP     R2,#12                          ;                       5328
000772  001014                                  BNE     38$
000774  105261  000050                          INCB    50(R1)                          ;                       5329
001000  032737  000001  001254'                BIT     #1,APT.MODE                     ;                       5331
001006  001403                                  BEQ     36$
001010  012746  000050                          MOV     #50,-(SP)                       ;                       5333
001014  000463                          35$:    BR      47$
001016  012746  000050                  36$:    MOV     #50,-(SP)                       ;                       5335
001022  000465                          37$:    BR      49$
001024  020227  000013                  38$:    CMP     R2,#13                          ;                       5339
001030  001023                                  BNE     42$
001032  023727  000000G 000003                  CMP     $B.CODE,#3                      ;                       5341
001040  001003                                  BNE     39$
001042  105261  000046                          INCB    46(R1)                          ;                       5343
001046  000402                                  BR      40$                             ;                       5341
001050  105261  000050                  39$:    INCB    50(R1)                          ;                       5345
001054  032737  000001  001254'        40$:    BIT     #1,APT.MODE                     ;                       5347
001062  001403                                  BEQ     41$
001064  012746  000051                          MOV     #51,-(SP)                       ;                       5349
001070  000435                                  BR      47$
001072  012746  000051                  41$:    MOV     #51,-(SP)                       ;                       5351
001076  000437                                  BR      49$
001100  020227  000037                  42$:    CMP     R2,#37                          ;                       5355
001104  001014                                  BNE     46$
001106  105261  000050                          INCB    50(R1)                          ;                       5356
001112  032737  000001  001254'                BIT     #1,APT.MODE                     ;                       5358
001120  001403                                  BEQ     44$
```

F1


```
001122  012746  000053                        MOV     #53,-(SP)                       ;                       5360
001126  000416                          43$:   BR      47$
001130  012746  000053                  44$:   MOV     #53,-(SP)                       ;                       5362
001134  000420                          45$:   BR      49$
001136  013700  000000G                 46$:   MOV     CCTLR,R0                        ;                       5367
001142  006300                                 ASL     R0
001144  105260  000000G                         INCB    C.ERR.TBL(R0)
001150  032737  000001  001254'                BIT     #1,APT.MODE                     ;                       5369
001156  001405                                 BEQ     48$
001160  012746  000055                         MOV     #55,-(SP)                       ;                       5371
001164  004737  000000V                 47$:   JSR     PC,ERR.HRD.RTNE.APT
001170  000404                                 BR      50$                             ;                       5369
001172  012746  000055                  48$:   MOV     #55,-(SP)                       ;                       5373
001176  004737  000000V                  49$:  JSR     PC,ERR.HRD.RTNE
001202  005726                          50$:   TST     (SP)+                           ;                       5366
001204  013700  000000G                 51$:   MOV     RP.ADDR,R0                      ;                       5379
001210  126027  000003  000002                CMPB    3(R0),#2
001216  001404                                 BEQ     52$
001220  123727  000000G 000001                CMPB    D.FAIL,#1                       ;                       5380
001226  001160                                 BNE     69$
001230  116001  000016                  52$:   MOVB    16(R0),R1                       ;                       5383
001234  0 701   177740                         BIC     #177740,R1
001240  001067                                 BNE     59$                             ;                       5385
001242  126027  000014  000201                CMPB    14(R0),#201                     ;                       5386
001250  001015                                 BNE     54$
001252  032760  001000  000022                BIT     #1000,22(R0)                    ;                       5387
001260  001011                                 BNE     54$
001262  012746  000074                         MOV     #74,-(SP)                       ;                       5390
001266  004737  000000V                 53$:   JSR     PC,ERR.HRD.RTNE
001272  013700  000000G                         MOV     T.ADDR,R0                       ;                       5391
001276  105260  000050                         INCB    50(R0)
001302  000531                                 BR      68$                             ;                       5389
001304  013700  000000G                 54$:   MOV     DUPPKT,R0                       ;                       5395
001310  042700  007777                         BIC     #7777,R0
001314  020027  050000                         CMP     R0,#50000
001320  001123                                 BNE     69$
001322  013701  000000G                         MOV     DUPPKT,R1                       ;                       5401
001326  042701  170000                         BIC     #170000,R1
001332  020127  000001                         CMP     R1,#1                           ;                       5403
001336  001003                                 BNE     55$
001340  012746  000076                         MOV     #76,-(SP)                       ;                       5404
001344  000470                                 BR      65$
001346  020127  000002                  55$:   CMP     R1,#2                           ;                       5407
001352  001003                                 BNE     56$
001354  012746  000077                         MOV     #77,-(SP)                       ;                       5408
001360  000462                                 BR      65$
001362  020127  000003                  56$:   CMP     R1,#3                           ;                       5411
001366  001003                                 BNE     57$
001370  012746  000100                         MOV     #100,-(SP)                      ;                       5412
001374  000734                                 BR      53$
001376  020127  000004                  57$:   CMP     R1,#4                           ;                       5415
001402  001003                                 BNE     58$
001404  012746  000101                         MOV     #101,-(SP)                      ;                       5416
```

```
001410  000446                          BR      65$
001412  012746  000102         58$:     MOV     #102,-(SP)              ;                       5420
001416  000454                          BR      67$
001420  020127  000001         59$:     CMP     R1,#1                   ;                       5427
001424  001003                          BNE     60$
001426  012746  000103                  MOV     #103,-(SP)              ;                       5428
001432  000715                          BR      53$
001434  020127  000002         60$:     CMP     R1,#2                   ;                       5431
001440  001003                          BNE     61$
001442  012746  000104                  MOV     #104,-(SP)              ;                       5432
001446  000707                          BR      53$
001450  020127  000003         61$:     CMP     R1,#3                   ;                       5435
001454  001003                          BNE     62$
001456  012746  000105                  MOV     #105,-(SP)              ;                       5436
001462  000421                          BR      65$
001464  020127  000004         62$:     CMP     R1,#4                   ;                       5439
001470  001003                          BNE     63$
001472  012746  000106                  MOV     #106,-(SP)              ;                       5440
001476  000413                          BR      65$
001500  020127  000005         63$:     CMP     R1,#5                   ;                       5443
001504  001003                          BNE     64$
001506  012746  000107                  MOV     #107,-(SP)              ;                       5444
001512  000665                          BR      53$
001514  020127  000006         64$:     CMP     R1,#6                   ;                       5447
001520  001011                          BNE     66$
001522  012746  000110                  MOV     #110,-(SP)              ;                       5448
001526  004737  000000V        65$:     JSR     PC,ERR.HRD.RTNE
001532  013700  000000G                  MOV     T.ADDR,R0               ;                       5449
001536  105260  000051                  INCB    51(R0)
001542  000411                          BR      68$                     ;                       5447
001544  012746  000111         66$:     MOV     #111,-(SP)              ;                       5452
001550  004737  000000V        67$:     JSR     PC,ERR.HRD.RTNE
001554  013700  000000G                  MOV     CCTLR,R0                ;                       5453
001560  006300                          ASL     R0
001562  105260  000000G                  INCB    C.ERR.TBL(R0)
001566  005726                  68$:     TST     (SP)+
001570  000207                  69$:     RTS     PC                      ;                       5451
                                                                        ;                       5158

; Routine Size:  445 words,     Routine Base:  $CODE$ + 22022
; Maximum stack depth per invocation:  7 words


;   5458  1
```

```
;    5459  1     GLOBAL routine UPD_IO_TALLY : novalue =
;    5460  1
;    5461  1     !+
;    5462  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;    5463  1     !       PACKETS WITH "SUCCESS" STATUS CODES. ITS PURPOSE IS TO UPDATE ALL THE
;    5464  1     !       APPROPRIATE STATISTICAL FIELDS FOR THE CURRENT UNIT. A CHECK IS ALSO
;    5465  1     !       MADE ON THE TOTAL NUMBER OF BYTES TRANSFERRED THUS FAR; IF THE
;    5466  1     !       OPERATOR-SPECIFIED LIMIT HAS BEEN REACHED, THEN THE UNIT IS DROPPED.
;    5467  1     !
;    5468  1     !       IMPLICIT INPUTS:
;    5469  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5470  1     !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    5471  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    5472  1     !               CUOFF - CST OFFSET FOR THE CURRENT UNIT
;    5473  1     !               L$LUN - CURRENT UNIT NUMBER
;    5474  1     !-
;    5475  1
;    5476  2         begin
;    5477  2
;    5478  2         local
;    5479  2             THOUSANDS : word,
;    5480  2             MILLIONS : word;                                        ! TOTAL NO. OF BYTES XFERRED TO/FROM A UNIT
;    5481  2
;    5482  3         if .RP_ADDR [ENDCOD] eql (OP_RD or OP_END)                  ! IF ENDCODE IS READ
;    5483  2         then
;    5484  3             begin
;    5485  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1;     ! INCREMENT NO. OF READS AND ADD BYTE COUNT
;    5486  3             T_ADDR [BYTES_READ_LO] = .T_ADDR [BYTES_READ_LO] + .RP_ADDR [BCNT_LO];
;    5487  3             T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + .RP_ADDR [BCNT_LO];
;    5488  3             OVF_CHK (T_ADDR [TOT_READS_LO]);                        ! CHECK FOR FIELD OVERFLOW
;    5489  3             OVF_CHK (T_ADDR [BYTES_READ_LO]);                       !
;    5490  3             OVF_CHK (T_ADDR [TOT_BYT_RED_LO]);                      !
;    5491  3             end
;    5492  2         else
;    5493  2
;    5494  3             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)             ! IF ENDCODE IS WRITE
;    5495  2             then
;    5496  3                 begin
;    5497  3                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1;   ! INCREMENT NO. OF WRITES, ADD BYTE COUNT
;    5498  3                 T_ADDR [BYTES_WRIT_LO] = .T_ADDR [BYTES_WRIT_LO] + .RP_ADDR [BCNT_LO];
;    5499  3                 T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + .RP_ADDR [BCNT_LO];
;    5500  3                 OVF_CHK (T_ADDR [TOT_WRITES_LO]);                   ! CHECK FOR FIELD OVERFLOW
;    5501  3                 OVF_CHK (T_ADDR [BYTES_WRIT_LO]);                   !
;    5502  3                 OVF_CHK (T_ADDR [TOT_BYT_WRT_LO]);                  !
;    5503  2                 end;
;    5504  2
;    5505  2         if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) or
;    5506  3            (.RP_ADDR [ENDCOD] eql (OP_WRT or OP_END))
;    5507  2         then
;    5508  3             begin
;    5509  3             MILLIONS = .T_ADDR [MBYTES_READ] + .T_ADDR [MBYTES_WRIT];   ! TOTAL BYTES TRANSFERRED
;    5510  3             THOUSANDS = .T_ADDR [BYTES_READ_HI] + .T_ADDR [BYTES_WRIT_HI];
;    5511  3
```

```
;    5512  3              if .THOUSANDS gequ 1000
;    5513  3              then
;    5514  4                  begin
;    5515  4                  MILLIONS = .MILLIONS + 1;                          ! COUNT THE LOWER OVERFLOW TOO!
;    5516  4                  THOUSANDS = .THOUSANDS - 1000;
;    5517  3                  end;
;    5518  3
;    5519  3          !
;    5520  3          ! THIS ADDED BECAUSE IT WILL TAKE FOREVER TO TRANSFER ON THE ORDER OF A MEGABYTE TO A FLOPPY
;    5521  3          ! BUT IT IS A MUCH MORE REASONABLE MEASURE FOR THE RD51/52 WINCHESTER.  THE QUESTION NOW REFERS TO
;    5522  3          ! THE TOTAL DATA TRANSFER TO THE CONTROLLER AND THIS IS PRETTY CLOSE SINCE THE FLOPPIES GET
;    5523  3          ! ABOUT 1/1000 THE DATA THE HARD DISK(S) GET.
;    5524  3          !
;    5525  3
;    5526  3              if .SWP_XFER eql 0                                     ! IF THERE IS A TRANSFER LIMIT
;    5527  3              then
;    5528  4                  begin
;    5529  4
;    5530  4                  if .THOUSANDS gtru 50                              !ZZZ
;    5531  4                  then
;    5532  4                      EOP_FLAG = TRUE;                               ! SET END-OF-PASS FLAG
;    5533  4
;    5534  4                  end
;    5535  3              else
;    5536  3
;    5537  3                  if .MILLIONS gequ .SWP_XFER                        ! IF TRANSFER LIMIT IS REACHED
;    5538  3                  then
;    5539  3                      EOP_FLAG = TRUE;                               ! SET END-OF-PASS FLAG
;    5540  3
;    5541  2              end;                                                   ! IF UNIT IS STILL ALIVE
;    5542  2
;    5543  2              !.....................................................
;    5544  2              !                                                     .
;    5545  2              ! THE FOLLOWING IS ADDED TO MAKE THE RUN TIME ABOUT 1.5 MINUTES FOR A  .
;    5546  2              ! QUICK PASS IF ALL UNITS UNDER TEST ARE FLOPPIES.                      .
;    5547  2              !.....................................................
;    5548  2
;    5549  2          !!ZZZ    IF .RD_COUNT EQL 0                                    !IF THERE ARE NO WINCHESTERS   ZZZ
;    5550  2          !!ZZZ       THEN                                               !                              ZZZ
;    5551  2          !!ZZZ         BEGIN                                            !                              ZZZ
;    5552  2          !!ZZZ         IF .THOUSANDS GTRU 44                            !IF ABOUT 1.5 MINUTES GONE BY  ZZZ
;    5553  2          !!ZZZ         THEN                                             !                              ZZZ
;    5554  2          !!ZZZ            EOP_FLAG = TRUE;                              !SET THE END OF PASS FLAG      ZZZ
;    5555  2          !!ZZZ         END;                                             !                              ZZZ
;    5556  2
;    5557  2
;    5558  2              ROUND_OUTPUT ();                                       ! ROUND TOTALS TO FIT PRINT POSITIONS
;    5559  1              end;                                                   ! ROUTINE UPD_IO_TALLY


                                        .SBTTL  UPD.IO.TALLY MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          UPD.IO.TALLY::
                                        JSR     R1,$SAVE2                      ;                                      5459
```

```
000004  013701  000000G                    MOV     RP.ADDR,R1              ;                               5482
000010  126127  000014  000241             CMPB    14(R1),#241
000016  001027                             BNE     1$
000020  013700  000000G                    MOV     T.ADDR,R0               ;                               5485
000024  005260  000016                     INC     16(R0)
000030  066110  000020                     ADD     20(R1),(R0)             ;                               5486
000034  066160  000020  000032             ADD     20(R1),32(R0)           ;                               5487
000042  012746  000016                     MOV     #16.,-(SP)              ;                               5488
000046  060016                             ADD     R0,(SP)
000050  004737  000000V                    JSR     PC,OVF.CHK
000054  013716  000000G                    MOV     T.ADDR,(SP)             ;                               5489
000060  004737  000000V                    JSR     PC,OVF.CHK
000064  013716  000000G                    MOV     T.ADDR,(SP)             ;                               5490
000070  062716  000032                     ADD     #32,(SP)
000074  000435                             BR      2$
000076  126127  000014  000242     1$:     CMPB    14(R1),#242             ;                               5494
000104  001034                             BNE     3$
000106  013700  000000G                    MOV     T.ADDR,R0               ;                               5497
000112  005260  000024                     INC     24(R0)
000116  066160  000020  000006             ADD     20(R1),6(R0)            ;                               5498
000124  066160  000020  000040             ADD     20(R1),40(R0)           ;                               5499
000132  012746  000024                     MOV     #24.,-(SP)              ;                               5500
000136  060016                             ADD     R0,(SP)
000140  004737  000000V                    JSR     PC,OVF.CHK
000144  013716  000000G                    MOV     T.ADDR,(SP)             ;                               5501
000130  062716  000006                     ADD     #6,(SP)
000154  004737  000000V                    JSR     PC,OVF.CHK
000160  013716  000000G                    MOV     T.ADDR,(SP)             ;                               5502
000164  062716  000040                     ADD     #40,(SP)
000170  004737  000000V             2$:    JSR     PC,OVF.CHK
000174  005726                             TST     (SP)+                   ;                               5496
000176  013700  000000G             3$:    MOV     RP.ADDR,R0              ;                               5505
000202  126027  000014  000241             CMPB    14(R0),#241
000210  001404                             BEQ     4$
000212  126027  000014  000242             CMPB    14(R0),#242             ;                               5506
000220  001034                             BNE     8$
000222  013700  000000G             4$:    MOV     T.ADDR,R0               ;                               5509
000226  016002  000004                     MOV     4(R0),R2                ; *,MILLIONS
000232  066002  000012                     ADD     12(R0),R2               ; *,MILLIONS
000236  016001  000002                     MOV     2(R0),R1                ; *,THOUSANDS                   5510
000242  066001  000010                     ADD     10(R0),R1               ; *,THOUSANDS
000246  020127  001750                     CMP     R1,#1750                ; THOUSANDS,*                   5512
000252  103403                             BLO     5$
000254  005202                             INC     R2                      ; MILLIONS                      5515
000256  162701  001750                     SUB     #1750,R1                ; *,THOUSANDS                   5516
000262  013700  000000G             5$:    MOV     SWP.XFER,R0             ;                               5526
000266  001004                             BNE     6$
000270  020127  000062                     CMP     R1,#62                  ; THOUSANDS,*                   5530
000274  101406                             BLOS    8$
000276  000402                             BR      7$                      ;                               5532
000300  020200                     6$:     CMP     R2,R0                   ; MILLIONS,*                    5537
000302  103403                             BLO     8$
000304  112737  000001  000000G     7$:     MOVB    #1,EOP.FLAG             ;                               5539
```

```
000312  004737  000000V                    8$:     JSR     PC,ROUND.OUTPUT                 ;                       5558
000316  000207                                    RTS     PC                              ;                       5459

; Routine Size:  104 words,     Routine Base:  $CODE$ + 23614
; Maximum stack depth per invocation:  5 words
```

```
;    5560  1     GLOBAL routine OVF_CHK (ADDR) : novalue =
;    5561  1     !+
;    5562  1     !       THIS ROUTINE IS CALLED FROM UPD_IO_TALLY TO CHECK FOR OVERFLOW IN
;    5563  1     !       CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
;    5564  1     !       LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
;    5565  1     !       EXCEEDING 1000. IF TRUE, THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
;    5566  1     !       THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IN INCREMENTED.
;    5567  1     !
;    5568  1     !       INPUTS:
;    5569  1     !               ADDR   ADDRESS OF THE BYTES_READ_LO OR BYTES_WRIT_LO FIELD FOR
;    5570  1     !                       THE CURRENT UNIT (SEE STATISTIC TABLE (TALLY) LAYOUT)
;    5571  1     !
;    5572  2         begin
;    5573  2
;    5574  2         while ..ADDR geau 1000 do                       ! IF LO ORDER OVERFLOW
;    5575  3             begin
;    5576  3             .ADDR = ..ADDR   1000;                      ! SUBTRACT 1000
;    5577  3             (.ADDR + 2) = .(.ADDR + 2) + 1;             ! INCR HI-ORDER
;    5578  2             end;
;    5579  2
;    5580  2         if .(.ADDR + 2) geau 1000                       ! IF HI ORDER OVERFLOW
;    5581  2         then
;    5582  3             begin
;    5583  3             (.ADDR + 2) = .(.ADDR + 2)   1000;          ! SUBTRACT 1000
;    5584  3             (.ADDR + 4) = .(.ADDR + 4) + 1;             ! INCREMENT MBYTES
;    5585  2             end;
;    5586  2
;    5587  1         end;                                            ! ROUTINE OVF_CHK


                                        .SBTTL  OVF.CHK MULTI DRIVE TEST ROUTINES
000000  010146                 OVF.CHK::
                                        MOV     R1,-(SP)                        ;
000002  016600  000004                  MOV     4(SP),R0                        ; ADDR,*    5560
000006  012701  000002                  MOV     #2,R1                           ;           5574
000012  060001                          ADD     R0,R1                           ;           5577
000014  021027  001750          1$:     CMP     (R0),#1750                      ;           5574
000020  103404                          BLO     2$                              ;
000022  162710  001750                  SUB     #1750,(R0)                      ;           5576
000026  005211                          INC     (R1)                            ;           5577
000030  000771                          BR      1$                              ;           5574
000032  021127  001750          2$:     CMP     (R1),#1750                      ;           5580
000036  103404                          BLO     3$                              ;
000040  162711  001750                  SUB     #1750,(R1)                      ;           5583
000044  005260  000004                  INC     4(R0)                           ;           5584
000050  012601                  3$:     MOV     (SP)+,R1                        ;           5560
000052  000207                          RTS     PC

; Routine Size:  22 words,      Routine Base:  $CODE$ + 24134
; Maximum stack depth per invocation:  2 words
```

```
;  5588  1     GLOBAL routine ROUND_OUTPUT : novalue =
;  5589  1
;  5590  1     !+
;  5591  1     !       THIS ROUTINE ROUNDS THE TOTALS TO FIT PRINT POSITIONS.
;  5592  1     !
;  5593  1
;  5594  2         begin
;  5595  2
;  5596  2         if .T_ADDR [TOT_READS_HI] gtru 9999
;  5597  2         then
;  5598  3             begin
;  5599  3
;  5600  3             if .T_ADDR [TOT_READS_LO] lssu 999
;  5601  3             then
;  5602  4                 begin
;  5603  4                 T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI]   1;
;  5604  4                 T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1000;
;  5605  3                 end;
;  5606  3
;  5607  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO]   999;
;  5608  3             T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI]   9999;
;  5609  2             end;
;  5610  2
;  5611  2         if .T_ADDR [TOT_WRITES_HI] gtru 9999
;  5612  2         then
;  5613  *             begin
;  5614  3
;  5615  3             if .T_ADDR [TOT_WRITES_LO] lssu 999
;  5616  3             then
;  5617  4                 begin
;  5618  4                 T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI]   1;
;  5619  4                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1000;
;  5620  3                 end;
;  5621  3
;  5622  3             T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO]   999;
;  5623  3             T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI]   9999;
;  5624  2             end;
;  5625  2
;  5626  2         if .T_ADDR [MTOT_BYT_RED] gtru 999
;  5627  2         then
;  5628  3             begin
;  5629  3
;  5630  3             if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;  5631  3             then
;  5632  4                 begin
;  5633  4                 T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED]   1;
;  5634  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;  5635  3                 end;
;  5636  3
;  5637  3             if .T_ADDR [TOT_BYT_RED_LO] lssu 999
;  5638  3             then
;  5639  4                 begin
;  5640  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI]   1;
```

```
;    5641  4                T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] * 1000;
;    5642  4
;    5643  4                IF .T_ADDR [TOT_BYT_RED_HI] LEQU 999
;    5644  4                THEN
;    5645  5                    BEGIN
;    5646  5                    T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;    5647  5                    T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] * 1000;
;    5648  4                    END;
;    5649  3                END;
;    5650  3
;    5651  3            T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] - 999;
;    5652  3            T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 999;
;    5653  3            T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 999;
;    5654  2            END;
;    5655  2
;    5656  2        IF .T_ADDR [MTOT_BYT_WRT] GTRU 999
;    5657  2        THEN
;    5658  3            BEGIN
;    5659  3
;    5660  3            IF .T_ADDR [TOT_BYT_WRT_HI] LEQU 999
;    5661  3            THEN
;    5662  4                BEGIN
;    5663  4                T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    5664  4                T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] * 1000;
;    5665  3                END;
;    5666  3
;    5667  3            IF .T_ADDR [TOT_BYT_WRT_LO] LEQU 999
;    5668  3            THEN
;    5669  4                BEGIN
;    5670  4                T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 1;
;    5671  4                T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] * 1000;
;    5672  4
;    5673  4                IF .T_ADDR [TOT_BYT_WRT_HI] LEQU 999
;    5674  4                THEN
;    5675  5                    BEGIN
;    5676  5                    T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    5677  5                    T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] * 1000;
;    5678  4                    END;
;    5679  3                END;
;    5680  3
;    5681  3            T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] - 999;
;    5682  3            T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 999;
;    5683  3            T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 999;
;    5684  2            END;
;    5685  2
;    5686  1        END;


                                    .SBTTL  ROUND.OUTPUT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G         ROUND.OUTPUT::
                                    JSR     R1,$SAVE3                       ;                       5588
000004  013700  000000G             MOV     T.ADDR,R0                       ;                       5596
000010  012702  000020              MOV     #20,R2
```

```
000014  060002                          ADD     R0,R2
000016  021227  023417                  CMP     (R2),#23417
000022  101415                          BLOS    2$
000024  012701  000016                  MOV     #16,R1          ;                       5600
000030  060001                          ADD     R0,R1
000032  021127  001747                  CMP     (R1),#1747
000036  103003                          BHIS    1$
000040  005312                          DEC     (R2)            ;                       5603
000042  062711  001750                  ADD     #1750,(R1)      ;                       5604
000046  162711  001747          1$:     SUB     #1747,(R1)      ;                       5607
000052  162712  023417                  SUB     #23417,(R2)     ;                       5608
000056  012702  000026          2$:     MOV     #26,R2          ;                       5611
000062  060002                          ADD     R0,R2
000064  021227  023417                  CMP     (R2),#23417
000070  101415                          BLOS    4$
000072  012701  000024                  MOV     #24,R1          ;                       5615
000076  060001                          ADD     R0,R1
000100  021127  001747                  CMP     (R1),#1747
000104  103003                          BHIS    3$
000106  005312                          DEC     (R2)            ;                       5618
000110  062711  001750                  ADD     #1750,(R1)      ;                       5619
000114  162711  001747          3$:     SUB     #1747,(R1)      ;                       5622
000120  162712  023417                  SUB     #23417,(R2)     ;                       5623
000124  012703  000036          4$:     MOV     #36,R3          ;                       5626
000130  060003                          ADD     R0,R3
000132  021327  001747                  CMP     (R3),#1747
000136  101436                          BLOS    7$
000140  012701  000034                  MOV     #34,R1          ;                       5630
000144  060001                          ADD     R0,R1
000146  021127  001747                  CMP     (R1),#1747
000152  103003                          BHIS    5$
000154  005313                          DEC     (R3)            ;                       5633
000156  062711  001750                  ADD     #1750,(R1)      ;                       5634
000162  012702  000032          5$:     MOV     #32,R2          ;                       5637
000166  060002                          ADD     R0,R2
000170  021227  001747                  CMP     (R2),#1747
000174  103011                          BHIS    6$
000176  005311                          DEC     (R1)            ;                       5640
000200  062712  001750                  ADD     #1750,(R2)      ;                       5641
000204  021127  001747                  CMP     (R1),#1747      ;                       5643
000210  103003                          BHIS    6$
000212  005313                          DEC     (R3)            ;                       5646
000214  062711  001750                  ADD     #1750,(R1)      ;                       5647
000220  162712  001747          6$:     SUB     #1747,(R2)      ;                       5651
000224  162711  001747                  SUB     #1747,(R1)      ;                       5652
000230  162713  001747                  SUB     #1747,(R3)      ;                       5653
000234  012702  000044          7$:     MOV     #44,R2          ;                       5656
000240  060002                          ADD     R0,R2
000242  021227  001747                  CMP     (R2),#1747
000246  101435                          BLOS    10$
000250  012701  000042                  MOV     #42,R1          ;                       5660
000254  060001                          ADD     R0,R1
000256  021127  001747                  CMP     (R1),#1747
```

C2


```
000262  103003                          BHIS    8$
000264  005312                          DEC     (R2)
000266  062711  001750                  ADD     #1750,(R1)          ;                   5663
000272  062700  000040          8$:     ADD     #40,R0              ;                   5664
000276  021027  001747                  CMP     (R0),#1747          ;                   5667
000302  103011                          BHIS    9$
000304  005311                          DEC     (R1)
000306  062710  001750                  ADD     #1750,(R0)          ;                   5670
000312  021127  001747                  CMP     (R1),#1747          ;                   5671
000316  103003                          BHIS    9$                  ;                   5673
000320  005312                          DEC     (R2)
000322  062711  001750                  ADD     #1750,(R1)          ;                   5676
000326  162710  001747          9$:     SUB     #1747,(R0)          ;                   5677
000332  162711  001747                  SUB     #1747,(R1)          ;                   5681
000336  162712  001747                  SUB     #1747,(R2)          ;                   5682
000342  000207                  10$:    RTS     PC                  ;                   5683
                                                                    ;                   5588

; Routine Size:  114 words,     Routine Base:  $CODE$ + 24210
; Maximum stack depth per invocation:  5 words
```

```
;  5687  1     GLOBAL routine HOST_WRT_CHK =
;  5688  1
;  5689  1     !+
;  5690  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;  5691  1     !       PACKETS WITH "SUCCESS" STATUS CODES, BUT ONLY IF THE HOST WRITE-COMPARE
;  5692  1     !       OPTION WAS SELECTED BY THE OPERATOR.
;  5693  1     !
;  5694  1     !       IF THE CURRENT RETPKT BEING PROCESSED IS A WRITE FUNCTION, THEN THE
;  5695  1     !       PACKET INDEX (RP_INDX) IS SAVED IN THE CONTROLLER'S RETURN PACKET SAVE
;  5696  1     !       AREA (RP_SAVE). OTHERWISE, THE PACKET IS A READ, SO ITS ASSOCIATED
;  5697  1     !       WRITE PACKET IS REMOVED FROM THE SAVE AREA, AND A BYTE-BY-BYTE
;  5698  1     !       COMPARISON IS PERFORMED ON THE TWO I/O BUFFERS. ANY DIFFERENCES
;  5699  1     !       ENCOUNTERED RESULTS IN THE DECLARATION OF A HARD ERROR.
;  5700  1     !
;  5701  1     !       IMPLICIT INPUTS:
;  5702  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  5703  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;  5704  1     !-
;  5705  1
;  5706  2         begin
;  5707  2
;  5708  2         local
;  5709  2             BUFF1 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;  5710  2             BUFF2 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;  5711  2             BUFFW,                                            ! I/O BUFFER ADDRESS
;  5712  2             COUNT : word,                                     ! BYTE COUNT
;  5713  2             FLAG : byte initial (byte (TRUE)),
;  5714  2             index : signed word;
;  5715  2
;  5716  3         if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)           ! IF WRITE OPERATION
;  5717  2         then
;  5718  2             FLAG = FALSE                                      ! DON'T CALL SWEEP FROM IO_RETPKT
;  5719  2         else
;  5720  2
;  5721  2             if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) and
;  5722  3                ((index = RPS_REM ()) geq 0)                   ! IF ASSOCIATED WRITE PACKET IS FOUND
;  5723  2             then
;  5724  3                 begin
;  5725  3                 BUFFW = RETPKT [.index, BUFF_0];              ! ADDR OF ADDR OF WRITE I/O BUFFER
;  5726  3                 BUFF1 = ..BUFFW;                              ! ADDR OF WRITE I/O BUFFER
;  5727  3                 BUFF2 = .RP_ADDR [BUFF_0];                    ! ADDR OF READ I/O BUFFER
;  5728  3                 COUNT = .RP_ADDR [BCNT_LO];                   ! BYTE COUNT
;  5729  3
;  5730  3                 incr I from 1 to .COUNT do                    ! FOR EACH BYTE IN BUFFERS
;  5731  3
;  5732  3                     if .(.BUFF1)<0, 8, 0> eql .(.BUFF2)<0, 8, 0>    ! IF BYTES COMPARE O.K.
;  5733  3                     then
;  5734  4                         begin
;  5735  4                         BUFF1 = .BUFF1 + 1;                   ! ADVANCE WRITE BUFFER ADDR
;  5736  4                         BUFF2 = .BUFF2 + 1;                   ! ADVANCE READ BUFFER ADDR
;  5737  4                         end
;  5738  3                     else
;  5739  4                         begin                                 ! ELSE   COMPARE ERROR
```

```
;    5740  4                      T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;
;    5741  4                      T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5742  4
;    5743  4                      if .APT_MODE
;    5744  4                      then
;    5745  4                          ERR_HRD_RTNE_APT (42)                   ! I/O REQUEST FAILED
;    5746  4                      else
;    5747  4                          ERR_HRD_RTNE (42);
;    5748  4
;    5749  4                      EMS_CMP (RETPKT + (.index * RP_LEN * 2));
;    5750  4
;    5751  4                      if .T_ADDR [ERR_HARD] gequ .SWP_ERROR
;    5752  4                      then
;    5753  5                          begin
;    5754  5                          DUR [.L$LUN] = DU_HERR;                 ! IF ERROR COUNT EXCEEDED
;    5755  5                          DODU (.L$LUN);                          ! DROP UNIT
;    5756  4                          end;
;    5757  4
;    5758  4                      exitloop;                                   ! NO NEED TO CONTINUE
;    5759  3                      end;                                        ! IF COMPARE ERROR
;    5760  3
;    5761  2                  end;                                            ! IF ASSOCIATED WRITE RETPKT WAS FOUND
;    5762  2
;    5763  2          return (.FLAG);
;    5764  1          end;                                                    ! ROUTINE HOST_WRT_CHK


                                .SBTTL  HOST.WRT.CHK MULTI-DRIVE TEST ROUTINES
000000  004137  000000G    HOST.WRT.CHK::
                            JSR     R1,$SAVE5                       ;                       5687
000004  005746              TST     -(SP)
000006  112705  000001      MOVB    #1,R5                           ; *,FLAG                5706
000012  013700  000000G     MOV     RP.ADDR,R0                      ;                       5716
000016  126027  000014  000242  CMPB    14(R0),#242
000024  001002              BNE     1$
000026  105005              CLRB    R5                              ; FLAG                  5718
000030  000511              BR      8$                              ;                       5716
000032  126027  000014  000241  1$:  CMPB    14(R0),#241             ;                       5721
000040  001105              BNE     8$
000042  004737  000000V     JSR     PC,RPS.REM                      ;                       5722
000046  005700              TST     R0                              ; INDEX
000050  002501              BLT     8$
000052  010046              MOV     R0,-(SP)                        ; INDEX,*               5725
000054  012746  000054      MOV     #54,-(SP)
000060  004737  000000G     JSR     PC,BL$MUL
000064  010066  000004      MOV     R0,4(SP)
000070  062700  000024G     ADD     #RETPKT+24,R0                   ; *,BUFFW
000074  011001              MOV     (R0),R1                         ; BUFFW,BUFF1           5726
000076  013700  000000G     MOV     RP.ADDR,R0                      ;                       5727
000102  016002  000024      MOV     24(R0),R2                       ; *,BUFF2
000106  016004  000020      MOV     20(R0),R4                       ; *,COUNT               5728
000112  005003              CLR     R3                              ; I                     5730
000114  000453              BR      6$
```

F2


```
000116  121112                          2$:     CMPB    (R1),(R2)               ; BUFF1,BUFF2                   5732
000120  001003                                  BNE     3$
000122  005201                                  INC     R1                      ; BUFF1                         5735
000124  005202                                  INC     R2                      ; BUFF2                         5736
000126  000446                                  BR      6$                      ;                               5732
000130  013700  000000G                  3$:     MOV     T.ADDR,R0               ;                               5740
000134  005260  000014                          INC     14(R0)
000140  105260  000051                          INCB    51(R0)                  ;                               5741
000144  032737  000001  001254'                BIT     #1,APT.MODE             ;                               5743
000152  001405                                  BEQ     4$
000154  012716  000052                          MOV     #52,(SP)                ;                               5745
000160  004737  000000V                        JSR     PC,ERR.HRD.RTNE.APT
000164  000404                                  BR      5$                      ;                               5743
000166  012716  000052                  4$:     MOV     #52,(SP)                ;                               5747
000172  004737  000000V                        JSR     PC,ERR.HRD.RTNE
000176  016616  000004                  5$:     MOV     4(SP),(SP)              ;                               5749
000202  062716  000000G                        ADD     #RETPKT,(SP)
000206  004737  000000G                        JSR     PC,EMS.CMP
000212  013700  000000G                        MOV     T.ADDR,R0               ;                               5751
000216  026037  000014  000000G               CMP     14(R0),SWP.ERROR
000224  103412                                  BLO     7$
000226  013700  000000G                        MOV     L$LUN,R0                ;                               5754
000232  112760  000004  000000G               MOVB    #4,DUR(R0)
000240  104451                                  TRAP    51                      ;                               5755
000242  000403                                  BR      7$                      ;                               5739
000244  005203                          6$:     INC     R3                      ; I                             5730
000246  020304                                  CMP     R3,R4                   ; I,COUNT
000250  003722                                  BLE     2$
000252  022626                          7$:     CMP     (SP)+,(SP)+             ;                               5724
000254  005000                          8$:     CLR     R0                      ;                               5763
000256  150500                                  BISB    R5,R0                   ; FLAG,*
000260  005726                                  TST     (SP)+                   ;                               5687
000262  000207                                  RTS     PC

; Routine Size:  90 words,      Routine Base:  $CODE$ + 24554
; Maximum stack depth per invocation:  11 words
```

```
;    5765  1     GLOBAL routine SWEEP : novalue =
;    5766  1
;    5767  1     !+
;    5768  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT AND OTHERS TO DEALLOCATE THE
;    5769  1     !       RESOURCES ASSOCIATED WITH THE CURRENT RETURN PACKET. THIS INCLUDES THE
;    5770  1     !       PACKET ITSELF AND THE I/O BUFFER. IN ADDITION, IF THE HOST IS
;    5771  1     !       PERFORMING WRITE-COMPARES, AND IF THE CURRENT RETURN PACKET IS A READ
;    5772  1     !       FUNCTION, THEN THE CURRENT CONTROLLER'S RP_SAVE AREA IS SEARCHED FOR
;    5773  1     !       THE ASSOCIATED WRITE RETPKT SO THAT ITS RESOURCES CAN ALSO BE
;    5774  1     !       DEALLOCATED.
;    5775  1     !
;    5776  1     !       IMPLICIT INPUTS:
;    5777  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    5778  1     !               RP_INDX - INDEX OF CURRENT RETURN PACKET
;    5779  1     !-
;    5780  1
;    5781  2         begin
;    5782  2
;    5783  2         local
;    5784  2             index : signed word;
;    5785  2
;    5786  2         if (.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD          ! IF READ OPCODE OR ENDCODE
;    5787  2         then
;    5788  2
;    5789  3             if BIT_TST (SWP_FLAGS, SWP_HWC)                  ! IF HOST IS DOING WRITE-COMPARES
;    5790  2             then
;    5791  2
;    5792  2                 if (index = RPS_REM ()) geq 0                ! IF ASSOCIATED WRITE RETPKT IS FOUND
;    5793  2                 then
;    5794  3                     begin
;    5795  3                     PUT_IO_BUFF (RETPKT [.index, BUFF_O]);   ! RETURN WRITE I/O BUFFER TO POOL
;    5796  3                     PUT_RETPKT (.index);                     ! RETURN WRITE PACKET TO POOL
;    5797  2                     end;
;    5798  2
;    5799  2             PUT_IO_BUFF (RP_ADDR [BUFF_O]);                  ! RETURN CURRENT I/O BUFFER TO POOL
;    5800  2             PUT_RETPKT (.RP_INDX);                           ! RETURN CURRENT RETPKT TO POOL
;    5801  1             end;                                             ! ROUTINE SWEEP


                                        .SBTTL  SWEEP MULTI-DRIVE TEST ROUTINES
000000  010146                 SWEEP:: MOV     R1,-(SP)                        ;                       5765
000002  013700  000000G                MOV     RP.ADDR,R0                      ;                       5786
000006  116000  000014                 MOVB    14(R0),R0
000012  042700  177600                 BIC     #177600,R0
000016  020027  000041                 CMP     R0,#41
000022  001026                        BNE     1$
000024  032737  000040  000000G        BIT     #40,SWP.FLAGS                   ;                       5789
000032  001422                        BEQ     1$
000034  004737  000000V                JSR     PC,RPS.REM                      ;                       5792
000040  010001                        MOV     R0,R1                           ; *,INDEX
000042  002416                        BLT     1$
000044  010146                        MOV     R1,-(SP)                        ; INDEX,*               5795
000046  012746  000054                MOV     #54,-(SP)
```

```
000052  004737  000000G                        JSR     PC,BL$MUL
000056  062700  000024G                        ADD     @RETPKT+24,R0
000062  010016                                 MOV     R0,(SP)
000064  004737  000000G                        JSR     PC,PUT.IO.BUFF
000070  010116                                 MOV     R1,(SP)          ; INDEX,*        5796
000072  004737  000000G                        JSR     PC,PUT.RETPKT
000076  022626                                 CMP     (SP)+,(SP)+      ;                5794
000100  013746  000000G                1$:     MOV     RP.ADDR,-(SP)    ;                5799
000104  062716  000024                        ADD     #24,(SP)
000110  004737  000000G                        JSR     PC,PUT.IO.BUFF
000114  013716  000000G                        MOV     RP.INDX,(SP)     ;                5800
000120  004737  000000G                        JSR     PC,PUT.RETPKT
000124  005726                                 TST     (SP)+            ;                5781
000126  012601                                 MOV     (SP)+,R1         ;                5765
000130  000207                                 RTS     PC

; Routine Size:  45 words,      Routine Base:  $CODE$ + 25040
; Maximum stack depth per invocation:  4 words
```

```
;    5802  1     GLOBAL routine RPS_REM =
;    5803  1
;    5804  1     !+
;    5805  1     !       THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP_SAVE AREA FOR A
;    5806  1     !       RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
;    5807  1     !       CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
;    5808  1     !       OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
;    5809  1     !       RP_SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
;    5810  1     !       OPERATION IS RETURNED TO THE CALLER.
;    5811  1     !
;    5812  1     !       IMPLICIT INPUTS:
;    5813  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5814  1     !
;    5815  1     !       OUTPUTS:
;    5816  1     !               INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
;    5817  1     !                       A CRN WHICH IS ONE LESS THAN THE CURRENT
;    5818  1     !-
;    5819  1
;    5820  2         begin
;    5821  2
;    5822  2         local
;    5823  2             index : signed word initial (-1);                          ! ASSUME NOT FOUND
;    5824  2
;    5825  2         incr COUNT from 0 to RP_CNT - 1 do                             ! FOR EACH ENTRY IN RP_SAVE
;    5826  2
;    5827  2             if (.RP_USE [.COUNT] eql .CCTLR) and                       ! IF THIS IS A VALID RETPKT INDEX
;    5828  3                (.RETPKT [.COUNT, ENDCOD] eql (OP_WRT or OP_END))
;    5829  2             then
;    5830  2
;    5831  3                 if ((.RETPKT [.COUNT, CRF_LO] eql (.RP_ADDR [CRF_LO] - 1)) and     ! IF CORRECT CRN
;    5832  2                     (.RETPKT [.COUNT, CRF_HI] eql .RP_ADDR [CRF_HI]); or
;    5833  3                    ((.RETPKT [.COUNT, CRF_HI] eql (.RP_ADDR [CRF_HI] - 1)) and
;    5834  3                     (.RETPKT [.COUNT, CRF_LO] eql %o'177777') and
;    5835  3                     (.RP_ADDR [CRF_LO] eql 0))
;    5836  2                 then
;    5837  3                     begin
;    5838  3                     index = .COUNT;                                    ! INDEX TO BE RETURNED
;    5839  3                     exitloop;                                          ! DONE
;    5840  2                     end;
;    5841  2
;    5842  2             return .index;
;    5843  1             end;                                                       ! ROUTINE RPS_REM


                                        .SBTTL  RPS.REM MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          RPS.REM::
                                        JSR     R1,$SAVE4                       ;                                       5802
000004  012704  177777                  MOV     #-1,R4                          ; *,INDEX                               5820
000010  005003                          CLR     R3                              ; COUNT                                 5825
000012  116300  000000G          1$:    MOVB    RP.USE(R3),R0                   ; *(COUNT),*                            5827
000016  020037  000000G                  CMP     R0,CCTLR
000022  001053                          BNE     4$
000024  010346                          MOV     R3, (SP)                        ; COUNT,*                               5828
```

J2


```
000026  012746  000054                          MOV     #54,-(SP)
000032  004737  000000G                         JSR     PC,BL$MUL
000036  022626                                  CMP     (SP)+,(SP)+
000040  126027  000014G 000242                  CMPB    RETPKT+14(R0),#242
000046  001041                                  BNE     4$
000050  010346                                  MOV     R3,-(SP)                ; COUNT,*               5831
000052  012746  000054                          MOV     #54,-(SP)
000056  004737  000000G                         JSR     PC,BL$MUL
000062  022626                                  CMP     (SP)+,(SP)+
000064  013701  000000G                         MOV     RP.ADDR,R1
000070  016102  000004                          MOV     4(R1),R2
000074  005302                                  DEC     R2
000076  026002  000004G                         CMP     RETPKT+4(R0),R2
000102  001004                                  BNE     2$
000104  026061  000006G 000006                  CMP     RETPKT+6(R0),6(R1)      ;                       5832
000112  001415                                  BEQ     3$
000114  016102  000006                  2$:     MOV     6(R1),R2                ;                       5833
000120  005302                                  DEC     R2
000122  026002  000006G                         CMP     RETPKT+6(R0),R2
000126  001011                                  BNE     4$
000130  026027  000004G 177777                  CMP     RETPKT+4(R0),#-1        ;                       5834
000136  001005                                  BNE     4$
000140  005761  000004                          TST     4(R1)                   ;                       5835
000144  001002                                  BNE     4$
000146  010304                          3$:     MOV     R3,R4                   ; COUNT,INDEX           5838
000150  000404                                  BR      5$                      ;                       5837
000152  005203                          4$:     INC     R3                      ; COUNT                 5825
000154  020327  000007                          CMP     R3,#7                   ; COUNT,*
000160  003714                                  BLE     1$
000162  010400                          5$:     MOV     R4,R0                   ; INDEX,*               5820
000164  000207                                  RTS     PC                      ;                       5802

; Routine Size:  59 words,    Routine Base:  $CODE$ + 25172
; Maximum stack depth per invocation:  8 words
```

```
;    5844  1     GLOBAL routine DR_RETPKT : novalue =
;    5845  1
;    5846  1     !+
;    5847  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT FOR ALL PACKETS ORIGINATING AT
;    5848  1     !       THE "DRIVER" PORTION OF THE PROGRAM. THIS INCLUDES PACKETS DESCRIBING
;    5849  1     !       FATAL DEVICE ERRORS.
;    5850  1     !
;    5851  1     !       FOR FATAL DEVICE ERRORS, THIS ROUTINE RELEASES ALL RESOURCES HELD BY
;    5852  1     !       THE CONTROLLER. THE CONTROLLER IS MARKED OFFLINE IN ITS CST, AND ALL
;    5853  1     !       UNITS ATTACHED TO THE CONTROLLER ARE DROPPED.
;    5854  1     !
;    5855  1     !       IMPLICIT INPUTS:
;    5856  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;    5857  1     !               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
;    5858  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    5859  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    5860  1     !-
;    5861  1
;    5862  2         begin
;    5863  2
;    5864  2
;    5865  2         PUTA_BUFF ();                           ! RELEASE ALL I/O BUFFERS HELD BY CONTROLLER
;    5866  2
;    5867  2         incr index from 0 to RP_CNT - 1 do      ! FOR EACH ENTRY IN CONTROLLER'S RP SAVE
;    5868  2
;    5869  2             if .RP_USE [.index] eql .CCTLR      ! IF VALID RETPKT INDEX
;    5870  2             then
;    5871  2                 PUT_RETPKT (.index);            ! RETURN RETPKT TO POOL
;    5872  2
;    5873  2         QIO [.CCTLR] = 0;                       ! CLEAR NO. OF OUTSTANDING QIOs
;    5874  2         CST_ADDR [STATE] = OFFLINE;             ! MARK CST OFFLINE
;    5875  2         DROP_CTLR (.CCTLR, DU_CFATAL);          ! DROP CONTROLLER'S UNITS
;    5876  2         PUT_RETPKT (.RP_INDX);                  ! PUT BACK RETPKT
;    5877  1         end;                                    ! ROUTINE DR RETPKT


                                        .SBTTL  DR.RETPKT MULTI DRIVE TEST ROUTINES
000000  010146                  DR.RETPKT::
                                        MOV     R1, (SP)                        ;                       5844
000002  004737  000000G                 JSR     PC,PUTA.BUFF                    ;                       5865
000006  005001                          CLR     R1                              ; INDEX                 5867
000010  116100  000000G         1$:     MOVB    RP.USE(R1),R0                   ; *(INDEX),*            5869
000014  020037  000000G                 CMP     R0,CCTLR
000020  001004                          BNE     2$
000022  010146                          MOV     R1,-(SP)                        ; INDEX,*               5871
000024  004737  000000G                 JSR     PC,PUT.RETPKT
000030  005726                          TST     (SP)+
000032  005201                  2$:     INC     R1                              ; INDEX                 5867
000034  020127  000007                  CMP     R1,#7                           ; INDEX,*
000040  003763                          BLE     1$
000042  013701  000000G                 MOV     CCTLR,R1                        ;                       5873
000046  105061  000000G                 CLRB    QIO(R1)
000052  013700  000000G                 MOV     CST.ADDR,R0                     ;                       5874
```

L2


```
000056  042760  100000  000002                  BIC     #100000,2(R0)
000064  010146                                  MOV     R1, -(SP)               ;                       5875
000066  012746  000006                          MOV     #6,-(SP)
000072  004737  000000G                         JSR     PC,DROP.CTLR
000076  013716  000000G                         MOV     RP.INDX,(SP)            ;                       5876
000102  004737  000000G                         JSR     PC,PUT.RETPKT
000106  022626                                  CMP     (SP)+,(SP)+             ;                       5862
000110  012601                                  MOV     (SP)+,R1                ;                       5844
000112  000207                                  RTS     PC

; Routine Size:  38 words,      Routine Base:  $CODE$ + 25360
; Maximum stack depth per invocation:  4 words
```

```
;    5878  1     %sbttl  RDRX INTERRUPT SERVICE ROUTINES'
;    5879  1
;    5880  1     !.
;    5881  1     !       THERE EXISTS AN RDRX INTERRUPT SERVICE ROUTINE FOR EACH DEVICE
;    5882  1     !       CONTROLLER. EACH SERVICE ROUTINE BEGINS BY SIMPLY SETTING THE
;    5883  1     !       APPROPRIATE CONTROLLER NUMBER INTO "ICTLR". ALL SERVICE ROUTINES THEN
;    5884  1     !       BRANCH TO A COMMON INTERRUPT PROCESSING ROUTINE.
;    5885  1     !.
;    5886  1
;    5887  2     BGNSRV (AZINTO);
;    5888  2     ICTLR = 0;
;    5889  2     AZINT ();
;    5890  1     ENDSRV;


                                        .SBTTL  AZINTO RDRX INTERRUPT SERVICE ROUTINES
000000  010046                 AZINTO::MOV     R0, (SP)                        ;                       5887
000002  005037  000104                 CLR     ICTLR                           ;                       5888
000006  004737  000000V                JSR     PC,AZINT                        ;                       5889
000012  012600                         MOV     (SP)+,R0                        ;                       5887
000014  000002                         RTI

; Routine Size:  7 words,       Routine Base:  $CODE$ + 25474
; Maximum stack depth per invocation:  2 words
```

```
;    5891  1     GLOBAL routine AZINT : novalue =
;    5892  1
;    5893  1     !+
;    5894  1     !       THIS IS THE COMMON INTERRUPT SERVICE ROUTINE FOR ALL RDRX CONTROLLERS.
;    5895  1     !       AFTER CALCULATING THE DCT ADDRESS FOR THE INTERRUPTING DEVICE, THIS
;    5896  1     !       ROUTINE WILL SAVE THE CURRENT CONTENTS OF THE SA REGISTER IN THE DCT.
;    5897  1     !       THEN, IF THE "IGNORE INTERRUPT" BIT IS SET, NO FURTHER ACTION IS TAKEN.
;    5898  1     !       OTHERWISE, THE SA VALUE IS CHECKED FOR A FATAL ERROR, AND THE COMMAND
;    5899  1     !       AND RESPONSE RINGS ARE POLLED.
;    5900  1     !-
;    5901  1
;    5902  2         begin
;    5903  2         IDCT_ADDR = DCT + (.ICTLR * DCT_LEN * 2);           ! GET DCT ADDRESS
;    5904  2         ICST_ADDR = CST + (.ICTLR * CST_LEN * 2);           ! GET CST ADDRESS
;    5905  2         IRDRX_ADDR = .ICST_ADDR [IP_ADDR];                  ! GET RDRX ADDRESS
;    5906  2         ICOM_ADDR = COMM_AREA + (.ICTLR * COMM_LEN * 2);    ! GET COMM_AREA ADDR
;    5907  2         IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];   ! SAVE SA REGISTER
;    5908  2
;    5909  2         if .IDCT_ADDR [IG_INT]                              ! IGNORE INTERRUPT?
;    5910  2         then
;    5911  2             return;                                         ! RETURN IF INTERRUPTS IGNORED
;    5912  2
;    5913  3         if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)            ! IF FATAL ERROR
;    5914  2         then
;    5915  2             FATAL_ERROR ()
;    5916  2         else
;    5917  3             begin
;    5918  3             POLL_CRING ();                                  ! POLL COMMAND RING
;    5919  3             POLL_RRING ();                                  ! POLL RESPONSE RING
;    5920  2             end;
;    5921  2
;    5922  1         end;


                                        .SBTTL  AZINT RDRX INTERRUPT SERVICE ROUTINES
000000  010146                 AZINT:: MOV     R1,-(SP)                        ;                       5891
000002  005746                         TST     -(SP)
000004  013701  000104'                MOV     ICTLR,R1                        ;                       5903
000010  010146                         MOV     R1,-(SP)
000012  012746  000022                 MOV     #22,-(SP)
000016  004737  000000G                JSR     PC,BL%MUL
000022  062700  000000G                ADD     #DCT,R0
000026  010037  000100'                MOV     R0,IDCT.ADDR
000032  010116                         MOV     R1,(SP)                         ;                       5904
000034  012746  000126                 MOV     #126,-(SP)
000040  004737  000000G                JSR     PC,BL%MUL
000044  062700  000000G                ADD     #CST,R0
000050  010037  000076'                MOV     R0,ICST.ADDR
000054  011037  000000G                MOV     (R0),IRDRX.ADDR                 ; ICST.ADDR,*           5905
000060  010116                         MOV     R1,(SP)                         ;                       5906
000062  012746  000050                 MOV     #50,-(SP)
000066  004737  000000G                JSR     PC,BL%MUL
000072  062700  000000'                ADD     #COMM.AREA,R0
```

```
000076  010037  000074'                        MOV     R0,ICOM.ADDR
000102  013701  000100'                        MOV     IDCT.ADDR,R1            ;                       5907
000106  013700  000000G                        MOV     IRDRX.ADDR,R0
000112  016066  000002  000010                 MOV     2(R0),10(SP)            ; *,RC.REG
000120  016661  000010  000002                 MOV     10(SP),2(R1)            ; RC.REG,*
000126  032711  040000                         BIT     #40000,(R1)             ; *,IDCT.ADDR           5909
000132  001016                                 BNE     2$                      ;                       5891
000134  016601  000010                         MOV     10(SP),R1               ;                       5913
000140  042701  077777                         BIC     #77777,R1
000144  020127  100000                         CMP     R1,#-100000
000150  001003                                 BNE     1$
000152  004737  000000V                        JSR     PC,FATAL.ERROR          ;                       5915
000156  000404                                 BR      2$                      ;                       5913
000160  004737  000000V             1$:        JSR     PC,POLL.CRING           ;                       5918
000164  004737  000000V                        JSR     PC,POLL.RRING           ;                       5919
000170  062706  000012             2$:        ADD     #12,SP                  ;                       5891
000174  012601                                 MOV     (SP)+,R1
000176  000207                                 RTS     PC

; Routine Size:  64 words,      Routine Base:  $CODE$ + 25512
; Maximum stack depth per invocation:  7 words


;   5923  1
```

```
;  5924
;  5925          !+
;  5926  '       GLOBAL ROUTINE DUP_RSP : NOVALUE =                                  !ZZZ
;  5927  1
;  5928  1       !-
;  5929  1       !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH DUP RESPONSE
;  5930  1       !       ITS GENERAL PURPOSE IS TO ACT ON A DATAGRAM OR SEQUENTIAL MESSAGE.
;  5931  1       !       IF THE MESSAGE TYPE IS SEQUENTIAL, THE ROUTINE COPIES THE
;  5932  1       !       CONTENTS OF THE MESSAGE ENVELOPE INTO A RETURN PACKET SO THAT THE
;  5933  1       !       ENVELOPE CAN BE RETURNED TO THE CONTROLLER.
;  5934  1       !
;  5935  1       !       IMPLICIT INPUTS:
;  5936  1       !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;  5937  1       !               IPKT_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING RESPONSE
;  5938  1       !-
;  5939  2           begin
;  5940  2
;  5941  2           local
;  5942  2               R_INDEX : signed word,
;  5943  2               DEBUG,                                                  !ZZZ
;  5944  2               SRC_ADDR,
;  5945  2               DST_ADDR,
;  5946  2               R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;  5947  2           !PRINTX (DER34);
;  5948  2
;  5949  2           incr COUNT from 0 to PKT_CNT - 1 do
;  5950  2
;  5951  2               if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and     ! IF THIS IS THE ASSOC CMD
;  5952  2                  (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  5953  2                  (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;  5954  2                  ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;  5955  2                  (.MSCP_PKT [.COUNT, CONNID] eql CID_DUP) and
;  5956  3                  ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END)
;  5957  2               then
;  5958  3                   begin
;  5959  3                   P_INDEX = .COUNT;                          ! SET PKT NUMBER
;  5960  3                   exitloop;
;  5961  2                   end;
;  5962  2
;  5963  2           if .P_INDEX lss 0                                  ! IF COMMAND NOT FOUND
;  5964  2           then
;  5965  3               begin
;  5966  3               PRINTF (DBM108, .IPKT_ADDR [CRN_LO]);          ! UNKNOWN COMMAND REF. NUMBER
;  5967  3               return;
;  5968  2               end;
;  5969  2
;  5970  2           if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0           ! IF RETPKT IS NOT AVAILABLE
;  5971  2           then
;  5972  2               DEBUG = TRUE                                   !TO SEE IF THIS PATH TAKEN          ZZZ
;  5973  2     !         PRINTF (DBM112)                                ! "DUP-RSP: RETPKT NOT AVAILABLE"   ZZZ
;  5974  2           else
;  5975  3               begin
;  5976  3               SRC_ADDR = .IPKT ADDR + 6;                     ! SET UP COPY (SKIP OVER PKT DESC)
```

```
;    5977  3          R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);   ! START OF ALLOCATED RETPKT
;    5978  3
;    5979  3          incr COUNT from 1 to RP_LEN do
;    5980  4              begin
;    5981  4              .DST_ADDR = ..SRC_ADDR;                 ! COPY 1 WORD
;    5982  4              DST_ADDR = .DST_ADDR + 2;               ! ADVANCE DESTINATION ADDR
;    5983  4              SRC_ADDR = .SRC_ADDR + 2;               ! ADVANCE SOURCE ADDR
;    5984  3              end;                                    ! COPY LOOP
;    5985  3
;    5986  3          IN_IODQ (.R_INDEX);                         ! PUT RETPKT INDEX INTO IODQ
;    5987  2          end;                                        ! IF RETPKT WAS ALLOCATED
;    5988  2
;    5989  2
;    5990  2      if .P_INDEX geq 0                               ! IF ASSOC CMD PKT WAS FOUND
;    5991  2      then
;    5992  2          PUT_PKT (.P_INDEX);                         ! RETURN COMMAND PACKET TO POOL
;    5993  2
;    5994  1      end;                                            ! ROUTINE DUP-RSP


                                        .SBTTL  DUP.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          DUP.RSP::
                                        JSR     R1,$SAVE3                       ;                               5926
000004  013701  000000G                 MOV     IPKT.ADDR,R1                    ;                               5951
000010  005002                          CLR     R2                              ; COUNT                         5949
000012  010246                  1$:     MOV     R2,-(SP)                        ; COUNT,*                       5951
000014  012746  000106                  MOV     #106,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  022626                          CMP     (SP)+,(SP)+
000026  026061  000012G 000012          CMP     MSCP.PKT+12(R0),12(R1)
000034  001024                          BNE     2$
000036  026061  000014G 000014          CMP     MSCP.PKT+14(R0),14(R1)          ;                               5952
000044  001020                          BNE     2$
000046  026011  000000G                 CMP     MSCP.PKT(R0),(R1)               ;                               5953
000052  001415                          BEQ     2$
000054  105760  000022G                 TSTB    MSCP.PKT+22(R0)                 ;                               5954
000060  100412                          BMI     2$
000062  126027  000011G 000002          CMPB    MSCP.PKT+11(R0),#2              ;                               5955
000070  001006                          BNE     2$
000072  105761  000022                  TSTB    22(R1)                          ;                               5956
000076  100003                          BPL     2$
000100  010237  000000G                 MOV     R2,P.INDEX                      ; COUNT,*                       5959
000104  000406                          BR      3$                              ;                               5958
000106  005202                  2$:     INC     R2                              ; COUNT                         5949
000110  020227  000013                  CMP     R2,#13                          ; COUNT,*
000114  003736                          BLE     1$
000116  005737  000000G                 TST     P.INDEX                         ;                               5963
000122  002013                  3$:     BGE     4$
000124  016146  000012                  MOV     12(R1),-(SP)                    ;                               5966
000130  012746  000000G                 MOV     #DBM108,-(SP)
000134  012746  000002                  MOV     #2,-(SP)
000140  010600                          MOV     SP,R0                           ; SP,*
000142  104417                          TRAP    17
```

```
000144  062706  000006                    ADD     #6,SP                   ;                               5967
000150  000207                            RTS     PC                      ;                               5965
000152  013746  000104'          4$:      MOV     ICTLR,-(SP)             ;                               5970
000156  004737  000000G                   JSR     PC,GET.RETPKT
000162  010001                            MOV     R0,R1                   ; *,R.INDEX
000164  005726                            TST     (SP)+
000166  005701                            TST     R1                      ; R.INDEX
000170  002003                            BGE     5$
000172  012700  000001                    MOV     #1,R0                   ; *,DEBUG                       5972
000176  000425                            BR      7$                      ;                               5970
000200  013702  000000G          5$:      MOV     IPKT.ADDR,R2            ; *,SRC.ADDR                    597-
000204  062702  000006                    ADD     #6,R2                   ; *,SRC.ADDR
000210  010146                            MOV     R1,-(SP)                ; R.INDEX,*                     5977
000212  012746  000054                    MOV     #54,-(SP)
000216  004737  000000G                   JSR     PC,BL$MUL
000222  062700  000000G                   ADD     #RETPKT,R0
000226  010003                            MOV     R0,R3                   ; *,DST.ADDR
000230  012700  000026                    MOV     #26,R0                  ; *,COUNT                       5979
000234  012223          6$:      MOV     (R2)+,(R3)+             ; SRC.ADDR,DST.ADDR             5981
000236  005300                            DEC     R0                      ; COUNT                         5979
000240  001375                            BNE     6$
000242  010116                            MOV     R1,(SP)                 ; R.INDEX,*                     5986
000244  004737  000000G                   JSR     PC,IN.IODQ
000250  022626                            CMP     (SP)+,(SP)+             ;                               5975
000252  013700  000000G          7$:      MOV     P.INDEX,R0              ;                               5990
000256  002404                            BLT     8$
000260  010046                            MOV     R0,-(SP)                ;                               5992
000262  004737  000000G                   JSR     PC,PUT.PKT
000266  005726                            TST     (SP)+
000270  000207                  8$:       RTS     PC                      ;                               5926

; Routine Size:  93 words,      Routine Base:  $CODE$ + 25712
; Maximum stack depth per invocation:  9 words


;   5995  1
```

```
;    5996  1     GLOBAL routine FATAL_ERROR : novalue =
;    5997  1
;    5998  1     !+
;    5999  1     !       THIS ROUTINE IS CALLED BY THE INTERRUPT SERVICE ROUTINE (AZINT) UPON
;    6000  1     !       DETECTING AN UNRECOVERABLE ERROR THROUGH THE DEVICE'S SA REGISTER.
;    6001  1     !       ITS PURPOSE IS TO CLEAN UP DEVICE DATA IN THE "DRIVER" PORTION OF
;    6002  1     !       THE EXERCISER, AND TO INFORM THE "PROGRAM' PORTION OF THE EVENT VIA
;    6003  1     !       RETURN PACKET.
;    6004  1     !
;    6005  1     !       IMPLICIT INPUTS:
;    6006  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    6007  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    6008  1     !               ICST_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S CST
;    6009  1     !               IRCRX_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S IP REGISTER
;    6010  1     !-
;    6011  1
;    6012  2         begin
;    6013  2
;    6014  2         local
;    6015  2             index : signed word,
;    6016  2             U_SAVE : word;
;    6017  2
;    6018  2         SA_REG = .IDCT_ADDR [SA_SAVE];
;    6019  2         U_SAVE = .L$LUN;                                    ! SAVE PRE-INTERRUPT CURRENT UNIT NUMBER
;    6020  2         C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;    6021  2
;    6022  2         if .APT_MODE
;    6023  2         then
;    6024  3             begin
;    6025  3             .MAIL_BOX_TESTNUM = 1;
;    6026  3             .MAIL_BOX_SUBTST = 0;
;    6027  2             end;
;    6028  2
;    6029  2         L$LUN = .ICST_ADDR [OF_UN + OF_DATA, D_UNIT];       ! SET CURRENT UNIT TO FIRST IN CONTROLLER
;    6030  2         ERRDF (14, EGD_14, EMS_14);                         ! FATAL CONTROLLER ERROR
;    6031  2         L$LUN = .U_SAVE;                                    ! RESTORE PRE-INTERRUPT CURRENT UNIT
;    6032  2         DRV_CTLERR (.ICTLR);                                ! CLEAN UP DRIVER DATA FOR CONTROLLER
;    6033  2
;    6034  2         if (index = GET_RETPKT (.ICTLR)) lss 0              ! TRY TO GET A RETPKT; IF FAILURE
;    6035  2         then
;    6036  3             PRINTF (DBM18)                                  ! "FATAL_ERROR: RETPKT NOT AVAILABLE"
;    6037  2         else
;    6038  3             begin                                           ! IF RETPKT WAS ALLOCATED
;    6039  3             RETPKT [.index, CONID] = CID_DRIVER;            ! SET CONNECTION ID TO "DRIVER"
;    6040  3             RETPKT [.index, MESTYP] = MT_FATAL;             ! FATAL ERROR
;    6041  3             RETPKT [.index, CTLR] = .ICTLR;                 ! CONTROLLER NUMBER
;    6042  3             IN_IODQ (.index);                               ! LOAD RETPKT INDEX INTO IODQ
;    6043  2             end;                                            ! IF RETPKT WAS ALLOCATED
;    6044  2
;    6045  1         end;                                                ! ROUTINE FATAL_ERR

                                .SBTTL  FATAL.ERROR RDRX INTERRUPT SERVICE ROUTINES
```

```
000000  004137  000000G                FATAL.ERROR::
                                                JSR     R1,$SAVE2               ;                               5996
000004  013700  000100'                        MOV     IDCT.ADDR,R0            ;                               6018
000010  016037  000002  000000G                MOV     2(R0),SA.REG
000016  013701  000000G                        MOV     L$LUN,R1                ; *,U SAVE                      6019
000022  013700  000104'                        MOV     ICTLR,R0                ;                               6020
000026  006300                                  ASL     R0
000030  105260  000000G                        INCB    C.ERR.TBL(R0)
000034  032737  000001  001254'                BIT     #1,APT.MODE             ;                               6022
000042  001405                                  BEQ     1$
000044  012777  000001  001256'                MOV     #1,@MAIL.BOX.TESTNUM    ;                               6025
000052  005077  001260'                        CLR     @MAIL.BOX.SUBTST        ;                               6026
000056  013700  000076'                1$:     MOV     ICST.ADDR,R0            ;                               6029
000062  016002  000006                        MOV     6(R0),R2
000066  000302                                SWAB    R2
000070  042702  177760                        BIC     #177760,R2
000074  010237  000000G                        MOV     R2,L$LUN
000100  104455                                TRAP    55                      ;                               6030
000102  000016                                .WORD   16
000104  000000G                               .WORD   EGD.14
000106  000000G                               .WORD   EMS.14
000110  010137  000000G                        MOV     R1,L$LUN                ; U.SAVE,*                      6031
000114  013746  000104'                        MOV     ICTLR,-(SP)             ;                               6032
000120  004737  000000G                        JSR     PC,DRV.CTLERR
000124  013716  000104'                        MOV     ICTLR,(SP)              ;                               6034
000130  004737  000000G                        JSR     PC,GET.RETPKT
000134  010001                                MOV     R0,R1                   ; *,INDEX
000136  002007                                BGE     2$
000140  012716  000000G                        MOV     #DBM18,(SP)             ;                               6036
000144  012746  000001                        MOV     #1,-(SP)
000150  010600                                MOV     SP,R0                   ; SP,*
000152  104417                                TRAP    17
000154  000424                                BR      3$                      ;                               6034
000156  010116                        2$:     MOV     R1,(SP)                 ; INDEX,*                       6039
000160  012746  000054                        MOV     #54,-(SP)
000164  004737  000000G                        JSR     PC,BL$MUL
000170  062700  000002G                        ADD     #RETPKT+2,R0
000174  112760  000003  000001               MOVB    #3,1(R0)
000202  013702  000104'                        MOV     ICTLR,R2                ;                               6041
000206  042702  177760                        BIC     #177760,R2
000212  112710  000060                        MOVB    #60,(R0)
000216  150210                                BISB    R2,(R0)
000220  010116                                MOV     R1,(SP)                 ; INDEX,*                       6042
000222  004737  000000G                        JSR     PC,IN.IODQ
000226  022626                        3$:     CMP     (SP)+,(SP)+             ;                               6012
000230  000207                                RTS     PC                      ;                               5996

; Routine Size:  77 words,      Routine Base:  $CODE$ + 26204
; Maximum stack depth per invocation:  7 words
```

```
;    6046  1     GLOBAL routine POLL_CRING : novalue =
;    6047  1
;    6048  1     !+
;    6049  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    6050  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    6051  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S COMMAND RING AND CHECK FOR ANY
;    6052  1     !       COMMAND SLOTS THAT HAVE BEEN "TAKEN" BY THE CONTROLLER. SUCH SLOTS
;    6053  1     !       HAVE BEEN RETURNED TO THE HOST, INDICATED BY A ZERO OWNERSHIP BIT. FOR
;    6054  1     !       EACH SLOT THAT HAS BEEN RETURNED TO THE HOST, THE CRING COUNT IS
;    6055  1     !       DECREMENTED, AND THE CR_POLL ADDRESS IS ADVANCED TO THE NEXT SLOT IN
;    6056  1     !       THE COMMAND RING.
;    6057  1     !
;    6058  1     !       IMPLICIT INPUTS:
;    6059  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    6060  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    6061  1     !               ICOM_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S COMM_AREA
;    6062  1     !-
;    6063  1
;    6064  2         begin
;    6065  2         ICOM_ADDR [CMD_INT] = 0;                                    ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER    ZZZ
;    6066  2
;    6067  3         while ((.IDCT_ADDR [CRING_CNT] gtru 0) and                  ! WHILE # OF COMMANDS IN CRING > 0 AND
;    6068  2             not (BIT_TST ((.IDCT_ADDR [CR_POLL] + 2), ED_OWN))) do  ! CURRENT SLOT IS HOST-OWNED
;    6069  3             begin
;    6070  3             IDCT_ADDR [CRING_CNT] = .IDCT_ADDR [CRING_CNT] - 1;     ! DECREMENT # CMDs IN CRING
;    6071  3             IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_POLL] + 4;         ! ADVANCE TO NEXT SLOT TO POLL
;    6072  3
;    6073  3             if .IDCT_ADDR [CR_POLL] gtre .IDCT_ADDR [CR_END]        ! IF BEYOND END OF RING
;    6074  3             then
;    6075  3                 IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_BEG];          ! SET POINTER TO TOP OF CRING
;    6076  3
;    6077  2             end;
;    6078  2
;    6079  2     !ZZZ    ICOM_ADDR [CMD_INT] = 0;                                    ! CLEAR COMMAND INTERRUPT WORD IN RING HEADE
R
;    6080  1         end;


                                        .SBTTL  POLL.CRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G        POLL.CRING::
                                JSR     R1,$SAVE2                       ;                       6046
000004  013700  000074'                 MOV     ICOM.ADDR,R0            ;                       6065
000010  005060  000004                  CLR     4(R0)
000014  013701  000100'                 MOV     IDCT.ADDR,R1            ;                       6067
000020  012702  000016                  MOV     #16,R2                  ;                       6071
000024  060102                          ADD     R1,R2
000026  105711                1$:       TSTB    (R1)                    ;                       6067
000030  001422                          BEQ     2$
000032  016100  000016                  MOV     16(R1),R0               ;                       6068
000036  016000  000002                  MOV     2(R0),R0
000042  042700  077777                  BIC     #77777,R0
000046  020027  100000                  CMP     R0,#-100000
000052  001411                          BEQ     2$
000054  105311                          DECB    (R1)                    ;                       6070
```

```
ZRQAM3          RD/RX EXERCISER                                 4-Apr-1985 13:23:31    VAX-11 Bliss-16 V4.1-582              SEQ 0446
V02.2           RDRX INTERRUPT SERVICE ROUTINES                 2-Apr-1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;19  Page 191
                                                                                                                             (52)

000056  062712  000004                          ADD     #4,(R2)                 ;                                       6071
000062  021261  000012                          CMP     (R2),12(R1)             ;                                       6073
000066  101757                                  BLOS    1$                                                              
000070  016112  000000                          MOV     10(R1),(R2)             ;                                       6075
000074  000754                                  BR      1$                      ;                                       6067
000076  000207                          2$:     RTS     PC                      ;                                       6046

; Routine Size:  32 words.      Routine Base:  $CODE$ + 26436
; Maximum stack depth per invocation:  4 words
```

```
;    6081  1     GLOBAL routine POLL_RRING : novalue =
;    6082  1
;    6083  1
;    6084  1     !+
;    6085  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    6086  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    6087  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S RESPONSE RING AND CHECK FOR ANY
;    6088  1     !       SLOTS WHICH HAVE BEEN RETURNED TO THE HOST (OWNERSHIP BIT = 0). FOR
;    6089  1     !       EACH SUCH SLOT, THE ASSOCIATED MESSAGE IS PROCESSED BASED ON ITS
;    6090  1     !       CONNECTION ID (DISK OR DUP). AFTER PROCESSING, THE MESSAGE PACKET
;    6091  1     !       IS RE-INITIALIZED AND RETURNED TO THE CONTROLLER (OWNERSHIP BIT SET
;    6092  1     !       TO 1).
;    6093  1     !
;    6094  1     !       IMPLICIT INPUTS:
;    6095  1     !               ICTLR - NUMBER OF INTERRUPTING CONTROLLER
;    6096  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    6097  1     !-
;    6098  1
;    6099  1
;    6100  2         begin
;    6101  2         ICOM_ADDR [RSP_INT] = 0;                                    ! CLR RESPONSE INTERRUPT WRD IN RING HEADER
    ZZZ
;    6102  2
;    6103  2
;    6104  2         while not (BIT_TST ((.IDCT_ADDR [RR_POLL] + 2), ED_OWN)) do ! WHILE 0 = 0
;    6105  3             begin
;    6106  3             IPKT_ADDR = ..IDCT_ADDR [RR_POLL] - 10;                 ! ADDRESS OF RESPONSE PACKET
;    6107  3
;    6108  3
;    6109  4     IF NOT (.IPKT_ADDR [CONNID] EQL CID_DUP)                        !                ZZZ
;    6110  3             THEN                                                    !                ZZZ
;    6111  3             (CREDIT_BAL = .CREDIT_BAL + .IPKT_ADDR [CREDITS]);      !ZZZ
;    6112  3     !IT WAS NOTICE THAT DUP WAS SENDIND BACK CREDITS WHICH IT SHOULD NOT.   !ZZZ
;    6113  3             selectoneu .IPKT_ADDR [CONNID] of
;    6114  3                 set
;    6115  3
;    6116  3                 [CID_DISK] :        DISK_RSP ();
;    6117  3
;    6118  3                 [CID_DUP]  :        DUP_RSP  ();                    !ZZZ
;    6119  3
;    6120  3                 [otherwise] :       PRINTF (DBM20, .IPKT_ADDR [CONNID], .IRDRX_ADDR);
;    6121  3                                                                     ! "BAD CONN ID = XXXXX FROM XXXXXX"
;    6122  3                 tes;
;    6123  3
;    6124  3
;    6125  3             IPKT_ADDR [MSGLEN] = MSG_LEN * 2;                       ! RE-INIT PKT FIELDS; MESSAGE LENGTH
;    6126  3             IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;         ! ADVANCE TO HI ORDER WORD OF RING SLOT
;    6127  3             .IDCT_ADDR [RR_POLL] = .IPKT_ADDR [PKT_HI];             ! RETURN SLOT TO CONTROLLER
;    6128  3             .IDCT_ADDR [RR_POLL] = ..IDCT_ADDR [RR_POLL] or ED_OWN or ED_FLAG; ! OWNERSHIP TOO
;    6129  3             IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;         ! ADVANCE TO NEXT RRING SLOT
;    6130  3
;    6131  3
;    6132  3             if .IDCT_ADDR [RR_POLL] gtra .IDCT_ADDR [RR END]        ! IF BEYOND END OF RING
;    6133  3
```

```
;   6134  3              then
;   6135  3                  IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_BEG];          ! CYCLE TO TOP OF RING
;   6136  3
;   6137  2              end;                                                    ! WHILE LOOP
;   6138  2
;   6139  2      !ZZZ    ICOM_ADDR [RSP_INT] = 0;                                        ! CLR RESPONSE INTERRUPT WRD IN RING
 HEADER
;   6140  1          end;


                                        .SBTTL  POLL.RRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G        POLL.RRING::
                                        JSR     R1,$SAVE3                       ;                                       6081
000004  013700  000074'                 MOV     ICOM.ADDR,R0                    ;                                       6101
000010  005060  000006                  CLR     6(R0)
000014  013701  000100'                 MOV     IDCT.ADDR,R1                    ;                                       6104
000020  062701  000014                  ADD     #14,R1
000024  011100                  1$:     MOV     (R1),R0
000026  016000  000002                  MOV     2(R0),R0
000032  042700  077777                  BIC     #77777,R0
000036  020027  100000                  CMP     R0,#-100000
000042  001504                          BEQ     6$
000044  017137  000000  000000G         MOV     @0(R1),IPKT.ADDR                ;                                       6106
000052  162737  000012  000000G         SUB     #12,IPKT.ADDR
000060  013700  000000G                 MOV     IPKT.ADDR,R0                    ;                                       6109
000064  005002                          CLR     R2
000066  156002  000011                  BISB    11(R0),R2
000072  020227  000002                  CMP     R2,#2
000076  001406                          BEQ     2$
000100  116003  000010                  MOVB    10(R0),R3                       ;                                       6111
000104  042703  177760                  BIC     #177760,R3
000110  060337  000000G                 ADD     R3,CREDIT.BAL
000114  005702                  2$:     TST     R2                              ;                                       6116
000116  001003                          BNE     3$
000120  004737  000000V                 JSR     PC,DISK.RSP
000124  000421                          BR      5$                              ;                                       6113
000126  020227  000002          3$:     CMP     R2,#2                           ;                                       6118
000132  001003                          BNE     4$
000134  004737  025712'                 JSR     PC,DUP.RSP
000140  000413                          BR      5$                              ;                                       6113
000142  013746  000000G         4$:     MOV     IRDRX.ADDR,-(SP)                ;                                       6120
000146  010246                          MOV     R2,-(SP)
000150  012746  000000G                 MOV     #DBM20,-(SP)
000154  012746  000003                  MOV     #3,-(SP)
000160  010600                          MOV     SP,R0                           ; SP,*
000162  104417                          TRAP    17
000164  062706  000010                  ADD     #10,SP
000170  013700  000000G         5$:     MOV     IPKT.ADDR,R0                    ;                                       6125
000174  012760  000074  000006         MOV     #74,6(R0)
000202  013702  000100'                 MOV     IDCT.ADDR,R2                    ;                                       6126
000206  010201                          MOV     R2,R1
000210  062701  000014                  ADD     #14,R1
000214  062711  000002                  ADD     #2,(R1)
000220  016071  000002  000000         MOV     2(R0),@0(R1)                    ;                                       6127
```

```
000226  052771  140000  000000                  BIS     # 40000,@0(R1)          ;                       6128
000234  062711  000002                          ADD     #2,(R1)                 ;                       6129
000240  021162  000006                          CMP     (R1),6(R2)              ;                       6132
000244  101667                                  BLOS    1$
000246  016211  000004                          MOV     4(R2),(R1)              ;                       6135
000252  000664                                  BR      1$                      ;                       6104
000254  000207                          6$:     RTS     PC                      ;                       6081

; Routine Size:  87 words,      Routine Base:  $CODE$ + 26536
; Maximum stack depth per invocation:  10 words


;   6141  1
```

```
;    6142  1
;    6143  1
;    6144  1     GLOBAL routine DISK_RSP : novalue =
;    6145  1
;    6146  1
;    6147  1     !+
;    6148  1     !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH RESPONSE MESSAGE
;    6149  1     !       WHICH HAS A CONNECTION ID INDICATING A DISK MSCP ORIGINATOR
;    6150  1     !       (I.E., ALL EXCEPT DUP RESPONSES). ITS PURPOSE IS TO PASS
;    6151  1     !       CONTROL TO THE APPROPRIATE ROUTINE BASED ON THE MESSAGE TYPE
;    6152  1     !       FIELD (SEQUENTIAL, DATAGRAM, OR CREDIT NOTIFICATION).
;    6153  1     !
;    6154  1     !       IMPLICIT INPUTS:
;    6155  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    6156  1     !                           MESSAGE
;    6157  1     !-
;    6158  1
;    6159  1
;    6160  1
;    6161  1         selectoneu .IPKT_ADDR [MSGTYP] of
;    6162  1
;    6163  1
;    6164  1             set
;    6165  1
;    6166  1             [MT_SEQ] :      SEQUEN ();
;    6167  1
;    6168  1             [MT_DG] :       DATAGM ();
;    6169  1
;    6170  1             [otherwise] :   PRINTF (DBM21, .IPKT_ADDR [MSGTYP]);    ! "MESSAGE TYPE XX RECEIVED'
;    6171  1             tes;


                                        .SBTTL  DISK.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  010146                  DISK.RSP::
                                MOV     R1,-(SP)                        ;                       6144
000002  013700  000000G                 MOV     IPKT.ADDR,R0            ;                       6161
000006  116001  000010                  MOVB    10(R0),R1
000012  006201                          ASR     R1
000014  006201                          ASR     R1
000016  006201                          ASR     R1
000020  006201                          ASR     R1
000022  042701  177760                  BIC     #177760,R1
000026  001003                          BNE     1$                      ;                       6166
000030  004737  000000V                 JSR     PC,SEQUEN
000034  000417                          BR      3$                      ;                       6161
000036  020127  000001          1$:     CMP     R1,#1                   ;                       6168
000042  001003                          BNE     2$
000044  004737  000000V                 JSR     PC,DATAGM
000050  000411                          BR      3$                      ;                       6161
000052  010146                  2$:     MOV     R1,-(SP)                ;                       6170
000054  012746  000000G                 MOV     #DBM21,-(SP)
000060  012746  000002                  MOV     #2,-(SP)
000064  010600                          MOV     SP,R0                   ; SP,*
```

```
ZRQAM3          RD/RX EXERCISER                                 4-Apr-1985 13:23:31    VAX-11 Bliss-16 V4.1-582              SEQ 0451
V02.2           RDRX INTERRUPT SERVICE ROUTINES                 2 Apr-1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;19  Page 196
                                                                                                                             (54)

000066  104417                                  TRAP    17
000070  062706  000006                          ADD     #6,SP
000074  012601                          3$:     MOV     (SP)+,R1                                                        6144
000076  000207                                  RTS     PC

; Routine Size:  32 words,      Routine Base:  $CODE$ + 27014
; Maximum stack depth per invocation:  6 words


;   6172  1
;   6173  1
;   6174  1
```

```
;    6175  1     GLOBAL routine SEQUEN : novalue =
;    6176  1
;    6177  1     !+
;    6178  1     !       THIS ROUTINE IS CALLED BY DISK_RSP FOR EACH DISK MSCP RESPONSE MESSAGE
;    6179  1     !       WITH THE "SEQUENTIAL" MESSAGE TYPE. ITS GENERAL PURPOSE IS TO COPY THE
;    6180  1     !       CONTENTS OF THE MESSAGE PACKET INTO A RETURN PACKET SO THAT THE
;    6181  1     !       PACKET CAN BE RETURNED TO THE CONTROLLER.   IN ADDITION,
;    6182  1     !       IF THE COMMAND WAS AN I/O TRANSFER (READ, WRITE, OR ACCESS), THEN SOME
;    6183  1     !       FIELDS OF THE COMMAND PACKET ARE COPIED INTO THE RETURN PACKET.
;    6184  1     !
;    6185  1     !       IMPLICIT INPUTS:
;    6186  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    6187  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    6188  1     !-
;    6189  1
;    6190  2         begin
;    6191  2
;    6192  2         local
;    6193  2             P_INDEX : signed word initial (-1),                          ! ASSUME NO ASSOCIATED COMMAND PKT
;    6194  2             R_INDEX : signed word,
;    6195  2             SRC_ADDR,
;    6196  2             DST_ADDR,
;    6197  2             R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;    6198  2
;    6199  2         incr COUNT from 0 to PKT_CNT - 1 do
;    6200  2
;    6201  2             if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and          ! IF THIS IS THE ASSOC CMD
;    6202  2                (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    6203  2                (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;    6204  2                ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;    6205  2                (.MSCP_PKT [.COUNT, MSGTYP] eql MT_SEQ) and
;    6206  2                ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END) and
;    6207  3                (.PKT_USE [.COUNT] eql .ICTLR)
;    6208  2             then
;    6209  3                 begin
;    6210  3                 P_INDEX = .COUNT;                                       ! SET PKT NUMBER
;    6211  3                 exitloop;
;    6212  2                 end;
;    6213  2
;    6214  2         if .P_INDEX lss 0                                               ! IF COMMAND NOT FOUND
;    6215  2         then
;    6216  3             begin
;    6217  3             PRINTF (DBM108, .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);          ! UNKNOWN COMMAND REF. NUMBER
;    6218  3             return;
;    6219  2             end;
;    6220  2
;    6221  3         if .MSCP_PKT [.P_INDEX, OPCODE] neq (.IPKT_ADDR [OPCODE] and (not OP_END))  ! IF OPCODE MISMATCH
;    6222  2         then
;    6223  2             PRINTF (DBM111, .MSCP_PKT [.P_INDEX, OPCODE], .IPKT_ADDR [OPCODE], .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO])
;
;    6224  2
;    6225  3         if ((.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or
;    6226  2             (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END))) and
;    6227  3            ((.IPKT_ADDR [STATUS_CODE] eql ST_SUC) and
```

```
;   6228  2              (.IPKT_ADDR [STATUS_SUBCODE] eql 0)) and
;   6229  3             ((.MSCP_PKT [.P_INDEX, BC_LO] neq .IPKT_ADDR [BC_LO]) or
;   6230  3              (.MSCP_PKT [.P_INDEX, BC_HI] neq .IPKT_ADDR [BC_HI]))
;   6231  2          then
; P 6232  2              PRINTF (DBM112,
; P 6233  2                      .MSCP_PKT [.P_INDEX, BC_HI], .MSCP_PKT [.P_INDEX, BC_LO], .IPKT_ADDR .BC_HI], .IPKT_ADDR [BC LO],
;   6234  2                      .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);
;   6235  2
;   6236  2          if .MSCP_PKT [.P_INDEX, RSP_RECEIVED]
;   6237  2          then
;   6238  3              begin
;   6239  3              PRINTF (DBM120, .MSCP_PKT [.P_INDEX, CRN_HI], .MSCP_PKT [.P_INDEX, CRN_LO]);
;   6240  3              PUT_PKT (.P_INDEX);
;   6241  3              return;
;   6242  3              end
;   6243  2          else
;   6244  2              MSCP_PKT [.P_INDEX, RSP_RECEIVED] = TRUE;                          ! MARK RESPONSE RECEIVED
;   6245  2
;   6246  2          if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0                           ! IF RETPKT IS NOT AVAILABLE
;   6247  2          then
;   6248  3              begin
;   6249  3              PRINTF (DBM22);                                                ! "SEQUEN: RETPKT NOT AVAILABLE"
;   6250  3              PUT_PKT (.P_INDEX);
;   6251  3              return;
;   6252  3              end
;   6253  2          else
;   6254  3              begin
;   6255  3              SRC_ADDR = .IPKT_ADDR + 6;                                     ! SET UP COPY (SKIP OVER PKT DESC)
;   6256  3              R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);          . START OF ALLOCATED RETPKT
;   6257  3
;   6258  3              incr COUNT from 1 to RP_LEN do
;   6259  4                  begin
;   6260  4                  .DST_ADDR = ..SRC_ADDR;                                    ! COPY 1 WORD
;   6261  4                  DST_ADDR = .DST_ADDR + 2;                                  ! ADVANCE DESTINATION ADDR
;   6262  4                  SRC_ADDR = .SRC_ADDR + 2;                                  ! ADVANCE SOURCE ADDR
;   6263  4
;   6264  5                  if .IPKT_ADDR [OPCODE] eql (OP_ONL or OP_END)              ! IF THIS IS THE ONLINE END MESSAGE
;   6265  4                  then
;   6266  4                      if .COUNT eql 10                                       ! SKIP OVER RESERVED WORDS
;   6267  4                      then
;   6268  4                          SRC_ADDR = .SRC_ADDR + 4;                          !       IN ONLINE END - MESSAGE
;   6269  3                  end;                                                       ! COPY LOOP
;   6270  3
;   6271  3              R_ADDR [CTLR] = .ICTLR;                                        ! LOAD CONTROLLER NUMBER INTO PKT
;   6272  3
;   6273  3              if .P_INDEX geq 0                                              ! IF ASSOC. CMD PKT WAS FOUND
;   6274  3              then
;   6275  3
;   6276  3                  if (.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or          ! IF END MESSAGE IS
;   6277  3                     (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END)) or         !       READ, WRITE, OR
;   6278  4                     (.IPKT_ADDR [OPCODE] eql (OP_ACC or OP_END))            !       ACCESS
;   6279  3                  then
;   6280  4                      begin
```

```
;    6281  4                    R_ADDR [CMDMOD] = .MSCP_PKT [.P_INDEX, MODIFY];          ! COPY
;    6282  4                    R_ADDR [CBCNT_LO] = .MSCP_PKT [.P_INDEX, BC_LO];         ! RELEVANT
;    6283  4                    R_ADDR [CBCNT_HI] = .MSCP_PKT [.P_INDEX, BC_HI];         ! FIELDS
;    6284  4                    R_ADDR [LBN_LO] = .MSCP_PKT [.P_INDEX, LBN_L];           ! FROM
;    6285  4                    R_ADDR [LBN_HI] = .MSCP_PKT [.P_INDEX, LBN_H];           ! COMMAND
;    6286  4                    R_ADDR [BUFF_0] = .MSCP_PKT [.P_INDEX, BUF_0];           ! PACKET
;    6287  4                    R_ADDR [BUFF_1] = .MSCP_PKT [.P_INDEX, BUF_1];           ! TO RETPKT
;    6288  3                    end;                                                     ! IF ENDCODE WAS READ/WRITE/ACCESS
;    6289  3
;    6290  3                IN_IODQ (.R_INDEX);                                          ! PUT RETPKT INDEX INTO IODQ
;    6291  2                end;                                                         ! IF RETPKT WAS ALLOCATED
;    6292  2
;    6293  2            if (.IPKT_ADDR [STATUS_CODE] neq  ST_SUC) or
;    6294  3               (.IPKT_ADDR [STATUS_SUBCODE] neq 0)
;    6295  2            then
;    6296  2                LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_OCCURED                ! SAVE ERROR CONDITION
;    6297  2            else
;    6298  2                LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_NOT_OCCURED;           !
;    6299  2
;    6300  2            LAST_PKT [.ICTLR, LAST_CRN_LO] = .IPKT_ADDR [CRN_LO];            ! SAVE COMMAND REFERENCE NUMBER
;    6301  2            LAST_PKT [.ICTLR, LAST_CRN_HI] = .IPKT_ADDR [CRN_HI];            !
;    6302  2            SCAN_ERRLOG ();                                                  ! PRINT ANY ASSOCIATED ERROR-LOGS
;    6303  2
;    6304  2            if .P_INDEX geq 0                                                ! IF ASSOC CMD PKT WAS FOUND
;    6305  2            then
;    6306  2                PUT_PKT (.P_INDEX);                                          ! RETURN COMMAND PACKET TO POOL
;    6307  2
;    6308  1            end;                                                             ! ROUTINE DISK_RSP
```

```
                                .SBTTL  SEQUEN RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G         SEQUEN::JSR     R1,$SAVE5                       ;                                6175
000004  005746                         TST     -(SP)
000006  012746  177777                 MOV     #-1,-(SP)                       ; *,P.INDEX                      6190
000012  013701  000000G                 MOV     IPKT.ADDR,R1                    ;                                6201
000016  005002                         CLR     R2                              ; COUNT                          6199
000020  010246                 1$:     MOV     R2,-(SP)                        ; COUNT,*                        6201
000022  012746  000106                 MOV     #106,-(SP)
000026  004737  000000G                 JSR     PC,BL$MUL
000032  022626                         CMP     (SP)+,(SP)+
000034  026061  000012G 000012         CMP     MSCP.PKT+12(R0),12(R1)
000042  001030                         BNE     2$
000044  026061  000014G 000014         CMP     MSCP.PKT+14(R0),14(R1)          ;                                6202
000052  001024                         BNE     2$
000054  026011  000000G                 CMP     MSCP.PKT(R0),(R1)               ;                                6203
000060  001421                         BEQ     2$
000062  105760  000022G                 TSTB    MSCP.PKT+22(R0)                 ;                                6204
000066  100416                         BMI     2$
000070  132760  000360  000010G        BITB    #360,MSCP.PKT+10(R0)            ;                                6205
000076  001012                         BNE     2$
000100  105761  000022                 TSTB    22(R1)                          ;                                6206
000104  100007                         BPL     2$
000106  116200  000000G                 MOVB    PKT.USE(R2),R0                  ; *(COUNT),*                     6207
```

```
000112  020037  000104'                        CMP     R0,ICTLR
000116  001002                                 BNE     2$
000120  010216                                 MOV     R2,(SP)         ; COUNT,P.INDEX                 6210
000122  000405                                 BR      3$              ;                               6209
000124  005202                          2$:    INC     R2              ; COUNT                         6199
000126  020227  000013                         CMP     R2,#13          ; COUNT,*
000132  003732                                 BLE     1$
000134  005716                                 TST     (SP)            ; P.INDEX                       6214
000136  002013                          3$:    BGE     4$
000140  016146  000012                         MOV     12(R1),-(SP)    ;                               6217
000144  016146  000014                         MOV     14(R1),-(SP)
000150  012746  000000G                        MOV     #DBM108,-(SP)
000154  012746  000003                         MOV     #3,-(SP)
000160  010600                                 MOV     SP,R0           ; SP,*
000162  104417                                 TRAP    17
000164  000545                                 BR      9$              ;                               6218
000166  011646                          4$:    MOV     (SP),-(SP)      ; P.INDEX,*                     6221
000170  012746  000106                         MOV     #106,-(SP)
000174  004737  000000G                        JSR     PC,BL$MUL
000200  010001                                 MOV     R0,R1
000202  022626                                 CMP     (SP)+,(SP)+
000204  013700  000000C                        MOV     IPKT.ADDR,R0
000210  116003  000022                         MOVB    22(R0),R3
000214  042703  177600                         BIC     #177600,R3
000220  005002                                 CLR     R2
000222  156102  000022G                        BISB    MSCP.PKT+22(R1),R2
000226  020203                                 CMP     R2,R3
000230  001422                                 BEQ     5$
000232  016046  000012                         MOV     12(R0),-(SP)    ;                               6223
000236  016046  000014                         MOV     14(R0),-(SP)
000242  005046                                 CLR     -(SP)
000244  116016  000022                         MOVB    22(R0),(SP)
000250  005046                                 CLR     -(SP)
000252  116116  000022G                        MOVB    MSCP.PKT+22(R1),(SP)
000256  012746  000000G                        MOV     #DBM111,-(SP)
000262  012746  000005                         MOV     #5,-(SP)
000266  010600                                 MOV     SP,R0           ; SP,*
000270  104417                                 TRAP    17
000272  062706  000014                         ADD     #14,SP
000276  013700  000000G                 5$:    MOV     IPKT.ADDR,R0    ;                               6225
000302  126027  000022  000241                 CMPB    22(R0),#241
000310  001404                                 BEQ     6$
000312  126027  000022  000242                 CMPB    22(R0),#242     ;                               6226
000320  001045                                 BNE     8$
000322  012702  000024                 6$:    MOV     #24,R2          ;                               6227
000326  060002                                 ADD     R0,R2
000330  132712  000037                         BITB    #37,(R2)
000334  001037                                 BNE     8$
000336  032712  177740                         BIT     #177740,(R2)    ;                               6228
000342  001034                                 BNE     8$
000344  026160  000026G 000026                 CMP     MSCP.PKT+26(R1),26(R0)  ;                       6229
000352  001004                                 BNE     7$
000354  026160  000030G 000030                 CMP     MSCP.PKT+30(R1),30(R0)  ;                       6230
```

```
000362  001424                          BEQ     8$
000364  016046  000012          7$:     MOV     12(R0),-(SP)            ;                       6234
000370  016046  000014                  MOV     14(R0),-(SP)
000374  016046  000026                  MOV     26(R0),-(SP)
000400  016046  000030                  MOV     30(R0),-(SP)
000404  016146  000026G                 MOV     MSCP.PKT+26(R1),-(SP)
000410  016146  000030G                 MOV     MSCP.PKT+30(R1),-(SP)
000414  012746  000000G                 MOV     #DBM112,-(SP)
000420  012746  000007                  MOV     #7,-(SP)
000424  010600                          MOV     SP,R0                   ; SP,*
000426  104417                          TRAP    17
000430  062706  000020                  ADD     #20,SP
000434  132761  000001  000005G  8$:     BITB    #1,MSCP.PKT+5(R1)       ;                       6236
000442  001422                          BEQ     10$
000444  016146  000012G                 MOV     MSCP.PKT+12(R1),-(SP)   ;                       6239
000450  016146  000014G                 MOV     MSCP.PKT+14(R1),-(SP)
000454  012746  000000G                 MOV     #DBM120,-(SP)
000460  012746  000003                  MOV     #3,-(SP)
000464  010600                          MOV     SP,R0                   ; SP,*
000466  104417                          TRAP    17
000470  016616  000010                  MOV     10(SP),(SP)             ; P.INDEX,*             6240
000474  004737  000000G                 JSR     PC,PUT.PKT
000500  062706  000010          9$:     ADD     #10,SP                  ;                       6241
000504  000137  030254'                 JMP     21$                     ;                       6238
000510  112761  000001  000005G  10$:    MOVB    #1,MSCP.PKT+5(R1)       ;                       6244
000516  013746  000104'                 MOV     ICTLR,-(SP)             ;                       6246
000522  004737  000000G                 JSR     PC,GET.RETPKT
000526  010066  000004                  MOV     R0,4(SP)                ; *,R.INDEX
000532  005726                          TST     (SP)+
000534  005766  000002                  TST     2(SP)                   ; R.INDEX
000540  002007                          BGE     11$
000542  012746  000000G                 MOV     #DBM22,-(SP)            ;                       6249
000546  012746  000001                  MOV     #1,-(SP)
000552  010600                          MOV     SP,R0                   ; SP,*
000554  104417                          TRAP    17
000556  000563                          BR      19$                     ;                       6250
000560  013704  000000G  11$:    MOV     IPKT.ADDR,R4            ; *,SRC.ADDR            6255
000564  062704  000006                  ADD     #6,R4                   ; *,SRC.ADDR
000570  016646  000002                  MOV     2(SP),-(SP)             ; R.INDEX,*             6256
000574  012746  000054                  MOV     #54,-(SP)
000600  004737  000000G                 JSR     PC,BL$MUL
000604  062700  000000G                 ADD     #RETPKT,R0
000610  010005                          MOV     R0,R5                   ; *,DST.ADDR
000612  013702  000000G                 MOV     IPKT.ADDR,R2            ;                       6264
000616  012703  000001                  MOV     #1,R3                   ; *,COUNT               6258
000622  012425          12$:    MOV     (R4)+,(R5)+             ; SRC.ADDR,DST.ADDR     6260
000624  126227  000022  000211          CMPB    22(R2),#211             ;                       6264
000632  001005                          BNE     13$
000634  020327  000012                  CMP     R3,#12                  ; COUNT,*               6266
000640  001002                          BNE     13$
000642  062704  000004                  ADD     #4,R4                   ; *,SRC.ADDR            6268
000646  005203          13$:    INC     R3                      ; COUNT                 6258
000650  020327  000026                  CMP     R3,#26                  ; COUNT,*
```

```
000654  003762                                  BLE     12$
000656  013703  000104'                         MOV     ICTLR,R3                ;                               6271
000662  042703  177760                          BIC     #177760,R3
000666  142760  000017  000002                  BICB    #17,2(R0)               ; *,*(R.ADDR)
000674  150360  000002                          BISB    R3,2(R0)                ; *,*(R.ADDR)
000700  005766  000004                          TST     4(SP)                   ; P.INDEX               6273
000704  002441                                  BLT     15$
000706  005003                                  CLR     R3                      ;                       6276
000710  156203  000022                          BISB    22(R2),R3
000714  020327  000241                          CMP     R3,#241
000720  001406                                  BEQ     14$
000722  020327  000242                          CMP     R3,#242                 ;                       6277
000726  001403                                  BEQ     14$
000730  020327  000220                          CMP     R3,#220                 ;                       6278
000734  001025                                  BNE     15$
000736  016160  000024G 000012          14$:    MOV     MSCP.PKT+24(R1),12(R0)  ; *,*(R.ADDR)           6281
000744  016160  000026G 000044                  MOV     MSCP.PKT+26(R1),44(R0)  ; *,*(R.ADDR)           6282
000752  016160  000030G 000046                  MOV     MSCP.PKT+30(R1),46(R0)  ; *,*(R.ADDR)           6283
000760  016160  000046G 000050                  MOV     MSCP.PKT+46(R1),50(R0)  ; *,*(R.ADDR)           6284
000766  016160  000050G 000052                  MOV     MSCP.PKT+50(R1),52(R0)  ; *,*(R.ADDR)           6285
000774  016160  000032G 000024                  MOV     MSCP.PKT+32(R1),24(R0)  ; *,*(R.ADDR)           6286
001002  016160  000034G 000026                  MOV     MSCP.PKT+34(R1),26(R0)  ; *,*(R.ADDR)           6287
001010  016616  000006                  15$:    MOV     6(SP),(SP)              ; R.INDEX,*             6290
001014  004737  000000G                          JSR     PC,IN.IODQ
001020  005726                                  TST     (SP)+                   ;                       6254
001022  013716  000104'                         MOV     ICTLR,(SP)              ;                       6296
001026  012746  000006                          MOV     #6,-(SP)
001032  004737  000000G                          JSR     PC,BL$MUL
001036  013701  000000G                          MOV     IPKT.ADDR,R1            ;                       6293
001042  012703  000024                          MOV     #24,R3
001046  060103                                  ADD     R1,R3
001050  132713  000037                          BITB    #37,(R3)
001054  001003                                  BNE     16$
001056  032713  177740                          BIT     #177740,(R3)            ;                       6294
001062  001404                                  BEQ     17$
001064  012760  000001  000120'          16$:    MOV     #1,LAST.PKT(R0)         ;                       6296
001072  000402                                  BR      18$                     ;                       6293
001074  005060  000120'                  17$:    CLR     LAST.PKT(R0)            ;                       6298
001100  016160  000012  000122'          18$:    MOV     12(R1),LAST.PKT+2(R0)   ;                       6300
001106  016160  000014  000124'                  MOV     14(R1),LAST.PKT+4(R0)   ;                       6301
001114  004737  000000V                          JSR     PC,SCAN.ERRLOG          ;                       6302
001120  005766  000004                          TST     4(SP)                   ; P.INDEX               6304
001124  002404                                  BLT     20$
001126  016616  000004                  19$:    MOV     4(SP),(SP)              ; P.INDEX,*             6306
001132  004737  000000G                          JSR     PC,PUT.PKT
001136  022626                          20$:    CMP     (SP)+,(SP)+             ;                       6190
001140  022626                          21$:    CMP     (SP)+,(SP)+             ;                       6175
001142  000207                                  RTS     PC

; Routine Size:  306 words,     Routine Base:  $CODE$ + 27114
; Maximum stack depth per invocation:  18 words
```

```
;    6309  1     GLOBAL routine SCAN_ERRLOG : novalue =
;    6310  1
;    6311  1     !+
;    6312  1     ! THIS ROUTINE SCANS THE ERROR-LOG SAVE AREA AND PRINTS ANY ERROR-LOGS RECEIVED FOR THE ASSOCIATED RESPONSE
;    6313  1     !-
;    6314  1
;    6315  2         begin
;    6316  2
;    6317  2         local
;    6318  2             TEMP_UNIT,
;    6319  2             SFT ERR_PRINTED : byte initial (byte (FALSE));
;    6320  2
;    6321  2         incr index from 0 to EP_CNT do                                          ! SCAN ERROR-LOG PACKET SAVE AREA
;    6322  3             begin
;    6323  3
;    6324  3             if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;    6325  3                (.ELOG_PKT [.index, EL_CRN_LO] eql .IPKT_ADDR [CRN_LO]) and
;    6326  3                (.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    6327  4                (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;    6328  3             then
;    6329  4                 begin                                                       ! ERROR-LOG PENDING THIS RESPONSE
;    6330  4
;    6331  4                 if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED     ! IF SOFT ERROR OCCURED
;    6332  4                 then
;    6333  4
;    6334  4                     if .ELOG_PKT [.index, EL_FORMAT] lequ 4
;    6335  4                     then
;    6336  5                         begin
;    6337  5                         SOFT_ERROR (.index);                                ! UPATE SOFT ERROR COUNT
;    6338  5                         TEMP_UNIT = .L$LUN;                                 ! SAVE UNIT NUMBER AS KNOWN TO DRS
;    6339  5
;    6340  5                         incr OFFSET from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do
;    6341  5
;    6342  5                             if (.ICST_ADDR [.OFFSET * OF_DATA, D_DISK_NUM] eql .ELOG_PKT [.index, EL_DK_NUM]) and
;    6343  6                                (.ICST_ADDR [.OFFSET * OF_DATA, D_PRES] eql PRESENT)
;    6344  5                             then
;    6345  6                                 begin
;    6346  6                                 L$LUN = .ICST_ADDR [.OFFSET * OF_DATA, D_UNIT];     ! CORECT UNIT NO. FOR ERROR MESSAGE
;    6347  6                                 exitloop;
;    6348  5                                 end;
;    6349  5
;    6350  5                         case .ELOG_PKT [.index, EL_FORMAT] from 0 to 4 of
;    6351  5                             set
;    6352  5
;    6353  5                             [0]:    if .APT_MODE                            ! CONTROLLER ERROR
;    6354  5                                     then
;    6355  5                                         ERR_SOFT_RTNE_APT (50, .index)
;    6356  5                                     else
;    6357  5                                         ERR_SOFT_RTNE (50);
;    6358  5
;    6359  5                             [1]:    if .APT_MODE                            ! HOST MEMORY ACCESS ERROR
;    6360  5                                     then
;    6361  5                                         ERR_SOFT_RTNE_APT (51, .index)
```

```
;    6362  5                                  else
;    6363  5                                      ERR_SOFT_RTNE (51);
;    6364  5
;    6365  5                              [2]:    if .APT_MODE                                    ! DISK TRANSFER ERROR
;    6366  5                                      then
;    6367  5                                          ERR_SOFT_RTNE_APT (52, .index)
;    6368  5                                      else
;    6369  5                                          ERR_SOFT_RTNE (52);
;    6370  5
;    6371  5                              [3]:    if .APT_MODE                                    ! SDI ERROR
;    6372  5                                      then
;    6373  5                                          ERR_SOFT_RTNE_APT (53, .index)
;    6374  5                                      else
;    6375  5                                          ERR_SOFT_RTNE (53);
;    6376  5
;    6377  5                              [4]:    if .APT_MODE                                    ! SMALL DISK ERROR
;    6378  5                                      then
;    6379  5                                          ERR_SOFT_RTNE_APT (54, .index)
;    6380  5                                      else
;    6381  5                                          ERR_SOFT_RTNE (54);
;    6382  5                              tes;
;    6383  5
;    6384  5                          L$LUN = .TEMP_UNIT;                                         ! RESTORE UNIT NUMBER
;    6385  5                          SFT_ERR_PRINTED = TRUE;                                     ! SOFT ERROR PRINTOUT OCCURED
;    6386  5                          end
;    6387  4                      else
;    6388  4                          PRINTF (D8M109, .ELOG_PKT [.index, EL_FORMAT]);             ! UNKNOWN ERROR-LOG FORMAT
;    6389  4
;    6390  5                      if not (.SFT_ERR_PRINTED)
;    6391  4                      then
;    6392  4                          PRINTB (CRLF);                                              ! EXTRA CARRIAGE-RETURN/LINE-FEED
;    6393  4
;    6394  4                      EMS_EL (.index);                                                ! PRINT ERROR-LOG CONTENTS
;    6395  4                      end
;    6396  3                  else
;    6397  3
;    6398  3                      if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;    6399  4                         ((.ELOG_PKT [.index, EL_CRN_HI] lssu .IPKT_ADDR [CRN_HI]) or
;    6400  5                          ((.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    6401  3                           (.ELOG_PKT [.index, EL_CRN_LO] lssu .IPKT_ADDR [CRN_LO]))) and
;    6402  4                         (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;    6403  3                      then
;    6404  4                          begin
;    6405  4                          PRINTB (CRLF);                                              ! CARRIAGE-RETURN/LINE-FEED
;    6406  4                          EMS_EL (.index);                                            ! PRINT ERROR-LOG CONTENTS
;    6407  3                          end;
;    6408  3
;    6409  2                  end;                                                                ! ERROR-LOG SAVE AREA SCAN
;    6410  2
;    6411  1          end;


                                        .SBTTL  SCAN.ERRLOG RDRX INTERRUPT SERVICE ROUTINES
```

```
000000  004137  000000G                SCAN.ERRLOG::
                                                JSR     R1,$SAVE5                       ;                                       6309
000004  005746                                  TST     -(SP)
000006  105005                                  CLRB    R5                              ; SFT.ERR.PRINTED                       6315
000010  005002                                  CLR     R2                              ; INDEX                                 6321
000012  010246                          1$:     MOV     R2,-(SP)                        ; INDEX,*                               6324
000014  012746  000102                          MOV     #102,-(SP)
000020  004737  000000G                        JSR     PC,BL$MUL
000024  010001                                  MOV     R0,R1
000026  022626                                  CMP     (SP)+,(SP)+
000030  012703  000000G                        MOV     #ELOG.PKT,R3
000034  060103                                  ADD     R1,R3
000036  005004                                  CLR     R4
000040  005000                                  CLR     R0
000042  151300                                  BISB    (R3),R0
000044  020037  000104'                        CMP     R0,ICTLR
000050  001016                                  BNE     2$
000052  005204                                  INC     R4
000054  013700  000000G                        MOV     IPKT.ADDR,R0                    ;                                       6325
000060  026160  000006G 000012                  CMP     ELOG.PKT+6(R1),12(R0)
000066  001007                                  BNE     2$
000070  026160  000010G 000014                  CMP     ELOG.PKT+10(R1),14(R0)          ;                                       6326
000076  001003                                  BNE     2$
000100  126327  000001  000001                  CMPB    1(R3),#1                        ;                                       6327
000106  001402                          2$:     BEQ     3$
000110  000137  031020'                        JMP     25$
000114  013746  000104'                  3$:     MOV     ICTLR,-(SP)                     ;                                       6331
000120  012746  000006                          MOV     #6,-(SP)
000124  004737  000000G                        JSR     PC,BL$MUL
000130  022626                                  CMP     (SP)+,(SP)+
000132  005760  000120'                        TST     LAST.PKT(R0)
000136  001161                                  BNE     23$
000140  126127  000016G 000004                  CMPB    ELOG.PKT+16(R1),#4              ;                                       6334
000146  101142                                  BHI     21$
000150  010246                                  MOV     R2,-(SP)                        ; INDEX,*                               6337
000152  004737  000000V                        JSR     PC,SOFT.ERROR
000156  013766  000000G 000002                  MOV     L$LUN,2(SP)                     ; *,TEMP.UNIT                           6338
000164  012703  000006                          MOV     #6,R3                           ; *,OFFSET                              6340
000170  010300                          4$:     MOV     R3,R0                           ; OFFSET,*                              6342
000172  063700  000076'                        ADD     ICST.ADDR,R0
000176  016146  000012G                        MOV     ELOG.PKT+12(R1),-(SP)
000202  111004                                  MOVB    (R0),R4
000204  042704  177760                        BIC     #177760,R4
000210  020426                                  CMP     R4,(SP)+
000212  001012                                  BNE     5$
000214  032710  040000                        BIT     #40000,(R0)                     ;                                       6343
000220  001407                                  BEQ     5$
000222  011004                                  MOV     (R0),R4                         ;                                       6346
000224  000304                                  SWAB    R4
000226  042704  177760                        BIC     #177760,R4
000232  010437  000000G                        MOV     R4,L$LUN
000236  000405                                  BR      6$                              ;                                       6345
000240  062703  000024                  5$:     ADD     #24,R3                          ; *,OFFSET                              6340
```

```
000244  020327  000102                        CMP     R3,#102                 ; OFFSET,*
000250  003747                                BLE     4$
000252  005000                        6$:     CLR     R0                      ;                       6353
000254  153700  001254'                       BISB    APT.MODE,R0
000260  116101  000016G                       MOVB    ELOG.PKT+16(R1),R1      ;                       6350
000264  042701  177400                        BIC     #177400,R1
000270  006301                                ASL     R1
000272  066107  000000'                       ADD     P.AAA(R1),PC            ; Case dispatch
000276  032700  000001                8$:     BIT     #1,R0                   ;                       6353
000302  001403                                BEQ     9$
000304  012716  000062                        MOV     #62,(SP)                ;                       6355
000310  000442                                BR      17$
000312  012716  000062                9$:     MOV     #62,(SP)                ;                       6357
000316  000446                                BR      19$
000320  032700  000001                10$:    BIT     #1,R0                   ;                       6359
000324  001403                                BEQ     11$
000326  012716  000063                        MOV     #63,(SP)                ;                       6361
000332  000431                                BR      17$
000334  012716  000063                11$:    MOV     #63,(SP)                ;                       6363
000340  000435                                BR      19$
000342  032700  000001                12$:    BIT     #1,R0                   ;                       6365
000346  001403                                BEQ     13$
000350  012716  000064                        MOV     #64,(SP)                ;                       6367
000354  000420                                BR      17$
000356  012716  000064                13$:    MOV     #64,(SP)                ;                       6369
000362  000424                                BR      19$
000364  032700  000001                14$:    BIT     #1,R0                   ;                       6371
000370  001403                                BEQ     15$
000372  012716  000065                        MOV     #65,(SP)                ;                       6373
000376  000407                                BR      17$
000400  012716  000065                15$:    MOV     #65,(SP)                ;                       6375
000404  000413                                BR      19$
000406  006000                        16$:    ROR     R0                      ;                       6377
000410  103007                                BCC     18$
000412  012716  000066                        MOV     #66,(SP)                ;                       6379
000416  010246                        17$:    MOV     R2,-(SP)                ; INDEX,*
000420  004737  000000V                       JSR     PC,ERR.SOFT.RTNE.APT
000424  005726                                TST     (SP)+
000426  000404                                BR      20$
000430  012716  000066                18$:    MOV     #66,(SP)                ;                       6377
000434  004737  000000V               19$:    JSR     PC,ERR.SOFT.RTNE        ;                       6381
000440  016637  000002  000000G       20$:    MOV     2(SP),L$LUN             ; TEMP.UNIT,*           6384
000446  112705  000001                        MOVB    #1,R5                   ; *,SFT.ERR.PRINTED     6385
000452  000412                                BR      22$                     ;                       6334
000454  005046                        21$:    CLR     -(SP)                   ;                       6388
000456  116116  000016G                       MOVB    ELOG.PKT+16(R1),(SP)
000462  012746  000000G                       MOV     #DBM109,-(SP)
000466  012746  000002                        MOV     #2,-(SP)
000472  010600                                MOV     SP,R0                   ; SP,*
000474  104417                                TRAP    17
000476  022626                                CMP     (SP)+,(SP)+
000500  005726                        22$:    TST     (SP)+                   ;                       6334
000502  032705  000001                23$:    BIT     #1,R5                   ; *,SFT.ERR.PRINTED     6390
```

```
000506  001007                          BNE     24$
000510  012746  000000G                MOV     #CRLF, (SP)                                     ;                                       6392
000514  012746  000001                  MOV     #1, (SP)
000520  010600                          MOV     SP,R0                   ; SP,*
000522  104414                          TRAP    14
000524  022626                          CMP     (SP)+,(SP)+
000526  010246                  24$:    MOV     R2, (SP)                ; INDEX,*               6394
000530  004737  000000G                JSR     PC,EMS.EL
000534  005726                          TST     (SP)+                   ;                       6329
000536  000433                          BR      27$                     ;                       6324
000540  006004                  25$:    ROR     R4                      ;                       6398
000542  103031                          BCC     27$
000544  013700  000000G                MOV     IPKT.ADDR,R0            ;                       6399
000550  026160  000010G 000014          CMP     ELOG.PKT+10(R1),14(R0)
000556  103405                          BLO     26$
000560  001022                          BNE     27$                     ;                       6400
000562  026160  000006G 000012          CMP     ELOG.PKT+6(R1),12(R0)   ;                       6401
000570  103016                          BHIS    27$
000572  126327  000001  000001  26$:    CMPB    1(R3),#1                ;                       6402
000600  001012                          BNE     27$
000602  012746  000000G                MOV     #CRLF,-(SP)             ;                       6405
000606  012746  000001                  MOV     #1, (SP)
000612  010600                          MOV     SP,R0                   ; SP,*
000614  104414                          TRAP    14
000616  010216                          MOV     R2,(SP)                 ; INDEX,*               6406
000620  004737  000000G                JSR     PC,EMS.EL
000624  022626                          CMP     (SP)+,(SP)+             ;                       6404
000626  005202                  27$:    INC     R2                      ; INDEX                 6321
000630  020227  000014                  CMP     R2,#14                  ; INDEX,*
000634  003002                          BGT     28$
000636  000137  030272                  JMP     1$
000642  005726                  28$:    TST     (SP)+                   ;                       6309
000644  000207                          RTS     PC

; Routine Size:  211 words,    Routine Base:  $CODE$ + 30260
; Maximum stack depth per invocation:  12 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:
000000  000000                  7$:     .WORD   0                       ; CASE Table for SCAN.ERRLOG+0272       6350
000002  000022                          .WORD   22                      ; [8$]
000004  000044                          .WORD   44                      ; [10$]
000006  000066                          .WORD   66                      ; [12$]
000010  000110                          .WORD   110                     ; [14$]
                                                                        ; [16$]
```

```
;  6412  1     GLOBAL routine DATAGM : novalue =
;  6413  1
;  6414  1     !+
;  6415  1     !       THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
;  6416  1     !       THE RDRX
;  6417  1     !
;  6418  1     !       IMPLICIT INPUTS:
;  6419  1     !               IPKT_ADDR   ADDRESS OF MSCP PACKET CONTAINING ERROR LOG
;  6420  1     !                           MESSAGE
;  6421  1     !               ICST_ADDR   ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;  6422  1     !
;  6423  1
;  6424  2         begin
;  6425  2
;  6426  2         local
;  6427  2             index : signed word initial ( 1),
;  6428  2             SAVE_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;  6429  2             SRC_ADDR,
;  6430  2             DST_ADDR,
;  6431  2             TEMP_UNIT,
;  6432  2             SFT_ERR_PRINTED : byte initial (byte (FALSE)),
;  6433  2             PACKET_LEN : word;
;  6434  2
;  6435  2     !
;  6436  2     ! FIND AN EMPTY SLOT IN THE ERROR LOG PACKET SAVE AREA
;  6437  2     !
;  6438  2
;  6439  2         incr COUNT from 0 to EP_CNT - 1 do
;  6440  2
;  6441  2             if .ELOG_PKT [.COUNT, EL_CONTENTS] eql EMPTY                    ! IF EMPTY SLOT FOUND
;  6442  2             then
;  6443  3                 begin
;  6444  3                 index = .COUNT;                                             ! SAVE INDEX INTO THE SAVE AREA
;  6445  3                 exitloop;
;  6446  2                 end;
;  6447  2
;  6448  2             if .index lss 0
;  6449  2             then
;  6450  2                 index = EP_CNT;                                             ! IF NO SLOT FOUND, USE LAST SPARE SLOT
;  6451  2
;  6452  2     !
;  6453  2     ! SAVE THE PACKET CONTENTS
;  6454  2     !
;  6455  2
;  6456  2             SAVE_ADDR = ELOG_PKT + (.index * EP_LEN * 2);               ! ADDRESS OF THE SAVE AREA
;  6457  2             SAVE_ADDR [EL_CONTENTS ] = FULL;                            ! MARK IT FULL
;  6458  2             SAVE_ADDR [EL_CNTR] = .ICTLR;                               ! OWNERSHIP
;  6459  2             SRC_ADDR = .IPKT_ADDR + 6;                                  ! SETUP COPY ADDRESSES
;  6460  2             DST_ADDR = .SAVE_ADDR + 2;                                  !
;  6461  2             PACKET_LEN = ((.IPKT_ADDR [MSGLEN] + 1) / 2) * 2;           ! LENGTH OF ERROR-LOG INCLUDING ENVELOPE
;  6462  2
;  6463  2             if .PACKET_LEN gtru EP_LEN - 1
;  6464  2             then
```

```
;  6465  2              PACKET_LEN = EP_LEN   1;                                    ! ADJUST LENGTH, IF TOO LONG
;  6466  2
;  6467  2          incr COUNT from 1 to .PACKET_LEN do
;  6468  3              begin
;  6469  3              .DST_ADDR = ..SRC_ADDR;                                     ! COPY A WORD
;  6470  3              SRC_ADDR = .SRC_ADDR + 2;                                   ! UPDATE ADDRESS POINTERS
;  6471  3              DST_ADDR = .DST_ADDR + 2;                                   !
;  6472  2              end;
;  6473  2
;  6474  2      !
;  6475  2      ! CHECK IF THE CORRESPONDING RESPONSE HAS ALREADY BEEN RECEIVED
;  6476  2      !
;  6477  2
;  6478  2          if (.SAVE_ADDR [EL_CRN_LO] eql .LAST_PKT [.ICTLR, LAST_CRN_LO]) and
;  6479  2             (.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI])
;  6480  2          then
;  6481  3              begin                                                       ! LOG REFERS TO THE LAST RESPONSE RECEIVED
;  6482  3
;  6483  3              if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED     ! IF SOFT ERROR HAD OCCURED
;  6484  3              then
;  6485  3
;  6486  3                  if .SAVE_ADDR [EL_FORMAT] lequ 4
;  6487  3                  then
;  6488  4                      begin
;  6489  4                      SOFT_ERROR (.index);                                ! UPDATE SOFT ERROR COUNT
;  6490  4                      TEMP_UNIT = .L$LUN;                                 ! SAVE UNIT NUMBER AS KNOWN TO DRS
;  6491  4
;  6492  4                      incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  6493  4
;  6494  4                          if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .SAVE_ADDR [EL_DK_NUM]) and
;  6495  5                             (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;  6496  4                          then
;  6497  5                              begin
;  6498  5                              L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT]; ! CORRECT UNIT NUMBER FOR ERROR MESSAGE
;  6499  5                              exitloop;
;  6500  4                              end;
;  6501  4
;  6502  4                      case .SAVE_ADDR [EL_FORMAT] from 0 to 4 of
;  6503  4                          set
;  6504  4
;  6505  4
;  6506  4                          [0] :       if .APT_MODE                        ! CONTROLLER ERROR
;  6507  4                                      then
;  6508  4                                          ERR_SOFT_RTNE_APT (50, .index)
;  6509  4                                      else
;  6510  4                                          ERR_SOFT_RTNE (50);
;  6511  4
;  6512  4                          [1] :       if .APT_MODE                        ! HOST MEMORY ACCESS ERROR
;  6513  4                                      then
;  6514  4                                          ERR_SOFT_RTNE_APT (51, .index)
;  6515  4                                      else
;  6516  4                                          ERR_SOFT RTNE (51);
;  6517  4
```

```
;    6518  4                        [2] :       if .APT_MODE                                    ! DISK TRANSFER ERROR
;    6519  4                                    then
;    6520  4                                        ERR_SOFT_RTNE_APT (52, .index)
;    6521  4                                    else
;    6522  4                                        ERR_SOFT_RTNE (52);
;    6523  4
;    6524  4                        [3] :       if .APT_MODE                                    ! SDI ERROR
;    6525  4                                    then
;    6526  4                                        ERR_SOFT_RTNE_APT (53, .index)
;    6527  4                                    else
;    6528  4                                        ERR_SOFT_RTNE (53);
;    6529  4
;    6530  4                        [4] :       if .APT_MODE                                    ! SMALL DISK ERROR
;    6531  4                                    then
;    6532  4                                        ERR_SOFT_RTNE_APT (54, .index)
;    6533  4                                    else
;    6534  4                                        ERR_SOFT_RTNE (54);
;    6535  4                        tes;
;    6536  4
;    6537  4
;    6538  4                    L$LUN = .TEMP_UNIT;                                             ! RESTORE UNIT NUMBER
;    6539  4                    SFT_ERR_PRINTED = TRUE;                                         ! SOFT ERROR PRINTOUT OCCURED
;    6540  4                    end
;    6541  4
;    6542  3                else
;    6543  3                    PRINTF (D8M109, .SAVE_ADDR [EL_FORMAT]);                        ! ERROR LOG FORMAT UNKNOWN
;    6544  3
;    6545  4            if not (.SFT_ERR_PRINTED)
;    6546  4
;    6547  3            then
;    6548  3                PRINTB (CRLF);                                                      ! EXTRA CARRIEGE-RETURN/LINE-FEED
;    6549  3
;    6550  3            EMS_EL (.index);                                                        ! PRINT PACKET CONTENTS
;    6551  3            end                                                                     ! CORRESPONDING RESPONSE RECEIVED
;    6552  3
;    6553  2        else
;    6554  2
;    6555  2            if (.SAVE_ADDR [EL_CRN_HI] lssu .LAST_PKT [.ICTLR, LAST_CRN_HI]) or
;    6556  3               ((.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI]) and
;    6557  3                (.SAVE_ADDR [EL_CRN_LO] lssu .LAST_PKT [.ICTLR, LAST_CRN_LO]))
;    6558  3
;    6559  2            then
;    6560  3                begin                                                               ! LOG REFERS TO SOME PREVIOUS RESPONSE
;    6561  3                PRINTB (CRLF);                                                      ! CARRIAGE-RETURN/LINE-FEED
;    6562  3                EMS_EL (.index);                                                    ! PRINT PACKET CONTENTS
;    6563  2                end;
;    6564  2
;    6565  1        end;


                                        .SBTTL  DATAGM RDRX INTERRUPT SERVICE ROUTINES
03:126                                  .PSECT  $CODE$,  RO
```

```
000000  004137  000000G          DATAGM::JSR     R1,$SAVE5                                                       ;
000004  012704  177777                   MOV     #-1,R4                                                          ; *,INDEX                                       6412
000010  105046                   CLRB    -(SP)                                                           ; SFT.ERR.PRINTED                               6424
000012  005001                   CLR     R1                                                              ; COUNT
000014  010146                1$:MOV     R1,-(SP)                                                        ; COUNT,*                                       6439
000016  012746  000102           MOV     #102,-(SP)                                                                                                      6441
000022  004737  000000G          JSR     PC,BL$MUL
000026  022626                   CMP     (SP)+,(SP)+
000030  105760  000001G          TSTB    ELOG.PKT+1(R0)
000034  001002                   BNE     2$
000036  010104                   MOV     R1,R4                                                           ; COUNT,INDEX                                   6444
000040  000405                   BR      3$                                                              ;                                               6443
000042  005201                2$:INC     R1                                                              ; COUNT                                         6439
000044  020127  000013           CMP     R1,#13                                                          ; COUNT,*
000050  003761                   BLE     1$
000052  005704                   TST     R4                                                              ; INDEX                                         6448
000054  002002                3$:BGE     4$
000056  012704  000014           MOV     #14,R4                                                          ; *,INDEX                                       6450
000062  010446                4$:MOV     R4,-(SP)                                                        ; INDEX,*                                       6456
000064  012746  000102           MOV     #102,-(SP)
000070  004737  000000G          JSR     PC,BL$MUL
000074  062700  000000G          ADD     #ELOG.PKT,R0
000100  010001                   MOV     R0,R1                                                           ; *,SAVE.ADDR
000102  111761  000001           MOVB    (PC),1(R1)                                                      ; *,*(SAVE.ADDR)                                6457
000106  113711  000104'          MOVB    ICTLR,(R1)                                                      ; *,SAVE.ADDR                                   6458
000112  013700  000000G          MOV     IPKT.ADDR,R0                                                    ;                                               6459
000116  012705  000006           MOV     #6,R5                                                           ; *,SRC.ADDR
000122  060005                   ADD     R0,R5                                                           ; *,SRC.ADDR
000124  012703  000002           MOV     #2,R3                                                           ; *,DST.ADDR                                    6460
000130  060103                   ADD     R1,R3                                                           ; SAVE.ADDR,DST.ADDR
000132  016016  000006           MOV     6(R0),(SP)                                                      ;                                               6461
000136  005216                   INC     (SP)
000140  012746  000002           MOV     #2,-(SP)
000144  004737  000000G          JSR     PC,BL$DIV
000150  062700  000002           ADD     #2,R0
000154  020027  000040           CMP     R0,#40                                                          ; PACKET.LEN,*                                  6463
000160  101402                   BLOS    5$
000162  012700  000040           MOV     #40,R0                                                          ; *,PACKET.LEN                                  6465
000166  005002                5$:CLR     R2                                                              ; COUNT                                         6467
000170  000401                   BR      7$
000172  012523                6$:MOV     (R5)+,(R3)+                                                     ; SRC.ADDR,DST.ADDR                             6469
000174  005202                7$:INC     R2                                                              ; COUNT                                         6467
000176  020200                   CMP     R2,R0                                                           ; COUNT,PACKET.LEN
000200  003774                   BLE     6$
000202  013716  000104'          MOV     ICTLR,(SP)                                                      ;                                               6478
000206  012746  000006           MOV     #6,-(SP)
000212  004737  000000G          JSR     PC,BL$MUL
000216  005726                   TST     (SP)+
000220  026160  000006  000122'   CMP     6(R1),LAST.PKT+2(R0)                                            ; *(SAVE.ADDR),*
000226  001402                   BEQ     8$
000230  000137  031770'          JMP     30$
000234  026160  000010  000124'  8$:CMP     10(R1),LAST.PKT+4(R0)                                           ; *(SAVE.ADDR),*                                6479
000242  001402                   BEQ     9$
```

```
000244  000137  031776'                    JMP     31$
000250  005760  000120'          9$:       TST     LAST.PKT(R0)                ;                                   6483
000254  001153                             BNE     28$
000256  005003                             CLR     R3                          ;                                   6486
000260  156103  000016                     BISB    16(R1),R3                   ; *(SAVE.ADDR),*
000264  020327  000004                     CMP     R3,#4
000270  101135                             BHI     27$
000272  010416                             MOV     R4,(SP)                     ; INDEX,*                           6489
000274  004737  000000V                    JSR     PC,SOFT.ERROR
000300  013705  000000G                    MOV     L$LUN,R5                    ; *,TEMP.UNIT                       6490
000304  012702  000006                     MOV     #6,R2                       ; *,OFFSET                          6492
000310  010200                   10$:      MOV     R2,R0                       ; OFFSET,*                          6494
000312  063700  000076'                    ADD     ICST.ADDR,R0
000316  016146  000012                     MOV     12(R1),-(SP)                ; *(SAVE.ADDR),*
000322  111046                             MOVB    (R0),-(SP)
000324  042716  177760                     BIC     #177760,(SP)
000330  022626                             CMP     (SP)+,(SP)+
000332  001012                             BNE     11$
000334  032710  040000                     BIT     #40000,(R0)                 ;                                   6495
000340  001407                             BEQ     11$
000342  011046                             MOV     (R0),-(SP)                  ;                                   6498
000344  000316                             SWAB    (SP)
000346  042716  177760                     BIC     #177760,(SP)
000352  012637  000000G                    MOV     (SP)+,L$LUN
000356  000405                             BR      12$
000360  062702  000024           11$:      ADD     #24,R2                      ;                                   6497
000364  020227  000102                     CMP     R2,#102                     ; *,OFFSET                          6492
000370  003747                             BLE     10$                         ; OFFSET,*
000372  005000                   12$:      CLR     R0
000374  153700  001254'                    BISB    APT.MODE,R0                 ;                                   6506
000400  006303                             ASL     R3
000402  066307  000012'                    ADD     P.AAB(R3),PC                ;                                   6502
000406  032700  000001           14$:      BIT     #1,R0                       ; Case dispatch
000412  001403                             BEQ     15$                         ;                                   6506
000414  012716  000062                     MOV     #62,(SP)
000420  000442                             BR      23$                         ;                                   6508
000422  012716  000062           15$:      MOV     #62,(SP)
000426  000446                             BR      25$                         ;                                   6510
000430  032700  000001           16$:      BIT     #1,R0
000434  001403                             BEQ     17$                         ;                                   6512
000436  012716  000063                     MOV     #63,(SP)
000442  000431                             BR      23$                         ;                                   6514
000444  012716  000063           17$:      MOV     #63,(SP)
000450  000435                             BR      25$                         ;                                   6516
000452  032700  000001           18$:      BIT     #1,R0
000456  001403                             BEQ     19$                         ;                                   6518
000460  012716  000064                     MOV     #64,(SP)
000464  000420                             BR      23$                         ;                                   6520
000466  012716  000064           19$:      MOV     #64,(SP)
000472  000424                             BR      25$                         ;                                   6522
000474  032700  000001           20$:      BIT     #1,R0
000500  001403                             BEQ     21$                         ;                                   6524
000502  012716  000065                     MOV     #65,(SP)                    ;                                   6526
```

```
ZRQAM3          RD/RX EXERCISER                              4-Apr-1985 13:23:31    VAX-11 Bliss-16 V4.1-582                SEQ 0468
V02.2           RDRX INTERRUPT SERVICE ROUTINES              2-Apr-1985 15:52:52    DISK$USER2:[POWERS.ZRQ]ZRQAGO.BL2;19     Page 213
                                                                                                                             (57)

000506  000407                                      BR      23$
000510  012716  000065                      21$:    MOV     #65,(SP)                ;                                       6528
000514  000413                                      BR      25$
000516  006000                              22$:    ROR     R0                      ;                                       6530
000520  103007                                      BCC     24$
000522  012716  000066                              MOV     #66,(SP)                ;
000526  010446                              23$:    MOV     R4,-(SP)                ; INDEX,*                               6532
000530  004737  000000V                             JSR     PC,ERR.SOFT.RTNE.APT
000534  005726                                      TST     (SP)+
000536  000404                                      BR      26$
000540  012716  000066                      24$:    MOV     #66,(SP)                ;                                       6530
000544  004737  000000V                     25$:    JSR     PC,ERR.SOFT.RTNE                                                6534
000550  010537  000000G                     26$:    MOV     R5,L$LUN                ; TEMP.UNIT,*
000554  112766  000001  000006                      MOVB    #1,6(SP)                ; *,SFT.ERR.PRINTED                     6538
000562  000410                                      BR      28$                     ;                                       6539
000564  010316                              27$:    MOV     R3,(SP)                 ;                                       6486
000566  012746  000000G                             MOV     #DBM109,-(SP)                                                   6543
000572  012746  000002                              MOV     #2,-(SP)
000576  010600                                      MOV     SP,R0                   ; SP,*
000600  104417                                      TRAP    17
000602  022626                                      CMP     (SP)+,(SP)+
000604  032766  000001  000006              28$:    BIT     #1,6(SP)                ; *,SFT.ERR.PRINTED                     6545
000612  001007                                      BNE     29$
000614  012716  000000G                             MOV     #CRLF,(SP)              ;                                       6548
000620  012746  000001                              MOV     #1,-(SP)
000624  010600                                      MOV     SP,R0                   ; SP,*
000626  104414                                      TRAP    14
000630  005726                                      TST     (SP)+
000632  010416                              29$:    MOV     R4,(SP)                 ; INDEX,*                               6550
000634  004737  000000G                             JSR     PC,EMS.EL
000640  000426                                      BR      33$                     ;                                       6478
000642  026160  000010  000124'          30$:    CMP     10(R1),LAST.PKT+4(R0)   ; *(SAVE.ADDR),*                        6555
000650  103410                              31$:    BLO     32$
000652  026160  000010  000124'                     CMP     10(R1),LAST.PKT+4(R0)   ; *(SAVE.ADDR),*                        6556
000660  001016                                      BNE     33$
000662  026160  000006  000122'                     CMP     6(R1),LAST.PKT+2(R0)    ; *(SAVE.ADDR),*                        6557
000670  103012                                      BHIS    33$
000672  012716  000000G                     32$:    MOV     #CRLF,(SP)              ;                                       6561
000676  012746  000001                              MOV     #1,-(SP)
000702  010600                                      MOV     SP,R0                   ; SP,*
000704  104414                                      TRAP    14
000706  010416                                      MOV     R4,(SP)                 ; INDEX,*                               6562
000710  004737  000000G                             JSR     PC,EMS.EL
000714  005726                                      TST     (SP)+                   ;                                       6560
000716  062706  000010                      33$:    ADD     #10,SP                  ;                                       6412
000722  000207                                      RTS     PC

; Routine Size:  234 words,     Routine Base:  $CODE$ + 31126
; Maximum stack depth per invocation:  14 words


000012                                              .PSECT  $PLIT$,  RO ,  D
```

```
                                P.AAB:
000012  000000                  13$:    .WORD   0               ; CASE Table for DATAGM.0402            6502
000014  000022                          .WORD   22              ; [14$]
000016  000044                          .WORD   44              ; [16$]
000020  000066                          .WORD   66              ; [20$]
000022  000110                          .WORD   110             ; [22$]


;    6566  1
;    6567  1
;    6568  1
```

```
;    6569  1     GLOBAL routine SOFT_ERROR (index) : novalue =
;    6570  1
;    6571  1     !+
;    6572  1     !       THIS ROUTINE UPDATES THE SOFT ERROR COUNT IN THE TALLY TABLE FOR EACH
;    6573  1     !       ERROR LOG MESSAGE RECEIVED
;    6574  1     !
;    6575  1     !       IMPLICIT INPUTS:
;    6576  1     !               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;    6577  1     !-
;    6578  1
;    6579  2         begin
;    6580  2
;    6581  2         local
;    6582  2             FOUND: byte initial (byte (FALSE)),
;    6583  2             SOFT_OCCURED : byte initial (byte (FALSE)),
;    6584  2             UNIT: word,
;    6585  2             ERROR_CODE : byte,
;    6586  2             ERROR_SUB : word,
;    6587  2             RETRIES : word,
;    6588  2             TALLY_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    6589  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;    6590  2
;    6591  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                          ! ADDR OF ERROR PKT
;    6592  2         ERROR_CODE = .ELOG_ADDR [EL_CODE];                                     ! ERROR CODE
;    6593  2         ERROR_SUB = .ELOG_ADDR [EL_SUBCODE];                                   ! ERROR SUBCODE
;    6594  2
;    6595  2         if (BIT_TST (SWP_FLAGS, SWF_TRY)) and
;    6596  3            (.ELOG_ADDR [EL_FORMAT] eql 2)
;    6597  2         then
;    6598  2             RETRIES = .ELOG_ADDR [EL_RETRY]                                    ! COUNT EACH RETRY
;    6599  2         else
;    6600  2             RETRIES = 1;                                                       ! IGNORE RETRIES
;    6601  2
;    6602  2         if .RETRIES eql 0                                                      ! IN CASE OF A BUG
;    6603  2         then
;    6604  2             RETRIES = 1;
;    6605  2
;    6606  2         incr OFFSET from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do
;    6607  2
;    6608  2             if (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and        ! DISK TO UNIT NO.
;    6609  3                (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_ADDR [EL_DK_NUM])
;    6610  2             then
;    6611  3                 begin
;    6612  3                 FOUND = TRUE;
;    6613  3                 UNIT = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];             ! DISK'S UNIT NO.
;    6614  3                 exitloop;
;    6615  2                 end;
;    6616  2
;    6617  2     !       if (.ELOG_ADDR [EL_SUCCESS]) or
;    6618  2     !          (.ELOG_ADDR [EL_CONTINUE])
;    6619  2     !       then
;    6620  2                 SOFT_OCCURED = TRUE;                                       ! SOFT ERROR FLAG
;    6621  2
```

```
;    6622  2          if .FOUND                                                                          ! IF UNIT FOUND
;    6623  2          then
;    6624  3              begin
;    6625  3              TALLY_ADDR = TALLY + (.UNIT * TALLY_LEN * 2);                                  ! ADDR OF TALLY TBL
;    6626  3
;    6627  3              if .SOFT_OCCURED                                                               ! FOR SOFT ERRORRS
;    6628  3              then
;    6629  3                  selectoneu .ERROR_CODE of
;    6630  3                      set
;    6631  3
;    6632  3                      [ST_MFE]:       TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES;! SOFT-MEDIA FORMAT
;    6633  3
;    6634  3                      [ST_DAT]:       if .ERROR_SUB eql 2                                    ! SOFT-DATA
;    6635  3                                      then
;    6636  3                                          TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
;    6637  3                                      else
;    6638  3                                          TALLY_ADDR [ERR_SFT_DAT] = .TALLY_ADDR [ERR_SFT_DAT] + .RETRIES;
;    6639  3
;    6640  3                      [ST_HST]:       TALLY_ADDR [ERR_SFT_HST] = .TALLY_ADDR [ERR_SFT_HST] + .RETRIES;! SOFT-HOST ACCESS
;    6641  3
;    6642  3                      [ST_CNT].       C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + .RETRIES;
;    6643  3                                                                                             ! SOFT-CONTROLLER
;    6644  3
;    6645  3                      [ST_DRV]:       if .ERROR_SUB eql 3                                    ! SOFT_DRIVE
;    6646  3                                      then
;    6647  3                                          TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
;    6648  3                                      else
;    6649  3                                          TALLY_ADDR [ERR_SFT_DRV] = .TALLY_ADDR [ERR_SFT_DRV] + .RETRIES;
;    6650  3                      tes
;    6651  3              else
;    6652  3
;    6653  3                  if (.ELOG_ADDR [EL_CRN_LO] eql 0) and
;    6654  4                     (.ELOG_ADDR [EL_CRN_HI] eql 0)
;    6655  3                  then
;    6656  3                      selectoneu .ERROR_CODE of
;    6657  3                          set
;    6658  3
;    6659  3                          [ST_MFE]:   TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1;      ! HARD-MEDIA FORMAT
;    6660  3
;    6661  3                          [ST_DAT]:   if .ERROR_SUB eql 2                                    ! HARD-DATA
;    6662  3                                      then
;    6663  3                                          TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
;    6664  3                                      else
;    6665  3                                          TALLY_ADDR [ERR_HRD_DAT] = .TALLY_ADDR [ERR_HRD_DAT] + 1;
;    6666  3
;    6667  3                          [ST_HST]:   TALLY_ADDR [ERR_HRD_HST] = .TALLY_ADDR [ERR_HRD_HST] + 1;      ! HARD-HOST ACCESS
;    6668  3
;    6669  3                          [ST_CNT]:   C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;    6670  3                                                                                             ! HARD-CONTROLLER
;    6671  3
;    6672  3                          [ST_DRV]:   if .ERROR_SUB eql 3                                    ! HARD-DRIVE
;    6673  3                                      then
;    6674  3                                          TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
```

```
;    6675  3                                    else
;    6676  3                                        TALLY_ADDR [ERR_HRD_DRV] = .TALLY_ADDR [ERR_HRD_DRV] + 1;
;    6677  3                            tes;
;    6678  3
;    6679  3                    end
;    6680  2                else                                                    ! UNIT NOT FOUND
;    6681  2
;    6682  2                    if .SOFT_OCCURED
;    6683  2                    then
;    6684  2                        C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + 1
;    6685  2                    else
;    6686  2                        C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;    6687  2
;    6688  1                end;                                                    ! RTNE SOFT_ERROR


                                        .SBTTL  SOFT.ERROR RDRX INTERRUPT SERVICE ROUTINES
032052                                  .PSECT  $CODE$,  RO

000000  004137  000000G          SOFT.ERROR::
                                        JSR     R1,$SAVE5               ;                                       6569
000004  005746                          TST     -(SP)
000006  105046                          CLRB    -(SP)
000010  105046                          CLRB    -(SP)                   ; FOUND                                 6579
000012  016646  000024                  MOV     24(SP),-(SP)            ; SOFT.OCCURED
000016  012746  000102                  MOV     #102,-(SP)              ; INDEX,*                               6591
000022  004737  000000G                 JSR     PC,BL$MUL
000026  062700  000000G                 ADD     #ELOG.PKT,RO
000032  010001                          MOV     RO,R1
000034  116100  000020                  MOVB    20(R1),RO               ; *,ELOG.ADDR
000040  042700  177740                  BIC     #177740,RO              ; *(ELOG.ADDR),*                        6592
000044  105003                          CLRB    R3
000046  050003                          BIS     RO,R3                   ; ERROR.CODE
000050  016105  000020                  MOV     20(R1),R5               ; *,ERROR.CODE
000054  006205                          ASR     R5                      ; *(ELOG.ADDR),ERROR.SUB                6593
000056  006205                          ASR     R5                      ; ERROR.SUB
000060  006205                          ASR     R5                      ; ERROR.SUB
000062  006205                          ASR     R5                      ; ERROR.SUB
000064  006205                          ASR     R5                      ; ERROR.SUB
000066  042705  174000                  BIC     #174000,R5              ; ERROR.SUB
000072  013700  000000G                 MOV     SWP.FLAGS,RO            ; *,ERROR.SUB
000076  042700  077777                  BIC     #77777,RO               ;                                       6595
000102  020027  100000                  CMP     RO,#-100000
000106  001010                          BNE     1$
000110  126127  000016  000002          CMPB    16(R1),#2               ; *(ELOG.ADDR),*                        6596
000116  001004                          BNE     1$
000120  005004                          CLR     R4                      ; RETRIES                               6598
000122  156104  000051                  BISB    51(R1),R4               ; *(ELOG.ADDR),RETRIES
000126  000402                          BR      2$                      ;                                       6595
000130  012704  000001          1$:     MOV     #1,R4                   ; *,RETRIES                             6600
000134  005704                  2$:     TST     R4                      ; RETRIES                               6602
000136  001002                          BNE     3$
000140  012704  000001                  MOV     #1,R4                   ; *,RETRIES                             6604
```

```
000144  012702  000006                3$:     MOV     #6,R2                   ; #,OFFSET                      6606
000150  010200                        4$:     MOV     R2,R0                   ; OFFSET,#                      6608
000152  063700  000076'                        ADD     ICST.ADDR,R0
000156  032710  040000                        BIT     #40000,(R0)
000162  001421                              BEQ     5$
000164  016146  000012                        MOV     12(R1),-(SP)            ; #(ELOG.ADDR),#                6609
000170  111046                              MOVB    (R0),-(SP)
000172  042716  177760                      BIC     #177760,(SP)
000176  022626                              CMP     (SP)+,(SP)+
000200  001012                              BNE     5$
000202  112766  000001  000006              MOVB    #1,6(SP)                ; #,FOUND                       6612
000210  011046                              MOV     (R0),-(SP)              ;                               6613
000212  000316                              SWAB    (SP)
000214  042716  177760                      BIC     #177760,(SP)
000220  012666  000010                      MOV     (SP)+,10(SP)            ; #,UNIT
000224  000405                              BR      6$                      ;                               6611
000226  062702  000024                5$:     ADD     #24,R2                  ; #,OFFSET                      6606
000232  020227  000102                        CMP     R2,#102                 ; OFFSET,#
000236  003744                              BLE     4$
000240  112766  000001  000004        6$:     MOVB    #1,4(SP)                ; #,SOFT.OCCURED                6620
000246  032766  000001  000006              BIT     #1,6(SP)                ; #,FOUND                       6622
000254  001002                              BNE     7$
000256  000137  032740'                      JMP     22$
000262  016616  000010                7$:     MOV     10(SP),(SP)             ; UNIT,#                        6625
000266  012746  000066                        MOV     #66,-(SP)
000272  004737  000000G                      JSR     PC,BL$MUL
000276  062700  000000G                      ADD     #TALLY,R0
000302  032766  000001  000006              BIT     #1,6(SP)                ; #,SOFT.OCCURED                6627
000310  001503                              BEQ     14$
000312  120327  000005                      CMPB    R3,#5                   ; ERROR.CODE,#                  6632
000316  001462                              BEQ     12$
000320  120327  000010                      CMPB    R3,#10                  ; ERROR.CODE,#                  6634
000324  001022                              BNE     9$
000326  012702  000052                      MOV     #52,R2                  ;                               6636
000332  060002                              ADD     R0,R2                   ; TALLY.ADDR,#
000334  020527  000002                      CMP     R5,#2                   ; ERROR.SUB,#                   6634
000340  001005                              BNE     8$
000342  005001                              CLR     R1                      ;                               6636
000344  151201                              BISB    (R2),R1
000346  060401                              ADD     R4,R1                   ; RETRIES,#
000350  110112                              MOVB    R1,(R2)
000352  000543                              BR      21$                     ;                               6634
000354  005001                        8$:     CLR     R1                      ;                               6638
000356  156201  000001                      BISB    1(R2),R1
000362  060401                              ADD     R4,R1                   ; RETRIES,#
000364  110162  000001                      MOVB    R1,1(R2)
000370  000534                              BR      21$                     ;                               6629
000372  120327  000011                9$:     CMPB    R3,#11                  ; ERROR.CODE,#                  6640
000376  001007                              BNE     10$
000400  005001                              CLR     R1
000402  156001  000055                      BISB    55(R0),R1               ; #(TALLY.ADDR),#
000406  060401                              ADD     R4,R1                   ; RETRIES,#
000410  110160  000055                      MOVB    R1,55(R0)               ; #,#(TALLY.ADDR)
```

```
000414  000522                          BR      21$                     ;                               6629
000416  120327  000012        10$:      CMPB    R3,#12                  ; ERROR.CODE,*                  6642
000422  001012                          BNE     11$
000424  013702  000104'                  MOV     ICTLR,R2
000430  006302                          ASL     P2
000432  005001                          CLR     R1
000434  156201  000001G                 BISB    C.ERR.TBL+1(R2),R1
000440  060401                          ADD     R4,R1                   ; RETRIES,*
000442  110162  000001G                 MOVB    R1,C.ERR.TBL+1(R2)
000446  000505                          BR      21$                     ;                               6629
000450  120327  000013        11$:      CMPB    R3,#13                  ; ERROR.CODE,*                  6645
000454  001102                          BNE     21$
000456  020527  000003                  CMP     R5,#3                   ; ERROR.SUB,*
000462  001007                          BNE     13$
000464  005001                  12$:    CLR     R1                      ;                               6647
000466  156001  000052                  BISB    52(R0),R1               ; *(TALLY.ADDR),*
000472  060401                          ADD     R4,R1                   ; RETRIES,*
000474  110160  000052                  MOVB    R1,52(R0)               ; *,*(TALLY.ADDR)
000500  000470                          BR      21$                     ;                               6645
000502  005001                  13$:    CLR     R1                      ;                               6649
000504  156001  000054                  BISB    54(R0),R1               ; *(TALLY.ADDR),*
000510  060401                          ADD     R4,R1                   ; RETRIES,*
000512  110160  000054                  MOVB    R1,54(R0)               ; *,*(TALLY.ADDR)
000516  000461                          BR      21$                     ;                               6629
000520  005761  000006        14$:      TST     6(R1)                   ; *(ELOG.ADDR)                  6653
000524  001056                          BNE     21$
000526  005761  000010                  TST     10(R1)                  ; *(ELOG.ADDR)                  6654
000532  001053                          BNE     21$
000534  120327  000005                  CMPB    R3,#5                   ; ERROR.CODE,*                  6659
000540  001443                          BEQ     19$
000542  120327  000010                  CMPB    R3,#10                  ; ERROR.CODE,*                  6661
000546  001013                          BNE     16$
000550  012704  000046                  MOV     #46,R4                  ;                               6663
000554  060004                          ADD     R0,R4                   ; TALLY.ADDR,*
000556  020527  000002                  CMP     R5,#2                   ; ERROR.SUB,*                   6663
000562  001002                          BNE     15$
000564  105214                          INCB    (R4)                    ;                               6661
000566  000435                          BR      21$                     ;                               6665
000570  105264  000001        15$:      INCB    1(R4)                   ;                               6656
000574  000432                          BR      21$                     ;                               6667
000576  120327  000011        16$:      CMPB    R3,#11                  ; ERROR.CODE,*
000602  001003                          BNE     17$
000604  105260  000051                  INCB    51(R0)                  ; *(TALLY.ADDR)
000610  000424                          BR      21$                     ;                               6656
000612  120327  000012        17$:      CMPB    R3,#12                  ; ERROR.CODE,*                  6669
000616  001006                          BNE     18$
000620  013702  000104'                  MOV     ICTLR,R2
000624  006302                          ASL     R2
000626  105262  000000G                 INCB    C.ERR.TBL(R2)
000632  000413                          BR      21$                     ;                               6656
000634  120327  000013        18$:      CMPB    R3,#13                  ; ERROR.CODE,*                  6672
000640  001010                          BNE     21$
000642  020527  000003                  CMP     R5,#3                   ; ERROR.SUB,*
```

```
000646  001003                                  BNE     20$
000650  105260  000046                  19$:    INCB    46(R0)                  ; *(TALLY.ADDR)                 6674
000654  000402                                  BR      21$                     ;                               6672
000656  105260  000050                  20$:    INCB    50(R0)                  ; *(TALLY.ADDR)                 6676
000662  005726                          21$:    TST     (SP)+                   ;                               6624
000664  000415                                  BR      24$                     ;                               6622
000666  013700  000104'                 22$:    MOV     ICTLR,R0                ;                               6684
000672  006300                                  ASL     R0
000674  062700  000000G                         ADD     #C.ERR.TBL,R0
000700  032766  000001  000004                  BIT     #1,4(SP)                ; *,SOFT.OCCURED                6682
000706  001403                                  BEQ     23$
000710  105260  000001                          INCB    1(R0)                   ;                               6684
000714  000401                                  BR      24$                     ;                               6682
000716  105210                          23$:    INCB    (R0)                    ;                               6686
000720  062706  000012                  24$:    ADD     #12,SP                  ;                               6569
000724  000207                                  RTS     PC

; Routine Size:  235 words,     Routine Base:  $CODE$ + 32052
; Maximum stack depth per invocation:  13 words
```

```
;   6689  1     routine ERR HRD RTNE (ERRNUM) : novalue =
;   6690  1
;   6691  1     !+
;   6692  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;   6693  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;   6694  1     !
;   6695  1
;   6696  2         begin
;   6697  2
;   6698  2         local
;   6699  2             CUR_PRIORITY : word;
;   6700  2
;   6701  2         builtin
;   6702  2             PC;
;   6703  2
;   6704  2         GETPRI (CUR_PRIORITY);
;   6705  2     !ZZZ    SETPRI (PRI04);                     ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;   6706  2         SETPRI (.BRLEVEL);                      ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW    ZZZ
;   6707  2
;   6708  2         if (.ERRNUM lequ 34) or                 ! FOR NON BAD BLOCK TYPE ERRORS
;   6709  2            (.ERRNUM gtru 38) or
;   6710  2            (.ERRNUM eql 36) or
;   6711  3            (.ERRNUM eql 37)
;   6712  2         then
;   6713  2
;   6714  3             if BIT_TST (SWP_FLAGS, SWF_HRD)     ! IF ERRORS TO BE TREATED NORMALLY
;   6715  2             then
;   6716  2
;   6717  2     !ZZZ        case .ERRNUM from 31 to 45 of
;   6718  2                 case .ERRNUM from 31 to 73 of   !INCLUDE DUP NUMBERS (60 73)          ZZZ
;   6719  2             set
;   6720  2
;   6721  2                     [31]:   ERRHRD (31, EGM_30, EMS 30),    ! INVALID COMMAND
;   6722  2
;   6723  2                     [32]:   ERRHRD (32, EGM_30, EMS 30);    ! COMMAND ABORTED
;   6724  2
;   6725  2                     [33]:   ;                               !
;   6726  2
;   6727  2                     [34]:   ;                               !
;   6728  2
;   6729  2                     [35]:   ;                               ! MEDIA FORMAT ERROR
;   6730  2
;   6731  2                     [36]:   ERRHRD (36, EGM_30, EMS_30);    ! WRITE PROTECTED
;   6732  2
;   6733  2                     [37]:   ERRHRD (37, EGM_30, EMS 30);    ! COMPARE ERROR
;   6734  2
;   6735  2                     [38]:   ;                               ! DATA ERROR
;   6736  2
;   6737  2                     [39]:   ERRHRD (39, EGM_30, EMS 30);    ! HOST BUFFER ACCESS ERROR
;   6738  2
;   6739  2                     [40]:   ERRHRD (40, EGM 30, EMS 30);    ! CONTROLLER ERROR
;   6740  2
;   6741  2                     [41]:   ERRHRD (41, EGM 30, EMS 30);    ! DRIVE ERROR
```

```
;   6742  2
;   6743  2                [42]:   ERRHRD (42, EGH_3G, 0);          ! HOST WRITE COMPARE ERROR
;   6744  2
;   6745  2                [43]:   ERRHRD (43, EGH_3O, EMS_3O);     ! MESSAGE FROM IN;ERNAL DIAGNOSTICS
;   6746  2
;   6747  2                [44]:   ERRHRD (44, EGH_3C, EMS_3O);     ! DUPLICATE UNIT NUMBER
;   6748  2
;   6749  2                [45].   ERRHRD (45, EGH_3O, EMS_3O);     ! INVALID END CODE
;   6750  2
;   6751  2                [46]:   ;                                !LEAVE ROOM FOR SOFT ERROR NUMBERS AND SOME PADDING  ZZZ
;   6752  2                [47]:   ;                                !
;   6753  2                [48]:   ;                                !
;   6754  2                [49]:   ;                                !
;   6755  2                [50]:   ;                                !
;   6756  2                [51]:   ;                                !
;   6757  2                [52]:   ;                                !
;   6758  2                [53]:   ;                                !
;   6759  2                [54]:   ;                                !
;   6760  2                [55]:   ;                                !
;   6761  2                [56]:   ;                                !
;   6762  2                [57]:   ;                                !
;   6763  2                [58]:   ;                                !
;   6764  2                [59]:   ;                                !
;   6765  2
;   6766  2                [60]:  ERRHRD (60, EH_12, EMS_30);       !NOT USED                                          ZZZ
;   6767  2                [61]:  ERRHRD (61, EH_13, EMS_30);       !SUCCESSFUL MESSAGE                                ZZZ
;   6768  2                [62]:  ERRHRD (62, EH_13, EMS_30);       !ILLEGAL UNIT NUMBER                               ZZZ
;   6769  2                [63]:  ERRHRD (63, EH_13, EMS_30);       !ILLEGAL RELATIVE OR PHYSICAL BLOCK                ZZZ
;   6770  2                [64]:  ERRHRD (64, EH_13, EMS_30);       !DEVICE ERROR                                      ZZZ
;   6771  2                [65]:  ERRHRD (65, EH_13, EMS_30);       !ZERO LENGTH MESSAGE                               ZZZ
;   6772  2                [66]:  ERRHRD (66, EH_8,  EMS_30);       !DUP UNKNOWN STATUS CODE                           ZZZ
;   6773  2                [67]:  ERRHRD (67, EH_7,  EMS_30);       !INVALID COMMAND                                   ZZZ
;   6774  2                [68]:  ERRHRD (68, EH_7,  EMS_30);       !NO REGION AVAILABLE                               ZZZ
;   6775  2                [69]:  ERRHRD (69, EH_7,  EMS_30);       !NO REGION SUITABI E                               ZZZ
;   6776  2                [70]:  ERRHRD (70, EH_7,  EMS_30);       !PROGRAM NOT KNOWN                                 ZZZ
;   6777  2                [71]:  ERRHRD (71, EH_7,  EMS_30);       !LOAD FAILURE                                      ZZZ
;   6778  2                [72]:  ERRHRD (72, EH_7,  EMS_30);       !STANDALONE                                        ZZZ
;   6779  2                [73]:  ERRHRD (73, EH_8,  EMS_30);       !DUP UNKNOWN STATUS CODE                           ZZZ
;   6780  2
;   6781  2                    tes
;   6782  2            else
;   6783  3                begin
;   6784  3                !****increment error count               ! INCREMENT TOTAL ERROR COUNT
;   6785  3                PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);  ! PRINT ERROR MESSAGE JUST LIKE DRS
;   6786  3
;   6787  3                if .ERRNUM neq 42
;   6788  3                then
;   6789  4                    begin
;   6790  4                    PRINTB (HRD_SUB);                    ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;   6791  4                    EMS ERR ();                          ! PRINT REST OF THE INFORMATION
;   6792  3                    end;
;   6793  2                end;
;   6794  2
```

```
;  6795  2          if (.ERRNUM eql 35) or                          ! FOR BAD-BLOCK TYPE ERRORS
;  6796  3             (.ERRNUM eql 38)
;  6797  2          then
;  6798  2
;  6799  3              if BIT_TST (SWP_FLAGS, SWF_BLK)             ! IF ERRORS TO BE TREATED NORMALLY
;  6800  2              then
;  6801  2
;  6802  2                  selectoneu .ERRNUM of
;  6803  2                      set
;  6804  2
;  6805  2                      [35]:   ERRHRD (35, EGH_30, EMS_30);    ! MEDIA FORMAT ERROR
;  6806  2
;  6807  2                      [38]:   ERRHRD (38, EGH_30, EMS_30);    ! DATA ERROR
;  6808  2                      tes
;  6809  2              else
;  6810  3                  begin
;  6811  3                  !****increment error count              ! INCREMENT TOTAL ERROR COUNT
;  6812  3                  PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC); ! PRINT ERROR LINE JUST LIKE DRS
;  6813  3                  PRINTB (HRD_SUB);                       ! PRINT NEXT LINE TOO
;  6814  3                  EMS_ERR ();                             ! PRINT REST OF THE INFORMATION
;  6815  2                  end;
;  6816  2
;  6817  2          SETPRI (.CUR_PRIORITY);                         ! PRIORITY BACK TO NORMAL
;  6818  2
;  6819  1          end;


                                .SBTTL  ERR.HRD.RTNE RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G   ERR.HRD.RTNE:
                        JSR     R1,$SAVE2                       ;                       6689
000004  104440             TRAP    40                              ;                       6704
000006  010002             MOV     R0,R2                           ; *,CUR.PRIORITY
000010  013700  000000G    MOV     @#BRLEVEL,R0                    ;                       6706
000014  104441             TRAP    41
000016  016601  000010     MOV     10(SP),R1                       ; ERRNUM,*              6708
000022  020127  000042     CMP     R1,#42
000026  101411             BLOS    1$
000030  020127  000046     CMP     R1,#46                          ;                       6709
000034  101006             BHI     1$
000036  020127  000044     CMP     R1,#44                          ;                       6710
000042  001403             BEQ     1$
000044  020127  000045     CMP     R1,#45                          ;                       6711
000050  001176             BNE     27$
000052  032737  010000  000000G  1$:  BIT     #10000,@#SWP.FLAGS              ;               6714
000060  001002             BNE     2$
000062  000137  033474'     JMP     31$
000066  010100         2$:  MOV     R1,R0                           ;                       6718
000070  162700  000037     SUB     #37,R0
000074  006300             ASL     R0
000076  066007  000024'     ADD     P.AAC(R0),PC                    ; Case dispatch
000102  104456         4$:  TRAP    56                              ;                       6721
000104  000037             .WORD   37
000106  000000G            .WORD   EGH.30
```

```
000110  000000G                         .WORD   EMS.30
000112  000567                         BR      30$
000114  104456                 5$:     TRAP    56                              ;       6718
000116  000040                         .WORD   40                              ;       6723
000120  000000G                         .WORD   EGH.30
000122  000000G                         .WORD   EMS.30
000124  000562                         BR      30$
000126  104456                 6$:     TRAP    56                              ;       6718
000130  000044                         .WORD   44                              ;       6731
000132  000000G                         .WORD   EGH.30
000134  000000G                         .WORD   EMS.30
000136  000535                         BR      30$
000140  104456                 7$:     TRAP    56                              ;       6718
000142  000045                         .WORD   45                              ;       6733
000144  000000G                         .WORD   EGH.30
000146  000000G                         .WORD   EMS.30
000150  000550                         BR      30$
000152  104456                 8$:     TRAP    56                              ;       6718
000154  000047                         .WORD   47                              ;       6737
000156  000000G                         .WORD   EGH.30
000160  000000G                         .WORD   EMS.30
000162  000574                         BR      33$
000164  104456                 9$:     TRAP    56                              ;       6718
000166  000050                         .WORD   50                              ;       6739
000170  000000G                         .WORD   EGH.30
000172  000000G                         .WORD   EMS.30
000174  000567                         BR      33$
000176  104456                 10$:    TRAP    56                              ;       6718
000200  000051                         .WORD   51                              ;       6741
000202  000000G                         .WORD   EGH.30
000204  000000G                         .WORD   EMS.30
000206  000562                         BR      33$
000210  104456                 11$:    TRAP    56                              ;       6718
000212  000052                         .WORD   52                              ;       6743
000214  000000G                         .WORD   EGH.30
000216  000000                         .WORD   0
000220  000555                         BR      33$
000222  104456                 12$:    TRAP    56                              ;       6718
000224  000053                         .WORD   53                              ;       6745
000226  000000G                         .WORD   EGH.30
000230  000000G                         .WORD   EMS.30
000232  000550                         BR      33$
000234  104456                 13$:    TRAP    56                              ;       6718
000236  000054                         .WORD   54                              ;       6747
000240  000000G                         .WORD   EGH.30
000242  000000G                         .WORD   EMS.30
000244  000543                         BR      33$
000246  104456                 14$:    TRAP    56                              ;       6718
000250  000055                         .WORD   55                              ;       6749
000252  000000G                         .WORD   EGH.30
000254  000000G                         .WORD   EMS.30
000256  000536                         BR      33$
000260  104456                 15$:    TRAP    56                              ;       6718
                                                                                ;       6766
```

```
000262  000074                          .WORD   74
000264  000000G                         .WORD   EH.12
000266  000000G                         .WORD   EMS.30
000270  000531                          BR      33$                     ;
000272  104456                  16$:    TRAP    56                      ;                       6718
000274  000075                          .WORD   75                      ;                       6767
000276  000000G                         .WORD   EH.13
000300  000000G                         .WORD   EMS.30
000302  000524                          BR      33$                     ;
000304  104456                  17$:    TRAP    56                      ;                       6718
000306  000076                          .WORD   76                      ;                       6768
000310  000000G                         .WORD   EH.13
000312  000000G                         .WORD   EMS.30
000314  000517                          BR      33$                     ;
000316  104456                  18$:    TRAP    56                      ;                       6718
000320  000077                          .WORD   77                      ;                       6769
000322  000000G                         .WORD   EH.13
000324  000000G                         .WORD   EMS.30
000326  000512                          BR      33$                     ;
000330  104456                  19$:    TRAP    56                      ;                       6718
000332  000100                          .WORD   100                     ;                       6770
000334  000000G                         .WORD   EH.13
000336  000000G                         .WORD   EMS.30
000340  000505                          BR      33$                     ;
000342  104456                  20$:    TRAP    56                      ;                       6718
000344  000101                          .WORD   101                     ;                       6771
000346  000000G                         .WORD   EH.13
000350  000000G                         .WORD   EMS.30
000352  000500                          BR      33$                     ;
000354  104456                  21$:    TRAP    56                      ;                       6718
000356  000102                          .WORD   102                     ;                       6772
000360  000000G                         .WORD   EH.8
000362  000000G                         .WORD   EMS.30
000364  000473                          BR      33$                     ;
000366  104456                  22$:    TRAP    56                      ;                       6718
000370  000103                          .WORD   103                     ;                       6773
000372  000000G                         .WORD   EH.7
000374  000000G                         .WORD   EMS.30
000376  000466                          BR      33$                     ;
000400  104456                  23$:    TRAP    56                      ;                       6718
000402  000104                          .WORD   104                     ;                       6774
000404  000000G                         .WORD   EH.7
000406  000000G                         .WORD   EMS.30
000410  000461                          BR      33$                     ;
000412  104456                  24$:    TRAP    56                      ;                       6718
000414  000105                          .WORD   105                     ;                       6775
000416  000000G                         .WORD   EH.7
000420  000000G                         .WORD   EMS.30
000422  000454                          BR      33$                     ;
000424  104456                  25$:    TRAP    56                      ;                       6718
000426  000106                          .WORD   106                     ;                       6776
000430  000000G                         .WORD   EH.7
000432  000000G                         .WORD   EMS.30
```

```
000434  000447                          BR      33$                             ;
000436  104456                  26$:    TRAP    56                              ;                       6718
000440  000107                          .WORD   107                             ;                       6777
000442  000000G                         .WORD   EH.7
000444  000000G                         .WORD   EMS.30
000446  000442                  27$:    BR      33$                             ;
000450  104456                  28$:    TRAP    56                              ;                       6718
000452  000110                          .WORD   110                             ;                       6778
000454  000000G                         .WORD   EH.7
000456  000000G                         .WORD   EMS.30
000460  000435                          BR      33$                             ;
000462  104456                  29$:    TRAP    56                              ;                       6718
000464  000111                          .WORD   111                             ;                       6779
000466  000000G                         .WORD   EH.8
000470  000000G                         .WORD   EMS.30
000472  000430                  30$:    BR      33$                             ;
000474  010746                  31$:    MOV     PC,-(SP)                        ;                       6714
000476  013746  000000G                 MOV     L$LUN,-(SP)                     ; PC,*                  6785
000502  010146                          MOV     R1,-(SP)
000504  012746  000000G                 MOV     #HRD.MSG,-(SP)
000510  012746  000004                  MOV     #4,-(SP)
000514  010600                          MOV     SP,R0
000516  104414                          TRAP    14                              ; SP,*
000520  020127  000052                  CMP     R1,#52                          ;                       6787
000524  001411                          BEQ     32$
000526  012716  000000G                 MOV     #HRD.SUB,(SP)                   ;                       6790
000532  012746  000001                  MOV     #1,-(SP)
000536  010600                          MOV     SP,R0                           ; SP,*
000540  104414                          TRAP    14
000542  004737  000000G                 JSR     PC,EMS.ERR                      ;                       6791
000546  005726                          TST     (SP)+                           ;                       6789
000550  062706  000012          32$:    ADD     #12,SP                          ;                       6783
000554  020127  000043          33$:    CMP     R1,#43                          ;                       6795
000560  001403                          BEQ     34$
000562  020127  000046                  CMP     R1,#46                          ;                       6796
000566  001050                          BNE     37$
000570  032737  040000  000000G 34$:    BIT     #40000,SWP.FLAGS                ;                       6799
000576  001420                          BEQ     36$
000600  020127  000043                  CMP     R1,#43                          ;                       6805
000604  001005                          BNE     35$
000606  104456                          TRAP    56
000610  000043                          .WORD   43
000612  000000G                         .WORD   EGH.30
000614  000000G                         .WORD   EMS.30
000616  000434                          BR      37$                             ;                       6802
000620  020127  000046          35$:    CMP     R1,#46                          ;                       6807
000624  001031                          BNE     37$
000626  104456                          TRAP    56
000630  000046                          .WORD   46
000632  000000G                         .WORD   EGH.30
000634  000000G                         .WORD   EMS.30
000636  000424                          BR      37$                             ;                       6802
000640  010746                  36$:    MOV     PC,-(SP)                        ; PC,*                  6812
```

```
000642  013746  000000G                        MOV     L$LUN,-(SP)
000646  010146                                  MOV     R1,-(SP)
000650  012746  000000G                        MOV     #HRD.MSG,-(SP)
000654  012746  000004                        MOV     #4,-(SP)
000660  010600                                  MOV     SP,R0                   ; SP,*
000662  104414                                  TRAP    14
000664  012716  000000G                        MOV     #HRD.SUB,(SP)           ;                               6813
000670  012746  000001                        MOV     #1,-(SP)
000674  010600                                  MOV     SP,R0                   ; SP,*
000676  104414                                  TRAP    14
000700  004737  000000G                        JSR     PC,EMS.ERR              ;                               6814
000704  062706  000014                        ADD     #14,SP                  ;                               6810
000710  010200                        37$:     MOV     R2,R0                   ; CUR.PRIORITY,*                6817
000712  104441                                  TRAP    41
000714  000207                                  RTS     PC                      ;                               6689

; Routine Size:  231 words,    Routine Base:  $CODE$ + 33000
; Maximum stack depth per invocation:  11 words


000024                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAC:
000024  000000                  3$:     .WORD   0                               ; CASE Table for ERR.HRD.RTNE+0076      6718
000026  000012                          .WORD   12                              ; [4$]
000030  000452                          .WORD   452                             ; [5$]
000032  000452                          .WORD   452                             ; [33$]
000034  000452                          .WORD   452                             ; [33$]
000036  000024                          .WORD   24                              ; [6$]
000040  000036                          .WORD   36                              ; [7$]
000042  000452                          .WORD   452                             ; [33$]
000044  000050                          .WORD   50                              ; [8$]
000046  000062                          .WORD   62                              ; [9$]
000050  000074                          .WORD   74                              ; [10$]
000052  000106                          .WORD   106                             ; [11$]
000054  000120                          .WORD   120                             ; [12$]
000056  000132                          .WORD   132                             ; [13$]
000060  000144                          .WORD   144                             ; [14$]
000062  000452                          .WORD   452                             ; [33$]
000064  000452                          .WORD   452                             ; [33$]
000066  000452                          .WORD   452                             ; [33$]
000070  000452                          .WORD   452                             ; [33$]
000072  000452                          .WORD   452                             ; [33$]
000074  000452                          .WORD   452                             ; [33$]
000076  000452                          .WORD   452                             ; [33$]
000100  000452                          .WORD   452                             ; [33$]
000102  000452                          .WORD   452                             ; [33$]
000104  000452                          .WORD   452                             ; [33$]
000106  000452                          .WORD   452                             ; [33$]
000110  000452                          .WORD   452                             ; [33$]
000112  000452                          .WORD   452                             ; [33$]
000114  000452                          .WORD   452                             ; [33$]
000116  000156                          .WORD   156                             ; [15$]
```

```
000120  000170                          .WORD   170             ; [16$]
000122  000202                          .WORD   202             ; [17$]
000124  000214                          .WORD   214             ; [18$]
000126  000226                          .WORD   226             ; [19$]
000130  000240                          .WORD   240             ; [20$]
000132  000252                          .WORD   252             ; [21$]
000134  000264                          .WORD   264             ; [22$]
000136  000276                          .WORD   276             ; [23$]
000140  000310                          .WORD   310             ; [24$]
000142  000322                          .WORD   322             ; [25$]
000144  000334                          .WORD   334             ; [26$]
000146  000346                          .WORD   346             ; [28$]
000150  000360                          .WORD   360             ; [29$]
```

```
;    6820  1     routine ERR_SOFT_RTNE (ERRNUM) : novalue =
;    6821  1
;    6822  1     !+
;    6823  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;    6824  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6825  1     !-
;    6826  1
;    6827  2         begin
;    6828  2
;    6829  2         builtin
;    6830  2             PC;
;    6831  2
;    6832  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                    ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    6833  2         then
;    6834  2
;    6835  2             case .ERRNUM from 50 to 54 of
;    6836  2                 set
;    6837  2
;    6838  2                 [50]:       ERRSOFT (50, 0, 0);            ! CONTROLLER ERROR
;    6839  2
;    6840  2                 [51]:       ERRSOFT (51, 0, 0);            ! HOST MEMORY ACCESS ERROR
;    6841  2
;    6842  2                 [52]:       ERRSOFT (52, 0, 0);            ! DISK TRANSFER ERROR
;    6843  2
;    6844  2                 [53]:       ERRSOFT (53, 0, 0);            ! SDI ERROR
;    6845  2
;    6846  2                 [54]:       ERRSOFT (54, 0, 0);            ! SMALL DISK ERROR
;    6847  2                 tes
;    6848  2         else
;    6849  3             begin
;    6850  3             !****increment error count                     ! INCREMENT TOTAL ERROR COUNT
;    6851  3             PRINTB (SFT_MSG, .ERRNUM, .L$LUN, .PC);        ! PRINT ERROR LINE JUST LIKE DRS
;    6852  2             end;
;    6853  2
;    6854  1         end;


                                        .SBTTL  ERR.SOFT.RTNE RDRX INTERRUPT SERVICE ROUTINES
033716                                  .PSECT  $CODE$,  RO

000000  032737  020000  000000G ERR.SOFT.RTNE:
                                        BIT     #20000,SWP.FLAGS        ;                       6832
000006  001440                          BEQ     7$
000010  016600  000002                  MOV     2(SP),RO                ; ERRNUM,*              6835
000014  162700  000062                  SUB     #62,RO
000020  006300                          ASL     RO
000022  066007  000152'                 ADD     P.AAD(RO),PC            ; Case dispatch
000026  104457                  2$:     TRAP    57                      ;                       6838
000030  000062                          .WORD   62
000032  000000                          .WORD   0
000034  000000                          .WORD   0
000036  000207                          RTS     PC                      ;                       6835
000040  104457                  3$:     TRAP    57                      ;                       6840
```

```
000042  000063                          .WORD   63
000044  000000                          .WORD   0
000046  000000                          .WORD   0
000050  000207                          RTS     PC                                      ;
000052  104457                  4$:     TRAP    57                                      ;                       6835
000054  000064                          .WORD   64                                      ;                       6842
000056  000000                          .WORD   0
000060  000000                          .WORD   0
000062  000207                          RTS     PC                                      ;
000064  104457                  5$:     TRAP    57                                      ;                       6835
000066  000065                          .WORD   65                                      ;                       6844
000070  000000                          .WORD   0
000072  000000                          .WORD   0
000074  000207                          RTS     PC                                      ;
000076  104457                  6$:     TRAP    57                                      ;                       6835
000100  000066                          .WORD   66                                      ;                       6846
000102  000000                          .WORD   0
000104  000000                          .WORD   0
000106  000207                          RTS     PC                                      ;
000110  010746                  7$:     MOV     PC,-(SP)                                ;                       6832
000112  013746  000000G                 MOV     L$LUN,-(SP)                             ; PC,*                  6851
000116  016646  000006                  MOV     6(SP),-(SP)                             ; ERRNUM,*
000122  012746  000000G                 MOV     #SFT.MSG,-(SP)
000126  012746  000004                  MOV     #4,-(SP)
000132  010600                          MOV     SP,R0                                   ; SP,*
000134  104414                          TRAP    14
000136  062706  000012                  ADD     #12,SP                                  ;                       6849
000142  000207                          RTS     PC                                      ;                       6820

; Routine Si      50 words,      Routine Base:  $CODE$ + 33716
; Maximum st...   depth per invocation:  7 words


000152                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAD:
000152  000000                1$:     .WORD   0                                       ; CASE Table for ERR.SOFT.RTNE+0022     6835
000154  000012                        .WORD   12                                      ; [2$]
000156  000024                        .WORD   24                                      ; [3$]
000160  000036                        .WORD   36                                      ; [4$]
000162  000050                        .WORD   50                                      ; [5$]
                                                                                        ; [6$]
```

J6


```
;    6855  1      routine ERR_HRD_RTNE_APT (ERRNUM) : novalue =
;    6856  1
;    6857  1
;    6858  1      !+
;    6859  1      ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;    6860  1      ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6861  1      !-
;    6862  1
;    6863  1
;    6864  2          begin
;    6865  2
;    6866  2
;    6867  2          local
;    6868  2              CUR_PRIORITY;
;    6869  2
;    6870  2
;    6871  2          builtin
;    6872  2              PC;
;    6873  2
;    6874  2          GETPRI (CUR_PRIORITY);
;    6875  2      !ZZZ    SETPRI (PRI04);                          ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;    6876  2          SETPRI (.BRLEVEL);                           ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW     ZZZ
;    6877  2
;    6878  2
;    6879  2          if .APT_MODE
;    6880  2          then
;    6881  2
;    6882  3              begin
;    6883  3              .MAIL_BOX_TESTNUM = .RP_ADDR [LBN_LO];   ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    6884  3              .MAIL_BOX_SUBTST = .RP_ADDR [DISK];      ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER UNDER APT ONLY
;    6885  2              end;
;    6886  2
;    6887  2
;    6888  2          if (.ERRNUM lequ 34) or                      ! FOR NON-BAD BLOCK TYPE ERRORS
;    6889  2             (.ERRNUM gtru 38) or
;    6890  2             (.ERRNUM eql 36) or
;    6891  3             (.ERRNUM eql 37)
;    6892  3
;    6893  2          then
;    6894  2
;    6895  3              if BIT_TST (SWP_FLAGS, SWF_HRD)          ! IF ERRORS TO BE TREATED NORMALLY
;    6896  2              then
;    6897  2
;    6898  2                  case .ERRNUM from 31 to 45 of
;    6899  2                      set
;    6900  2
;    6901  2                      [31]:   ERRDF (31, EGH_30, EMS_30);     ! INVALID COMMAND
;    6902  2
;    6903  2                      [32]:   ERRDF (32, EGH_30, EMS_30);     ! COMMAND ABORTED
;    6904  2
;    6905  2                      [33]:   ;                               !
;    6906  2
;    6907  2                      [34]:   ;                               !
```

```
;    6908  2
;    6909  2                    [35]:   ;                                   ! MEDIA FORMAT ERROR
;    6910  2
;    6911  2                    [36]:   ERRDF (36, EGH_30, EMS_30);         ! WRITE PROTECTED
;    6912  2
;    6913  2                    [37]:   ERRDF (37, EGH_30, EMS 30);         ! COMPARE ERROR
;    6914  2
;    6915  2                    [38].   ;                                   ! DATA ERROR
;    6916  2
;    6917  2                    [39]:   ERRDF (39, EGH_30, EMS_30);         ! HOST BUFFER ACCESS ERROR
;    6918  2
;    6919  2                    [40]:   ERRDF (40, EGH_30, EMS_30);         ! CONTROLLER ERROR
;    6920  2
;    6921  2                    [41]:   ERRDF (41, EGH_30, EMS_30);         ! DRIVE ERROR
;    6922  2
;    6923  2                    [42]:   ERRDF (42, EGH_30, 0);              ! HOST WRITE COMPARE ERROR
;    6924  2
;    6925  2                    [43]:   ERRDF (43, EGH_30, EMS_30);         ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    6926  2
;    6927  2                    [44]:   ERRDF (44, EGH_30, EMS_30);         ! DUPLICATE UNIT NUMBER
;    6928  2
;    6929  2                    [45]:   ERRDF (45, EGH_30, EMS 30);         ! INVALID END CODE
;    6930  2                    tes
;    6931  2
;    6932  2                else
;    6933  2
;    6934  3                    begin
;    6935  3                    !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;    6936  3                    PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);      ! PRINT ERROR MESSAGE JUST LIKE DRS
;    6937  3
;    6938  3
;    6939  3                    if .ERRNUM neq 42
;    6940  3
;    6941  3                    then
;    6942  4                        begin
;    6943  4                        PRINTB (HRD_SUB);                       ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;    6944  4                        EMS_ERR ();                             ! PRINT REST OF THE INFORMATION
;    6945  3                        end;
;    6946  2                    end;
;    6947  2
;    6948  2                if (.ERRNUM eql 35) or                          ! FOR BAD-BLOCK TYPE ERRORS
;    6949  3                   (.ERRNUM eql 38)
;    6950  3
;    6951  2                then
;    6952  2
;    6953  3                    if BIT_TST (SWP_FLAGS, SWF BLK)             ! IF ERRORS TO BE TREATED NORMALLY
;    6954  2                    then
;    6955  2
;    6956  2                        selectoneu .ERRNUM of
;    6957  2                        set
;    6958  2
;    6959  2                        [35]:   ERRDF (35, EGH 30, EMS 30);     ! MEDIA FORMAT ERROR
;    6960  2
```

```
;    6961  2                    [38]:   ERRD+ (38, EGH 30, EMS 30);     ! DATA ERROR
;    6962  2                    tes
;    6963  2
;    6964  2               else
;    6965  2
;    6966  3                    begin
;    6967  3                    !****increment error count                ! INCREMENT TOTAL ERROR COUNT
;    6968  3                    PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);    ! PRINT ERROR LINE JUST LIKE DRS
;    6969  3                    PRINTB (HRD_SUB);                         ! PRINT NEXT LINE TOO
;    6970  3                    EMS_ERR ();                               ! PRINT REST OF THE INFORMATION
;    6971  2                    end;
;    6972  2
;    6973  2
;    6974  2               SETPRI (.CUR PRIORITY);                        ! PRIORITY BACK TO NORMAL
;    6975  2
;    6976  2
;    6977  1               end;


034062                                          .SBTTL  ERR.HRD.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
                                                .PSECT  $CODE$,  RO

000000  004137  000000G                 ERR.HRD.RTNE.APT:
                                                JSR     R1,$SAVE2                       ;                       6855
000004  104440                                  TRAP    40                              ;                       6874
000006  010002                                  MOV     R0,R2                           ; *,CUR.PRIORITY
000010  013700  000000G                         MOV     BRLEVEL,R0                      ;                       6876
000014  104441                                  TRAP    41
000016  032737  000001  001254'                 BIT     #1,APT.MODE                     ;                       6879
000024  001412                                  BEQ     1$
000026  013700  000000G                         MOV     RP.ADDR,R0                      ;                       6883
000032  016077  000050  001256'                 MOV     50(R0),@MAIL.BOX.TESTNUM
000040  013700  000000G                         MOV     RP.ADDR,R0                      ;                       6884
000044  016077  000010  001260'                 MOV     10(R0),@MAIL.BOX.SUBTST
000052  016601  000010                  1$:     MOV     10(SP),R1                       ; ERRNUM,*              6888
000056  020127  000042                          CMP     R1,#42
000062  101411                                  BLOS    2$
000064  020127  000046                          CMP     R1,#46                          ;                       6889
000070  101006                                  BHI     2$
000072  020127  000044                          CMP     R1,#44                          ;                       6890
000076  001403                                  BEQ     2$
000100  020127  000045                          CMP     R1,#45                          ;                       6891
000104  001131                                  BNE     17$
000106  032737  010000  000000G         2$:     BIT     #10000,SWP.FLAGS                ;                       6895
000114  001475                                  BEQ     15$
000116  010100                                  MOV     R1,R0                           ;                       6898
000120  162700  000037                          SUB     #37,R0
000124  006300                                  ASL     R0
000126  066007  000164'                         ADD     P.AAE(R0),PC                    ; Case dispatch
000132  104455                          4$:     TRAP    55                              ;                       6901
000134  000037                                  .WORD   37
000136  000000G                                 .WORD   EGH.30
000140  000000G                                 .WORD   EMS.30
```

```
000142  000512                          BR      17$                             ;
000144  104455                  5$:     TRAP    55                              ;                       6898
000146  000040                          .WORD   40                              ;                       6903
000150  000000G                         .WORD   EGM.30
000152  000000G                         .WORD   EMS.30
000154  000505                          BR      17$
000156  104455                  6$:     TRAP    55                              ;                       6898
000160  000044                          .WORD   44                              ;                       6911
000162  000000G                         .WORD   EGM.30
000164  000000G                         .WORD   EMS.30
000166  000500                          BR      17$
000170  104455                  7$:     TRAP    55                              ;                       6898
000172  000045                          .WORD   45                              ;                       6913
000174  000000G                         .WORD   EGM.30
000176  000000G                         .WORD   EMS.30
000200  000473                          BR      17$
000202  104455                  8$:     TRAP    55                              ;                       6898
000204  000047                          .WORD   47                              ;                       6917
000206  000000G                         .WORD   EGM.30
000210  000000G                         .WORD   EMS.30
000212  000466                          BR      17$
000214  104455                  9$:     TRAP    55                              ;                       6898
000216  000050                          .WORD   50                              ;                       6919
000220  000000G                         .WORD   EGM.30
000222  000000G                         .WORD   EMS.30
000224  000461                          BR      17$
000226  104455                  10$:    TRAP    55                              ;                       6898
000230  000051                          .WORD   51                              ;                       6921
000232  000000G                         .WORD   EGM.30
000234  000000G                         .WORD   EMS.30
000236  000454                          BR      17$
000240  104455                  11$:    TRAP    55                              ;                       6898
000242  000052                          .WORD   52                              ;                       6923
000244  000000G                         .WORD   EGM.30
000246  000000                          .WORD   0
000250  000447                          BR      17$
000252  104455                  12$:    TRAP    55                              ;                       6898
000254  000053                          .WORD   53                              ;                       6925
000256  000000G                         .WORD   EGM.30
000260  000000G                         .WORD   EMS.30
000262  000442                          BR      17$
000264  104455                  13$:    TRAP    55                              ;                       6898
000266  000054                          .WORD   54                              ;                       6927
000270  000000G                         .WORD   EGM.30
000272  000000G                         .WORD   EMS.30
000274  000435                          BR      17$
000276  104455                  14$:    TRAP    55                              ;                       6898
000300  000055                          .WORD   55                              ;                       6929
000302  000000G                         .WORD   EGM.30
000304  000000G                         .WORD   EMS.30
000306  000430                          BR      17$
000310  010746                  15$:    MOV     PC, (SP)                        ; PC.*                  6895
000312  013746  000000G                 MOV     L$LUN, (SP)                                             6936
```

```
000316  010146                                  MOV     R1, (SP)
000320  012746  000000G                         MOV     #DF.MSG, (SP)
000324  012746  000004                          MOV     #4,-(SP)
000330  010600                                  MOV     SP,R0                   ; SP,*
000332  104414                                  TRAP    14
000334  020127  000052                          CMP     R1,#52                  ;                       6939
000340  001411                                  BEQ     16$
000342  012716  000000G                         MOV     #HRD.SUB,(SP)           ;                       6943
000346  012746  000001                          MOV     #1,-(SP)
000352  010600                                  MOV     SP,R0                   ; SP,*
000354  104414                                  TRAP    14
000356  004737  000000G                         JSR     PC,EMS.ERR              ;                       6944
000362  005726                                  TST     (SP)+                   ;                       6942
000364  062706  000012              16$:        ADD     #12,SP                  ;                       6934
000370  020127  000043              17$:        CMP     R1,#43                  ;                       6948
000374  001403                                  BEQ     18$
000376  020127  000046                          CMP     R1,#46                  ;                       6949
000402  001050                                  BNE     21$
000404  032737  040000  000000G     18$:        BIT     #40000,SWP.FLAGS        ;                       6953
000412  001420                                  BEQ     20$
000414  020127  000043                          CMP     R1,#43                  ;                       6959
000420  001005                                  BNE     19$
000422  104455                                  TRAP    55
000424  000043                                  .WORD   43
000426  000000G                                 .WORD   EGH.30
000430  000000G                                 .WORD   EMS.30
000432  000434                                  BR      21$
000434  020127  000046              19$:        CMP     R1,#46                  ;                       6956
000440  001031                                  BNE     21$                     ;                       6961
000442  104455                                  TRAP    55
000444  000046                                  .WORD   46
000446  000000G                                 .WORD   EGH.30
000450  000000G                                 .WORD   EMS.30
000452  000424                                  BR      21$                     ;                       6956
000454  010746                  20$:            MOV     PC,-(SP)                ; PC,*                  6968
000456  013746  000000G                         MOV     L$LUN,-(SP)
000462  010146                                  MOV     R1,-(SP)
000464  012746  000000G                         MOV     #DF.MSG,-(SP)
000470  012746  000004                          MOV     #4,-(SP)
000474  010600                                  MOV     SP,R0                   ; SP,*
000476  104414                                  TRAP    14
000500  012716  000000G                         MOV     #HRD.SUB,(SP)           ;                       6969
000504  012746  000001                          MOV     #1,-(SP)
000510  010600                                  MOV     SP,R0                   ; SP,*
000512  104414                                  TRAP    14
000514  004737  000000G                         JSR     PC,EMS.ERR              ;                       6970
000520  062706  000014                          ADD     #14,SP                  ;                       6966
000524  010200                  21$:            MOV     R2,R0                   ; CUR.PRIORITY,*        6974
000526  104441                                  TRAP    41
000530  000207                                  RTS     PC                      ;                       6855

; Routine Size:  173 words,     Routine Base:  $CODE$ + 34 62
; Maximum stack depth per invocation:  11 words
```

```
00016'                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAE:
000164  000000                    3$:     .WORD   0                       ; CASE Table for ERR.HRD.RTNE.AP+0126   6898
000166  000012                            .WORD   12                      ; [4$]
000170  000236                            .WORD   236                     ; [5$]
000172  000236                            .WORD   236                     ; [17$]
000174  000236                            .WORD   236                     ; [17$]
000176  000024                            .WORD   24                      ; [17$]
000200  000036                            .WORD   36                      ; [6$]
000202  000236                            .WORD   236                     ; [7$]
000204  000050                            .WORD   50                      ; [17$]
000206  000062                            .WORD   62                      ; [8$]
000210  000074                            .WORD   74                      ; [9$]
000212  000106                            .WORD   106                     ; [10$]
000214  000120                            .WORD   120                     ; [11$]
000216  000132                            .WORD   132                     ; [12$]
000220  000144                            .WORD   144                     ; [13$]
                                                                        ; [14$]


;   6978  1
;   6979  1
```

```
;    6980  1     routine ERR_SOFT_RTNE_APT (ERRNUM, index) : novalue =
;    6981  1
;    6982  1     !+
;    6983  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERPSOFT' MACRO CALL TO DRS OR TO FAKE
;    6984  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6985  1     !-
;    6986  1
;    6987  2         begin
;    6988  2
;    6989  2         local
;    6990  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;    6991  2
;    6992  2         builtin
;    6993  2             PC;
;    6994  2
;    6995  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);       ! ADDRESS OF THE SAVED ERROR-LOG INFORMATION
;    6996  2
;    6997  2         if .APT_MODE
;    6998  2         then
;    6999  3             begin
;    7000  3             .MAIL_BOX_TESTNUM = .ELOG_ADDR [EL_BLOCK];      ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    7001  3             .MAIL_BOX_SUBTST = .ELOG_ADDR [EL_DK_NUM];      ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER IN APT ONLY
;    7002  2             end;
;    7003  2
;    7004  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                     ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    7005  2         then
;    7006  2
;    7007  2             case .ERRNUM from 50 to 54 of
;    7008  2                 set
;    7009  2
;    7010  2                 [50]:       ERRDF (50, 0, 0);               ! CONTROLLER ERROR
;    7011  2
;    7012  2                 [51]:       ERRDF (51, 0, 0);               ! HOST MEMORY ACCESS ERROR
;    7013  2
;    7014  2                 [52]:       ERRDF (52, 0, 0);               ! DISK TRANSFER ERROR
;    7015  2
;    7016  2                 [53]:       ERRDF (53, 0, 0);               ! SDI ERROR
;    7017  2
;    7018  2                 [54]:       ERRDF (54, 0, 0);               ! SMALL DISK ERROR
;    7019  2                 tes
;    7020  2         else
;    7021  3             begin
;    7022  3             !****increment error count                      ! INCREMENT TOTAL ERROR COUNT
;    7023  3             PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);          ! PRINT ERROR LINE JUST LIKE DRS
;    7024  2             end;
;    7025  2
;    7026  1         end;


                                        .SBTTL  ERR.SOFT.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
034614                                  .PSECT  $CODE$,  RO

000000  016646  000002              ERR.SOFT.RTNE.APT:
```

```
                                        MOV     2(SP),-(SP)             ; INDEX,*                       6995
000004  012746  000102                  MOV     #102,-(SP)
000010  004737  000000G                 JSR     PC,BL$MUL
000014  062700  000000G                 ADD     #ELOG.PKT,R0
000020  032737  000001  001254'         BIT     #1,APT.MODE             ;                               6997
000026  001406                          BEQ     1$
000030  016077  000056  001256'         MOV     56(R0),@MAIL.BOX.TESTNUM ; *(ELOG.ADDR),*               7000
000036  016077  000012  001260'         MOV     12(R0),@MAIL.BOX.SUBTST ; *(ELOG.ADDR),*                7001
000044  032737  020000  000000G 1$:     BIT     #20000,SWP.FLAGS        ;                               7004
000052  001440                          BEQ     8$
000054  016600  000010                  MOV     10(SP),R0               ; ERRNUM,*                      7007
000060  162700  000062                  SUB     #62,R0
000064  006300                          ASL     R0
000066  066007  000222'                 ADD     P.AAF(R0),PC            ; Case dispatch
000072  104455              3$:         TRAP    55                      ;                               7010
000074  000062                          .WORD   62
000076  000000                          .WORD   0
000100  000000                          .WORD   0
000102  000441                          BR      9$                      ;                               7007
000104  104455              4$:         TRAP    55                      ;                               7012
000106  000063                          .WORD   63
000110  000000                          .WORD   0
000112  000000                          .WORD   0
000114  000434                          BR      9$                      ;                               7007
000116  104455              5$:         TRAP    55                      ;                               7014
000120  000064                          .WORD   64
000122  000000                          .WORD   0
000124  000000                          .WORD   0
000126  000427                          BR      9$                      ;                               7007
000130  104455              6$:         TRAP    55                      ;                               7016
000132  000065                          .WORD   65
000134  000000                          .WORD   0
000136  000000                          .WORD   0
000140  000422                          BR      9$                      ;                               7007
000142  104455              7$:         TRAP    55                      ;                               7018
000144  000066                          .WORD   66
000146  000000                          .WORD   0
000150  000000                          .WORD   0
000152  000415                          BR      9$                      ;                               7004
000154  010716              8$:         MOV     PC,(SP)                 ; PC,*                          7023
000156  013746  000000G                 MOV     L$LUN,-(SP)
000162  016646  000012                  MOV     12(SP),-(SP)            ; ERRNUM,*
000166  012746  000000G                 MOV     #DF.MSG,-(SP)
000172  012746  000004                  MOV     #4,-(SP)
000176  010600                          MOV     SP,R0                   ; SP,*
000200  104414                          TRAP    14
000202  062706  000010                  ADD     #10,SP                  ;                               7021
000206  022626              9$:         CMP     (SP)+,(SP)+             ;                               6987
000210  000207                          RTS     PC                      ;                               6980

; Routine Size:  69 words,      Routine Base:  $CODE$ + 34614
; Maximum stack depth per invocation:  8 words
```

```
000222                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAF:
000222  000000                  2$:     .WORD   0                       ; CASE Table for ERR.SOFT.RTNE.A+0066   7007
000224  000012                          .WORD   12                      ; [3$]
000226  000024                          .WORD   24                      ; [4$]
000230  000036                          .WORD   36                      ; [5$]
000232  000050                          .WORD   50                      ; [7$]


;    7027  1
;    7028  1
;    7029  1        end
;    7030  1
;    7031  0        eludom


;                               OTS external references
                                        .GLOBL  $SAVE5, $SAVE4, $SAVE3, $SAVE2
                                        .GLOBL  BL$SHF, BL$DIV, BL$MOD, BL$MUL


;                               PSECT SUMMARY
;
;         Psect Name            Words     Attributes
;          $GGG$                  355       RO , I , LCL, REL, CON
;          $CODE$                7435       RO , I , LCL, REL, CON
;          $PLIT$                  78       RO , D , LCL, REL, CON


;                                       Library Statistics
;
;                                       -------- Symbols --------      Pages       Processing
;         File                          Total    Loaded   Percent      Mapped      Time
;
;    DISK$USER2:[POWERS.ZRQ]ZRQAG0.L16;10    407      337        82        21        00:00.1


;                                               COMMAND QUALIFIERS

;         BLISS/PDP11 ZRQAG0.BL2/LIST=ZRQAG0.LS2/OBJECT=ZRQAG0.OB2/SOURCE=PAGE:53
```

```
;    7032  0     module ZRQAM4 (
;    7033  0
;    7034  0     %title 'RD/RX EXERCISER'
;    7035  0                  ident = 'V01.9',
;    7036  0                  addressing_mode (absolute),
;    7037  0                  environment (noeis)
;    7038  0                  ) =
;    7039  0
;    7040  1     begin
;    7041  1
;    7042  1     %sbttl 'LASTAD AND SETUP'
;    7043  1
;    7044  1     library 'ZRQAG0.L16';
;    7045  1
;    7046  1     !ZZZ require 'BLSMAC.REQ';                    ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    7047  1     require 'HSAXA0.BLB';                         ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    8788  1
;    8789  2     LASTAD
;    8790  2
;    8791  2     BGNSETUP (4)                                                                       !ZZZ
;    8792  2
; P  8793  2             BGNPTAB
; P  8794  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000020', 0, 0, RD52_MAX_LBN, 0  !ZZZ
; P  8795  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
; P  8796  2
;    8797  2             ENDPTAB
;    8798  2
; P  8799  2             BGNPTAB
; P  8800  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000001', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  8801  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;    8802  2             ENDPTAB
;    8803  2
; P  8804  2             BGNPTAB
; P  8805  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000002', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  8806  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;    8807  2             ENDPTAB
; P  8808  2             BGNPTAB
; P  8809  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000003', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  8810  2                             !HERE'S ONE FOR THE 4TH DRIVE                                          !ZZZ
;    8811  2             ENDPTAB                                                                                !ZZZ
;    8812  2
;    8813  1     ENDSETUP


                                        .TITLE  ZRQAM4 RD/RX EXERCISER
                                        .IDENT  /V01.9/
                                        .ENABL  AMA


000000                                  .PSECT  $XYZ$,  RO
000000  000124'                 BL$LAS::.WORD   T$FREE
000002  000000C                         .WORD   <<T$FREE-<BL$LAS+4>>/2>
000004  000034'                 P.AAA:  .WORD   L$LAST+30
```

```
000006  000010                          .WORD   10                      ; Plit count word
000010  172150          P.AAB:  .WORD   -5630
000012  000154                  .WORD   154
000014  000004                  .WORD   4
000016  000020                  .WORD   20
000020  000000                  .WORD   0
000022  000000                  .WORD   0
000024  150477                  .WORD   -27301
000026  000000                  .WORD   0
000030  000060'         P.AAC:  .WORD   L$LAST+54
000032  000010                  .WORD   10                      ; Plit count word
000034  172150          P.AAD:  .WORD   -5630
000036  000154                  .WORD   154
000040  000004                  .WORD   4
000042  000001                  .WORD   1
000044  000000                  .WORD   0
000046  000000                  .WORD   0
000050  001437                  .WORD   1437
000052  000000                  .WORD   0
000054  000104'         P.AAE:  .WORD   L$LAST+100
000056  000010                  .WORD   10                      ; Plit count word
000060  172150          P.AAF:  .WORD   -5630
000062  000154                  .WORD   154
000064  000004                  .WORD   4
000066  000002                  .WORD   2
000070  000000                  .WORD   0
000072  000000                  .WORD   0
000074  001437                  .WORD   1437
000076  000000                  .WORD   0
000100  000000          P.AAG:  .WORD   0
000102  000010                  .WORD   10                      ; Plit count word
000104  172150          P.AAH:  .WORD   -5630
000106  000154                  .WORD   154
000110  000004                  .WORD   4
000112  000003                  .WORD   3
000114  000000                  .WORD   0
000116  000000                  .WORD   0
000120  001437                  .WORD   1437
000122  000000                  .WORD   0
000124  000000          T$FREE::.WORD   0


        000004'         L$LAST=         BL$LAS+4
        000004          T$PTHV=         4
        00000'          $LAS5=          P.AAA
        000010'         $REM5=          P.AAB
        000030'         $LAS4=          P.AAC
        000034'         $REM4=          P.AAD
        000054'         $LAS3=          P.AAE
        000060'         $REM3=          P.AAF
        000100'         $$LAS1=         P.AAG
        000104'         $REM2=          P.AAH
```

```
                                        .SBTTL  $END.LINK LASTAD AND SETUP
000000  000207                  $END.LINK::
                                        RTS     PC                                      ;                                               8787

; Routine Size:  1 word,        Routine Base:  $XYZ$ + 0126
; Maximum stack depth per invocation:  0 words


;   8814  1     end
;   8815  1
;   8816  0     eludom



;                                       PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;        $XYZ$                            44       RO , I  , LCL,  REL,  CON



;                               Library Statistics
;
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[POWERS.ZRQ]ZRQAG0.L16;10   407        7         1         21          00:00.1



;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAG0.BL2/LIST=ZRQAG0.LS2/OBJECT=ZRQAG0.OB2/SOURCE=PAGE:53

; Size:          7436 code + 476 data words
; Run Time:        03:43.6
; Elapsed Time: 01:06:55.4
; Lines/CPU Min:    2365
; Lexemes/CPU-Min: 21165
; Memory Used:  551 pages
; Compilation Complete
```

```
Partition name : DUMMY
Identification : V02.2
Task  UIC      : [202,24]
Task attributes: -HD
Total address windows: 1.
Task  image  size  : 17952. words
Task address limits: 002000 110063
R-W disk blk limits: 000002 000110 000107 00071.

*** Root segment: ZRQAGO


R/W mem  limits: 002000 110063 106064 35892.
Disk blk limits: 000002 000110 000107 00071.


Memory allocation synopsis:

Section                                                  Title  Ident  File
-------                                                  -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON)      002000 000000 00000.
$CODE$:(RO,I,LCL,REL,CON)      002000 075676 31678.
                               002000 023224 09876.  ZRQAM1 V02.2  ZRQAGO.OB1;10
                               025224 015000 06656.  ZRQAM2 V02.2  ZRQAGO.OB1;10
                               042224 035026 14870.  ZRQAM3 V02.2  ZRQAGO.OB2;15
                               077252 000316 00206.  B16MUL 2.8    NOEIS.OLB;1
                               077570 000106 00070.  B16SAV 2.4    NOEIS.OLB;1
$FFF$ :(RW,D,GBL,REL,CON)      077676 006020 03088.
                               077676 006020 03088.  ZRQAM1 V02.2  ZRQAGO.OB1;10
$GGG$ :(RO,I,LCL,REL,CON)      105716 001306 00710.
                               105716 001306 00710.  ZRQAM3 V02.2  ZRQAGO.OB2;15
$OWN$ :(RW,D,LCL,REL,CON)      107224 000224 00148.
                               107224 000224 00148.  ZRQAM2 V02.2  ZRQAGO.OB1;10
$PLIT$:(RO,D,LCL,REL,CON)      107450 000264 00180.
                               107450 000030 00024.  ZRQAM2 V02.2  ZRQAGO.OB1;10
                               107500 000234 00156.  ZRQAM3 V02.2  ZRQAGO.OB2;15
$XYZ$ :(RO,I,LCL,REL,CON)      107734 000130 00088.
                               107734 000130 00088.  ZRQAM4 V01.9  ZRQAGO.OB2;15

Global symbols:

ADDR.V 105715-R  BIT06  000100    BIT3   000010    BOE    000400    CMD.TI 105642-R  DASH   025066-R  DBM121 007116-R
ADR    000020    BIT07  000200    BIT4   000020    BRLEVE 105704-R  CNTR.E 023772-R  DATAGM 073352-R  DBM15  005244-R
ASTERI 025074-R  BIT08  000400    BIT5   000040    BST    100056-R  CREDIT 105656-R  DBM101 006266-R  DBM18  005274-R
AZINT  067736-R  BIT09  001000    BIT6   000100    BUFF.A 105536-R  CRLF   025062-R  DBM104 006314-R  DBM19  005346-R
AZINTO 067720-R  BIT1   000002    BIT7   000200    BUFF.O 105556-R  CRN.HI 105650-R  DBM105 006356-R  DBM20  005432-R
BIT0   000001    BIT10  002000    BIT8   000400    BYTS.P 105626-R  CRN.LO 105646-R  DBM107 006414-R  DBM21  005506-R
BIT00  000001    BIT11  004000    BIT9   001000    CCTLR  105606-R  CST    077676-R  DBM108 006452-R  DBM22  005570-R
BIT01  000002    BIT12  010000    BL$DIV 077476-R  CDISK  105610-R  CST.AD 100024-R  DBM109 006542-R  DBM23  005634-R
BIT02  000004    BIT13  020000    BL$LAS 107734-R  CER.01 016736-R  CTLR.C 105614-R  DBM111 006622-R  DBM25  005672-R
BIT03  000010    BIT14  040000    BL$MOD 077510-R  CER.02 017002-R  CTLR.I 044050-R  DBM112 006722-R  DBM26  005740-R
BIT04  000020    BIT15  100000    BL$MUL 077252-R  CLK.PR 105672-R  CUOFF  105612-R  DBM12  005170-R  DBM27  005772-R
BIT05  000040    BIT2   000004    BL$SHF 077522-R  CLK.TI 105664-R  C.ERR. 101500-R  DBM120 007024-R  DBM28A 006044-R
```

```
DBM28B 006104-R  EGD.22 012234-R  EX.CBC 015664-R  GP$28  025644-R  IO.RET 063652-R  L$REPP 002062-R  PUT.RE 033746-R
DBM29  006144-R  EGD.23 012274-R  EX.CBR 015730-R  GP$29  025660-R  IPKT.A 103212-R  L$REV  002010-R  P.INDE 105700-R
DBM32  006212-R  EGD.24 012336-R  EX.CBW 016002-R  GP$3   025310-R  IRDRX. 100054-R  L$RPT  027042-R  QIO    105622-R
DBM5   005142-R  EGM.30 012402-R  EX.CMD 015014-R  GP$30  025670-R  ISR    000100    L$SFTL 025462-R  QIO.FU 055606-R
DCT    100026-R  EGS.01 011366-R  EX.CMP 015054-R  GP$31  025704-R  IXE    004000    L$SOFT 025464-R  QIO.GE 053214-R
DCT.AD 100050-R  EGS.02 011406-R  EX.LBN 015374-R  GP$32  025722-R  LOE    040000    L$SPC  002056-R  QIO.LB 061544-R
DFPTBL 025104-R  EH.0   013060-R  EX.LBR 015470-R  GP$33  025734-R  LOT    000010    L$SPCP 002020-R  QIO.OK 053046-R
DF.MSG 024530-R  EH.1   013116-R  EX.LBW 015536-R  GP$4   025322-R  L$ACP  002110-R  L$SPTP 002024-R  QIO.OU 053142-R
DIO.RE 063246-R  EH.10  013472-R  EX.ONL 015070-R  GP$5   025334-R  L$APT  002036-R  L$STA  002030-R  QIO.SI 062410-R
DISK.R 071240-R  EH.12  013522-R  EX.OP  015114-R  GP$6   025344-R  L$AU   032614-R  L$SW   025134-R  QIO.UN 054202-R
DRIVER 043516-R  EH.13  013556-R  EX.PBN 015432-R  GP$7   025354-R  L$AUT  002070-R  L$SWLE 025132-R  RANDOM 101440-R
DROP.C 034266-R  EH.2   013154-R  EX.RBN 015604-R  GP$8   025364-R  L$AUTO 031622-R  L$TEST 002114-R  RANDY  053724-R
DRV.CT 034374-R  EH.3   013214-R  EX.RD  013034-R  GP$9   025376-R  L$CCP  002106-R  L$TIML 002014-R  RDM.CN 101436-R
DR.ERR 051232-R  EH.4   013252-R  EX.RP  016366-R  HARD.E 064246-R  L$CLEA 032116-R  L$UNIT 002012-R  RDRX.A 100052-R
DR.RET 067604-R  EH.5   013276-R  EX.SA  014720-R  HARD.I 045514-R  L$CO   002032-R  MD.INI 052502-R  RDRX.E 024512-R
DUP    056466-R  EH.6   013326-R  EX.SB  014772-R  HOE    100000    L$DEPO 002011-R  MINUTE 105662-R  RD.COU 105702-R
DUPCOM 061152-R  EH.7   013356-R  EX.SBO 014766-R  HOE.FL 105673-R  L$DESC 025244-R  MODULA 035126-R  REG.EX 044476-R
DUPIDL 061320-R  EH.A   013406-R  EX.SC  014736-R  HOST.W 067000-R  L$DESP 002076-R  MSCP.P 101502-R  RETPKT 103230-R
DUPPKT 100430-R  EH.9   013442-R  EX.TIM 016464-R  HOURS  105661-R  L$DEVP 002060-R  MSG.01 007754-R  ROUND. 066434-R
DUPRED 060110-R  ELG.FM 014706-R  EX.WRD 016454-R  HRD.MS 024626-R  L$DISP 002124-R  MSG.02 010006-R  RPS.RE 067416-R
DUPROU 025150-R  ELG.00 014354-R  EX.WRT 015044-R  HRD.SU 025024-R  L$DLY  002116-R  MSG.03 010042-R  RPT1   010074-R
DUPWRT 057032-R  ELOG.P 104004-R  FATAL. 070430-R  HWPTE0 025120-R  L$DTP  002040-R  MULTI. 052026-R  RPT10  010604-R
DUP.CO 063554-R  EMS.BL 037022-R  FER.BC 105712-R  HWPTE1 025122-R  L$DTYP 002034-R  NAME.H 025126-R  RPT11  010672-R
DUP.FL 105604-R  EMS.CM 041052-R  FER.LB 105710-R  HWPTS0 025114-R  L$DU   032522-R  NAME.L 025124-R  RPT12  010740-R
DUP.RS 070136-R  EMS.DB 036624-R  FERO.L 105666-R  HWPTS1 025116-R  L$DUT  002072-R  NEX    105644-R  RPT13  011006-R
DUR    105616-R  EMS.EL 037730-R  FER1.L 105670-R  HWPT.B 025110-R  L$DVTY 025224-R  NEXT.P 105660-R  RPT14  011106-R
DU.MSG 007206-R  EMS.ER 041400-R  FILL.B 062620-R  HWPT.D 025112-R  L$EF   002052-R  NEX.TR 032624-R  RPT15  011204-R
DU.RSN 007726-R  EMS.RP 037672-R  FORCED 105707-R  HWPT.I 025104-R  L$ENVI 002044-R  NULL   005140-R  RPT16  011304-R
D$PCNT 002122-R  EMS.01 041604-R  FREE.M 105624-R  HWPT.V 025106-R  L$ERRT 002126-R  OFF    000002    RPT2   010160-R
D.FAIL 105706-R  EMS.10 041642-R  FSET.U 064122-R  HWQ1   002460-R  L$ETP  002102-R  OF.RC  105636-R  RPT3   010224-R
EBD.10 012470-R  EMS.12 041704-R  GET.IO 033762-R  HWQ10  003260-R  L$EXP1 002046-R  ON     000001    RPT4   010310-R
EBD.12 012530-R  EMS.13 041742-R  GET.PK 033112-R  HWQ11  003310-R  L$EXP4 002064-R  OUT.IO 034142-R  RPT5   010354-R
EBD.13 012576-R  EMS.14 042004-R  GET.RA 053614-R  HWQ2   002474-R  L$EXP5 002066-R  OVF.CH 066360-R  RPT6   010442-R
EBD.14 012630-R  EMS.18 042046-R  GET.RE 033640-R  HWQ3   002504-R  L$HARD 025270-R  PKT.US 103214-R  RPT7   010506-R
EBD.18 012670-R  EMS.21 042110-R  GP$DIS 025640-R  HWQ4   002546-R  L$HIME 002120-R  PNT    001000    RPT8   010524-R
EBD.19 012724-R  EMS.22 042126-R  GP$1   025270-R  HWQ5   002564-R  L$HPCP 002016-R  POLL.C 070662-R  RPT9   010552-R
EBD.24 013004-R  EMS.24 042144-R  GP$10  025410-R  HWQ6A  002634-R  L$HPTP 002022-R  POLL.R 070762-R  RP ADD 104002-R
EBS.01 012426-R  EMS.30 042206-R  GP$11  025424-R  HWQ6B  002706-R  L$HRDL 025266-R  PRI    002000    RP.IND 104000-R
EF.CON 000036    ENTRY. 105602-R  GP$12  025440-R  HWQ7A  002762-R  L$HW   025104-R  PRIO0  000000    RP.USE 103770-R
EF.NEW 000035    EOP.FL 105603-R  GP$13  025450-R  HWQ7B  003032-R  L$HWLE 025102-R  PRIO1  000040    SA.REG 105640-R
EF.PWR 000034    ERRBLK 002134-R  GP$14  025464-R  HWQ8   003102-R  L$ICP  002104-R  PRIO2  000100    SB.COD 105632-R
EF.RES 000037    ERRMSG 002132-R  GP$15  025476-R  HWQ9   003160-R  L$INIT 031610-R  PRIO3  000140    SCAN.E 072504-R
EF.STA 000040    ERRNBR 002130-R  GP$16  025510-R  IBE    010000    L$LADP 002026-R  PRIO4  000200    SC.CLK 022070-R
EGD.10 011500-R  ERRTYP 002126-R  GP$17  025522-R  IDU    000040    L$LAST 107740-R  PRIO5  000240    SC.CON 017102-R
EGD.11 011540-R  ERR.CO 014320-R  GP$18  025534-R  IER    020000    L$LOAD 002100-R  PRIO6  000300    SC.CTO 021332-R
EGD.12 011564-R  ERR.00 013630-R  GP$19  025542-R  INIT.I 052642-R  L$LUN  002074-R  PRIO7  000340    SC.DIS 017430-R
EGD.13 011612-R  EVL    000004    GP$2   025300-R  INIT.O 105714-R  L$MREV 002050-R  PROC.R 063052-R  SC.DST 020040-R
EGD.14 011640-R  EX.ACC 015102-R  GP$20  025550-R  INIT.T 043360-R  L$NAME 002000-R  PTCH1  002136-R  SC.DS2 020114-R
EGD.15 011670-R  EX.BB  015120-R  GP$21  025556-R  INI.CT 044352-R  L$NDHR 025460-R  PTCH2  002210-R  SC.DUP 017124-R
EGD.16 011706-R  EX.BBU 015304-R  GP$22  025564-R  INI.RR 046364-R  L$NDHW 025130-R  PTCH3  002262-R  SC.ECC 020166-R
EGD.17 011736-R  EX.BB1 015210-R  GP$23  025572-R  INT.GE 045234-R  L$NDSF 025752-R  PTCH4  002334-R  SC.ECD 020250-R
EGD.18 011754-R  EX.BC  016054-R  GP$24  025600-R  IN.IOD 034200-R  L$NDSW 025214-R  PTCH5  002406-R  SC.EC1 020520-R
EGD.19 011774-R  EX.BD  016130-R  GP$25  025614-R  IODQ   105566-R  L$PRIO 002042-R  PUTA.B 034102-R  SC.EC2 020550-R
EGD.20 012034-R  EX.BDR 016206-R  GP$26  025624-R  IODQ.I 105576-R  L$PROT 025216-R  PUT.IO 034036-R  SC.EC3 020600-R
EGD.21 012122-R  EX.BDW 016276-R  GP$27  025632-R  IODQ.O 105600-R  L$PRT  002112-R  PUT.PK 033500-R  SC.EC4 020632-R
```

```
SC.EC5 020662-R  SC.NXM 021242-R  SC.SUR 022164-R  ST.COD 105630-R  SWQ13  004010-R  SWQ9   003540-R  WAIT   035110-R
SC.EC6 020712-R  SC.ODA 021170-R  SC.SWP 021070-R  SWEEP  067264-R  SWQ14  004066-R  S.DUPP 105676-R  XX13   016524-R
SC.EC7 020742-R  SC.ODB 021220-R  SC.UNK 017216-R  SWM1   005042-R  SWQ15  004140-R  S.PATT 105674-R  XX23   016546-R
SC.EC8 020774-R  SC.ONL 017154-R  SC.VOL 017276-R  SWP.DP 025142-R  SWQ17  004210-R  TALLY  100076-R  XX32   016602-R
SC.EC9 021026-R  SC.PAR 021276-R  SC.576 020376-R  SWP.ER 025134-R  SWQ19  004306-R  TEMP1  105652-R  XX33   016630-R
SC.EDC 021462-R  SC.POE 021752-R  SEND   034452-R  SWP.FL 025140-R  SWQ2   003364-R  TEMP2  105654-R  XX34   016666-R
SC.FCT 020452-R  SC.PSP 022214-R  SEQUEN 071340-R  SWP.RA 025144-R  SWQ20  004376-R  TIME   032634-R  $END.L 110062-R
SC.FER 017522-R  SC.RCT 020302-R  SET.CP 032716-R  SWP.TI 025146-R  SWQ21  004464-R  TRK.SG 101432-R  $SAVE2 077570-R
SC.FE2 017610-R  SC.RDY 022006-R  SET.CT 046466-R  SWP.UC 025152-R  SWQ22  004550-R  T$FREE 110060-R  $SAVE3 077604-R
SC.FUL 020322-R  SC.RSP 022116-R  SET.UP 033012-R  SWP.UD 025154-R  SWQ23  004610-R  T$PTHV 000004     $SAVE4 077622-R
SC.HWP 021130-R  SC.SDI 017056-R  SFPTBL 025134-R  SWP.XF 025136-R  SWQ24  004662-R  T.ADDR 100426-R  $SAVE5 077642-R
SC.IDS 021522-R  SC.SDS 021404-R  SFT.MS 024724-R  SWQ1   003342-R  SWQ25  004726-R  T1     043344-R
SC.IOP 017356-R  SC.SON 017176-R  SOFT.E 074276-R  SWQ10  003614-R  SWQ26  005000-R  UAM    000200
SC.ISH 017700-R  SC.SRI 021664-R  SPACE4 025056-R  SWQ11  003660-R  SWQ4   003444-R  UPD.IO 066040-R
SC.IS2 017760-R  SC.SRT 021572-R  STEP   105634-R  SWQ12  003712-R  SWQ7   003466-R  VEC.BR 044710-R


*** Task builder statistics:

    Total work file references: 156336.
    Work  file  reads: 0.
    Work  file writes: 0.
    Size of core pool: 23454. words (91. pages)
    Size of work file: 5120. words (20. pages)

    Elapsed time:00:02:59
```

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
ADDR.V	105715-R	• ZRQAM1	ZRQAM2	ZRQAM3
ADR	000020	• ZRQAM1	• ZRQAM2	• ZRQAM3
ASTERI	025074-R	• ZRQAM1	ZRQAM2	
AZINT	067736-R	• ZRQAM3		
AZINTO	067720-R	• ZRQAM3		
BIT0	000001	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT00	000001	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT01	000002	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT02	000004	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT03	000010	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT04	000020	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT05	000040	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT06	000100	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT07	000200	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT08	000400	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT09	001000	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT1	000002	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT10	002000	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT11	004000	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT12	010000	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT13	020000	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT14	040000	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT15	100000	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT2	000004	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT3	000010	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT4	000020	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT5	000040	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT6	000100	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT7	000200	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT8	000400	• ZRQAM1	• ZRQAM2	• ZRQAM3
BIT9	001000	• ZRQAM1	• ZRQAM2	• ZRQAM3
BL$DIV	077476-R	• B16MUL	ZRQAM2	ZRQAM3
BL$LAS	107734-R	• ZRQAM4		
BL$MOD	077510-R	• B16MUL	ZRQAM2	ZRQAM3
BL$MUL	077252-R	• B16MUL	ZRQAM2	ZRQAM3
BL$SHF	077522-R	• B16MUL	ZRQAM3	
BOE	000400	• ZRQAM1	• ZRQAM2	• ZRQAM3
B$LEVE	105704-R	• ZRQAM1	ZRQAM2	ZRQAM3
BST	100056-R	• ZRQAM1	ZRQAM2	ZRQAM3
BUFF.A	105536-R	• ZRQAM1	ZRQAM2	ZRQAM3
BUFF.O	105556-R	• ZRQAM1	ZRQAM2	ZRQAM3
BYTS.P	105626-R	• ZRQAM1	ZRQAM2	ZRQAM3
CCTLR	105606-R	• ZRQAM1	ZRQAM2	ZRQAM3
CDISK	105610-R	• ZRQAM1	ZRQAM2	ZRQAM3
CER.01	016736-R	• ZRQAM1	ZRQAM2	
CER.02	017002-R	• ZRQAM1	ZRQAM2	
CLK.PR	105672-R	• ZRQAM1	ZRQAM2	ZRQAM3
CLK.TI	105664-R	• ZRQAM1	ZRQAM2	ZRQAM3
CMD.TI	105642-R	• ZRQAM1	ZRQAM2	ZRQAM3
CNTR.E	023772-R	• ZRQAM1	ZRQAM2	
CREDIT	105656 R	• ZRQAM1	ZRQAM2	ZRQAM3
CRLF	025062 R	• ZRQAM1	ZRQAM2	ZRQAM3

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
CRN.HI	105650 R	# ZRQAM1	ZRQAM2	ZRQAM3
CRN.LO	105646-R	# ZRQAM1	ZRQAM2	ZRQAM3
CST	077676-R	# ZRQAM1	ZRQAM2	ZRQAM3
CST.AD	100024-R	# ZRQAM1	ZRQAM2	ZRQAM3
CTLR.C	105614-R	# ZRQAM1	ZRQAM2	ZRQAM3
CTLR.I	044050-R	# ZRQAM3		
CUOFF	105612-R	# ZRQAM1	ZRQAM2	ZRQAM3
C.ERR.	101500-R	# ZRQAM1	ZRQAM2	ZRQAM3
DASM	025066-R	# ZRQAM1	ZRQAM2	
DATAGM	073352-R	# ZRQAM3		
DBM101	006766-R	# ZRQAM1		
DBM104	006314-R	# ZRQAM1		
DBM105	006356-R	# ZRQAM1		
DBM107	006414-R	# ZRQAM1	ZRQAM2	
DBM108	006452-R	# ZRQAM1	ZRQAM3	
DBM109	006542-R	# ZRQAM1	ZRQAM3	
DBM111	006622-R	# ZRQAM1	ZRQAM3	
DBM112	006722-R	# ZRQAM1	ZRQAM3	
DBM12	005170-R	# ZRQAM1	ZRQAM3	
DBM120	007024-R	# ZRQAM1	ZRQAM3	
DBM121	007116-R	# ZRQAM1	ZRQAM3	
DBM15	005244-R	# ZRQAM1		
DBM18	005274-R	# ZRQAM1	ZRQAM3	
DBM19	005346-R	# ZRQAM1	ZRQAM3	
DBM20	005432-R	# ZRQAM1	ZRQAM3	
DBM21	005506-R	# ZRQAM1	ZRQAM3	
DBM22	005570-R	# ZRQAM1	ZRQAM3	
DBM23	005634-R	# ZRQAM1	ZRQAM3	
DBM25	005672-R	# ZRQAM1	ZRQAM3	
DBM26	005740-R	# ZRQAM1	ZRQAM3	
DBM27	005772-R	# ZRQAM1	ZRQAM3	
DBM28A	006044-R	# ZRQAM1		
DBM28B	006104-R	# ZRQAM1		
DBM29	006144-R	# ZRQAM1	ZRQAM3	
DBM32	006212-R	# ZRQAM1		
DBM5	005142-R	# ZRQAM1	ZRQAM2	
DCT	100026-R	# ZRQAM1	ZRQAM2	ZRQAM3
DCT.AD	100050-R	# ZRQAM1	ZRQAM2	ZRQAM3
DFPTBL	025104-R	# ZRQAM1		
DF.MSG	024530-R	# ZRQAM1	ZRQAM3	
DIO.RE	063246-R	# ZRQAM3		
DISK.R	071240-R	# ZRQAM3		
DRIVER	043516-R	# ZRQAM3		
DROP.C	034266-R	# ZRQAM2	ZRQAM3	
DRV.CT	034374-R	# ZRQAM2	ZRQAM3	
DR.ERR	051232-R	# ZRQAM3		
DR.RET	067604-R	# ZRQAM3		
DUP	056466-R	# ZRQAM3		
DUPCOM	061152-R	# ZRQAM3		
DUPIDL	061320-R	# ZRQAM3		
DUPPKT	100430-R	# ZRQAM1	ZRQAM2	ZRQAM3
DUPRED	060110 R	# ZRQAM3		

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
DUPROU	025150 R	# ZRQAM1	ZRQAM3	
DUPWRT	057032-R	# ZRQAM3		
DUP.CO	063554-R	# ZRQAM3		
DUP.FL	105604-R	# ZRQAM1	ZRQAM2	ZRQAM3
DUP.RS	070136-R	# ZRQAM3		
DUR	105616-R	# ZRQAM1	ZRQAM2	ZRQAM3
DU.MSG	007206-R	# ZRQAM1	ZRQAM2	
DU.RSN	007726-R	# ZRQAM1	ZRQAM2	
D#PCNT	002122-R	# ZRQAM1		
D.FAIL	105706-R	# ZRQAM1	ZRQAM2	ZRQAM3
EBD.10	012470-R	# ZRQAM1	ZRQAM2	
EBD.12	012530-R	# ZRQAM1	ZRQAM2	
EBD.13	012576-R	# ZRQAM1	ZRQAM2	
EBD.14	012630-R	# ZRQAM1	ZRQAM2	
EBD.18	012670-R	# ZRQAM1	ZRQAM2	
EBD.19	012724-R	# ZRQAM1	ZRQAM2	
EBD.24	013004-R	# ZRQAM1	ZRQAM2	
EBS.01	012426-R	# ZRQAM1	ZRQAM2	
EF.CON	000036	# ZRQAM1	# ZRQAM2	# ZRQAM3
EF.NEW	000035	# ZRQAM1	# ZRQAM2	# ZRQAM3
EF.PWR	000034	# ZRQAM1	# ZRQAM2	# ZRQAM3
EF.RES	000037	# ZRQAM1	# ZRQAM2	# ZRQAM3
EF.STA	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
EGD.10	011500-R	# ZRQAM1	ZRQAM3	
EGD.11	011540-R	# ZRQAM1	ZRQAM3	
EGD.12	011564-R	# ZRQAM1	ZRQAM3	
EGD.13	011612-R	# ZRQAM1	ZRQAM3	
EGD.14	011640-R	# ZRQAM1	ZRQAM3	
EGD.15	011670-R	# ZRQAM1	ZRQAM3	
EGD.16	011706-R	# ZRQAM1	ZRQAM3	
EGD.17	011736-R	# ZRQAM1	ZRQAM3	
EGD.18	011754-R	# ZRQAM1	ZRQAM3	
EGD.19	011774-R	# ZRQAM1	ZRQAM3	
EGD.20	012034-R	# ZRQAM1	ZRQAM3	
EGD.21	012122-R	# ZRQAM1	ZRQAM3	
EGD.22	012234-R	# ZRQAM1	ZRQAM3	
EGD.23	012274-R	# ZRQAM1	ZRQAM3	
EGD.24	012336-R	# ZRQAM1	ZRQAM3	
EGH.30	012402-R	# ZRQAM1	ZRQAM3	
EGS.01	011366-R	# ZRQAM1	ZRQAM2	
EGS.02	011406-R	# ZRQAM1	ZRQAM3	
EH.0	013060-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.1	013116-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.10	013472-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.12	013522-R	# RQAM1	ZRQAM2	ZRQAM3
EH.13	013556-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.2	013154-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.3	013214-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.4	013252-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.5	013276-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.6	013326-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.7	013356-R	# ZRQAM1	ZRQAM2	ZRQAM3

ZRQAGO CREATED BY TKB ON 4-APR-85 AT 14:35 PAGE 4 SEQ 0504

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
EH.8	013406-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.9	013442-R	# ZRQAM1	ZRQAM2	ZRQAM3
ELG.FM	014706-R	# ZRQAM1	ZRQAM2	
ELG.OO	014354-R	# ZRQAM1	ZRQAM2	
ELOG.P	104004-R	# ZRQAM1	ZRQAM2	ZRQAM3
EMS.BL	037022-R	# ZRQAM2		
EMS.CM	041052-R	# ZRQAM2	ZRQAM3	
EMS.DB	036624-R	# ZRQAM2		
EMS.EL	037730-R	# ZRQAM2	ZRQAM3	
EMS.ER	041400-R	# ZRQAM2	ZRQAM3	
EMS.RP	037672-R	# ZRQAM2	ZRQAM3	
EMS.01	041604-R	# ZRQAM2		
EMS.10	041642-R	# ZRQAM2	ZRQAM3	
EMS.12	041704-R	# ZRQAM2	ZRQAM3	
EMS.13	041742-R	# ZRQAM2	ZRQAM3	
EMS.14	042004-R	# ZRQAM2	ZRQAM3	
EMS.18	042046-R	# ZRQAM2	ZRQAM3	
EMS.21	042110-R	# ZRQAM2	ZRQAM3	
EMS.22	042126-R	# ZRQAM2	ZRQAM3	
EMS.24	042144-R	# ZRQAM2	ZRQAM3	
EMS.30	042206-R	# ZRQAM2	ZRQAM3	
ENTRY.	105602-R	# ZRQAM1	ZRQAM2	ZRQAM3
EOP.FL	105603-R	# ZRQAM1	ZRQAM2	ZRQAM3
ERRBLK	002134-R	# ZRQAM1		
ERRMSG	002132-R	# ZRQAM1		
ERRNBR	002130-R	# ZRQAM1		
ERRTYP	002126-R	# ZRQAM1		
ERR.CO	014320-R	# ZRQAM1	ZRQAM2	
ERR.OO	013630-R	# ZRQAM1	ZRQAM2	
EVL	000004	# ZRQAM1	# ZRQAM2	# ZRQAM3
EX.ACC	015102-R	# ZRQAM1	ZRQAM2	
EX.BB	015120-P	# ZRQAM1	ZRQAM2	
EX.BBU	015304-R	# ZRQAM1	ZRQAM2	
EX.BB1	015210-R	# ZRQAM1	ZRQAM2	
EX.BC	016054-R	# ZRQAM1	ZRQAM2	
EX.BD	016130-R	# ZRQAM1	ZRQAM2	
EX.BDR	016206-R	# ZRQAM1	ZRQAM2	
EX.BDW	016276-R	# ZRQAM1	ZRQAM2	
EX.CBC	015664-R	# ZRQAM1	ZRQAM2	
EX.CBR	015730-R	# ZRQAM1	ZRQAM2	
EX.CBW	016002-R	# ZRQAM1	ZRQAM2	
EX.CMD	015014-R	# ZRQAM1	ZRQAM2	
EX.CMP	015054-R	# ZRQAM1	ZRQAM2	
EX.LBN	015374-R	# ZRQAM1	ZRQAM2	
EX.LBR	015470-R	# ZRQAM1	ZRQAM2	
EX.LBW	015536-R	# ZRQAM1	ZRQAM2	
EX.ONL	015070-R	# ZRQAM1	ZRQAM2	
EX.OP	015114-R	# ZRQAM1	ZRQAM2	
EX.PBN	015432-R	# ZRQAM1	ZRQAM2	
EX.RBN	015604-R	# ZRQAM1	ZRQAM2	
EX.RD	015034-R	# ZRQAM1	ZRQAM2	
EX.RP	016366-R	# ZRQAM1	ZRQAM2	

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
EX.SA	014720-R	# ZRQAM1	ZRQAM2	
EX.SB	014772-R	# ZRQAM1	ZRQAM2	
EX.SBO	014766-R	# ZRQAM1	ZRQAM2	
EX.SC	014736-R	# ZRQAM1	ZRQAM2	
EX.TIM	016464-R	# ZRQAM1	ZRQAM2	
EX.WRD	016454-R	# ZRQAM1	ZRQAM2	
EX.WRT	015044-R	# ZRQAM1	ZRQAM2	
FATAL.	070430-R	# ZRQAM3		
FER.BC	105712-R	# ZRQAM1	ZRQAM2	ZRQAM3
FER.LB	105710-R	# ZRQAM1	ZRQAM2	
FERO.L	105666-R	# ZRQAM1	ZRQAM2	ZRQAM3
FER1.L	105670-R	# ZRQAM1	ZRQAM2	ZRQAM3
FILL.B	062620-R	# ZRQAM3		
FORCED	105707-R	# ZRQAM1	ZRQAM2	ZRQAM3
FREE.M	105624-R	# ZRQAM1	ZRQAM2	ZRQAM3
FSET.U	064122-R	# ZRQAM3		
GET.IO	033762-R	# ZRQAM2	ZRQAM3	
GET.PK	033112-R	# ZRQAM2	ZRQAM3	
GET.RA	053614-R	# ZRQAM3		
GET.RE	033640-R	# ZRQAM2	ZRQAM3	
GP$DIS	025640-R	# ZRQAM2		
GP$1	025270-R	# ZRQAM2		
GP$10	025410-R	# ZRQAM2		
GP$11	025424-R	# ZRQAM2		
GP$12	025440-R	# ZRQAM2		
GP$13	025450-R	# ZRQAM2		
GP$14	025464-R	# ZRQAM2		
GP$15	025476-R	# ZRQAM2		
GP$16	025510-R	# ZRQAM2		
GP$17	025522-R	# ZRQAM2		
GP$18	025534-R	# ZRQAM2		
GP$19	025542-R	# ZRQAM2		
GP$2	025300-R	# ZRQAM2		
GP$20	025550-R	# ZRQAM2		
GP$21	025556-R	# ZRQAM2		
GP$22	025564-R	# ZRQAM2		
GP$23	025572-R	# ZRQAM2		
GP$24	025600-R	# ZRQAM2		
GP$25	025614-R	# ZRQAM2		
GP$26	025624-R	# ZRQAM2		
GP$27	025632-R	# ZRQAM2		
GP$28	025644-R	# ZRQAM2		
GP$29	025660-R	# ZRQAM2		
GP$3	025310-R	# ZRQAM2		
GP$30	025670-R	# ZRQAM2		
GP$31	025704-R	# ZRQAM2		
GP$32	025722-R	# ZRQAM2		
GP$33	025734-R	# ZRQAM2		
GP$4	025322-R	# ZRQAM2		
GP$5	025334-R	# ZRQAM2		
GP$6	025344-R	# ZRQAM2		
GP$7	025354-R	# ZRQAM2		

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...
GP$8	025364-R	# ZRQAM2
GP$9	025376-R	# ZRQAM2
HARD.E	064246-R	# ZRQAM3
HARD.I	045514-R	# ZRQAM3
HOE	100000	# ZRQAM1 # ZRQAM2 # ZRQAM3
HOE.FL	105673-R	# ZRQAM1 ZRQAM2 ZRQAM3
HOST.W	067000-R	# ZRQAM3
HOURS	105661-R	# ZRQAM1 ZRQAM2 ZRQAM3
HRD.MS	024626-R	# ZRQAM1 ZRQAM3
HRD.SU	025024-R	# ZRQAM1 ZRQAM3
HWPTE0	025120-R	# ZRQAM1
HWPTE1	025122-R	# ZRQAM1
HWPTS0	025114-R	# ZRQAM1
HWPTS1	025116-R	# ZRQAM1
HWPT.B	025110-R	# ZRQAM1
HWPT.D	025112-R	# ZRQAM1
HWPT.I	025104-R	# ZRQAM1
HWPT.V	025106-R	# ZRQAM1
HWQ1	002460-R	# ZRQAM1 ZRQAM2
HWQ10	003260-R	# ZRQAM1 ZRQAM2
HWQ11	003310-R	# ZRQAM1 ZRQAM2
HWQ2	002474-R	# ZRQAM1 ZRQAM2
HWQ3	002504-R	# ZRQAM1 ZRQAM2
HWQ4	002546-R	# ZRQAM1 ZRQAM2
HWQ5	002564-R	# ZRQAM1 ZRQAM2
HWQ6A	002634-R	# ZRQAM1 ZRQAM2
HWQ6B	002706-R	# ZRQAM1 ZRQAM2
HWQ7A	002762-R	# ZRQAM1 ZRQAM2
HWQ7B	003032-R	# ZRQAM1 ZRQAM2
HWQ8	003102-R	# ZRQAM1 ZRQAM2
HWQ9	003160-R	# ZRQAM1 ZRQAM2
IBE	010000	# ZRQAM1 # ZRQAM2 # ZRQAM3
IDU	000040	# ZRQAM1 # ZRQAM2 # ZRQAM3
IER	020000	# ZRQAM1 # ZRQAM2 # ZRQAM3
INIT.I	052642-R	# ZRQAM3
INIT.O	105714-R	# ZRQAM1 ZRQAM2 ZRQAM3
INIT.T	043360-R	# ZRQAM3
INI.CT	044352-R	# ZRQAM3
INI.RR	046364-R	# ZRQAM3
INT.GE	045234-R	# ZRQAM3
IN.IOD	034200-R	# ZRQAM2 ZRQAM3
IOQ	105566-R	# ZRQAM1 ZRQAM2 ZRQAM3
IOQ.I	105576-R	# ZRQAM1 ZRQAM2 ZRQAM3
IOQ.O	105600-R	# ZRQAM1 ZRQAM2 ZRQAM3
IO.RET	063652-R	# ZRQAM3
IPKT.A	103212-R	# ZRQAM1 ZRQAM2 ZRQAM3
IRDRX.	100054-R	# ZRQAM1 ZRQAM2 ZRQAM3
ISR	000100	# ZRQAM1 # ZRQAM2 # ZRQAM3
IXE	004000	# ZRQAM1 # ZRQAM2 # ZRQAM3
LOE	040000	# ZRQAM1 # ZRQAM2 # ZRQAM3
LOT	000010	# ZRQAM1 # ZRQAM2 # ZRQAM3
L$ACP	002110-R	# ZRQAM1

ZRQAGO CREATED BY TKB ON 4-APR-85 AT 14:35 PAGE 7 SEQ 0507

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
L$APT	002036-R	# ZRQAM1		
L$AU	032614-R	ZRQAM1	# ZRQAM2	
L$AUT	002070-R	# ZRQAM1		
L$AUTU	031622-R	ZRQAM1	# ZRQAM2	
L$CCP	002106-R	# ZRQAM1		
L$CLEA	032116-R	ZRQAM1	# ZRQAM2	
L$CO	002032-R	# ZRQAM1		
L$DEPO	002011-R	# ZRQAM1		
L$DESC	025244-R	ZRQAM1	# ZRQAM2	
L$DESP	002076-R	# ZRQAM1		
L$DEVP	002060-R	# ZRQAM1		
L$DISP	002124-R	# ZRQAM1		
L$DLY	002116-R	# ZRQAM1	ZRQAM2	ZRQAM3
L$DTP	002040-R	# ZRQAM1		
L$DTYP	002034-R	# ZRQAM1		
L$DU	032522-R	ZRQAM1	# ZRQAM2	
L$DUT	002072-R	# ZRQAM1		
L$DVTY	025224-R	ZRQAM1	# ZRQAM2	
L$EF	002052-R	# ZRQAM1		
L$ENVI	002044-R	# ZRQAM1		
L$ERRT	002126-R	# ZRQAM1		
L$ETP	002102-R	# ZRQAM1		
L$EXP1	002046-R	# ZRQAM1		
L$EXP4	002064-R	# ZRQAM1		
L$EXP5	002066-R	# ZRQAM1		
L$HARD	025270-R	ZRQAM1	# ZRQAM2	
L$HIME	002120-R	# ZRQAM1	ZRQAM2	
L$HPCP	002016-R	# ZRQAM1		
L$HPTP	002022-R	# ZRQAM1		
L$HRDL	025266-R	# ZRQAM2		
L$HW	025104-R	# ZRQAM1		
L$HWLE	025102-R	# ZRQAM1		
L$ICP	002104-R	# ZRQAM1		
L$INIT	031610-R	ZRQAM1	# ZRQAM2	
L$LADP	002026-R	# ZRQAM1		
L$LAST	107740-R	ZRQAM1	# ZRQAM4	
L$LOAD	002100-R	# ZRQAM1		
L$LUN	002074-R	# ZRQAM1	ZRQAM2	ZRQAM3
L$MREV	002050-R	# ZRQAM1		
L$NAME	002000-R	# ZRQAM1		
L$NDHR	025460-R	# ZRQAM2		
L$NDHW	025130-R	# ZRQAM1		
L$NDSF	025752-R	# ZRQAM2		
L$NDSW	025214-R	# ZRQAM1		
L$PRIO	002042-R	# ZRQAM1		
L$PROT	025216-R	# ZRQAM1		
L$PRT	002112-R	# ZRQAM1		
L$REPP	002062-R	# ZRQAM1		
L$REV	002010-R	# ZRQAM1		
L$RPT	027042-R	ZRQAM1	# ZRQAM2	
L$SFTL	025462-R	# ZRQAM2		
L$SOFT	025464-R	ZRQAM1	# ZRQAM2	

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
L$SPC	002056-R	# ZRQAM1		
L$SPCP	002020-R	# ZRQAM1		
L$SPTP	002024-R	# ZRQAM1		
L$STA	002030-R	# ZRQAM1		
L$SW	025134-R	# ZRQAM1		
L$SWLE	025132-R	# ZRQAM1		
L$TEST	002114-R	# ZRQAM1		
L$TIML	002014-R	# ZRQAM1		
L$UNIT	002012-R	# ZRQAM1	ZRQAM2	ZRQAM3
MD.INI	052502-R	# ZRQAM3		
MINUTE	105662-R	# ZRQAM1	ZRQAM2	ZRQAM3
MODULA	035126-R	# ZRQAM2	ZRQAM3	
MSCP.P	101502-R	# ZRQAM1	ZRQAM2	ZRQAM3
MSG.01	007754-R	# ZRQAM1	ZRQAM2	
MSG.02	010006-R	# ZRQAM1	ZRQAM3	
MSG.03	010042-R	# ZRQAM1	ZRQAM3	
MULTI.	052026-R	# ZRQAM3		
NAME.H	025126-R	# ZRQAM1		
NAME.L	025124-R	# ZRQAM1		
NEX	105644-R	# ZRQAM1	ZRQAM2	ZRQAM3
NEXT.P	105660-R	# ZRQAM1	ZRQAM2	ZRQAM3
NEX.TR	032624-R	# ZRQAM2	ZRQAM3	
NULL	005140-R	# ZRQAM1	ZRQAM2	
OFF	000002	# ZRQAM1	# ZRQAM2	# ZRQAM3
OF.RC	105636-R	# ZRQAM1	ZRQAM2	ZRQAM3
ON	000001	# ZRQAM1	# ZRQAM2	# ZRQAM3
OUT.IO	034142-R	# ZRQAM2	ZRQAM3	
OVF.CH	066360-R	# ZRQAM3		
PKT.US	103214-R	# ZRQAM1	ZRQAM2	ZRQAM3
PNT	001000	# ZRQAM1	# ZRQAM2	# ZRQAM3
POLL.C	070662-R	# ZRQAM3		
POLL.R	070762-R	# ZRQAM3		
PRI	002000	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI00	000000	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI01	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI02	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI03	000140	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI04	000200	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI05	000240	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI06	000300	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI07	000340	# ZRQAM1	# ZRQAM2	# ZRQAM3
PROC.R	063052-R	# ZRQAM3		
PTCH1	002136-R	# ZRQAM1		
PTCH2	002210-R	# ZRQAM1		
PTCH3	002262-R	# ZRQAM1		
PTCH4	002334-R	# ZRQAM1		
PTCH5	002406-R	# ZRQAM1		
PUTA.B	034102-R	# ZRQAM2	ZRQAM3	
PUT.IO	034036-R	# ZRQAM2	ZRQAM3	
PUT.PK	033500-R	# ZRQAM2	ZRQAM3	
PUT.RE	033746-R	# ZRQAM2	ZRQAM3	
P.INDE	105700-R	# ZRQAM1	ZRQAM2	ZRQAM3

ZRQAGO CREATED BY TKB ON 4 APR-85 AT 14:35 PAGE 9 SEQ 0509

GLOBAL CROSS REFERENCE CPEF V02

SYMBOL	VALUE	REFERENCES...		
QIO	105622-R	# ZRQAM1	ZRQAM2	ZRQAM3
QIO.FU	055606-R	# ZRQAM3		
QIO.GE	053214-R	# ZRQAM3		
QIO.LB	061544-R	# ZRQAM3		
QIO.OK	053046-R	# ZRQAM3		
QIO.OU	053142-R	# ZRQAM3		
QIO.SI	062410-R	# ZRQAM3		
QIO.UN	054202-R	# ZRQAM3		
RANDOM	101440-R	# ZRQAM1	ZRQAM2	ZRQAM3
RANDY	053724-R	# ZRQAM3		
RDM.CN	101436-R	# ZRQAM1	ZRQAM2	ZRQAM3
RDRX.A	100052-R	# ZRQAM1	ZRQAM2	ZRQAM3
RDRX.E	024512-R	# ZRQAM1	ZRQAM2	
RC.COU	105702-R	# ZRQAM1	ZRQAM2	ZRQAM3
REG.EX	044476-R	# ZRQAM3		
RETPKT	103230-R	# ZRQAM1	ZRQAM2	ZRQAM3
ROUND.	066434-R	# ZRQAM3		
RPS.RE	067416-R	# ZRQAM3		
RPT1	010074-R	# ZRQAM1	ZRQAM2	
RPT10	010604-R	# ZRQAM1	ZRQAM2	
RPT11	010672-R	# ZRQAM1	ZRQAM2	
RPT12	010740-R	# ZRQAM1	ZRQAM2	
RPT13	011006-R	# ZRQAM1	ZRQAM2	
RPT14	011106-R	# ZRQAM1	ZRQAM2	
RPT15	011204-R	# ZRQAM1	ZRQAM2	
RPT16	011304-R	# ZRQAM1	ZRQAM2	
RPT2	010160-R	# ZRQAM1	ZRQAM2	
RPT3	010224-R	# ZRQAM1	ZRQAM2	
RPT4	010310-R	# ZRQAM1	ZRQAM2	
RPT5	010354-R	# ZRQAM1	ZRQAM2	
RPT6	010442-R	# ZRQAM1	ZRQAM2	
RPT7	010506-R	# ZRQAM1	ZRQAM2	
RPT8	010524-R	# ZRQAM1	ZRQAM2	
RPT9	010552-R	# ZRQAM1	ZRQAM2	
RP.ADD	104002-R	# ZRQAM1	ZRQAM2	ZRQAM3
RP.IND	104000-R	# ZRQAM1	ZRQAM2	ZRQAM3
RP.USE	103770-R	# ZRQAM1	ZRQAM2	ZRQAM3
SA.REG	105640-R	# ZRQAM1	ZRQAM2	ZRQAM3
SB.COD	105632-R	# ZRQAM1	ZRQAM2	ZRQAM3
SCAN.E	072504-R	# ZRQAM3		
SC.CLK	022070-R	# ZRQAM1	ZRQAM2	
SC.CON	017102-R	# ZRQAM1	ZRQAM2	
SC.CTO	021332-R	# ZRQAM1	ZRQAM2	
SC.DIS	017430-R	# ZRQAM1	ZRQAM2	
SC.DST	020040-R	# ZRQAM1	ZRQAM2	
SC.DS2	020114-R	# ZRQAM1	ZRQAM2	
SC.DUP	017124-R	# ZRQAM1	ZRQAM2	
SC.ECC	020166-R	# ZRQAM1	ZRQAM2	
SC.ECD	020250-R	# ZRQAM1	ZRQAM2	
SC.EC1	020520-R	# ZRQAM1	ZRQAM2	
SC.EC2	020550-R	# ZRQAM1	ZRQAM2	
SC.EC3	020600-R	# ZRQAM1	ZRQAM2	

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
SC.EC4	020632-R	# ZRQAM1	ZRQAM2	
SC.EC5	020662-R	# ZRQAM1	ZRQAM2	
SC.EC6	020712-R	# ZRQAM1	ZRQAM2	
SC.EC7	020742-R	# ZRQAM1	ZRQAM2	
SC.EC8	020774-R	# ZRQAM1	ZRQAM2	
SC.EC9	021026-R	# ZRQAM1	ZRQAM2	
SC.EDC	021462-R	# ZRQAM1	ZRQAM2	
SC.FCT	020452-R	# ZRQAM1	ZRQAM2	
SC.FER	017522-R	# ZRQAM1	ZRQAM2	
SC.FE2	017610-R	# ZRQAM1	ZRQAM2	
SC.FUL	020322-R	# ZRQAM1	ZRQAM2	
SC.HMP	021130-R	# ZRQAM1	ZRQAM2	
SC.IDS	021522-R	# ZRQAM1	ZRQAM2	
SC.IOP	017356-R	# ZRQAM1	ZRQAM2	
SC.ISH	017700-R	# ZRQAM1	ZRQAM2	
SC.IS2	017760-R	# ZRQAM1	ZRQAM2	
SC.NXM	021242-R	# ZRQAM1	ZRQAM2	
SC.ODA	021170-R	# ZRQAM1	ZRQAM2	
SC.ODB	021220-R	# ZRQAM1	ZRQAM2	
SC.ONL	017154-R	# ZRQAM1	ZRQAM2	
SC.PAR	021276-R	# ZRQAM1	ZRQAM2	
SC.POE	021752-R	# ZRQAM1	ZRQAM2	
SC.PSP	022214-R	# ZRQAM1	ZRQAM2	
SC.RCT	020302-R	# ZRQAM1	ZRQAM2	
SC.RDY	022006-R	# ZRQAM1	ZRQAM2	
SC.RSP	022116-R	# ZRQAM1	ZRQAM2	
SC.SDI	017056-R	# ZRQAM1	ZRQAM2	
SC.SDS	021404-R	# ZRQAM1	ZRQAM2	
SC.SON	017176-R	# ZRQAM1	ZRQAM2	
SC.SRI	021664-R	# ZRQAM1	ZRQAM2	
SC.SRT	021572-R	# ZRQAM1	ZRQAM2	
SC.SUR	022164-R	# ZRQAM1	ZRQAM2	
SC.SWP	021070-R	# ZRQAM1	ZRQAM2	
SC.UNK	017216-R	# ZRQAM1	ZRQAM2	
SC.VOL	017276-R	# ZRQAM1	ZRQAM2	
SC.576	020376-R	# ZRQAM1	ZRQAM2	
SEND	034452-R	# ZRQAM2	ZRQAM3	
SEQUEN	071340-R	# ZRQAM3		
SET.CP	032716-R	# ZRQAM2	ZRQAM3	
SET.CT	046466-R	# ZRQAM3		
SET.UP	033012-R	# ZRQAM2	ZRQAM3	
SFPTBL	025134-R	# ZRQAM1		
SFT.MS	024724-R	# ZRQAM1	ZRQAM3	
SOFT.E	074276-R	# ZRQAM3		
SPACE4	025056-R	# ZRQAM1	ZRQAM2	
STEP	105634-R	# ZRQAM1	ZRQAM2	ZRQAM3
ST.COD	105630-R	# ZRQAM1	ZRQAM2	ZRQAM3
SWEEP	067264-R	# ZRQAM3		
SWM1	005042-R	# ZRQAM1	ZRQAM2	
SWP.DP	025142-R	# ZRQAM1	ZRQAM3	
SWP.ER	025134-R	# ZRQAM1	ZRQAM3	
SWP.FL	025140-R	# ZRQAM1	ZRQAM2	ZRQAM3

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...			
SWP.RA	025144-R	# ZRQAM1	ZRQAM3		
SWP.TI	025146-R	# ZRQAM1	ZRQAM3		
SWP.UC	025152-R	# ZRQAM1	ZRQAM3		
SWP.UD	025154-R	# ZRQAM1	ZRQAM3		
SWP.XF	025136-R	# ZRQAM1	ZRQAM3		
SWQ1	003342-R	# ZRQAM1	ZRQAM2		
SWQ10	003614-R	# ZRQAM1	ZRQAM2		
SWQ11	003660-R	# ZRQAM1	ZRQAM2		
SWQ12	003712-R	# ZRQAM1	ZRQAM2		
SWQ13	004010-R	# ZRQAM1	ZRQAM2		
SWQ14	004066-R	# ZRQAM1	ZRQAM2		
SWQ15	004140-R	# ZRQAM1	ZRQAM2		
SWQ17	004210-R	# ZRQAM1	ZRQAM2		
SWQ19	004306-R	# ZRQAM1	ZRQAM2		
SWQ2	003364-R	# ZRQAM1	ZRQAM2		
SWQ20	004376-R	# ZRQAM1	ZRQAM2		
SWQ21	004464-R	# ZRQAM1	ZRQAM2		
SWQ22	004550-R	# ZRQAM1	ZRQAM2		
SWQ23	004610-R	# ZRQAM1	ZRQAM2		
SWQ24	004662-R	# ZRQAM1	ZRQAM2		
SWQ25	004726-R	# ZRQAM1	ZRQAM2		
SWQ26	005000-R	# ZRQAM1	ZRQAM2		
SWQ4	003444-R	# ZRQAM1	ZRQAM2		
SWQ7	003466-R	# ZRQAM1	ZRQAM2		
SWQ9	003540-R	# ZRQAM1	ZRQAM2		
S.DUPP	105676-R	# ZRQAM1	ZRQAM2	ZRQAM3	
S.PATT	105674-R	# ZRQAM1	ZRQAM2	ZRQAM3	
TALLY	100076-R	# ZRQAM1	ZRQAM2	ZRQAM3	
TEMP1	105652-R	# ZRQAM1	ZRQAM2	ZRQAM3	
TEMP2	105654-R	# ZRQAM1	ZRQAM2	ZRQAM3	
TIME	032634-R	# ZRQAM2	ZRQAM3		
TRK.SG	101432-R	# ZRQAM1	ZRQAM2	ZRQAM3	
T$FREE	110060-R	# ZRQAM4			
T$PTHV	000004	ZRQAM1	# ZRQAM4		
T.ADDR	100426-R	# ZRQAM1	ZRQAM2	ZRQAM3	
T1	043344-R	ZRQAM1	# ZRQAM3		
UAM	000200	# ZRQAM1	# ZRQAM2	# ZRQAM3	
UPD.IO	066040-R	# ZRQAM3			
VEC.BR	044710-R	# ZRQAM3			
WAIT	035110-R	# ZRQAM2	ZRQAM3		
XX13	016524-R	# ZRQAM1	ZRQAM2		
XX23	016546-R	# ZRQAM1	ZRQAM2		
XX32	016602-R	# ZRQAM1	ZRQAM2		
XX33	016630-R	# ZRQAM1	ZRQAM2		
XX34	016666-R	# ZRQAM1	ZRQAM2		
$END.L	110062-R	# ZRQAM4			
$SAVE2	077570-R	B16MUL	# B16SAV	ZRQAM2	ZRQAM3
$SAVE3	077604-R	# B16SAV	ZRQAM2	ZRQAM3	
$SAVE4	077622-R	# B16SAV	ZRQAM2	ZRQAM3	
$SAVE5	077642-R	B16MUL	# B16SAV	ZRQAM2	ZRQAM3