RQDX1 RX50 RQDX1/RUX50 EXER
RD51/52 RUX50 CZRQAF0

AH-T399F-MC
FICHE 01 OF 03

APR 1985
digital
Made In USA

RQDX1 RX50
RD51/52 RUX50

RQDX1/RUX50 EXER
CZRQAF0

AH-T399F-MC
FICHE 02 OF 03

APR 1985
digital
Made In USA

RQDX1 RX50 RQDX1/RUX50 EXER
RD51/52 RUX50 CZRQAF0

AH-T399F-MC
FICHE 03 OF 03

APR 1985
digital
Made In USA

IDENTIFICATION

PRODUCT CODE: AC T398F-MC

PRODUCT NAME: CZRQAF0 RQDX1/RUX50 EXERCISER

PRODUCT DATE: 27-DEC-84

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: RAVINDER K. KARWAN
BOB POWERS

```
;  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             RD51_MAX_TRACK  = 1200,                         ! MAXIMUM NUMBER OF TRACKS FOR RD51
;  0028  0             RD51_SEC_PER_TRK= 18,                           ! NUMBER OF SECTORS PER TRACK FOR RD51
;  0029  0             RD51_MAX_LBN    = RD51_MAX_TRACK * RD51_SEC_PER_TRK - 1,! MAX LBN FOR RD51
;  0030  0
;  0031  0             RD52_MAX_TRACK  = 2976,                         ! MAXIMUM NUMBER OF TRACKS FOR RD52
;  0032  0             RD52_SEC_PER_TRK= 18,                           ! NUMBER OF SECTORS PER TRACK FOR RD52
;  0033  0             RD52_MAX_LBN    = RD52_MAX_TRACK * RD52_SEC_PER_TRK - 1,! MAX LBN FOR RD52
;  0034  0
;  0035  0             RX50_MAX_TRACK  = 80,                           ! MAXIMUM NUMBER OF TRACKS FOR RX50
;  0036  0             RX50_SEC_PER_TRK= 10,                           ! NUMBER OF SECTORS PER TRACK FOR RX50
;  0037  0             RX50_MAX_LBN    = RX50_MAX_TRACK * RX50_SEC_PER_TRK - 1,! MAX LBN FOR RX50
;  0038  0
;  0039  0             BYTES_PER_SECT  = 512,                          ! BYTES/SECTOR (AT PRESENT SAME FOR RDs AND RXs)
;  0040  0     !       MAX_XFER_SIZE   = 2 * BYTES_PER_SECT,           ! ARTBITRARY MAX SIZE OF EACH DISK I/O
;  0041  0             MAX_XFER_SIZE   = BYTES_PER_SECT * 3 / 2,
;  0042  0     !
;  0043  0     ! NOTE - BOTH OF THESE NUMBERS ARE NOW ARBITRARILY CHOSEN AS THE NUMBER OF LBNS CONTAINED PER UNIT/10 .
;  0044  0
;  0045  0     !
;  0046  0     !++++++ RING SIZES
;  0047  0     !
;  0048  0             CR_LOG          = 2,                            ! LOG2 LENGTH OF COMMAND RING
;  0049  0             RR_LOG          = 2,                            ! LOG2 LENGTH OF RESPONSE RING
;  0050  0             CRING_LEN       = 1 ^ CR_LOG,                   ! COMMAND RING LENGTH
;  0051  0             RRING_LEN       = 1 ^ RR_LOG,                   ! RESPONSE RING LENGTH
;  0052  0     !
;  0053  0     !++++++ OFFSETS (IN WORDS)
```

```
;    0054  0     !
;    0055  0             OF_UN           = 3,                                    ! OFFSET FROM START OF CST TO FIRST UNIT
;    0056  0             OF_DATA         = 0,                                    ! OFFSET TO DISK UNIT FLAGS   WITHIN UNIT'S CST
;    0057  0             OF_BEG          = 1,                                    ! OFFSET TO BEGINNING BLK NO. WITHIN UNIT'S CST
;    0058  0             OF_BEG1         = 2,        !OFFSET TO START BK HI      ZZZ
;    0059  0             OF_END          = 3,        !OFFSET TO END BLOCK LO     ZZZ
;    0060  0             OF_END1         = 4,        !OFFSET TO END BK HI        ZZZ
;    0061  0             OF_NAME_0       = 5,        !OFFSET TO 1st 2 CHARS OF NAME  ZZZ
;    0062  0             OF_NAME_2       = 6,        !OFFSET TO 2nd 2 CHARS OF NAME  ZZZ
;    0063  0             OF_DUPFLAGS     = 8,        !OFFSET TO DUP FLAGS        ZZZ
;    0064  0             OF_COUNT        = 9,        !OFFSET TO MSCP FUNCTION COUNTER ZZZ
;    0065  0             OF_DBN          = 8,        !OFFSET TO RELATIVE DBN     ZZZ
;    0066  0     !
;    0067  0     !
;    0068  0     !****** TABLE AND OTHER STRUCTURE SIZES
;    0069  0     !
;    0070  0             LBNADR_LEN      = 2,                                    !MAX_LBN'S ARE 2 WD ADDRESSES
 ZZZ
;    0071  0             HWPT_LEN        = 8,                                    ! SIZE (WORDS) OF HW P-TABLE
 ZZZ
;    0072  0             COMM_LEN        = (RRING_LEN * 2) + (CRING_LEN * 2) + 4,! SIZE (WORDS) OF COMMUNICATION AREA PER CONTROLLER
;    0073  0             UNIT_SIZE       = 10,                                   ! SIZE (WORDS) OF CST UNIT ENTRY
 ZZZ
;    0074  0             CST_LEN         = UNITS_PER_CNTR * UNIT_SIZE + OF_UN,   ! SIZE (WORDS) OF A CONTROLLER STATUS TABLE
;    0075  0             TALLY_CLEAR     = 7,                                    ! SIZE (WORDS) OF STATISTICS TBL CLEARED EVERY PASS
;    0076  0             TALLY_TOTALS    = 20,                                   ! SIZE (WORDS) OF STATISTICS TABLE FOR TOTALS
 ZZZ
;    0077  0             TALLY_LEN       = TALLY_CLEAR + TALLY_TOTALS,           ! SIZE (WORDS) OF A STATISTICS TABLE
;    0078  0             C_ERR_LEN       = 1,                                    ! SIZE (WORDS) OF CONTROLLER ERROR TABLE
;    0079  0             RP_LEN          = 22,                                   ! SIZE (WORDS) OF A RETURN PACKET
;    0080  0             MSG_LEN         = 30,                                   ! SIZE (WORDS) OF AN MSCP MESSAGE (TEXT PORTION)
;    0081  0             PKT_LEN         = MSG_LEN + 5,                          ! SIZE (WORDS) OF AN MSCP PACKET
;    0082  0             DCT_LEN         = 9,                                    ! SIZE (WORDS) OF A DRIVER CONTROLLER TABLE
;    0083  0             RDM_LEN         = 16,                                   ! SIZE (WORDS) OF THE RANDOM NUMBER TABLE
;    0084  0             MAX_UDP_CNT     = 16,                                   ! MAX SIZE OF USER DATA PATTERN
;    0085  0             MAX_BUF_CNT     = (CRING_LEN * 2) * MAX_CTLR,           ! MAX NO. OF I/O BUFFERS (BUFF_ADDR & BUFF_OWN)
;    0086  0             PKT_CNT         = ((CRING_LEN * 2) + RRING_LEN) * MAX_CTLR,
;    0087  0                                                                     ! NO. OF MSCP PACKETS IN POOL
;    0088  0             RP_CNT          = PKT_CNT - (RRING_LEN * MAX_CTLR),     ! NO. OF RETURN PACKETS IN POOL
;    0089  0             IODQ_LEN        = RP_CNT,                               ! NO. OF ENTRIES IN I/O DONE QUEUE (IODQ)
;    0090  0             OUTC_CNT        = CRING_LEN * 2,                        ! NO. OF ENTRIES/CONTROLLER'S OUTSTANDING CMD LIST
;    0091  0             DP_CNT          = 21,                                   ! NO. OF PRE-DEFINED DATA PATTERNS
;    0092  0             EP_CNT          = MAX_CTLR * RRING_LEN * 3,             ! NO. OF ERROR-LOG PACKET SAVE BUFFERS
;    0093  0             EP_LEN          = PKT_LEN - 3 + 1,                      ! LENGTH OF EACH ERROR-LOG SAVE BUFFER
;    0094  0             LAST_PKT_LEN    = 3,                                    ! BUFFER LENGTH TO SAVE INFO. ABOUT LAST RESPONSE
;    0095  0             TOO_MANY_READS  = 2,                                    !FOR READ/WRITE BALANCE WITH HOST READ COMPARES  ZZZ
;    0096  0             DESC_SIZ        = 4,                                    !NO. OF BYTES IN A PACKET DESCRIPTOR             ZZZ
;    0097  0     !
;    0098  0     !****** SW P-TABLE FLAGS (SWP_FLAGS)
;    0099  0     !
;    0100  0     !ZZZ    SWF_TRC         = %o'000001',                           ! DIAGNOSTIC TRACE
;    0101  0             SWF_APT         = %o'000001',                           !RUNNING UNDER A.P.T. MONITOR                    ZZZ
;    0102  0             SWF_RDM         = %o'000002',                           ! RANDOM SEEK MODE
;    0103  0             SWF_CRC         = %o'000004',                           ! READ-COMPARE AT CONTROLLER
;    0104  0     !       SWF_DCC         = %o'000010',                           ! DRIVE COMPLEMENT COMPLETE
;    0105  0             SWF_CWC         = %o'000020',                           ! WRITE-COMPARE AT CONTROLLER
;    0106  0             SWF_HWC         = %o'000040',                           ! WRITE-COMPARE AT HOST
```

```
;  0107  0          SWF_UDP         = %o'000100',                  ! USER-DEFINED DATA PATTERN
;  0108  0          SWF_CST         = %o'000200',                  ! CLEAR STATISTICAL TABLES
;  0109  0    !     SWF_DIA         = %o'000400',                  ! DIAGNOSTIC PACKAGE, WHEN THIS IS SELECTED
;  0110  0                                                         ! ALL INTERRUPTS ARE WAITED FOR, E.G. ONLY
;  0111  0                                                         ! ONE MSCP PACKET IS OUTSTANDING AT A TIME
;  0112  0          SWF_SEQ         = %o'001000',                  ! RANDOM OR FIXED SEQUENTIAL STEPPING
;  0113  0    !     SWF_DUP         = %o'002000',                  ! RUN DUP DIAGNOSTIC
;  0114  0          SWF_FER         = %o'004000',                  ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT DETECTED
;  0115  0          SWF_HRD         = %o'010000',                  ! HALT ON HARD ERRORS ALSO WITH 'HOE' DRS FLAG?
;  0116  0          SWF_SFT         = %o'020000',                  ! HALT ON SOFT ERRORS ALSO WITH 'HOE' DRS FLAG?
;  0117  0          SWF_BLK         = %o'040000',                  ! HALT ON BAD-BLOCK ERRORS ALSO WITH 'HOE' DRS FLAG?
;  0118  0          SWF_TRY         = %o'100000',                  ! COUNT EACH RETRY AS ANOTHER EXTRA SOFT-ERROR
;  0119  0    !
;  0120  0    !****** FLAGS FOR DUP EXERCISER (DUP_FLAGS)                                ZZZ
;  0121  0    !                                                                         ZZZ
;  0122  0          SWP_DINT        = %o'2',          !DUP CAUSED INIT                ZZZ
;  0123  0
;  0124  0    !
;  0125  0    !****** ENTRY_REASON VALUES
;  0126  0    !            (HOW PROGRAM WAS INVOKED)
;  0127  0    !
;  0128  0          START           = 1,                           ! START
;  0129  0          RESTART         = 2,                           ! RESTART
;  0130  0          CONT            = 3,                           ! CONTINUE
;  0131  0          PWR_FAIL        = 4,                           ! POWER FAIL
;  0132  0          NEW_PASS        = 5,                           ! NEW PASS
;  0133  0    !
;  0134  0    !****** DROP UNIT REASONS
;  0135  0    !            (LOADED INTO DUR VECTOR)
;  0136  0    !
;  0137  0          DU_USER         = 0,                           ! USER COMMAND
;  0138  0          DU_CONF         = 1,                           ! CONFIGURATION ERROR
;  0139  0          DU_INIT         = 2,                           ! INITIALIZATION ERROR
;  0140  0          DU_XFER         = 3,                           ! TRANSFER LIMIT REACHED
;  0141  0          DU_HERR         = 4,                           ! HARD ERROR LIMIT REACHED
;  0142  0          DU_DFATAL       = 5,                           ! UNRECOVERABLE DEVICE ERROR
;  0143  0          DU_CFATAL       = 6,                           ! UNRECOVERABLE CONTROLLER ERROR
;  0144  0          DU_ONLINE       = 7,                           ! ONLINE FAILED
;  0145  0          DU_ACCESS       = 8,                           ! ACCESS TO LAST TRACK FAILED
;  0146  0          DU_PROTECT      = 9,                           ! WRITE PROTECT CONFLICT
;  0147  0          DU_TIME         = 10,                          ! COMMAND TIME OUT
;  0148  0    !
;  0149  0    !****** MISCELLANEOUS LITERALS
;  0150  0    !
;  0151  0          INI_ATT         = 2,                           ! NO. OF HW INIT ATTEMPTS BEFORE FAILURE IS ASSUMED
;  0152  0          WR_RING         = ((%o'200') or (CR_LOG + 3) or (RR_LOG)),
;  0153  0                                                         ! WR-BIT-AND-RING-LENGTH (STEP 1 WRITE/STEP 2 READ)
;  0154  0          QIO_PER_CTLR    = CRING_LEN * 2,               ! MAXIMUM NUMBER OF OUTSTANDING QIOS PER CONTROLLER
;  0155  0    !     MAX_XFER        = 256,                         ! MAXIMUM SIZE (WORDS) OF AN I/O TRANSFER
;  0156  0          REMOVABLE_BIT   = %o'0',                       ! BIT IN HARDWARE TABLES MARKING A REMOVABLE DISK
;  0157  0          FIXED_BIT       = %o'20',                      ! BIT IN HARDWARE TABLES MARKING A FIXED     DISK
;  0158  0          REMOVABLE       = 0,                           ! NUMBER FOR REMOVABLE DISK WHEN SHIFTED RIGHT
;  0159  0          FIXED           = 1,                           ! NUMBER FOR FIXED     DISK WHEN SHIFTED RIGHT
```

F 1


```
;  0160  0                 RX_50           = 0,                    !D_TYPE FLAG = 0 FOR RX50 ZZZ
;  0161  0                 RD_51           = 1,                    !D_TYPE FLAG = 1 FOR RD51 ZZZ
;  0162  0                 RD_52           = 2,                    !                2 FOR RD52 ZZZ
;  0163  0         !
;  0164  0         !****** MSCP PACKET DESCRIPTOR
;  0165  0         !
;  0166  0                 ED_OWN          = %o'100000',                   ! OWNERSHIP BIT
;  0167  0                 ED_FLAG         = %o'040000',                   ! FLAG BIT
;  0168  0         !
;  0169  0         !****** MSCP COMMAND PACKET OPCODES
;  0170  0         !
;  0171  0                 OP_MSK          = %o'177',                      ! OPCODE MASK
;  0172  0                 OP_END          = %o'200',                      ! ENDCODE DESIGNATOR
;  0173  0                 OP_ACC          = %o'20',                       ! ACCESS COMMAND
;  0174  0                 OP_ONL          = %o'11',                       ! ONLINE COMMAND
;  0175  0                 OP_RD           = %o'41',                       ! READ COMMAND
;  0176  0                 OP_SCC          = %o'4',                        ! SET CONTROLLER CHARACTERISTICS COMMAND
;  0177  0                 OP_WRT          = %o'42',                       ! WRITE COMMAND
;  0178  0                 OP_GDS          = %o'1',        !get dust status        ZZZ
;  0179  0                 OP_ESP          = %o'2',        !execute supplied prog  ZZZ
;  0180  0                 OP_ELP          = %o'3',        !execute local program  ZZZ
;  0181  0                 OP_SDD          = %o'4',        !send data              ZZZ
;  0182  0                 OP_RCD          = %o'5',        !receive data           ZZZ
;  0183  0                 OP_ABT          = %o'6',        !abort program          ZZZ
;  0184  0
;  0185  0         !
;  0186  0         !****** PACKET SIZES
;  0187  0         !
;  0188  0                 SZ_ACC = %decimal '32',                         ! ACCESS
;  0189  0                 SZ_ONL = %decimal '36',                         ! ON LINE COMMAND
;  0190  0                 SZ_RD  = %decimal '32',                         ! READ
;  0191  0                 SZ_SCC = %decimal '32',                         ! SET CONTROLLER CHARACTERISTICS
;  0192  0                 SZ_WRT = %decimal '32',                         ! WRITE
;  0193  0                 SZ_GEN = %decimal '32',                         ! GENERAL PACKET SIZE
;  0194  0                 SZ_REC = %DECIMAL '28',         !                       ZZZ
;  0195  0                 SZ_SEN = %DECIMAL '28',         !                       ZZZ
;  0196  0                 SZ_ELP = %DECIMAL '18',         !                       ZZZ
;  0197  0                 SZ_ABT = %DECIMAL '12',         !                       ZZZ
;  0198  0                 SZ_GDS = %DECIMAL '12',         !                       ZZZ
;  0199  0         !
;  0200  0         !****** MSCP COMMAND MODIFIERS
;  0201  0         !
;  0202  0                 MD_CMP          = %o'040000',                   ! COMPARE
;  0203  0                 MD_EXP          = %o'100000',                   ! EXPRESS REQUEST
;  0204  0         !
;  0205  0         !****** CONNECTION ID VALUES (MSCP_PKT, RETPKT)
;  0206  0         !               (SERVE AS SOURCES AND DESTINATIONS OF MSCP MESSAGES)
;  0207  0         !
;  0208  0                 CID_DISK        = 0,                            ! DISK MSCP
;  0209  0                 CID_MSCP        = 0,                            !DISK MSCP
 !ZZZ
;  0210  0                 CID_TAPE        = 1,                            ! TAPE MSCP
;  0211  0                 CID_DUP         = 2,                            ! DIAGNOSTIC AND UTILITIES PROTOCOL
;  0212  0                 CID_DRIVER      = 3,                            ! EXERCISER "DRIVER'
```

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

```
;  0266  0              SA_S1           = %o'004000',                  ! STEP 1 STATUS BIT
;  0267  0              SA_S2           = %o'010000',                  !   :  2
;  0268  0              SA_S3           = %o'020000',                  !   :  3
;  0269  0              SA_S4           = %o'040000',                  !   V  4
;  0270  0              SA_ERR          = %o'100000',                  ! ERROR INDICATOR
;  0271  0              SA_INT          = %o'000200',                  ! INTERRUPT ENABLE DURING INITIALIZATION
;  0272  0              SA_GO           = %o'000001',                  ! GO BIT TO START FIRMWARE
;  0273  0      !
;  0274  0      !****** INITIALIZATION STEP READ MASKS
;  0275  0      !
;  0276  0              S1_MASK         = %o'176000',                  ! STEP 1 READ BITS
;  0277  0              S2_MASK         = %o'174377',                  !   :  2
;  0278  0              S3_MASK         = %o'174377',                  !   :  3
;  0279  0              S4_MASK         = %o'174000',                  !   V  4
;  0280  0      !
;  0281  0      !****** COMMAND TYPES
;  0282  0      !
;  0283  0              IMM_CMD         = 0,                           ! IMMEDIATE COMMAND
;  0284  0              SEQ_CMD         = 1,                           ! SEQUENTIAL COMMAND
;  0285  0              NON_SEQ_CMD     = 2,                           ! NON-SEQUENTIAL COMMAND
;  0286  0      !
;  0287  0      !****** ERROR-LOG FORMAT TYPES
;  0288  0      !
;  0289  0              FORMAT_CNTR     = %o'0',                       ! CONTROLLER ERROR
;  0290  0              FORMAT_HOST     = %o'1',                       ! HOST MEMORY ACCESS ERROR
;  0291  0              FORMAT_XFER     = %o'2',                       ! DISK TRANSFER ERROR
;  0292  0              FORMAT_SDI      = %o'3',                       ! 'STANDARD DISK INTECONNECT' ERROR
;  0293  0              FORMAT_SDE      = %o'4',                       ! SMALL DISK ERROR
;  0294  0      !
;  0295  0      !****** ERROR-LOG BLOCK NUMBER INFORMATION
;  0296  0      !
;  0297  0              TYPE_LBN        = %o'0000',                    ! LOGICAL BLOCK NUMBER
;  0298  0              TYPE_RBN        = %o'0110',                    ! REPLACEMENT BLOCK NUMBER
;  0299  0      !
;  0300  0      !****** MSCP DISK MODEL CODES
;  0301  0      !
;  0302  0      !       MODEL_RX50      = 7,                           ! RX50
;  0303  0      !       MODEL_RD51      = 6,                           ! RD51
;  0304  0      !       MODEL_RD52      = 8,                           ! RD52
;  0305  0      !
;  0306  0      !****** LITERALS FOR READABILITY
;  0307  0      !
;  0308  0              YES             = 1,
;  0309  0              NO              = 0,
;  0310  0              TRUE            = 1,
;  0311  0              FALSE           = 0,
;  0312  0              SUCCESS         = 1,
;  0313  0              FAILURE         = 0,
;  0314  0              FOUND           = 1,
;  0315  0              NOT_FOUND       = 0,
;  0316  0              PRESENT         = 1,                           ! DISK IS PRESENT IN CONTROLLER
;  0317  0              NOT_PRESENT     = 0,                           ! DISK IS NOT PRESENT IN CONTROLLER
;  0318  0              UNPROTECTED     = 1,                           ! DISK HAS UNPROTECTED CUSTOMER LBN'S
```

```
;   0319  0          PROTECTED        = 0,                    ! DISK HAS PROTECTED CUSTOMER LBN'S
;   0320  0          ONLINE           = 1,
;   0321  0          OFFLINE          = 0,
;   0322  0          IDLE             = 0,                    !IDLE
 !ZZZ
;   0323  0          ACTIVE           = 1,                    !ACTIVE
 !ZZZ
;   0324  0          FULL             = 1,                    ! ERROR-LOG SAVE PACKET FILLED
;   0325  0          EMPTY            = 0,                    ! ERROR-LOG SAVE PACKET PRINTED
;   0326  0          HRD_OCCURED      = 1,                    ! HARD ERROR DETECTED IN RESPONSE PACKET
;   0327  0          HRD_NOT_OCCURED  = 0,                    ! HARD ERROR NOT DETECTED
;   0328  0          ALL_ONES         = %o'177777';
```

```
;  0329  0     !*****************:*************************************************************
;  0330  0     !                                                                            *
;  0331  0     !                          F I E L D S                                       *
;  0332  0     !                                                                            *
;  0333  0     !*****************************************************************************
;  0334  0     !
;  0335  0     FIELD
;  0336  0     !
;  0337  0     !****** HARDWARE P-TABLE FIELDS
;  0338  0     !
;  0339  0     HWP_FIELDS =
;  0340  0             set
;  0341  0             HWP_IP_ADDR     = [0, 0, 16, 0],                ! IP ADDRESS
;  0342  0             HWP_VECTOR      = [1, 0, 16, 0],                ! VECTOR ADDRESS
;  0343  0             HWP_BR_LEVEL    = [2, 0, 16, 0],                ! BUS REQUEST LEVEL
;  0344  0             HWP_DISK        = [3, 0, 16, 0],                ! DISK (ALL FIELDS)
;  0345  0             HWP_DISK_NUM    = [3, 0, 4, 0],                 ! DISK NUMBER
;  0346  0             HWP_DISK_TYPE   = [3, 4, 1, 0],                 ! DISK TYPE
;  0347  0             HWP_DISK_DUPEX  = [3, 5, 1, 0],                 !RUN DUP EXERCISER              !ZZZ
;  0348  0             HWP_DISK_DUPWT  = [3, 6, 1, 0],                 !DUP WRITE FLAG                 !ZZZ
;  0349  0             HWP_ENTIRE      = [3, 7, 1, 0],                 !TEST ENTIRE DISK               !ZZZ
;  0350  0             HWP_DISK_CP     = [3, 15, 1, 0],                ! PROTECT CUSTOMER DATA BIT
;  0351  0             HWP_BEG_TRK     = [4, 0, 16, 0],                ! BEGINNING TRACK LO            !ZZZ
;  0352  0             HWP_BEG_TRK1    = [5, 0, 16, 0],                ! BEGINNING TRACK HI            !ZZZ
;  0353  0             HWP_END_TRK     = [6, 0, 16, 0],                ! ENDING TRACK LO               !ZZZ
;  0354  0             HWP_END_TRK1    = [7, 0, 16, 0]                 ! ENDING TRACK HI               !ZZZ
;  0355  0             tes,
;  0356  0     !
;  0357  0     !****** COMMUNICATION AREA HEADER FIELDS
;  0358  0     !
;  0359  0     COM_FIELDS =
;  0360  0             set
;  0361  0             ADAP_CH         = [1, 8, 8, 0],                 ! ADAPTER CHANNEL NUMBER FOR PURGES
;  0362  0             CMD_INT         = [2, 0, 16, 0],                ! COMMAND RING INTERRUPT
;  0363  0             RSP_INT         = [3, 0, 16, 0]                 ! RESPONSE RING INTERRUPT
;  0364  0             tes,
;  0365  0
;  0366  0     !                                                                       ZZZ
;  0367  0     !       DUP BUFFER FIELD                                                ZZZ
;  0368  0     !                                                                       ZZZ
;  0369  0     DP_FIELDS =                                                            !ZZZ
;  0370  0             SET                                                            !ZZZ
;  0371  0             DUPBF0          = [0, 0, 16, 0],                           !ZZZ
;  0372  0             DUPBF1          = [1, 0, 16, 0],                           !ZZZ
;  0373  0             DUPBF2          = [2, 0, 16, 0],                           !ZZZ
;  0374  0             DUPTYPE         = [0, 12, 4, 0],                           !ZZZ
;  0375  0             DUPMESG         = [0, 0, 12, 0]                            !ZZZ
;  0376  0             TES,                                                       !ZZZ
;  0377  0
;  0378  0
;  0379  0     !****** CONTROLLER STATUS TABLE (CST) FIELDS
;  0380  0     !
;  C381  0     CST_FIELDS =
```

```
;  0382  0          set
;  0383  0          IP_ADDR         = [0, 0, 16, 0],                ! IP ADDRESS
;  0384  0          VEC_ADDR        = [1, 0, 9, 0],                 ! VECTOR ADDRESS
;  0385  0          STATE           = [1, 15, 1, 0],                ! CONTROLLER STATUS
;  0386  0          BR_LEV          = [2, 0, 8, 0],                 ! BUS REQUEST LEVEL
;  0387  0          U_CNT           = [2, 8, 8, 0]                  ! NUMBER OF UNITS (DISKS) FOR THIS CONTROLLER
;  0388  0      !
;  0389  0      !   DU_ALL          = [3, 0, 16, 0],                ! DISK 0 (ALL FIELDS)
;  0390  0      !   DO_DISK_NUM     = [3, 0, 4, 0],                 ! DISK NUMBER
;  0391  0      !   DO_TYPE         = [3, 4, 1, 0],                 ! DISK TYPE
;  0392  0      !   DO_UNIT         = [3, 8, 4, 0],                 ! DISK 0 UNIT NUMBER (DRS UNIT)
;  0393  0      !   DO_FATAL        = [3, 12, 1, 0],                ! DISK 0 FATAL ERROR BIT
;  0394  0      !   DO_STAT         = [3, 13, 1, 0],                ! DISK 0 STATUS BIT
;  0395  0      !   DO_PRES         = [3, 14, 1, 0],                ! DISK 0 PRESENT BIT
;  0396  0      !   DO_PROT         = [3, 15, 1, 0],                ! DK 0 PROTECT CUSTOMER DATA
;  0397  0      !   DO_BEG0         = [4, 0, 16, 0],    !DK 0 BEGIN TK LO          ZZZ
;  0398  0      !   DO_BEG1         = [5, 0, 16, 0],    !DK 0 BEGIN TK HI          ZZZ
;  0399  0      !   DO_END0         = [6, 0, 16, 0],    !DK 0 END TK LO            ZZZ
;  0400  0      !   DO_END1         = [7, 0, 16, 0],    !DK 0 END TK HI            ZZZ
;  0401  0      !   DO_NAME0        = [8, 0, 8, 0],     !DK 0 MANE BYTE 0          ZZZ
;  0402  0      !   DO_NAME1        = [8, 8, 8, 0],     !DK 0 NAME BYTE 1          ZZZ
;  0403  0      !   DO_NAME2        = [9, 0, 8, 0],     !DK 0 NAME BYTE 2          ZZZ
;  0404  0      !   DO_NAME3        = [9, 8, 8, 0],     !DK 0 NAME BYTE 3          ZZZ
;  0405  0      !   DO_NUL          = [10, 0, 16, 0]    !NUL AFTER NAME            ZZZ
;  0406  0      !   DO_DBN          = [11, 0, 8, 0],    !DK 0 RELATIVE DBN         ZZZ
;  0407  0      !   DO_WRITE        = [11, 12, 1, 0],   !DK 0 DUP WRITE FLAG       ZZZ
;  0408  0      !   DO_ACTIVE       = [11, 13, 1, 0],   !DK 0 ACTIVE FLAG          ZZZ
;  0409  0      !   DO_DUPERROR     = [11, 14, 1, 0],   !DK 0 DUP ERROR FLAG       ZZZ
;  0410  0      !   DONODUPMED      = [11, 15, 1, 0],   !DK 0 NO DUP MEDIA FLAG    ZZZ
;  0411  0      !   DO_COUNT        = [12, 0, 16, 0],   !DK 0 RELATIVE MSCP FUN-   ZZZ
;  0412  0      !                                           CTION COUNTER          ZZZ
;  0413  0      !                                                                  ZZZ
;  0414  0      !   ! REPEAT WORDS 3 THROUGH 12 ABOVE AS: !                        ZZZ
;  0415  0      !   ! WORDS 13 THROUGH 21 FOR DRIVE 1     !                        ZZZ
;  0416  0      !   ! WORDS 22 THROUGH 30 FOR DRIVE 2     !                        ZZZ
;  0417  0      !   ! WORDS 31 THROUGH 39 FOR DRIVE 3     !                        ZZZ
;  0418  0      !   !                                     !                        ZZZ
;  0419  0      !                                                                  ZZZ
;  0420  0          tes,
;  0421  0      !
;  0422  0      !****** MSCP PACKET FIELDS
;  0423  0      !           (NOTE: BASE ADDRESS OF PACKET REFERENCES THE PACKET'S OWN
;  0424  0      !           BUFFER DESCRIPTOR, RATHER THAN THE MESSAGE BODY (TEXT + 0).
;  0425  0      !           SEE DOCUMENTATION FOR LAYOUT OF MSCP PACKETS )
;  0426  0      !
;  0427  0      PKT_FIELDS =
;  0428  0          set
;  0429  0      !
;  0430  0      !   HEADER FIELDS
;  0431  0      !
;  0432  0          PKT_LO          = [0, 0, 16, 0],                ! PACKET DESCRIPTOR (LO ORDER)
;  0433  0          PKT_HI          = [1, 0, 16, 0],                ! PACKET DESCRIPTOR (HI ORDER - ALL FIELDS)
;  0434  0          PKT_U           = [1, 0, 2, 0],                 ! PACKET DESCRIPTOR (HI ORDER UNIBUS BITS)
```

```
;  0435  0          PKT_Q           = [1, 2, 4, 0],                 ! PACKET DESCRIPTOR (HI ORDER Q-BUS BITS)
;  0436  0          PKT_F           = [1, 14, 1, 0],                ! PACKET DESCRIPTOR FLAG BIT
;  0437  0          PKT_O           = [1, 15, 1, 0],                ! PACKET DESCRIPTOR OWNERSHIP BIT
;  0438  0          CMD_TYPE        = [2, 0, 8, 0],                 ! COMMAND TYPE
;  0439  0          RSP_RECEIVED    = [2, 8, 8, 0],                 ! FLAG SET IF RESPONSE TO COMMAND RECEIVED
;  0440  0          MSGLEN          = [3, 0, 16, 0],                ! MESSAGE LENGTH
;  0441  0          CREDITS         = [4, 0, 4, 0],                 ! CREDITS
;  0442  0          MSGTYP          = [4, 4, 4, 0],                 ! MESSAGE TYPE
;  0443  0          CONNID          = [4, 8, 8, 0],                 ! CONNECTION ID
;  0444  0      !
;  0445  0      !   GENERIC COMMAND PACKET AND END PACKET HEADER FILEDS
;  0446  0      !
;  0447  0          CRN_LO          = [5, 0, 16, 0],                ! COMMAND REF NUMBER (LO ORDER)
;  0448  0          CRN_HI          = [6, 0, 16, 0],                ! COMMAND REF NUMBER (HI ORDER)
;  0449  0          DK_NUM          = [7, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0450  0          OPCODE          = [9, 0, 8, 0],                 ! OPCODE AND ENDCODE
;  0451  0          MODIFY          = [10, 0, 16, 0],               ! COMMAND MODIFIERS
;  0452  0          STATUS_CODE     = [10, 0, 5, 0],                ! STATUS (PART OF RESPONSE PACKET)
;  0453  0          STATUS_SUBCODE  = [10, 5, 11, 0],               ! SUBCODE (PART OF RESPONSE PACKET)
;  0454  0      !
;  0455  0      !   READ, WRITE, AND ACCESS COMMAND FIELDS (FOR COMMAND AND END PACKETS)
;  0456  0      !
;  0457  0          BC_LO           = [11, 0, 16, 0],               ! BYTE COUNT (LO ORDER)
;  0458  0          BC_HI           = [12, 0, 16, 0],               ! BYTE COUNT (HI ORDER)
;  0459  0          BUF_0           = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR
;  0460  0          BUF_1           = [14, 0, 16, 0],               !        :
;  0461  0          BUF_2           = [15, 0, 16, 0],               !        :
;  0462  0          BUF_3           = [16, 0, 16, 0],               !        :
;  0463  0          BUF_4           = [17, 0, 16, 0],               !        :
;  0464  0          BUF_5           = [18, 0, 16, 0],               !        V
;  0465  0          LBN_L           = [19, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
;  0466  0          LBN_H           = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
;  0467  0
;  0468  0      !
;  0469  0      ! DUP PROGRAM LETTER FIELDS (FOR EXECUTE LOCAL PROGRAM CMD)          ZZZ
;  0470  0      !                                                                    ZZZ
;  0471  0          L1              = [11, 0, 8, 0],        !LETTER NO 1            ZZZ
;  0472  0          L2              = [11, 8, 8, 0],        !LETTER NO 2            ZZZ
;  0473  0          L3              = [12, 0, 8, 0],        !LETTER NO 3            ZZZ
;  0474  0          L4              = [12, 8, 8, 0],        !LETTER NO 4            ZZZ
;  0475  0          L5              = [13, 0, 8, 0],        !LETTER NO 5            ZZZ
;  0476  0          L6              = [13, 8, 8, 0],        !LETTER NO 6            ZZZ
;  0477  0      !
;  0478  0      !   SET CONTROLLER CHARACTERISTICS COMMAND FIELDS
;  0479  0      !
;  0480  0          C_FLAGS         = [12, 0, 16, 0],               ! CONTROLLER FLAGS
;  0481  0      !
;  0482  0      !   ONLINE COMMAND FIELDS
;  0483  0      !
;  0484  0          U_FLAGS         = [12, 0, 16, 0],               ! UNIT FLAGS
;  0485  0          DDPAR           = [19, 0, 16, 0]                ! DEVICE DEPENDENT PARAMETERS
;  0486  0          tes,
;  0487  0      !
```

```
;  0488  0      !****** RETURN PACKET (RETPKT) FIELDS
;  0489  0      !               (SIMILAR, BUT NOT IDENTICAL, TO MSCP PACKET FIELDS)
;  0490  0      !
;  0491  0      RP_FIELDS =
;  0492  0              set
;  0493  0      !
;  0494  0      !       COMMON TO ALL RETURN PACKETS FROM DISK MSCP
;  0495  0      !
;  0496  0              MESLEN          = [0, 0, 16, 0],                ! MESSAGE LENGTH
;  0497  0              CTLR            = [1, 0, 4, 0],                 ! CONTROLLER NUMBER (CREDITS OVERWRITTEN)
;  0498  0              MESTYP          = [1, 4, 4, 0],                 ! MESSAGE TYPE
;  0499  0              CONID           = [1, 8, 8, 0],                 ! CONNECTION ID
;  0500  0              CRF_LO          = [2, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (LO ORDER)
;  0501  0              CRF_HI          = [3, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (HI ORDER)
;  0502  0              DISK            = [4, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0503  0              CMDMOD          = [5, 0, 16, 0],                ! COMMAND MODIFIERS
;  0504  0              ENDCOD          = [6, 0, 8, 0],                 ! END CODE
;  0505  0              FLAGS           = [6, 8, 8, 0],                 ! FLAGS
;  0506  0              STATUS          = [7, 0, 16, 0],                ! STATUS AND SUB-CODE
;  0507  0              STSCOD          = [7, 0, 5, 0],                 ! STATUS CODE
;  0508  0              SUBCOD          = [7, 5, 11, 0],                ! SUB-CODE
;  0509  0      !
;  0510  0      !       READ, WRITE, AND ACCESS COMMAND RETURN PACKETS
;  0511  0      !
;  0512  0              BCNT_LO         = [8, 0, 16, 0],                ! BYTE COUNT (LO ORDER)
;  0513  0              BCNT_HI         = [9, 0, 16, 0],                ! BYTE COUNT (HI ORDER)
;  0514  0              BUFF_0          = [10, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 0)
;  0515  0              BUFF_1          = [11, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 1)
;  0516  0              BUFF_2          = [12, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 2)
;  0517  0              BUFF_3          = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 3)
;  0518  0              BUFF_4          = [14, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 4)
;  0519  0              BUFF_5          = [15, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 5)
;  0520  0              BBLK_LO         = [16, 0, 16, 0],               ! FIRST BAD BLOCK (LO ORDER)
;  0521  0              BBLK_HI         = [17, 0, 16, 0],               ! FIRST BAD BLOCK (HI ORDER)
;  0522  0              CBCNT_LO        = [18, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (LO ORDER)
;  0523  0              CBCNT_HI        = [19, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (HI ORDER)
;  0524  0              LBN_LO          = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
;  0525  0              LBN_HI          = [21, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
;  0526  0      !
;  0527  0      !       SET CONTROLLER CHARACTERISTICS RETURN PACKET
;  0528  0      !
;  0529  0              C_FLGS          = [9, 0, 16, 0],                ! CONTROLLER FLAGS
;  0530  0              C_TIME          = [10, 0, 16, 0],               ! CONTROLLER TIMEOUT
;  0531  0      !
;  0532  0      !       UNIT ONLINE RETURN PACKET
;  0533  0      !
;  0534  0              U_FLGS          = [9, 0, 16, 0],                ! UNIT FLAGS
;  0535  0              R_MODEL         = [13, 0, 8, 0],                ! 2 DIGIT MODEL NUMBER                          ZZZ
;  0536  0              NAME_NUM        = [14, 0, 6, 0],                ! MODEL NAME - 2 DIGIT NUMBER
;  0537  0              NAME_1_LO       = [14, 12, 4, 0],               ! MODEL NAME - 2ND CHARACTER (LOW  ORDER 4 BITS)
;  0538  0              NAME_1_HI       = [15, 0, 1, 0],                ! MODEL NAME - 2ND CHARACTER (HIGH ORDER 1 BIT)
;  0539  0              NAME_0          = [15, 1, 5, 0],                ! MODEL NAME - 1ST CHARACTER
;  0540  0      !ZZZ    USIZ_LO         = [18, 0, 16, 0],               ! UNIT SIZE (LO ORDER)
```

```
;  0541  0     !ZZZ    USIZ_HI         = [19, 0, 16, 0],                ! UNIT SIZE (HI ORDER)
;  0542  0             SIZE0           = [18, 0, 16, 0],                !LOWER WD OF MAX LBNS OR UNIT SIZE          ZZZ
;  0543  0             SIZE1           = [19, 0, 16, 0]                 !UPPER WD                                   ZZZ
;  0544  0             tes,
;  0545  0     !
;  0546  0     !****** STATISTICS TABLE (TALLY) FIELDS
;  0547  0     !
;  0548  0     T_FIELDS =
;  0549  0             set
;  0550  0             BYTES_READ_LO   = [0, 0, 16, 0],                 ! NUMBER OF BYTES READ (LO ORDER)
;  0551  0             BYTES_READ_HI   = [1, 0, 16, 0],                 ! NUMBER OF BYTES READ (HI ORDER)
;  0552  0             MBYTES_READ     = [2, 0, 16, 0],                 ! MEGABYTES READ
;  0553  0             BYTES_WRIT_LO   = [3, 0, 16, 0],                 ! NUMBER OF BYTES WRITTEN (LO ORDER)
;  0554  0             BYTES_WRIT_HI   = [4, 0, 16, 0],                 ! NUMBER OF BYTES WRITTEN (HI ORDER)
;  0555  0             MBYTES_WRIT     = [5, 0, 16, 0],                 ! MEGABYTES WRITTEN
;  0556  0             ERR_HARD        = [6, 0, 16, 0],                 ! NUMBER OF HARD ERRORS
;  0557  0     !
;  0558  0             TOT_READS_LO    = [7, 0, 16, 0],                 ! TOTAL NUMBER OF READS (LO ORDER)
;  0559  0             TOT_READS_HI    = [8, 0, 16, 0],                 ! TOTAL NUMBER OF READS (HI ORDER)
;  0560  0             TOT_WRITES_LO   = [10, 0, 16, 0],                ! TOTAL NUMBER OF WRITES (LO ORDER)
;  0561  0             TOT_WRITES_HI   = [11, 0, 16, 0],                ! TOTAL NUMBER OF WRITES (HI ORDER)
;  0562  0             TOT_BYT_RED_LO  = [13, 0, 16, 0],                ! TOTAL BYTES READ (LO ORDER)
;  0563  0             TOT_BYT_RED_HI  = [14, 0, 16, 0],                ! TOTAL BYTES READ (HI ORDER)
;  0564  0             MTOT_BYT_RED    = [15, 0, 16, 0],                ! TOTAL MEGABYTES READ
;  0565  0             TOT_BYT_WRT_LO  = [16, 0, 16, 0],                ! TOTAL BYTES WRITTEN (LO ORDER)
;  0566  0             TOT_BYT_WRT_HI  = [17, 0, 16, 0],                ! TOTAL BYTES WRITTEN (HI ORDER)
;  0567  0             MTOT_BYT_WRT    = [18, 0, 16, 0],                ! TOTAL MEGABYTES WRITTEN
;  0568  0             ERR_HRD_SEK     = [19, 0, 8, 0],                 ! TOTAL HARD ERRORS - SEEK
;  0569  0             ERR_HRD_DAT     = [19, 8, 8, 0],                 ! TOTAL HARD ERRORS - DATA
;  0570  0             ERR_HRD_DRV     = [20, 0, 8, 0],                 ! TOTAL HARD ERRORS   DRIVE
;  0571  0             ERR_HRD_HST     = [20, 8, 8, 0],                 ! TOTAL HARD ERRORS - HOST
;  0572  0             ERR_SFT_SEK     = [21, 0, 8, 0],                 ! TOTAL SOFT ERRORS - SEEK
;  0573  0             ERR_SFT_DAT     = [21, 8, 8, 0],                 ! TOTAL SOFT ERRORS - DATA
;  0574  0             ERR_SFT_DRV     = [22, 0, 8, 0],                 ! TOTAL SOFT ERRORS - DRIVE
;  0575  0             ERR_SFT_HST     = [22, 8, 8, 0],                 ! TOTAL SOFT ERRORS   HOST
;  0576  0             T_BLK_WT        = [23, 0,16, 0],        !                          ZZZ
;  0577  0             T_DBN_WT        = [24, 0,16, 0],        !DBNS WRITTEN              ZZZ
;  0578  0             T_BLK_RD        = [25, 0,16, 0],        !                          ZZZ
;  0579  0             T_DBN_RD        = [26, 0,16, 0]         !DBNS READ                 ZZZ
;  0580  0
;  0581  0             tes,
;  0582  0     !
;  0583  0     !****** CONTROLLER ERROR TALLY FIELDS
;  0584  0     !
;  0585  0     C_ERR_FIELDS =
;  0586  0             set
;  0587  0             C_ERR_HRD       = [0, 0, 8, 0],                  ! HARD ERRORRS
;  0588  0             C_ERR_SFT       = [0, 8, 8, 0]                   ! SOFT ERRORS
;  0589  0             tes,
;  0590  0     !
;  0591  0     !****** DRIVER CONTROLLER TABLE (DCT) FIELDS
;  0592  0     !
;  0593  0     DCT_FIELDS =
```

```
;  0594  0          set
;  0595  0          WORD0          = [0, 0, 16, 0],     ! ALL FIELDS IN WORD 0
;  0596  0          CRING_CNT      = [0, 0, 8, 0],      ! NUMBER OF SLOTS IN CRING NOT YET RETURNED TO HOST
;  0597  0          IG_INT         = [0, 14, 1, 0],     ! IGNORE INTERRUPT BIT
;  0598  0          STAT           = [0, 15, 1, 0],     ! ONLINE / OFFLINE STATUS
;  0599  0          SA_SAVE        = [1, 0, 16, 0],     ! SA REGISTER SAVE WORD
;  0600  0          RR_BEG         = [2, 0, 16, 0],     ! FIXED ADDRESSES OF START AND
;  0601  0          RR_END         = [3, 0, 16, 0],     !       END OF EACH RING
;  0602  0          CR_BEG         = [4, 0, 16, 0],     !              :
;  0603  0          CR_END         = [5, 0, 16, 0],     !              V
;  0604  0          RR_POLL        = [6, 0, 16, 0],     ! ADDR OF NEXT RRING SLOT TO BE POLLED
;  0605  0          CR_POLL        = [7, 0, 16, 0],     ! ADDR OF NEXT CRING SLOT TO BE POLLED
;  0606  0          CR_NEXT        = [8, 0, 16, 0]      ! ADDR OF NEXT AVAIL CRING SLOT
;  0607  0          tes,
;  0608  0      !
;  0609  0      !****** ERROR LOG PACKET SAVE AREA FIELDS
;  0610  0      !
;  0611  0      EP_FIELDS =
;  0612  0          set
;  0613  0          EL_CNTR        = [0, 0, 8, 0],      ! CONTROLLER NUMBER
;  0614  0          EL_CONTENTS    = [0, 8, 8, 0],      ! FLAG INDICATES IF PACKET CONTENTS ALREADY PRINTED
;  0615  0          EL_MSGLEN      = [1, 0, 16, 0],     ! PACKET LENGTH
;  0616  0          EL_CRN_LO      = [3, 0, 16, 0],     ! COMMAND REFERENCE NUMBER
;  0617  0          EL_CRN_HI      = [4, 0, 16, 0],     !
;  0618  0          EL_DK_NUM      = [5, 0, 16, 0],     ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0619  0          EL_FORMAT      = [7, 0, 8, 0],      ! FORMAT
;  0620  0          EL_CONTINUE    = [7, 14, 1, 0],     ! CONTINUE FLAG
;  0621  0          EL_SUCCESS     = [7, 15, 1, 0],     ! SUCCESS FLAG
;  0622  0          EL_CODE        = [8, 0, 5, 0],      ! ERROR CODE
;  0623  0          EL_SUBCODE     = [8, 5, 11, 0],     ! SUB CODE
;  0624  0          EL_RETRY       = [20, 8, 8, 0],     ! RETRY COUNT
;  0625  0          EL_BLOCK       = [23, 0, 16, 0],    ! BLOCK NUMBER
;  0626  0          EL_BLOCK_TYPE  = [24, 12, 4, 0]     ! TYPE OF BLOCK NUMBER INFO RETURNED
;  0627  0          tes,
;  0628  0      !
;  0629  0      !****** INFORMATION ABOUT LAST RESPONSE PACKET
;  0630  0      !
;  0631  0      LAST_PKT_FIELDS =
;  0632  0          set
;  0633  0          LAST_HRD_ERR   = [0, 0, 16, 0],     ! FLAG INDICATES IF HARD ERROR OCCURED
;  0634  0          LAST_CRN_LO    = [1, 0, 16, 0],     ! COMMAND REFERENCE NUMBER
;  0635  0          LAST_CRN_HI    = [2, 0, 16, 0]      !
;  0636  0          tes,
;  0637  0      !
;  0638  0      !****** RDRX REGISTER FIELDS
;  0639  0      !
;  0640  0      RC_REG =
;  0641  0          set
;  0642  0          RC_ALL         = [0, 16, 0]         ! DEFINE ALL BITS
;  0643  0          tes;
```

```
;  0644  0     !*************************************************************************
;  0645  0     !                                                                        *
;  0646  0     !                       M A C R O S                                      *
;  0647  0     !                                                                        *
;  0648  0     !*************************************************************************
;  0649  0     !
;  0650  0     macro
;  0651  0     !
;  0652  0     !****** CST FIELDS. MODEL FOR WDS 3-12, 13-21, 22-30, AND 31-39.       ZZZ
;  0653  0     !
;  0654  0             D_ALL           = 0, 16, 0%,                    ! ALL FIELDS
;  0655  0             D_DISK_NUM      = 0, 4, 0%,                     ! DISK ADDRESS
;  0656  0             D_TYPE          = 4, 1, 0%,                     !DISK TYPE - 1 BIT      ZZZ
;  0657  0             D_UNIT          = 8, 4, 0%,                     ! DISK UNIT NUMBER (DRS UNIT)
;  0658  0             D_FATAL         = 12, 1, 0%,                    ! FATAL ERROR BIT
;  0659  0             D_STAT          = 13, 1, 0%,                    ! DISK STATUS BIT
;  0660  0             D_PRES          = 14, 1, 0%,                    ! DISK PRESENT BIT
;  0661  0             D_PROT          = 15, 1, 0%,                    ! DISK PROTECTION BIT
;  0662  0             D_BEG0          = 0, 16, 0%,        !BEGIN TRACK LO         ZZZ
;  0663  0             D_BEG1          = 0, 16, 0%,        !BEGIN TRACK HI         ZZZ
;  0664  0             D_END0          = 0, 16, 0%,        !END TRACK LO           ZZZ
;  0665  0             D_END1          = 0, 16, 0%,        !END TRACK HI           ZZZ
;  0666  0             D_NAME_0        = 0, 8, 0%,         ! NAME (FIRST  CHARACTER)
;  0667  0             D_NAME_1        = 8, 8, 0%,         ! NAME (SECOND CHARACTER)
;  0668  0             D_NAME_2        = 0, 8, 0%,         ! NAME (THIRD  CHARACTER)
;  0669  0             D_NAME_3        = 8, 8, 0%,         ! NAME (FOURTH CHARACTER)
;  0670  0     !       D_NUL           = 0, 16, 0%,        !NUL AFTER NAME         ZZZ
;  0671  0             D_DBN           = 0, 8, 0%,         !RELATIVE DBN           ZZZ
;  0672  0             DUPWRITE        = 12, 1, 0%,        !DUP WRITE FLAG         ZZZ
;  0673  0             D_ACTIVE        = 13, 1, 0%,        !ACTIVE STATE           ZZZ
;  0674  0             DUPERROR        = 14, 1, 0%,        !DUP ERROR FLAG         ZZZ
;  0675  0             NODUPMEDIA      = 15, 1, 0%,        !NO DUP MEDIA           ZZZ
;  0676  0             D_COUNT         = 0, 16, 0%,        !MSCP FUNCTION COUNTER  ZZZ
;  0677  0
;  0678  0
;  0679  0     !*********** BST FIELDS ************                                   ZZZ
;  0680  0     !                                                                    ZZZ
;  0681  0             HI_WRD          = 1, 0, 16, 0%,     !HI LBN                 ZZZ
;  0682  0             LO_WRD          = 0, 0, 16, 0%,     !LO LBN                 ZZZ
;  0683  0     !
;  0684  0     !****** BIT TEST
;  0685  0     !               (CAUTION: THE FIRST ARGUMENT IS THE ADDRESS AND NOT THE CONTENTS)
;  0686  0     !
; M 0687  0     BIT_TST (ADDR, EXPECTED) =
; M 0688          (if (.ADDR and EXPECTED) eql EXPECTED
; M 0689  0       then
; M 0690  0           TRUE
; M 0691  0       else
;  0692  0           FALSE )%,
;  0693  0     !
;  0694  0     !****** RDRX WRITE
;  0695  0     !
; M 0696  0     WRT_RDRX (0, FIELDNAM, IMAGE) =
```

```
; M 0697  0       begin
; M 0698  0       local
; M 0699  0           RC_REG;
; M 0700  0       RC_REG <%fieldexpand (FIELDNAM)> = IMAGE;
; M 0701  0       (.RDRX_ADDR + (%upval * 0)) = .RC_REG;
;   0702  0       end%;
```

```
;  0703  0     !**********************************************************************
;  0704  0     !                                                                     *
;  0705  0     !                       S T R U C T U R E S                           *
;  0706  0     !                                                                     *
;  0707  0     !**********************************************************************
;  0708  0     !
;  0709  0     !****** NIBBLE (4-BIT) VECTOR STRUCTURE
;  0710  0     !
;  0711  0     !structure
;  0712  0     !    NIBVECTOR [I; N] =
;  0713  0     !        [(N + 1) / 2]
;  0714  0     !        (NIBVECTOR + I / 2) <(I ^ 2) and 4, 4>;
;  0715  0     !
;  0716  0     !****** RDRX ACCESS ALGORITHM
;  0717  0     !
;  0718  0     structure
;  0719  0         RDRX [O, P, S, E] =
;  0720  1             begin
;  0721  1             local
;  0722  1                 RC_REG;
;  0723  1             RC_REG = .(RDRX + %upval * O) <O, %bpval, O>;
;  0724  1             RC_REG
;  0725  1             end
;  0726  0             <P, S, E>;
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAF0.REQ/LIST=ZRQAF0.LIS/LIBRARY=ZRQAF0.L16/SOURCE=PAGE:53

; Run Time:         00:06.5
; Elapsed Time:     00:25.9
; Lines/CPU Min:     6670
; Lexemes/CPU-Min: 35411
; Memory Used:  76 pages
; Library Precompilation Complete
```

```
;    0001  0     module ZRQAM1 (
;    0002  0
;    0003  0     %title 'RD/RX EXERCISER'
;    0004  0                     ident = 'V02.1',
;    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-T398F-MC
; C  0017  1
; C  0018  1             PRODUCT NAME:           CZRQAF0 RQDX1/RUX50 EXERCISER
; C  0019  1
; C  0020  1             PRODUCT DATE:           27-DEC-84
; 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
```

H2


```
;C 0074  1
;C 0075  1                       TABLE OF CONTENTS
;C 0076  1
;C 0077  1
;C 0078  1
;C 0079  1               1.0     GENERAL INFORMATION
;C 0080  1               1.1     PROGRAM ABSTRACT
;C 0081  1               1.2     SYSTEM REQUIREMENTS
;C 0082  1               1.2.1   HARDWARE REQUIREMENTS
;C 0083  1               1.2.2   SOFTWARE REQUIREMENTS
;C 0084  1               1.3     RELATED DOCUMENTS AND STANDARDS
;C 0085  1               1.4     DIAGNOSTIC HIERARCHY PREREQUISITES
;C 0086  1               1.5     ASSUMPTIONS
;C 0087  1               ..6     MEMORY MAP
;C 0088  1
;C 0089  1               2.0     OPERATING INSTRUCTIONS
;C 0090  1               2.1     HARDWARE QUESTIONS
;C 0091  1               2.2     SOFTWARE QUESTIONS
;C 0092  1
;C 0093  1               3.0     ERROR TYPES
;C 0094  1               3.1     ERROR INFORMATION
;C 0095  1               3.2     INITIALIZATION ERRORS
;C 0096  1               3.3     EXERCISER ERRORS
;C 0097  1               3.4     ERROR LOG MESSAGES
;C 0098  1               3.5     MSCP ERRORS
;C 0099  1               3.6     SAMPLE ERROR STATEMENT
;C 0100  1
;C 0101  1               4.0     PERFORMANCE AND PROGRESS REPORTS
;C 0102  1
;C 0103  1               5.0     TEST SUMMARY
;C 0104  1               5.1     INITIALIZATION SUBTEST
;C 0105  1               5.2     EXERCISER
;C 0106  1               5.3     DROP UNIT SUMMARY
;C 0107  1
;C 0108  1               6.0     ERROR LIST
;C 0109  1
;C 0110  1               7.0     DATA PATTERNS
```

```
; C 0111  1
; C 0112  1
; C 0113  1
; C 0114  1
; 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
```

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

This program will functionally verify and exercise RQDX1 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 RQDX1 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 0162  1               1.6     MEMORY MAP
; C 0163  1                       ------ -
; C 0164  1
; C 0165  1
; C 0166  1               Memory layout on 28k machine - XXDP environment
; C 0167  1               ------ -- -------------------- ----- ---------
; C 0168  1
; C 0169  1                                                       ADDRESS
; C 0170  1
; C 0171  1                       +--------- ----------------+     160000
; C 0172  1                       !                          !
; C 0173  1                       !      XXDP+ MONITOR       !
; C 0174  1                       !                          !
; C 0175  1                       +-------------- ---------- +
; C 0176  1                       !                          !
; C 0177  1                       !  XXDP+   TTY I/O AREA    !
; C 0178  1                       !                          !
; C 0179  1                       +----------------- -    --+
; C 0180  1                       !                          !
; C 0181  1                       !                          !
; C 0182  1                       !          DRS             !
; C 0183  1                       !                          !
; C 0184  1                       !                          !
; C 0185  1                       + ------------------    --+     121000
; C 0186  1                       !                          !
; C 0187  1                       !       EXERCISER          !
; C 0188  1                       !                          !
; C 0189  1                       +          ---------------+     2000
; C 0190  1                       !                          !
; C 0191  1                       !         STACKS           !
; C 0192  1                       !                          !
; C 0193  1                       +        -  -- ----------  +     400
; C 0194  1                       !                          !
; C 0195  1                       !       VECTOR AREA        !
; C 0196  1                       !                          !
; C 0197  1                       +--- ----- - -  -          +     0
; C 0198  1
; C 0199  1
; C 0200  1               In a machine  with more memory,  free space will occur
; C 0201  1               between the exerciser and the DRS.
```

```
; C 0202  1
; C 0203  1
; C 0204  1
; C 0205  1
; 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
```

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. The 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 RQDX1 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 0251  1              4.  VECTOR ADDRESS (O) 154 ?
; C 0252  1
; C 0253  1                  Answer with the interrupt vector ofthe same RQDX1 or RUX50
; C 0254  1                  controller described in the above question. A vector address
; C 0255  1                  in the range of 4 to 774 may be specified. 154 is the default.
; C 0256  1
; C 0257  1              5.  BR LEVEL [USUALLY 4-RQDX1 5-RUX50] (D) 4 ?
; C 0258  1
; C 0259  1                  Answer with the bus request interrupt level used by  the
; C 0260  1                  above controller. Levels 4 through 7 are acceptable. 4 is the
; C 0261  1                  default.
; C 0262  1
; C 0263  1
; C 0264  1              6.  DRIVE NUMBER (D) 0 ?
; C 0265  1
; C 0266  1                  Enter the logical unit number for one drive associated
; C 0267  1                  with the IP address above. Drive numbers are in the
; C 0268  1                  range of 0 through 15. The number entered here must
; C 0269  1                  match the unit plug on the front  panel of the drive, and
; C 0270  1                  must be within the range implied by the jumper (LUN0-7)
; C 0271  1                  on the RQDX1 or RUX50 controller board. 0 is the default
; C 0272  1                  answer.
; C 0273  1
; C 0274  1
; C 0275  1              7.  ALSO RUN DUP EXERCISER (L) N ?
; C 0276  1
; C 0277  1                  ANSWER Y TO HAVE TESTS PERFORMED SPECIFICALLY WITH THE
; C 0278  1                  DIAGNOSTIC BLOCKS. SUCH DUP TESTING, IF SELECTED, IS
; C 0279  1                  INTERLEAVED WITH NORMAL EXERCISER TESTING.
; C 0280  1
; C 0281  1
; C 0282  1
; C 0283  1              8.  WRITE ON DIAGNOSTIC AREA (L) N ?
; C 0284  1
; C 0285  1                  IF THE DUP EXERCISER IS CHOSEN TO BE RUN, ANSWERING Y
; C 0286  1                  TO THIS QUESTION ADDS WRITE TESTING IN THE DIAGNOSTIC
; C 0287  1                  BLOCK AREA. THIS CAN BE USED TO DETERMINE WHETHER A
; C 0288  1                  UNIT IS WRITING PROPERLY, WITHOUT USING THE CUSTOMER AREA.
; C 0289  1
; C 0290  1
; C 0291  1
; C 0292  1              9.  TEST ENTIRE CUSTOMER DATA AREA OF THIS DISK (L) Y?
; C 0293  1
; C 0294  1                  This question is asked to give the opportunity of limiting the
; C 0295  1                  addressing range over which the testing will be performed.
; C 0296  1                  An affirmative answer will cause no limits to be imposed for
; C 0297  1                  the unit in question. A negative answer will cause limits
; C 0298  1                  to be imposed, as defined by the following four questions.
; C 0299  1
; C 0300  1
; C 0301  1             10.  LOWER OCTAL WORD OF BEGINNING LBN ADDRESS (O) 0?
; C 0302  1
; C 0303  1                  Enter in octal the less significant 16-bit word of the lowest
```

```
; C 0304  1               LBN address in the test range. The value may be from 000000
; C 0305  1               to 177777.
; C 0306  1
; C 0307  1
; C 0308  1           11.  HIGHER OCTAL WORD OF BEGINNING LBN ADDRESS (0) 0?
; C 0309  1
; C 0310  1               Enter in octal the more significant 16-bit word of the
; C 0311  1               lowest LBN address in the test range.
; C 0312  1
; C 0313  1
; C 0314  1           12.  LOWER OCTAL WORD OF ENDING LBN ADDRESS (0) 150477?
; C 0315  1
; C 0316  1               Enter in octal the less significant 16-bit word of the
; C 0317  1               highest LBN address in the test range. 150477 is the
; C 0318  1               highest LBN address for an RD52.
; C 0319  1
; C 0320  1
; C 0321  1           13.  HIGHER OCTAL WORD OF ENDING LBN ADDRESS (0) 0?
; C 0322  1
; C 0323  1               Enter in octal the more significant 16-bit word of the
; C 0324  1               highest LBN address in the test range.
; C 0325  1
; C 0326  1
; C 0327  1                Note:
; C 0328  1                     The four previous questions are usually software
; C 0329  1                     Parameter questions, but since three different disk
; C 0330  1                     drives exist on the subsystem, this becomes a unit
; C 0331  1                     by unit question.  It is possible to specify an LBN
; C 0332  1                     which is too large since we are dealing with different
; C 0333  1                     drives. The program will check for block number bounds,
; C 0334  1                     and, if they are exceeded, will assign the maximum
; C 0335  1                     bounds for that drive.
; C 0336  1
; C 0337  1
; C 0338  1
; C 0339  1       14.  WRITE ON CUSTOMER DATA AREA ON THIS DISK UNIT (L) ?
; C 0340  1
; C 0341  1           Answering YES will destroy any customer data that is on
; C 0342  1           the disk; therefore, the following warning message will
; C 0343  1           appear, followed by a confirmation prompt:
; C 0344  1
; C 0345  1           ** WARNING - CUSTOMER DATA AREA WILL BE OVERWRITTEN! ...
; C 0346  1            CONFIRM (L) ?
; C 0347  1
; C 0348  1           This question will default to NO if the operator has de
; C 0349  1           cided to bypass the hardware questions. Otherwise,
; C 0350  1           there is no default.
```

```
; C 0351  1
; C 0352  1
; C 0353  1
; C 0354  1
; 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
```

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 #### (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 0397  1          5.  PERCENTAGE OF 'FIXED DISK' OPERATIONS OUT OF TOTAL OPERATIONS (D) 99 ?
; C 0398  1
; C 0399  1              In  order to maintain  typical usage for the devices of
; C 0400  1              this  exercise,  a certain percentage of operations must
; C 0401  1              be directed to the RD51/52s  (the rest go to the RX50s).
; C 0402  1              It turns out that this percentage is very high (as ind'
; C 0403  1              cated by the 99% figure given as the default). It may be
; C 0404  1              desirable in  some cases to  direct more activity to the
; C 0405  1              RX50s.  This is easily  done by directing a smaller per
; C 0406  1              centage of the  operations to the RD51/52s.  The numbers
; C 0407  1              associated  with  usage  are  adjusted internally by the
; C 0408  1              program according to drive type and percentage.
; C 0409  1
; C 0410  1          6.  CLEAR STATISTICAL TABLES AFTER PRINTING (L) N ?
; C 0411  1
; C 0412  1              Answering YES causes the statistical fields to be clear-
; C 0413  1              ed to zero after the report is printed (either at end of
; C 0414  1              pass,  or at  operator  request).  Otherwise, cumulative
; C 0415  1              totals are maintained.
; C 0416  1
; C 0417  1          7.  REWRITE BLOCKS WHEN "FORCED ERROR" DETECTED ON READS (L) Y ?
; C 0418  1
; C 0419  1              On encountering a bad block on the RD51 or RD52 disk
; C 0420  1              (during either a read or a write operation), the RQDX1
; C 0421  1              or RUX50 controller will revector the logical block to
; C 0422  1              another physical location on the disk. This operation is
; C 0423  1              transparent to the user. However, if the revectoring was
; C 0424  1              done  subsequent to  a write  operation  (i.e  the write
; C 0425  1              operation detected the bad block),  the data is  flagged
; C 0426  1              with a "Forced Error" code,  signifying that the data at
; C 0427  1              the  revectored location is suspect.  The controller re-
; C 0428  1              turns an error code  whenever the block is re-read.  An
; C 0429  1              swer 'Yes' to the question to force a WRITE operation on
; C 0430  1              the same block whenever a "Forced Error" flag is detect
; C 0431  1              ed on a read.  This is to avoid the same error code (the
; C 0432  1              "Forced  Error") being  reported  for  the  same  block
; C 0433  1              repeatedly. The re-write will, however,  take place only
; C 0434  1              if writes are enabled for the particular disk unit.
; C 0435  1
; C 0436  1          8.  HALT ON BAD-BLOCK HARD ERRORS (#s 35, 38) (L) Y ?
; C 0437  1
; C 0438  1              When the Exerciser is  run with the DRS  "Halt on Error"
; C 0439  1              switch set (eg. START/FLAGS:HOE), the Exerciser halts on
; C 0440  1              an encountering  ANY error.  If it is  desired that  the
; C 0441  1              testing continue on a bad-block error, even with the HOE
; C 0442  1              switch set, answer No to the question.
; C 0443  1
; C 0444  1          9.  HALT ON OTHER HARD ERRORS (#s 31-34, 36-37, 39-45) (L) Y ?
; C 0445  1
; C 0446  1              This  question is similar  to question 8,  but refers to
; C 0447  1              non-bad block type of Hard Errors.
```

```
; C 0448  1          10.  HALT ON SOFT ERRORS (0s 50-54) (L) N ?
; C 0449  1
; C 0450  1               This question  is similar to  question 8,  but refers to
; C 0451  1               Soft Errors.
; C 0452  1
; C 0453  1          11.  COUNT EACH RETRY AS A SEPARATE SOFT ERROR (L) N ?
; C 0454  1
; C 0455  1               On encountering any error on a read/write,  the control
; C 0456  1               ler  retries the operation  a number  of times.  If  the
; C 0457  1               operation is eventually succeseful,  this is reported as
; C 0458  1               a Soft Error.  The error log packet contains  the number
; C 0459  1               of  retries performed  before the operation was success-
; C 0460  1               ful.  Normally, the whole sequence of retries is classi
; C 0461  1               fied as one Soft Error. Answer Yes to the question if it
; C 0462  1               is  desired to  count each internal  retry attempt  as a
; C 0463  1               seperate Soft Error.
; C 0464  1
; C 0465  1          12.  RANDOM SEEK MODE (L) Y ?
; C 0466  1
; C 0467  1               Answer YES to cause block numbers to be chosen randomly.
; C 0468  1               Answer  NO) to cause block numbers to be selected sequen-
; C 0469  1               tially.
; C 0470  1
; C 0471  1          13.  UNITS TO BE SELECTED AT RANDOM (NO, IMPLIES SEQUENTIAL) (L) N ?
; C 0472  1
; C 0473  1               This  question is  optionally asked if the answer to the
; C 0474  1               previous  question is N[o].  The  selection of units for
; C 0475  1               sequential  operations is affected by the answer to this
; C 0476  1               question.  If the default  answer is chosen (N[o]), then
; C 0477  1               units  shall be  selected in a  predetermined manner  in
; C 0478  1               accordance  with the typical  seek time margins for each
; C 0479  1               drive.  If the alternate answer is chosen (Y[es]),  then
; C 0480  1               the  units will be chosen  at random  in accordance with
; C 0481  1               the percentages specified in Software question 4.
; C 0482  1
; C 0483  1          14.  READ-COMPARES PERFORMED AT THE CONTROLLER (L) Y ?
; C 0484  1
; C 0485  1               Answering YES causes all read commands  to  include  the
; C 0486  1               "compare"  modifier.   This  essentially forces the con-
; C 0487  1               troller to perform two read operations on the same  disk
; C 0488  1               address, and to compare the results.
; C 0489  1
; C 0490  1               The following message will appear after the operator has
; C 0491  1               answered this question:
; C 0492  1
; C 0493  1
; C 0494  1          15.  RUNNING UNDER THE A.P.T. MONITOR (L) N ?
; C 0495  1
; C 0496  1               THIS QUESTION SHOULD BE ANSWERED N (DEFAULT) IN THE FIELD.
; C 0497  1               IT ENABLES THE PROGRAM TO KNOW THAT IT IS RUNNING UNDER A
; C 0498  1               SPECIAL (AUTOMATED PRODUCT TEST) MONITOR.
```

```
; C 0499  1
; C 0500  1
; C 0501  1
; C 0502  1          THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED DISK UNITS.
; C 0503  1
; C 0504  1
; C 0505  1          16.  WRITE-COMPARES PERFORMED AT THE CONTROLLER (L) N ?
; C 0506  1
; C 0507  1               Answering YES causes all write I/O requests to be
; C 0508  1               changed to write-compare.  After each write, the con
; C 0509  1               troller will read the data and compare it to data
; C 0510  1               re-obtained from the host.
; C 0511  1
; C 0512  1
; C 0513  1          17.  CHECK ALL WRITES AT HOST BY READING (L) Y ?
; C 0514  1
; C 0515  1               This question will only be asked if the previous ques-
; C 0516  1               tion was answered NO.  Answering YES causes all writes
; C 0517  1               to be checked by the host by reading the data immediate-
; C 0518  1               ly after the write operation.  This option consumes
; C 0519  1               extra CPU time, and doubles the amount of storage re-
; C 0520  1               quired for writes.  Therefore, it is only recommended
; C 0521  1               when drive write-compare operations are suspect.
; C 0522  1
; C 0523  1
; C 0524  1          18.  USER-DEFINED DATA PATTERN (L) N ?
; C 0525  1
; C 0526  1               An answer of YES allows the operator to define his/her
; C 0527  1               own data pattern to be used in all write operations.  A
; C 0528  1               NO answer will allow the operator to select a
; C 0529  1               pre-defined data pattern in the next question.
; C 0530  1
; C 0531  1
; C 0532  1          19.  SELECT PRE-DEFINED DATA PATTERN (0 FOR SEQUENTIAL SELEC-
; C 0533  1               TION) (D) 0 ?
; C 0534  1
; C 0535  1               There are 21 pre-defined data patterns available,
; C 0536  1               selected as 1 to 21 (see section 4.9).  A zero answer
; C 0537  1               will cause patterns 1 to 21 to be sequentially selected
; C 0538  1               for each write.  (Note that pattern 1 consists entirely
; C 0539  1               of random numbers).
; C 0540  1
; C 0541  1
; C 0542  1          20.  NUMBER OF WORDS IN DATA PATTERN (16 MAXIMUM)  (D)  16  ?
; C 0543  1               PATTERN VALUES (O) ?
; C 0544  1
; C 0545  1               These questions will only be asked if the operator has
; C 0546  1               decided to define his/her own data pattern.  The actual
; C 0547  1               bit patterns will be entered as octal (PDP-11).
```

```
; C 0548  1                3.0   ERROR TYPES
; C 0549  1                      - --- -
; C 0550  1
; C 0551  1                      This program has four types of error  classifications;
; C 0552  1                      system fatal, drive fatal, hard and soft.
; C 0553  1
; C 0554  1                      SYSTEM FATAL ERRORS
; C 0555  1                      ------------------
; C 0556  1
; C 0557  1                      System fatal errors are used to indicate that an error
; C 0558  1                      was detected by the  Diagnostic Supervisor in relation
; C 0559  1                      to loading/controlling the diagnostic process.
; C 0560  1
; C 0561  1                      The content  of each  error is  such that it should be
; C 0562  1                      self explanatory.  However,  the messages utilize some
; C 0563  1                      terms that are specific to the disk subsystem, and may
; C 0564  1                      require some getting use to.
; C 0565  1
; C 0566  1
; C 0567  1                      DRIVE FATAL ERRORS
; C 0568  1                      ------------------
; C 0569  1
; C 0570  1                      Drive fatal errors are a result of:
; C 0571  1
; C 0572  1                      an error that is  considered fatal to  the drive,  but
; C 0573  1                      testing will continue.
; C 0574  1
; C 0575  1
; C 0576  1                      HARD ERRORS
; C 0577  1                      -----------
; C 0578  1
; C 0579  1                      Hard errors are a result of:
; C 0580  1
; C 0581  1                              1.  retries of a soft error or *
; C 0582  1                              2.  a non-recoverable error
; C 0583  1                              3.  a soft error if retries are not set.
; C 0584  1
; C 0585  1                              *  Note: Retries are executed in the controller
; C 0586  1
; C 0587  1                      SOFT ERRORS
; C 0588  1                      -------- --
; C 0589  1
; C 0590  1                      Soft errors  are media related errors.  All soft errors
; C 0591  1                      will be retried by the controller.
; C 0592  1
; C 0593  1                      Note: Soft errors are retrieved from the controller via
; C 0594  1                      the error log  capabilities of  MSCP.
```

```
; C 0595  1
; C 0596  1
; C 0597  1
; C 0598  1
; 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
```

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 effect 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

```
; C 0644  1
; C 0645  1
; C 0646  1
; C 0647  1
; C 0648  1
; C 0649  1
; C 0650  1
; C 0651  1
; C 0652  1
; C 0653  1
; C 0654  1
; C 0655  1
; C 0656  1
; C 0657  1
; C 0658  i
; C 0659  1
; C 0660  1
; C 0661  1
; C 0662  1
; C 0663  1
; C 0664  1
; C 0665  1
; C 0666  1
; C 0667  1
; C 0668  1
; C 0669  1
; C 0670  1
; C 0671  1
; C 0672  1
; C 0673  1
; C 0674  1
; C 0675  1
; C 0676  1
; C 0677  1
; C 0678  1
; C 0679  1
; C 0680  1
; C 0681  1
; C 0682  1
; C 0683  1
; C 0684  1
; C 0685  1
; C 0686  1
; C 0687  1
; C 0688  1
; C 0689  1
; C 0690  1
```

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.

```
; C 0691  1          3.6   SAMPLE ERROR STATEMENT
; C 0692  1                --------   -- -------
; C 0693  1
; C 0694  1
; C 0695  1          The errors listed by the exerciser are usually very des-
; C 0696  1          criptive and are self explanatory.  The following is an example
; C 0697  1          error statement.  This error statement is the extended error
; C 0698  1          message.
; C 0699  1
; C 0700  1
; C 0701  1          (example)                          (comments)
; C 0702  1
; C 0703  1          DISK XXX                           !DISK UNIT NUMBER
; C 0704  1          INVALID COMMAND                    !MAJOR STATUS CODE RECEIVED BACK
; C 0705  1          SUB-CODE XXXX                      !SUB-CODE OF GIVEN COMMAND
; C 0706  1          COMMAND: READ                      !COMMAND GIVEN TO DRIVE
; C 0707  1          LBN: XXXXX                         !LOGICAL BLOCK NUMBER GIVEN
; C 0708  1          BYTE COUNT IN COMMAND XXXXX        !NUMBER OF BYTES WANTED TO READ
; C 0709  1          ACTUAL # OF BYTES TRANSFERRED XXXXX !NUMBER OF BYTES ACTUALLY READ
; C 0710  1
; C 0711  1
; C 0712  1
; C 0713  1          The status code in an end messages is broken into two
; C 0714  1          pieces. The first 5 bits represent the major status which is given
; C 0715  1          by the "invalid command" message.  The 11 remaining bits represent
; C 0716  1          the sub-code, which tells in greater detail the error in the con-
; C 0717  1          troller.  The LBN is the logical block on the disk the controller
; C 0718  1          was trying to read.  The byte count refers to the number of bytes
; C 0719  1          the controller was going to read off the LBN. The actual number of
; C 0720  1          bytes transferred refers to the number of bytes read before the
; C 0721  1          error.
```

```
; C 0722  1              4.0   PERFORMANCE AND PROGRESS REPORTS
; C 0723  1                    - -- ----- - ----     - -
; C 0724  1
; C 0725  1
; C 0726  1                 A summary report is printed at the end of  each  pass of
; C 0727  1                 the  Exerciser or upon demand by the operator. The fields may
; C 0728  1                 be cleared to zero after the  report is printed  depending on
; C 0729  1                 the operator's  response to this option in the software ques
; C 0730  1                 tions. Any units added to the test cycle will also begin with
; C 0731  1                 cleared statistics.
; C 0732  1
; C 0733  1                 Errors are grouped into two  basic categories:  hard and
; C 0734  1                 soft.  Each is sub divided into four more categories, depend
; C 0735  1                 ing on the most probable classification for that error.
; C 0736  1
; C 0737  1                      The sub categories are:
; C 0738  1
; C 0739  1                 1. disk related errors
; C 0740  1
; C 0741  1                 2. seek (or format) related errors
; C 0742  1
; C 0743  1                 3. controller or drive related errors
; C 0744  1
; C 0745  1                 4. host (the CPU) related errors.
; C 0746  1
; C 0747  1                      All numeric values are in decimal radix.
; C 0748  1
; C 0749  1           UNT         # OF BYTS  # OF    BYTES    -- HRD ERS --    -- SFT ERS  -
; C 0750  1            #  TYPE READS READ WRITES WRITTEN DAT SEK DRV HST DAT SEK DRV HST
; C 0751  1           --- ---- ----- ---- ------ ------- ----------  -   ---  - -  --  -
; C 0752  1
; C 0753  1            X  XXXX  XXXX XXXX  XXXXX  XXXXXX  X   X   X   X   X   X   X   X
; C 0754  1            .   ..    ..   ..    ...    ....   .   .   .   .   .   .   .   .
; C 0755  1            .   ..    ..   ..    ...    ....   .   .   .   .   .   .   .   .
```

```
; C 0756  1               5.0   TEST SUMMARY
; C 0757  1                     ----- -- --
; C 0758  1
; C 0759  1                  This exerciser consists of two parts: the initialization subtest,
; C 0760  1                  and the performance exerciser. The operator is not able to select
; C 0761  1                  which of these two parts he/she wishes to run; they both must be
; C 0762  1                  executed.
; C 0763  1
; C 0764  1                5.1  INITIALIZATION SUBTEST
; C 0765  1                     ----------------------
; C 0766  1
; C 0767  1                  The  purpose of  this subtest  is to verify the hardware
; C 0768  1                  configuration as specified by the operator, and to bring each
; C 0769  1                  unit online.  The Initialization  Subtest will always precede
; C 0770  1                  the execution of any other test.
; C 0771  1
; C 0772  1                  First, the presence of each drive register will be veri-
; C 0773  1                  fied,  along with a  check on the  BR level  specified by the
; C 0774  1                  operator.  Then,  an  initialization  will be issued to  each
; C 0775  1                  controller configured for testing.  When  the initialization
; C 0776  1                  sequence has been completed, an attempt will be made to bring
; C 0777  1                  each unit online.  If this succeeds,  one  or two MSCP reads
; C 0778  1                  will be issued to the inner-most LBN of each selected disk to
; C 0779  1                  to ensure that each disk drive can seek and be read.
; C 0780  1
; C 0781  1                  Any drive-fatal or hard errors  encountered  during this
; C 0782  1                  test  will  cause the  appropriate unit(s) to be dropped.  If
; C 0783  1                  basic error messages are enabled, then the program will print
; C 0784  1                  out the specific reason for dropping the unit(s). Henceforth,
; C 0785  1                  the  failed  unit(s) will  not be  tested unless the operator
; C 0786  1                  intervenes (adds unit(s) or restarts Exerciser).
; C 0787  1
; C 0788  1                  Upon  successful completion  of the  Initialization Sub
; C 0789  1                  test, the program will begin executing the Exerciser.
; C 0790  1
; C 0791  1              5.2  EXERCISER
; C 0792  1                   --------
; C 0793  1
; C 0794  1                  The purpose of this subtest is to exercise the  disk  drives
; C 0795  1                  in a manner similar to the typical usage under standard oper-
; C 0796  1                  ating systems.  Execution of this test should give an indica-
; C 0797  1                  tion of the operating performance of the disk drive subunits.
; C 0798  1                  This test  will utilize  random disk addresses,  random  word
; C 0799  1                  counts,  and data  patterns,  all subject  to the  limits and
; C 0800  1                  specifications made by the operator. All protected disks will
; C 0801  1                  be subject to read-only  operations,  while unprotected disks
; C 0802  1                  may be read or written, depending on the answers given to the
; C 0803  1                  software parameter questions.  End-of-pass  will be declared
; C 0804  1                  when the specified number  of bytes have been transferred for
; C 0805  1                  all the disks taken as a whole.
```

```
;C 0806  1          If a read/write error occurs during this test,  then the
;C 0807  1          controller will initiate an  appropriate number of re-
;C 0808  1          tries.  If all retries fail, then a hard error will be repor
;C 0809  1          ted to the host,  an error message  will  be displayed on the
;C 0810  1          console  terminal and  the error will be tallied for the sum
;C 0811  1          mary report. The unit will be dropped if the hard error count
;C 0812  1          has exceeded the specified limit.
;C 0813  1
;C 0814  1
;C 0815  1
;C 0816  1      5.3   DROP UNIT SUMMARY
;C 0817  1            ----------------
;C 0818  1          During the Initialization Subtest, individual units will
;C 0819  1          be dropped from  the test  sequence if they are unable  to be
;C 0820  1          brought online or the operator specified drive does not match
;C 0821  1          the hardware.
;C 0822  1
;C 0823  1          During  the Exercise,  the program will  drop a unit for
;C 0824  1          one of  three reasons.  The normal path is for  each unit  to
;C 0825  1          complete the transfer of N megabytes, where N is specified by
;C 0826  1          the  operator  during SW  questioning  and  be  soft-dropped.
;C 0827  1          Otherwise,  a unit will be hard-dropped if the number of hard
;C 0828  1          errors encountered exceeds  the operator-specified limit,  or
;C 0829  1          if a fatal error is detected. Units hard-dropped may later be
;C 0830  1          added to the  test cycle.  However,  statistics for the hard-
;C 0831  1          added  unit  will be  cleared to  zero;  if a transfer  limit
;C 0832  1          was specified,  in which case the unit was soft-dropped,  the
;C 0833  1          statistics may  or may not be cleared  depending on the oper-
;C 0834  1          ators answer to Software question 12.
```

6.0 ERROR CODES GENERATED BY THIS EXERCISER

SYSTEM FATAL ERRORS

1	More than 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 seauence 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 correpeated.
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.

N7


```
; C 0937  1
; C 0938  1
; C 0939  1                41  Drive error.                        See #40.
; C 0940  1
; C 0941  1                42  Host write compare error.           Error detected when Host
; C 0942  1                                                        CPU compared the data
; C 0943  1                                                        written and read back. May
; C 0944  1                                                        be a problem with the Host
; C 0945  1                                                        or Controller RAM.
; C 0946  1
; C 0947  1                43 Message from internal diagnostics    See #40.
; C 0948  1
; C 0949  1                44  Duplicate unit number detected      -
; C 0950  1                    by the Controller.
; C 0951  1
; C 0952  1                45  Unknown end code received.          Problem with the Controller
; C 0953  1                                                        microcode or the port/
; C 0954  1                                                        DMA interface.
; C 0955  1
; C 0956  1
; C 0957  1
; C 0958  1
; C 0959  1                SOFT ERRORS
; C 0960  1                ----------
; C 0961  1
; C 0962  1                50  Controller error.                  See error-log packet for
; C 0963  1                                                        details as the exact cause
; C 0964  1                                                        may not be evident.
; C 0965  1
; C 0966  1                51  Host memory access error.          See #50.
; C 0967  1
; C 0968  1                52  Disk transfer error.               See #50.
; C 0969  1
; C 0970  1                53  'Standard  Disk  Interconnect'     See #50.
; C 0971  1                    error.
; C 0972  1
; C 0973  1                54  'Small Disk' error.                See #50.
; C 0974  1
; C 0975  1
; C 0976  1
; C 0977  1
; C 0978  1                DUP ERRORS
; C 0979  1                ----------
; C 0980  1
; C 0981  1                60 Unable to load local controller DUP media.
; C 0982  1
. C 0983  1                61 (Not used)
; C 0984  1
; C 0985  1                62 Illegal unit number.
; C 0986  1
; C 0987  1                63 Illegal relative or physical block.
; C 0988  1
; C 0989  1                64 Device error.
```

```
; C 0990  1
; C 0991  1                65 Zero length message.
; C 0992  1
; C 0993  1                66 Unknown DUP status code.
; C 0994  1
; C 0995  1                67 Invalid command.
```

```
; C 0996  1
; C 0997  1
; C 0998  1
; C 0999  1                DUP ERRORS  (CONTINUED)
; C 1000  1
; C 1001  1
; C 1002  1
; C 1003  1                68 No region available.
; C 1004  1
; C 1005  1                69 No region suitable.
; C 1006  1
; C 1007  1                70 Program not known.
; C 1008  1
; C 1009  1                71 Load failure.
; C 1010  1
; C 1011  1                72 Standalone.
; C 1012  1
; C 1013  1                73 Unknown DUP status code.
; C 1014  1
; C 1015  1
```

```
;  C 1016  1               7.0   DATA PATTERNS
;  C 1017  1                     ----- - ---
;  C 1018  1
;  C 1019  1
;  C 1020  1                          HEX      OCTAL                BINARY
;  C 1021  1                          ---      ---                  ---
;  C 1022  1
;  C 1023  1          Pattern 1            R A N D O M   N U M B E R S
;  C 1024  1
;  C 1025  1          Pattern 2       0000     000000          0 000 000 000 000 000
;  C 1026  1
;  C 1027  1          Pattern 3       FFFF     177777          1 111 111 111 111 111
;  C 1028  1
;  C 1029  1          Pattern 4       8888     105613          1 000 101 110 001 011
;  C 1030  1
;  C 1031  1          Pattern 5       3333     031463          0 011 001 100 110 011
;  C 1032  1
;  C 1033  1          Pattern 6       3091     030221          0 011 000 010 010 001
;  C 1034  1
;  C 1035  1          Pattern 7       0001     000001          0 000 000 000 000 001
;  C 1036  1                          0003     000003          0 000 000 000 000 011
;  C 1037  1                          0007     000007          0 000 000 000 000 111
;  C 1038  1                          000F     000017          0 000 000 000 001 111
;  C 1039  1                          001F     000037          0 000 000 000 011 111
;  C 1040  1                          003F     000077          0 000 000 000 111 111
;  C 1041  1                          007F     000177          0 000 000 001 111 111
;  C 1042  1                          00FF     000377          0 000 000 011 111 111
;  C 1043  1                          01FF     000777          0 000 000 111 111 111
;  C 1044  1                          03FF     001777          0 000 001 111 111 111
;  C 1045  1                          07FF     003777          0 000 011 111 111 111
;  C 1046  1                          0FFF     007777          0 000 111 111 111 111
;  C 1047  1                          1FFF     017777          0 001 111 111 111 111
;  C 1048  1                          3FFF     037777          0 011 111 111 111 111
;  C 1049  1                          7FFF     077777          0 111 111 111 111 111
;  C 1050  1                          FFFF     177777          1 111 111 111 111 111
;  C 1051  1
;  C 1052  1          Pattern 8       FFFE     177776          1 111 111 111 111 110
;  C 1053  1                          FFFC     177774          1 111 111 111 111 100
;  C 1054  1                          FFF8     177770          1 111 111 111 111 000
;  C 1055  1                          FFF0     177760          1 111 111 111 110 000
;  C 1056  1                          FFE0     177740          1 111 111 111 100 000
;  C 1057  1                          FFC0     177700          1 111 111 111 000 000
;  C 1058  1                          FF80     177600          1 111 111 110 000 000
;  C 1059  1                          FF00     177400          1 111 111 100 000 000
;  C 1060  1                          FE00     177000          1 111 111 000 000 000
;  C 1061  1                          FC00     176000          1 111 110 000 000 000
;  C 1062  1                          F800     174000          1 111 100 000 000 000
;  C 1063  1                          F000     170000          1 111 000 000 000 000
;  C 1064  1                          E000     160000          1 110 000 000 000 000
;  C 1065  1                          C000     140000          1 100 000 000 000 000
;  C 1066  1                          8000     100000          1 000 000 000 000 000
;  C 1067  1                          0000     000000          0 000 000 000 000 000
```

```
; C 1068  1          Pattern 9       0000    000000          0 000 000 000 000 000
; C 1069  1                          0000    000000          0 000 000 000 000 000
; C 1070  1                          0000    000000          0 000 000 000 000 000
; C 1071  1                          FFFF    177777          1 111 111 111 111 111
; C 1072  1                          FFFF    177777          1 111 111 111 111 111
; C 1073  1                          FFFF    177777          1 111 111 111 111 111
; 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                          0000    000000          0 000 000 000 000 000
; C 1079  1                          FFFF    177777          1 111 111 111 111 111
; 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                          0000    000000          0 000 000 000 000 000
; C 1083  1                          FFFF    177777          1 111 111 111 111 111
; C 1084  1
; C 1085  1          Pattern 10      B6D9    133331          1 011 011 011 011 001
; C 1086  1
; C 1087  1          Pattern 11      5555    052525          0 101 010 101 010 101
; C 1088  1                          5555    052525          0 101 010 101 010 101
; C 1089  1                          5555    052525          0 101 010 101 010 101
; C 1090  1                          AAAA    125252          1 010 101 010 101 010
; C 1091  1                          AAAA    125252          1 010 101 010 101 010
; C 1092  1                          AAAA    125252          1 010 101 010 101 010
; 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                          5555    052525          0 101 010 101 010 101
; C 1098  1                          AAAA    125252          1 010 101 010 101 010
; 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                          5555    052525          0 101 010 101 010 101
; C 1102  1                          AAAA    125252          1 010 101 010 101 010
```

```
; C 1103  1          Pattern 12      2020    026455      0 010 110 100 101 101
; C 1104  1                          2020    026455      0 010 110 100 101 101
; C 1105  1                          2020    026455      0 010 110 100 101 101
; C 1106  1                          D2D2    151322      1 101 001 011 010 010
; C 1107  1                          D2D2    151322      1 101 001 011 010 010
; C 1108  1                          D2D2    151322      1 101 001 011 010 010
; C 1109  1                          2020    026455      0 010 110 100 101 101
; C 1110  1                          2020    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                          2020    026455      0 010 110 100 101 101
; C 1114  1                          2020    026455      0 010 110 100 101 101
; C 1115  1                          D2D2    151322      1 101 001 011 010 010
; C 1116  1                          2020    026455      0 010 110 100 101 101
; C 1117  1                          D2D2    151322      1 101 001 011 010 010
; C 1118  1                          2020    026455      0 010 110 100 101 101
; C 1119  1                          D2D2    151322      1 101 001 011 010 010
; C 1120  1                          2020    026455      0 010 110 100 101 101
; C 1121  1                          D2D2    151322      1 101 001 011 010 010
; C 1122  1                          2020    026455      0 010 110 100 101 101
; C 1123  1
; C 1124  1          Pattern 13      6DB6    066666      0 110 110 110 110 110
; C 1125  1
; C 1126  1          Pattern 14      0001    000001      0 000 000 000 000 001
; C 1127  1                          0002    000002      0 000 000 000 000 010
; C 1128  1                          0004    000004      0 000 000 000 000 100
; C 1129  1                          0008    000010      0 000 000 000 001 000
; C 1130  1                          0010    000020      0 000 000 000 010 000
; C 1131  1                          0020    000040      0 000 000 000 100 000
; C 1132  1                          0040    000100      0 000 000 001 000 000
; C 1133  1                          0080    000200      0 000 000 010 000 000
; C 1134  1                          0100    000400      0 000 000 100 000 000
; C 1135  1                          0200    001000      0 000 001 000 000 000
; C 1136  1                          0400    002000      0 000 010 000 000 000
; C 1137  1                          0800    004000      0 000 100 000 000 000
; C 1138  1                          1000    010000      0 001 000 000 000 000
; C 1139  1                          2000    020000      0 010 000 000 000 000
; C 1140  1                          4000    040000      0 100 000 000 000 000
; C 1141  1                          8000    100000      1 000 000 000 000 000
```

```
; C 1142  1          Pattern 15      FFFE    177776      1 111 111 111 111 110
; C 1143  1                          FFFD    177775      1 111 111 111 111 101
; C 1144  1                          FFFB    177773      1 111 111 111 111 011
; C 1145  1                          FFF7    177767      1 111 111 111 110 111
; C 1146  1                          FFEF    177757      1 111 111 111 101 111
; C 1147  1                          FFDF    177737      1 111 111 111 011 111
; C 1148  1                          FFBF    177677      1 111 111 110 111 111
; C 1149  1                          FF7F    177577      1 111 111 101 111 111
; C 1150  1                          FEFF    177377      1 111 111 011 111 111
; C 1151  1                          FDFF    176777      1 111 110 111 111 111
; C 1152  1                          FBFF    175777      1 111 101 111 111 111
; C 1153  1                          F7FF    173777      1 111 011 111 111 111
; C 1154  1                          EFFF    167777      1 110 111 111 111 111
; C 1155  1                          DFFF    157777      1 101 111 111 111 111
; C 1156  1                          BFFF    137777      1 011 111 111 111 111
; C 1157  1                          7FFF    077777      0 111 111 111 111 111
; C 1158  1
; C 1159  1          Pattern 16      B6D9    133331      1 011 011 011 011 001
; C 1160  1                          B6D9    133331      1 011 011 011 011 001
; C 1161  1                          B6D9    133331      1 011 011 011 011 001
; C 1162  1                          DB6C    155554      1 101 101 101 101 100
; C 1163  1                          DB6C    155554      1 101 101 101 101 100
; C 1164  1                          DB6C    155554      1 101 101 101 101 100
; 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                          B6D9    133331      1 011 011 011 011 001
; C 1170  1                          DB6C    155554      1 101 101 101 101 100
; 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                          B6D9    133331      1 011 011 011 011 001
; C 1174  1                          DB6C    155554      1 101 101 101 101 100
```

```
; C 1175  1          Pattern 17      (LBN)*   (LBN)          (LBN)
; C 1176  1                          8D36     106466    1 000 110 100 110 110
; C 1177  1                          8D36     106466    1 000 110 100 110 110
; C 1178  1                          72C9     071311    0 111 001 011 001 001
; C 1179  1                          72C9     071311    0 111 001 011 001 001
; C 1180  1                          72C9     071311    0 111 001 011 001 001
; 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                          8D36     106466    1 000 110 100 110 110
; C 1184  1                          8D36     106466    1 000 110 100 110 110
; C 1185  1                          72C9     071311    0 111 001 011 001 001
; C 1186  1                          72C9     071311    0 111 001 011 001 001
; C 1187  1                          72C9     071311    0 111 001 011 001 001
; C 1188  1                          72C9     071311    0 111 001 011 001 001
; C 1189  1                          72C9     071311    0 111 001 011 001 001
; C 1190  1                          8D36     106466    1 000 110 100 110 110
; C 1191  1                          8D36     106466    1 000 110 100 110 110
; C 1192  1                          8D36     106466    1 000 110 100 110 110
; C 1193  1                          8D36     106466    1 000 110 100 110 110
; C 1194  1                          8D36     106466    1 000 110 100 110 110
; C 1195  1                          8D36     106466    1 000 110 100 110 110
; C 1196  1
; C 1197  1              * This word position contains the number of the locical
; C 1198  1                block to be written.
; C 1199  1
; C 1200  1
; C 1201  1
; C 1202  1          Pattern 18      8D36     106466    1 000 110 100 110 110
; C 1203  1                          (LBN)    (LBN)          (LBN)
; C 1204  1                          72C9     071311    0 111 001 011 001 001
; C 1205  1                          8D36     106466    1 000 110 100 110 110
; C 1206  1                          8D36     106466    1 000 110 100 110 110
; C 1207  1                          8D36     106466    1 000 110 100 110 110
; C 1208  1                          72C9     071311    0 111 001 011 001 001
; C 1209  1                          72C9     071311    0 111 001 011 001 001
; C 1210  1                          72C9     071311    0 111 001 011 001 001
; C 1211  1                          72C9     071311    0 111 001 011 001 001
; C 1212  1                          8D36     106466    1 000 110 100 110 110
; C 1213  1                          8D36     106466    1 000 110 100 110 110
; C 1214  1                          8D36     106466    1 000 110 100 110 110
; C 1215  1                          8D36     106466    1 000 110 100 110 110
; C 1216  1                          8D36     106466    1 000 110 100 110 110
; C 1217  1                          72C9     071311    0 111 001 011 001 001
; C 1218  1                          72C9     071311    0 111 001 011 001 001
; C 1219  1                          72C9     071311    0 111 001 011 001 001
; C 1220  1                          72C9     071311    0 111 001 011 001 001
; C 1221  1                          72C9     071311    0 111 001 011 001 001
; C 1222  1                          72C9     071311    0 111 001 011 001 001
```

```
;  C 1223  1          Pattern 19      (LBN)   (LBN)          (LBN)
;  C 1224  1                          B999    134631   1 011 100 110 011 001
;  C 1225  1                          B999    134631   1 011 100 110 011 001
;  C 1226  1                          4666    043146   0 100 011 001 100 110
;  C 1227  1                          4666    043146   0 100 011 001 100 110
;  C 1228  1                          4666    043146   0 100 011 001 100 110
;  C 1229  1                          B999    134631   1 011 100 110 011 001
;  C 1230  1                          B999    134631   1 011 100 110 011 001
;  C 1231  1                          B999    134631   1 011 100 110 011 001
;  C 1232  1                          B999    134631   1 011 100 110 011 001
;  C 1233  1                          4666    043146   0 100 011 001 100 110
;  C 1234  1                          4666    043146   0 100 011 001 100 110
;  C 1235  1                          4666    043146   0 100 011 001 100 110
;  C 1236  1                          4666    043146   0 100 011 001 100 110
;  C 1237  1                          4666    043146   0 100 011 001 100 110
;  C 1238  1                          B999    134631   1 011 100 110 011 001
;  C 1239  1                          B999    134631   1 011 100 110 011 001
;  C 1240  1                          B999    134631   1 011 100 110 011 001
;  C 1241  1                          B999    134631   1 011 100 110 011 001
;  C 1242  1                          B999    134631   1 011 100 110 011 001
;  C 1243  1                          B999    134631   1 011 100 110 011 001
;  C 1244  1
;  C 1245  1          Pattern 20      B999    134631   1 011 100 110 011 001
;  C 1246  1                          (LBN)   (LBN)          (LBN)
;  C 1247  1                          4666    043146   0 100 011 001 100 110
;  C 1248  1                          B999    134631   1 011 100 110 011 001
;  C 1249  1                          B999    134631   1 011 100 110 011 001
;  C 1250  1                          B999    134631   1 011 100 110 011 001
;  C 1251  1                          4666    043146   0 100 011 001 100 110
;  C 1252  1                          4666    043146   0 100 011 001 100 110
;  C 1253  1                          4666    043146   0 100 011 001 100 110
;  C 1254  1                          4666    043146   0 100 011 001 100 110
;  C 1255  1                          B999    134631   1 011 100 110 011 001
;  C 1256  1                          B999    134631   1 011 100 110 011 001
;  C 1257  1                          B999    134631   1 011 100 110 011 001
;  C 1258  1                          B999    134631   1 011 100 110 011 001
;  C 1259  1                          B999    134631   1 011 100 110 011 001
;  C 1260  1                          4666    043146   0 100 011 001 100 110
;  C 1261  1                          4666    043146   0 100 011 001 100 110
;  C 1262  1                          4666    043146   0 100 011 001 100 110
;  C 1263  1                          4666    043146   0 100 011 001 100 110
;  C 1264  1                          4666    043146   0 100 011 001 100 110
;  C 1265  1                          4666    043146   0 100 011 001 100 110
;  C 1266  1
;  C 1267  1               Pattern 21 (LBN)   (LBN)          (LBN)
;  C 1268  1
;     1269  1      )$
```

J4


```
;   1270  1     %sbttl 'PROGRAM HEADER
;   1271  1
;   1272  1     library 'ZRQAF0.L16';                   ! RDRX EXERCISER GLOBAL LIBRARY
;   1273  1
;   1274  1     require 'BLSMAC.REQ';                   ! DIAGNOSTIC SUPERVISOR LIBRARY
;   2765  1
;   2766  1     literal
;   2767  1         DS$NBR_OF_TESTS = 1;                ! NUMBER OF TESTS IN THIS DIAGNOSTIC
;   2768  1
;   2769  1     EQUALS;
;   2770  1
;   2771  1     POINTER (ALL);
;   2772  1
;   2773  1     !+
;   2774  1     ! THE PROGRAM HEADER IS THE INTERFACE BETWEEN
;   2775  1     ! THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
;   2776  1     !
;   2777  1
;   2778  1     !ZZZ HEADER (%ascii'ZRQA', %ascii'F', %ascii'0', 32767, 1, PRI00);
;   2779  1     HEADER (%ascii'ZRQA', %ascii'F', %ascii'0', 32000, 1, PRI00);       !ZZZ NEED POSITIVE NUMBER
;   2780  1
```

```
;   2781  1     %sbttl 'DISPATCH TABLE'
;   2782  1
;   2783  1     !+
;   2784  1     ! THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
;   2785  1     ! IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
;   2786  1     !
;   2787  1
;   2788  1     DISPATCH (DS$NBR_OF_TESTS);
```

```
;  2789  1     %sbttl 'GLOBAL DATA SECTION'
;  2790  1
;  2791  1     !+
;  2792  1     ! THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
;  2793  1     ! IN MORE THAN ONE TEST.
;  2794  1     !
;  2795  1
;  2796  1     psect
;  2797  1         global = $FFF$ (read, write, noexecute, global, concatenate);
;  2798  1
;  2799  1     global
;  2800  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;  2801  1                         ! RUN-TIME CONTROLLER STATUS TABLES
;  2802  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;  2803  1                         ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;  2804  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;  2805  1                         ! DRIVER CONTROLLER TABLES
;  2806  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;  2807  1                         ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;  2808  1         RDRX ADDR : ref rdrx field (RC_REG),
;  2809  1                         ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;  2810  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;  2811  1                         ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;  2812  1
;  2813  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                          !ZZZ
;  2814  1                         !CONTAINS LO+ HI LBN FIELDS FOR SEQUENTIAL       !ZZZ
;  2815  1                         !I/O TRANSFER FOR EACH UNIT.                     !ZZZ
;  2816  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;  2817  1                         ! STATISTICS TABLES
;  2818  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;  2819  1                         ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;  2820  1
;  2821  1         DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;  2822  1                         !INFO FROM RECEIVE + SEND CMDS                   ZZZ
;  2823  1         TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED] INITIAL (BYTE (REP    !ZZZ
;  2824  1                 MAX_UNITS OF (1))),          !CURRENT TRACK DIRECTION     ZZZ
;  2825  1         RDM_CNT : WORD INITIAL (RDM_LEN),    !NO OF RANDOM NOS   \\KEEP   ZZZ
;  2826  1         RANDOM : VECTOR [RDM_LEN, WORD],     !RANDOM NO. TABLE   //TOGETHER   ZZZ
;  2827  1
;  2828  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;  2829  1                         ! STATISTICS TABLE FOR CONTROLLER ERRORS
;  2830  1         MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;  2831  1                         ! MSCP PACKET POOL
;  2832  1         IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  2833  1                         ! ADDRESS OF AN MSCP PACKET (INTERUPT PROCESSING)
;  2834  1         PKT_USE : vector [PKT_CNT, byte, signed],
;  2835  1                         ! MSCP PACKET POOL ALLOCATION TABLE
;  2836  1         RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;  2837  1                         ! RETURN PACKET POOL
;  2838  1         RP_USE : vector [RP_CNT, byte, signed],
;  2839  1                         ! RETURN PACKET POOL ALLOCATION TABLE
;  2840  1         RP_INDX : word,      ! CURRENT RETURN PACKET INDEX
;  2841  1         RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
```

```
;    2842  1                          ! CURRENT RETURN PACKET ADDRESS
;    2843  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;    2844  1                          ! ERROR-LOG PACKET SAVE AREA
;    2845  1          BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
;    2846  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;    2847  1          IODQ : vector [IODQ_LEN, byte],                      ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;    2848  1          IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
;    2849  1          IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
;    2850  1          ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
;    2851  1          EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
;    2852  1          DUP_FLAGS : WORD,                                    !DUP FLAGS       ZZZ
;    2853  1          CCTLR : word,                                        ! NUMBER OF "CURRENT" CONTROLLER
;    2854  1          CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    2855  1          CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
;    2856  1          CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;    2857  1          DUR : vector [MAX UNITS, byte],                      ! DROP UNIT REASON
;    2858  1          QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;    2859  1          FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
;    2860  1          BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
;    2861  1          ST_CODE : word,                                      ! CURRENT STATUS CODE
;    2862  1          SB_CODE : word,                                      ! CURRENT SUB-CODE
;    2863  1          STEP : word,                                         ! CURRENT STEP IN HARD_INIT
;    2864  1          OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
;    2865  1          SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
;    2866  1          CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
;    2867  1          NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
;    2868  1          CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
;    2869  1          CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
;    2870  1          TEMP1 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;    2871  1          TEMP2 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;    2872  1          CREDIT_BAL : word,                                   ! CREDIT BALANCE
;    2873  1          NEXT_PKT_USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;    2874  1          HOURS : byte,                                        ! TIME OF DAY (HOURS)
;    2875  1          MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
;    2876  1          CLK_TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
;    2877  1          CLK_PRESENT : byte,                                  ! FLAG INDICATES IF LINE-CLOCK PRESENT
;    2878  1          HOE_FLAG : byte,                                     ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;    2879  1
;    2880  1          S_PATTERN : WORD,                  !PATTERN FOR DUP WRITES          ZZZ
;    2881  1          S_DUPPKT : WORD,                   !DBN BYTE COUNTER                ZZZ
;    2882  1          P_INDEX : SIGNED WORD,             !CURRENT MESSAGE PACKET INDEX    ZZZ
;    2883  1          RD_COUNT : WORD INITIAL (0),       ! NUMBER OF WINCHESTER UNITS     ZZZ
;    2884  1          BRLEVEL : WORD,                    !BUS REQUEST LEVEL FROM OPERATOR ZZZ
;    2885  1          D_FAIL : BYTE,                     !SIGNIFIES DUP TYPE ERROR        ZZZ
;    2886  1          FORCED_ERROR : byte,                                 ! "FORCED ERROR" DETECTED IN LAST READ
;    2887  1          FER_LBN : word,                                      ! LBN OF THE "FORCED ERROR" BLOCK
;    2888  1          FER_BC : word,                                       ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    2889  1          INIT_OCCURED : byte initial (byte (FALSE)),          ! EXERCISER INITIALIZATION COMPLETE
;    2890  1          ADDR_VECT_OK : byte initial (byte (FALSE));          ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    2891  1
;    2892  1      ERRTBL;
```

```
;  2893  1     %sbttl 'GLOBAL TEXT SECTION
;  2894  1
;  2895  1     !+
;  2896  1     ! THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
;  2897  1     ! MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
;  2898  1     ! MORE THAN ONE TEST.
;  2899  1     !
;  2900  1
;  2901  1     global bind
;  2902  1     !
;  2903  1     ! HARDWARE DIALOG
;  2904  1     !
;  2905  1         HWQ1 = uplit (%asciz'IP address'),
;  2906  1         HWQ2 = uplit (%asciz'Vector'),
;  2907  1         HWQ3 = uplit (%asciz'BR Level [usually 4-RQDX1 5-RUX50]'),          !ZZZ
;  2908  1         HWQ4 = uplit (%asciz'Drive number'),                               !ZZZ
;  2909  1         HWQ5 = uplit (%asciz'Test entire customer area of this disk'),     !ZZZ
;  2910  1         HWQ6A = uplit (%asciz'Lower octal word of beginning LBN address'), !ZZZ
;  2911  1         HWQ6B = uplit (%asciz'Higher octal word of beginning LBN address'),!ZZZ
;  2912  1         HWQ7A = uplit (%asciz'Lower octal word of ending LBN address'),    !ZZZ
;  2913  1         HWQ7B = uplit (%asciz'Higher octal word of ending LBN address'),   !ZZZ
;  2914  1         HWQ8 = uplit (%asciz'Write on customer data area of this disk unit'), !ZZZ
;  2915  1         HWQ9 = uplit (%asciz'** WARNING - CUSTOMER DATA AREA MAY BE OVERWRITTEN! ... CONFIRM'),
;  2916  1         HWQ10 = uplit (%asciz'Also run DUP exerciser'),                    !ZZZ
;  2917  1         HWQ11 = uplit (%asciz'Write on diagnostic area'),                  !ZZZ
;  2918  1     !
;  2919  1     ! SOFTWARE DIALOG
;  2920  1     !
;  2921  1         SWQ1  = uplit (%asciz'Hard error limit'),                                  !ZZZ
;  2922  1         SWQ2  = uplit (%asciz'Transfer limit in megabytes (0 for quick pass) '),   !ZZZ
;  2923  1         SWQ4  = uplit (%asciz'Random seek mode'),
;  2924  1         SWQ7  = uplit (%asciz'Read-compares performed at the controller'),
;  2925  1         SWQ9  = uplit (%asciz'Write-compares performed at the controller'),
;  2926  1         SWQ10 = uplit (%asciz'Check all writes at host by reading'),
;  2927  1         SWQ11 = uplit (%asciz'User-defined data pattern'),
;  2928  1         SWQ12 = uplit (%asciz'Select pre-defined data pattern (0 for sequential selection)'),
;  2929  1         SWQ13 = uplit (%asciz'Number of words in data pattern (16 maximum)'),
;  2930  1         SWQ14 = uplit (%asciz'Pattern value (no leading zeros allowed)'),
;  2931  1         SWQ15 = uplit (%asciz'Clear statistical tables after printing'),
;  2932  1         SWQ17 = uplit (%asciz'Percentage of "Fixed Disk" operations out of total operations'),
;  2933  1         SWQ19 = uplit (%asciz'Units to be selected at random (No, implies sequential)'),
;  2934  1         SWQ20 = uplit (%asciz'Rewrite blocks when "Forced Error" detected on reads'),
;  2935  1         SWQ21 = uplit (%asciz'Halt on other hard errors (#s 31-34, 36-37, 39-45)'),  !ZZZ
;  2936  1         SWQ22 = uplit (%asciz'Halt on soft errors (#s 50-54)'),                      !ZZZ
;  2937  1         SWQ23 = uplit (%asciz'Halt on bad-block hard errors (#s 35, 38)'),           !ZZZ
;  2938  1         SWQ24 = uplit (%asciz'Enter time as HHMM (example: 1305)'),                  !ZZZ
;  2939  1         SWQ25 = uplit (%asciz'Count each retry as a seperate soft error'),           !ZZZ
;  2940  1         SWQ26 = uplit (%asciz'Running under the A.P.T. Monitor'),                    !ZZZ
;  2941  1         SWM1  = uplit (%asciz'The remaining questions only apply to unprotected disk un ts'), !ZZZ
;  2942  1         NULL  = uplit (%asciz''),
;  2943  1
;  2944  1     !++
;  2945  1     !  THE FOLLOWING DBMs ARE DEBUG MESSAGES, AND SHOULD BE REMOVED BEFORE
```

```
;  2946  1     !  RELEASING THE PROGRAM.  THEY INCLUDE THE NAMES OF EACH ROUTINE, PLUS
;  2947  1     !  FORMAT STATEMENTS FOR PRINTING OUT OTHER INFORMATION.
;  2948  1     !
;  2949  1
;  2950  1         DBM5   = uplit (%asciz'%N%AAA Drop unit %D2' ),
;  2951  1         DBM12  = uplit (%asciz'%N%AAA PROC_RETPKT: Conn ID %O6%A received ),
;  2952  1         DBM15  = uplit (%asciz'%N%AAA Multi-drive test' ),
;  2953  1         DBM18  = uplit (%asciz'%N%AAA FATAL_ERROR: RETPKT not available' ),
;  2954  1         DBM19  = uplit (%asciz'%N%AAA FSET_UPAR: Can''t find disk %D3%A in CST %D1' ),
;  2955  1         DBM20  = uplit (%asciz'%N%AAA Bad conn ID %O6%A received from %O6 ),
;  2956  1         DBM21  = uplit (%asciz'%N%AAA Message type %O2%A received in MSCP packet ),
;  2957  1         DBM22  = uplit (%asciz'%N%AAA SEQUEN: RETPKT not available' ),
;  2958  1         DBM23  = uplit (%asciz'%N%AAA Error in SET_CTLR_CHAR' ),
;  2959  1         DBM25  = uplit (%asciz'%N%AAA Ctlr timeout = %D3%A. seconds' ),
;  2960  1         DBM26  = uplit (%asciz'%N%AAA Error in UNIT_INIT' ),
;  2961  1         DBM27  = uplit (%asciz'%N%AAA UNIT_INIT: RETPKT has bad ENDCODE ),
;  2962  1         DBM28A = uplit (%asciz'%N%AAA Unit size (Lo) = %D5%A.' ),
;  2963  1         DBM28B = uplit (%asciz'%N%AAA Unit size (Hi) = %D5%A.' ),
;  2964  1         DBM29  = uplit (%asciz'%N%AAA ACCESS: RETPKT has bad ENDCODE' ),
;  2965  1         DBM32  = uplit (%asciz'%N%AAA QIO_UNIT: CST %D1%A no unit selected' ),
;  2966  1         DBM101 = uplit (%asciz'%N%AAA Unit # is: %O6' ),
;  2967  1         DBM104 = uplit (%asciz'%N%AAA Removable disk is selected' ),
;  2968  1         DBM105 = uplit (%asciz'%N%AAA Fixed disk is selected' ),
;  2969  1         DBM107 = uplit (%asciz'%N%AAA Illegal function: %O6' ),
;  2970  1         DBM108 = uplit (%asciz'%N%AAA Command ref # %O6%A/%O6%A (Oct) not sent by Host' ),
;  2971  1         DBM109 = uplit (%asciz'%N%AAA Unknown Error Log format %O3%A received' ),
;  2972  1     !   DBM110 = uplit (%asciz'%N%AAA Error-Log save area full' ),
;  2973  1         DBM111 = uplit (%asciz'%N%AAA Op-code %O3%A, End-code %O3%A for ref # %O6%A/%O6%A (8)' ),
;  2974  1         DBM112 = uplit (%asciz'%N%AAA Cmd-bc %O6%^'%O6%A Rsp-bc %O6%A/%O6%A for %O6%A/%O6%A (8)' ),
;  2975  1         DBM120 = uplit (%asciz'%N%AAA Response already received for cmd %O6%A/%O6%A (8)' ),
;  2976  1         DBM121 = uplit (%asciz'%N%AAA Failure to send command after # %O6%A/%O6%A (8)' ),
;  2977  1     !
;  2978  1     ! DROP UNIT MESSAGES
;  2979  1     !
;  2980  1         DU_MSG = uplit (%asciz'%N%AUNIT%D2%A DROPPED - ' ),
;  2981  1         DU_RSN = uplit (
;  2982  1                  uplit (%asciz'%AUSER COMMAND%N' ),
;  2983  1                  uplit (%asciz'%ACONFIGURATION ERROR%N' ),
;  2984  1                  uplit (%asciz'%AINIT ERROR%N' ),
;  2985  1                  uplit (%asciz'%ATRANSFER LIMIT REACHED%N' ),
;  2986  1                  uplit (%asciz'%AERROR LIMIT REACHED%N' ),
;  2987  1                  uplit (%asciz'%AUNRECOVERABLE DRIVE ERROR%N' ),
;  2988  1                  uplit (%asciz'%AUNRECOVERABLE CONTROLLER ERROR%N' ),
;  2989  1                  uplit (%asciz'%AFAILED TO COME ONLINE%N' ),
;  2990  1                  uplit (%asciz'%AFAILED TO ACCESS EITHER FIRST OR LAST TRACK DURING INIT%N ),
;  2991  1                  uplit (%asciz'%ADISK WRITE PROTECTED%N' ),
;  2992  1                  uplit (%asciz'%ACOMMAND TIME OUT%N' )) : vector [11],
;  2993  1     !
;  2994  1     ! SYSTEM MESSAGES (PRINTF)
;  2995  1     !
;  2996  1         MSG_01 = uplit (%asciz'%N%APOWER DELAY - WAITING' ),
;  2997  1         MSG_02 = uplit (%asciz'%N%AFUNCTIONAL TEST STARTED' ),
;  2998  1         MSG_03 = uplit (%asciz'%N%N%AEXERCISER STARTED%N' ),
```

```
;  2999  1     !
;  3000  1     ! REPORT MESSAGES (PRINTS)
;  3001  1     !
;  3002  1         RPT1  = uplit (%asciz'%N%N%AUNT DSK%S8%A# OF    # BYTES    # OF     # BYTES '),
;  3003  1         RPT2  = uplit (%asciz'%A  --HARD ERRORS-- --SOFT ERRORS--'),
;  3004  1         RPT3  = uplit (%asciz'%N%A #    #  TYPE  READS     READ   WRITES   WRITTEN'),
;  3005  1         RPT4  = uplit (%asciz'%A  SEK DAT DRV HST SEK DAT DRV HST'),
;  3006  1         RPT5  = uplit (%asciz'%N%A--- --- ----  --- --- --- --  --  ---------'),
;  3007  1         RPT6  = uplit (%asciz'%A ---  -- --- --- ---  -  --- ---'),
;  3008  1         RPT7  = uplit (%asciz'%N%D2%D4%S2%T'),
;  3009  1         RPT8  = uplit (%asciz'%D4%Z3%D3%A,%Z3%A,%Z3'),
;  3010  1         RPT9  = uplit (%asciz'%D4%D4%D4%D4%D4%D4%D4%D4'),
;  3011  1         RPT10 = uplit (%asciz'%N%A .   . CNTR      .  .........     .  .........'),
;  3012  1         RPT11 = uplit (%asciz'%A   .   .%D4%A   .   .   .%D4%A   .'),
;  3013  1         RPT12 = uplit (%asciz'%A   .   .%D4%A   .   .   .%D4%A   .'),
;  3014  1         RPT13 = UPLIT(%ASCIZ'%N%N%AUNIT  DISK          # OF   # BLKS      # OF    # BLKS '),
;  3015  1         RPT14 = UPLIT(%ASCIZ'%N%A  #    #     TYPE   READS     READ     WRITES  WRITTEN '),
;  3016  1         RPT15 = UPLIT(%ASCIZ'%N%A---  ---- -------  -----  ------  ------  ------ '),
;  3017  1         RPT16 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBN I/O  %D6%S3%D6%S5%D6%S3%D6'),
;  3018  1     !ZZZ    RPT17  = uplit (%asciz'%N%D2%D4%A  RD52'),
;  3019  1     !ZZZ    RPT18 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBNRD52  %D6%S3%D6%S5%D6%S3%D6'),
;  3020  1     !ZZZ    RPT19  = uplit (%asciz'%N%D2%D4%A  ????'),
;  3021  1
;  3022  1     !
;  3023  1     ! GENERAL ERROR MESSAGES
;  3024  1     !
;  3025  1     !       SYSTEM FATAL (ERRSF)
;  3026  1     !
;  3027  1         EGS_01 = uplit (%asciz'TOO MANY UNITS'),
;  3028  1         EGS_02 = uplit (%asciz'NOT ENOUGH FREE MEMORY FOR ALLOCATING READ/WRITE BUFFERS'),
;  3029  1     !
;  3030  1     !       DRIVE FATAL (ERRDF)
;  3031  1     !
;  3032  1         EGD_10 = uplit (%asciz'REGISTER EXISTENCE TEST FAILED'),
;  3033  1         EGD_11 = uplit (%asciz'VECTOR TEST FAILED'),
;  3034  1         EGD_12 = uplit (%asciz'BR LEVEL TEST FAILED'),
;  3035  1         EGD_13 = uplit (%asciz'INIT SEQUENCE FAILED'),
;  3036  1         EGD_14 = uplit (%asciz'FATAL CONTROLLER ERROR'),
;  3037  1         EGD_15 = uplit (%asciz'ONLINE FAILED'),
;  3038  1         EGD_16 = uplit (%asciz'WRITE-PROTECT CONFLICT'),
;  3039  1         EGD_17 = uplit (%asciz'ACCESS FAILED'),
;  3040  1         EGD_18 = uplit (%asciz'FATAL I/O ERROR'),
;  3041  1     !   EGD_19 = uplit (%asciz'CONTROLLER TIMEOUT'),
;  3042  1         EGD_19 = uplit (%asciz'DISK TYPE UNKNOWN TO EXERCISER'),
;  3043  1         EGD_20 = uplit (%asciz'FAILED TO SEND SET-CONTROLLER-CHARACTERISTICS COMMAND '),
;  3044  1         EGD_21 = uplit (%asciz'SET-CONTROLLER-CHARACTERISTICS RESPONSE HAS BAD ENDCODE OR FLAGS IN ERROR'),
;  3045  1         EGD_22 = uplit (%asciz'FAILED TO SEND ON-LINE COMMAND'),
;  3046  1         EGD_23 = uplit (%asciz'ON-LINE RESPONSE HAS BAD ENDCODE'),
;  3047  1         EGD_24 = uplit (%asciz'ON-LINE RESPONSE HAS UNKNOWN DEVICE'),
;  3048  1     !
;  3049  1     !       HARD or SOFT (ERRHRD or ERRSOFT)
;  3050  1     !
;  3051  1         EGH_30 = uplit (%asciz'I/O REQUEST FAILED'),
```

```
;    3052  1     !
;    3053  1     ! BASIC ERROR MESSAGES (PRINTB)
;    3054  1     !
;    3055  1     !       SYSTEM FATAL (ERRSF)
;    3056  1     !
;    3057  1        EBS_01 = uplit (%esciz'%AMORE THAN %D2%A UNITS SPECIFIED'),
;    3058  1     !
;    3059  1     !       DRIVE FATAL (ERRDF)
;    3060  1     !
;    3061  1        EBD_10 = uplit (%esciz'%A* NO RESPONSE AT ADDRESS %O6'),
;    3062  1        EBD_12 = uplit (%esciz'%A* INCORRECT BR LEVEL FOR DRIVE %O6'),
;    3063  1        EBD_13 = uplit (%esciz'%A* STEP %D1%A READ ERROR'),
;    3064  1        EBD_14 = uplit (%esciz'%A* BAD SA CODE FROM DRIVE %O6'),
;    3065  1        EBD_18 = uplit (%esciz'%A* DISK%D2%A WENT OFFLINE'),
;    3066  1        EBD_19 = uplit (%esciz'%A* DRIVE %O6%A NOT PROCESSING COMMAND PACKETS'),
;    3067  1        EBD_24 = uplit (%esciz'%A* DISK%D2%A WENT TO THE "AVAILABLE" STATE'),
;    3068  1     !
;    3069  1     !
;    3070  1     !       HARD or SOFT (ERRHRD or ERRSOFT)
;    3071  1     !
;    3072  1     EH_0 = UPLIT (%ASCIZ' - UNRECOGNIZED MESSAGE TYPE '),                    !ZZZ
;    3073  1     EH_1 = UPLIT (%ASCIZ' - UNRECOGNIZED CONNECTION ID'),                    !ZZZ
;    3074  1     EH_2 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN MESSAGE'),                   !ZZZ
;    3075  1     EH_3 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN PACKET'),                    !ZZZ
;    3076  1     EH_4 = UPLIT (%ASCIZ' - UNRECOGNIZED CRN'),                              !ZZZ
;    3077  1     EH_5 = UPLIT (%ASCIZ' - UNRECOGNIZED OPCODE'),                           !ZZZ
;    3078  1     EH_6 = UPLIT (%ASCIZ' - MSCP STATUS CODE ERR'),                          !ZZZ
;    3079  1     EH_7 = UPLIT (%ASCIZ' - DUP STATUS CODE ERR'),                           !ZZZ
;    3080  1     EH_8 = UPLIT (%ASCIZ' - UNRECOGNIZED STATUS CODE'),                      !ZZZ
;    3081  1     EH_9 = UPLIT (%ASCIZ' - LBN HOST COMPARE ERR'),                          !ZZZ
;    3082  1     EH_10 = UPLIT (%ASCIZ' - DBN HOST COMPARE ERR'),                         !ZZZ
;    3083  1     EH_12 = UPLIT (%ASCIZ' - UNABLE TO LOAD DUP MEDIA'),                     !ZZZ
;    3084  1     EH_13 = UPLIT (%ASCIZ' - ERR IN DUP PKT WHEN USING CTLR LC PROG'),       !ZZZ
;    3085  1     !
;    3086  1         ERR_00  = uplit (%esciz'%A* DISK%D2'),
;    3087  1         ERR_COD = uplit (
;    3088  1                   uplit (%esciz'%AINVALID COMMAND'),
;    3089  1                   uplit (%esciz'%ACOMMAND ABORTED'),
;    3090  1                   uplit (%esciz'%AUNIT OFFLINE'),
;    3091  1                   uplit (%esciz'%ATRANSITION TO AVAILABLE STATE'),
;    3092  1                   uplit (%esciz'%AMEDIA FORMAT ERROR'),
;    3093  1                   uplit (%esciz'%AWRITE-PROTECTED'),
;    3094  1                   uplit (%esciz'%ADEVICE COMPARE ERROR'),
;    3095  1                   uplit (%esciz'%ADATA ERROR'),
;    3096  1                   uplit (%esciz'%AHOST BUFFER ACCESS ERROR'),
;    3097  1                   uplit (%esciz'%ACONTROLLER ERROR'),
;    3098  1                   uplit (%esciz'%ADRIVE ERROR'),
;    3099  1                   uplit (%esciz'%AMESSAGE FROM INTERNAL DIAGNOSTICS'),
;    3100  1                   uplit (%esciz'%AHOST COMPARE ERROR'),
;    3101  1                   uplit (%esciz'%ACOMMAND TIMEOUT')) : vector [14],
;    3102  1     !
;    3103  1     !       ERROR LOG MESSAGE (ERRSOFT)
;    3104  1     !
```

```
;    3105  1          ELG_00  = uplit (%ascic'%AFRROR LOG MESSAGE RECEIVED:%N'),
;    3106  1          ELG_FMT = uplit (
;    3107  1                    uplit (%ascic'%A* CONTROLLER ERROR%N'),
;    3108  1                    uplit (%ascic'%A* HOST MEMORY ACCESS ERROR%N'),
;    3109  1                    uplit (%ascic'%A* DISK%D2%A - DISK TRANSFER ERROR%N'),
;    3110  1                    uplit (%ascic'%A* DISK%D2%A - "STANDARD DISK INTERCONNECT" ERROR%N'),
;    3111  1                    uplit (%ascic'%A* DISK%D2%A - "SMALL DISK" ERROR%N')) : vector [5],
;    3112  1      !
;    3113  1      ! EXTENDED ERROR MESSAGES (PRINTX)
;    3114  1      !
;    3115  1          EX_SA  = uplit (%ascic'%N%A* SA: %O6'),
;    3116  1          EX_SC  = uplit (%ascic'%N%A* STATUS CODE: %O2'),
;    3117  1          EX_SBO = uplit (%ascic'%O4'),
;    3118  1          EX_SB  = uplit (%ascic'%N%A* SUB_CODE: '),
;    3119  1          EX_CMD = uplit (%ascic'%N%A* COMMAND  '),
;    3120  1          EX_RD  = uplit (%ascic'%AREAD'),
;    3121  1          EX_WRT = uplit (%ascic'%AWRITE'),
;    3122  1          EX_CMP = uplit (%ascic'%A-COMPARE'),
;    3123  1          EX_ONL = uplit (%ascic'%AONLINE'),
;    3124  1          EX_ACC = uplit (%ascic'%AACCESS'),
;    3125  1          EX_OP  = uplit (%ascic'%O3'),
;    3126  1          EX_BB  = uplit (%ascic'%N%A* BAD BLOCK (Host replaceable): %D5%A. (OCT %O6%A)'),
;    3127  1          EX_BB1 = uplit (%ascic'%N%A* 1st BAD BLOCK (Host replaceable): %D5%A. (OCT %O6%A)'),
;    3128  1          EX_BBU = uplit (%ascic'%N%A* BAD BLOCK REPORTED (Replaced): %D%A. (OCT %O6%A)'),
;    3129  1          EX_LBN = uplit (%ascic'%N%A* LBN: %D5%A. (OCT %O6%A)'),
;    3130  1          EX_PBN = uplit (%ascic'%N%A* PBN: %D5%A. (OCT %O6%A)'),
;    3131  1          EX_LBR = uplit (%ascic'%N%A* LBN: (READ) %D5%A. (OCT %O6%A)'),
;    3132  1          EX_LBW = uplit (%ascic'%N%A* LBN: (WRITE) %D5%A. (OCT %O6%A)'),
;    3133  1          EX_RBN = uplit (%ascic'%N%A* REPLACEMENT BLOCK NO. %D5%A. (OCT %O6%A)'),
;    3134  1          EX_CBC = uplit (%ascic'%N%A* BYTE COUNT IN COMMAND: %D5%A.'),
;    3135  1          EX_CBR = uplit (%ascic'%N%A* BYTE COUNT IN READ COMMAND: %D5%A.'),
;    3136  1          EX_CBW = uplit (%ascic'%N%A* BYTE COUNT IN WRITE COMMAND: %D5%A.'),
;    3137  1          EX_BC  = uplit (%ascic'%N%A* ACTUAL # OF BYTES TRANSFERRED: %D5%A.'),
;    3138  1          EX_BD  = uplit (%ascic'%N%A* I/O BUFFER ADDRESS (32 bits): %O6%A %O6'),
;    3139  1          EX_BDR = uplit (%ascic'%N%A* I/O BUFFER ADDRESS FOR READ (32 bits): %O6%A %O6'),
;    3140  1          EX_BDW = uplit (%ascic'%N%A* I/O BUFFER ADDRESS FOR WRITE (32 bits): %O6%A %O6'),
;    3141  1          EX_RP  = uplit (%ascic'%N%ACONTENTS OF COMMAND/RESPONSE PACKET SAVE AREA:%N'),
;    3142  1          EX_WRD = uplit (%ascic'%A %O6'),
;    3143  1          EX_TIM = uplit (%ascic'%N%A* TIME: %Z2%A:%Z2%A HOURS%N'),
;    3144  1
;    3145  1
;    3146  1          XX13 = UPLIT (%ASCIZ'%N%A * DISK : %D2'),                          !ZZZ
;    3147  1          XX23 = UPLIT (%ASCIZ'%N%ADBN: %D5%A. (OCT %O6%A)'),                !ZZZ
;    3148  1          XX32 = UPLIT (%ASCIZ'%N%ABYTE NUMBER: %D3'),                       !ZZZ
;    3149  1          XX33 = UPLIT (%ASCIZ'%N%ARANDOM WRITTEN WORD :%B16'),              !ZZZ
;    3150  1          XX34 = UPLIT (%ASCIZ'%N%ARANDOM READ WORD bin:%B16%A oct:%O6'),    !ZZZ
;    3151  1
;    3152  1      !
;    3153  1      ! CONFIGURATION ERROR MESSAGES (PRINTF)
;    3154  1      !
;    3155  1          CER_01 = uplit (%ascic'%N%ADUPLICATE UNIT:%D2%A AT IP: %O6'),
;    3156  1          CER_02 = uplit (%ascic'%N%AMORE THAN %D1%A DIFFERENT IP ADDRESSES'),
;    3157  1      !
```

```
;    3158  1      ! ERROR/EVENT SUB CODES (PRINTX)
;    3159  1      !
;    3160  1          SC_SDI = uplit (%asciz'%ASPIN-DOWN IGNORED'),
;    3161  1          SC_CON = uplit (%asciz'%ASTILL CONNECTED'),
;    3162  1          SC_DUP = uplit (%asciz'%ADUPLICATE UNIT NUMBER'),
;    3163  1          SC_ONL = uplit (%asciz'%AALREADY ONLINE'),
;    3164  1          SC_SON = uplit (%asciz'%ASTILL ONLINE'),
;    3165  1          SC_UNK = uplit (%asciz'%AUNIT UNKNOWN OR ONLINE TO ANOTHER CONTROLLER'),
;    3166  1          SC_VOL = uplit (%asciz'%AND VOLUME MOUNTED OR DRIVE DISABLED BY SWITCH'),
;    3167  1          SC_IOP = uplit (%asciz'%AUNIT INOPERATIVE (RD51/52 write fault)'),
;    3168  1          SC_DIS = uplit (%asciz'%AUNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTICS '),
;    3169  1          SC_FER = uplit (%asciz'%A"FORCED ERROR" DETECTED WHILE ACCESSING FCT OR RCT'),
;    3170  1          SC_FE2 = uplit (%asciz'%ASECTOR HAD BEEN WRITTEN WITH "FORCED ERROR" MODIFIER '),
;    3171  1          SC_ISH = uplit (%asciz'%AFCT OR RCT UNREADABLE - INVALID SECTOR HEADER'),
;    3172  1          SC_IS2 = uplit (%asciz'%AHEADER COMPARE ERROR (Valid header not found)'),
;    3173  1          SC_DST = uplit (%asciz'%AFCT OR RCT UNREADABLE - DATA SYNC TIMEOUT'),
;    3174  1          SC_DS2 = uplit (%asciz'%ADATA SYNC NOT FOUND (Data sync timeout)'),
;    3175  1          SC_ECC = uplit (%asciz'%AFCT OR RCT UNREADABLE - UNCORRECTABLE ECC ERROR'),
;    3176  1          SC_ECD = uplit (%asciz'%AUNCORRECTABLE ECC ERROR'),
;    3177  1          SC_RCT = uplit (%asciz'%ARCT CORRUPTED'),
;    3178  1          SC_FUL = uplit (%asciz'%ANO REPLACEMENT BLOCK AVAILABLE (RCT full)'),
;    3179  1          SC_576 = uplit (%asciz'%ADISK NOT FORMATTED WITH 512 BYTE SECTORS'),
;    3180  1          SC_FCT = uplit (%asciz'%ADISK NOT FORMATTED OR FCT CORRUPTED'),
;    3181  1          SC_EC1 = uplit (%asciz'%AONE SYMBOL ECC ERROR'),
;    3182  1          SC_EC2 = uplit (%asciz'%ATWO SYMBOL ECC ERROR'),
;    3183  1          SC_EC3 = uplit (%asciz'%ATHREE SYMBOL ECC ERROR'),
;    3184  1          SC_EC4 = uplit (%asciz'%AFOUR SYMBOL ECC ERROR'),
;    3185  1          SC_EC5 = uplit (%asciz'%AFIVE SYMBOL ECC ERROR'),
;    3186  1          SC_EC6 = uplit (%asciz'%ASIX SYMBOL ECC ERROR'),
;    3187  1          SC_EC7 = uplit (%asciz'%ASEVEN SYMBOL ECC ERROR'),
;    3188  1          SC_EC8 = uplit (%asciz'%AEIGHT SYMBOL ECC ERROR'),
;    3189  1          SC_EC9 = uplit (%asciz'%ACORRECTABLE ERROR IN ECC FIELD'),
;    3190  1          SC_SWP = uplit (%asciz'%AUNIT SOFTWARE WRITE PROTECTED'),
;    3191  1          SC_HWP = uplit (%asciz'%AUNIT HARDWARE WRITE PROTECTED'),
;    3192  1          SC_ODA = uplit (%asciz'%AODD TRANSFER ADDRESS'),
;    3193  1          SC_ODB = uplit (%asciz'%AODD BYTE COUNT'),
;    3194  1          SC_NXM = uplit (%asciz'%ANON-EXISTENT HOST MEMORY'),
;    3195  1          SC_PAR = uplit (%asciz'%AHOST MEMORY PARITY ERROR'),
;    3196  1          SC_CTO = uplit (%asciz'%ACOMMAND TIMOUT OR RETRY LIMIT EXCEEDED'),
;    3197  1          SC_SDS = uplit (%asciz'%ASERIALIZER/DESERIALIZER OVERRUN OR UNDERRUN'),
;    3198  1          SC_EDC = uplit (%asciz'%A"ERROR DETECTION CODE" ERROR'),
;    3199  1          SC_IDS = uplit (%asciz'%AINCONSISTENT INTERNAL DATA STRUCTURE'),
;    3200  1          SC_SRT = uplit (%asciz'%ADRIVE COMMAND TIMEOUT (No response or seek incomplete) '),
;    3201  1          SC_SRI = uplit (%asciz'%ACONTROLLER DETECTED TRANSMISSION OR PROTOCOL ERROR'),
;    3202  1          SC_POE = uplit (%asciz'%APOSITION ERROR (Mis-seek)'),
;    3203  1          SC_RDY = uplit (%asciz'%ALOST READ/WRITE READY DURING/BETWEEN TRANSFERS'),
;    3204  1          SC_CLK = uplit (%asciz'%ADRIVE CLOCK DROPOUT'),
;    3205  1          SC_RSP = uplit (%asciz'%ALOST RECEIVER READY BETWEEN SECTORS'),
;    3206  1          SC_SUR = uplit (%asciz'%ADRIVE DETECTED ERROR'),
;    3207  1          SC_PSP = uplit (%asciz'%ACONTROLLER DETECTED PULSE OR STATE PARITY ERROR '),
;    3208  1      !
;    3209  1      ! CONTROLLER GENERIC ERROR CODES
;    3210  1      !
```

```
;    3211  1          CNTR_ERR = uplit (
;    3212  1                     uplit (%asciz'%ACONTROLLER TIMEOUT'),
;    3213  1                     uplit (%asciz'%AENVELOPE/PACKET READ ERROR (Parity or timeout)'),
;    3214  1                     uplit (%asciz'%AENVELOPE/PACKET WRITE ERROR (Parity or timeout)'),
;    3215  1                     uplit (%asciz'%ACONTROLLER ROM AND RAM PARITY ERROR'),
;    3216  1                     uplit (%asciz'%ACONTROLLER RAM PARITY ERROR'),
;    3217  1                     uplit (%asciz'%ACONTROLLER ROM PARITY ERROR'),
;    3218  1                     uplit (%asciz'%ARING READ ERROR (Parity or timeout)'),
;    3219  1                     uplit (%asciz'%ARING WRITE ERROR (Parity or timeout)'),
;    3220  1                     uplit (%asciz'INTERRUPT MASTER FAILURE'),
;    3221  1                     uplit (%asciz'%AHOST ACCESS TIMEOUT (Higher level protocol dependent)'),
;    3222  1                     uplit (%asciz'%ACREDIT LIMIT EXCEEDED'),
;    3223  1                     uplit (%asciz'%AQ-BUS MASTER ERROR'),
;    3224  1                     uplit (%asciz'%ACONTROLLER FATAL ERROR'),
;    3225  1                     uplit (%asciz'%AINSTRUCTION LOOP TIMEOUT'),
;    3226  1                     uplit (%asciz'%AILLEGAL VIRTUAL CIRCUIT ID'),
;    3227  1                     uplit (%asciz'%AINTERRUPT VECTOR ILLEGAL'),
;    3228  1                     uplit (%asciz'%AMAINTENANCE READ/WRITE INVALID REGION IDENTIFIER '),
;    3229  1                     uplit (%asciz'%AMAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER'),
;    3230  1                     uplit (%asciz'%ACONTROLLER RAM ERROR (Non-parity)'),
;    3231  1                     uplit (%asciz'%AINIT SEQUENCE ERROR'),
;    3232  1                     uplit (%asciz'%AHIGHER LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
;    3233  1                     uplit (%asciz'%APURGE/POLL HARDWARE FAILURE'),
;    3234  1                     uplit (%asciz'%AMAPPING REGISTER READ FAILURE (Parity or timeout)')) : vector [23],
;    3235  1          !
;    3236  1          ! RD/RX CONTROLLER DEPENDENT ERRORS CODES
;    3237  1          !
;    3238  1             RDRX_ERR = uplit (
;    3239  1                        uplit (%asciz'%AT11 CPU FAILURE'),
;    3240  1                        uplit (%asciz'%ANON-PARITY RAM ERROR'),
;    3241  1                        uplit (%asciz'%ASTATE MACHINE FAILURE - T11 ADDRESS REGISTER'),
;    3242  1                        uplit (%asciz'%ASTATE MACHINE FAILURE - Q-BUS ADDRESS REGISTER'),
;    3243  1                        uplit (%asciz'%ASTATE MACHINE FAILURE - CRC REGISTER'),
;    3244  1                        uplit (%asciz'%ASTATE MACHINE FAILURE - SERIALIZER/DESERIALIZER REGISTER'),
;    3245  1                        uplit (%asciz'%ASTATE MACHINE FAILURE - WRONG HARDWARE VERSION')) . vector [7],
;    3246  1          !
;    3247  1          ! PRINTOUTS THAT FAKE THE DRS ERROR MESSAGES
;    3248  1          !
;    3249  1             DF_MSG  = uplit (%asciz'%N%AZRQA DEV FTL  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;    3250  1             HRD_MSG = uplit (%asciz'%N%AZRQA HRD ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;    3251  1             SFT_MSG = uplit (%asciz'%N%AZRQA SFT ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06%N'),
;    3252  1             HRD_SUB = uplit (%asciz'%N%AI/O REQUEST FAILED%N'),
;    3253  1
;    3254  1
;    3255  1          !
;    3256  1          ! MISCELLANEOUS
;    3257  1          !
;    3258  1             SPACE4   = uplit (%asciz'%S4'),
;    3259  1             CRLF     = uplit (%asciz'%N'),
;    3260  1             DASH     = uplit (%asciz'%A - '),
;    3261  1             ASTERISK = uplit (%asciz'%A* ');
```

H5


```
;  3262  1     %sbttl 'DEFAULT HARDWARE P-TABLE
;  3263  1
;  3264  1     !+
;  3265  1     ! THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
;  3266  1     ! THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
;  3267  1     ! IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
;  3268  1     ! AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
;  3269  1     !-
;  3270  1
;  3271  1     BGNHW (DFPTBL);
;  3272  1
;  3273  1     global
;  3274  1         HWPT_IP_ADDR : word initial (INIT_IP_ADDR),          ! IP ADDRESS
;  3275  1         HWPT_VECTOR : word initial (INIT_INTR_VECT),         ! VECTOR ADDRESS
;  3276  1         HWPT_BR_LEVEL : word initial (INIT_BR_LEVEL),        ! BR LEVEL
;  3277  1         HWPT_DISK : WORD INITIAL (%o'000200'),               !PROTECT, WHOLE DISK, NO DUP   ZZZ
;  3278  1                                                              ! DK 0       ZZZ
;  3279  1         HWPTS0_LBN : word initial (0),                       ! STARTING TRACK LO ZZZ
;  3280  1         HWPTS1_LBN : word initial (0),                       ! STARTING TRACK HI ZZZ
;  3281  1         HWPTE0_LBN : word initial (%o'177777'),              ! ENDING TRACK LO   ZZZ
;  3282  1         HWPTE1_LBN : word initial (0),                       ! ENDING TRACK HI   ZZZ
;  3283  1         NAME_LO    : WORD INITIAL (%o'020040'),              !DISK TYPE          ZZZ
;  3284  1         NAME_HI    : WORD INITIAL (%o'020040');              !DISK TYPE          ZZZ
;  3285  1
;  3286  1     ENDHW;
```

```
;  3287  1     %sbttl 'SOFTWARE P-TABLE'
;  3288  1
;  3289  1     !+
;  3290  1     ! THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
;  3291  1     ! PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
;  3292  1     ! SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
;  3293  1     ! AT RUN TIME.
;  3294  1     !
;  3295  1
;  3296  1     BGNSW (SFPTBL);
;  3297  1
;  3298  1     global
;  3299  1         SWP_ERROR : word initial (32),                              ! HARD ERROR LIMIT FOR DROPPING UNIT
;  3300  1         SWP_XFER : WORD INITIAL (0),                                !XFER LIMIT. DEFAULT = QUICK PASS       !ZZZ
;  3301  1         SWP_FLAGS : word initial (SWF_RDM or SWF_CRC or SWF_HWC or SWF_FER  ! FLAGS (SEE DOCUMENTATION)
;  3302  1                                    or SWF_HRD or SWF_BLK),          !                                       !ZZZ
;  3303  1         SWP_DPAT : word initial (0),                                ! DATA PATTERN NUMBER
;  3304  1         SWP_RAT : word initial (99),                                ! RD51/52 OPERATION RATIO
;  3305  1         SWP_TIME : word initial (0),                                ! START TIME (HHMM)
;  3306  1         DUPROUND : WORD INITIAL (11),              !NO OF I/Os PER DBN TEST ZZZ
;  3307  1
;  3308  1     !       THE NEXT TWO LOCATIONS SHOULD BE TOGETHER
;  3309  1
;  3310  1         SWP_UCNT : word initial (MAX_UDP_CNT),                      ! USER DATA PATTERN COUNT
;  3311  1         SWP_UDPAT : vector [MAX_UDP_CNT, word];                     ! USER DATA PATTERN
;  3312  1
;  3313  1     ENDSW;
```

```
;    3314  1     %sbttl 'PROTECTION TABLE
;    3315  1
;    3316  1     !+
;    3317  1     ! THIS TABLE IS USED BY THE RUNTIME SERVICES
;    3318  1     ! TO PROTECT THE LOAD MEDIA.
;    3319  1     !
;    3320  1
;    3321  1     BGNPROT (0,  1, 6);
;    3322  1
;    3323  1     !1ST ARG =      OFFSET INTO P-TABLE FOR CSR ADDRESS
;    3324  1     !2ND ARG =      OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;    3325  1     !3RD ARG =      OFFSET INTO P-TABLE FOR DRIVE NUMBER
;    3326  1
;    3327  1     ENDPROT;
;    3328  1     end
;    3329  1
;    3330  0     eludom

                                        .TITLE  ZRQAM1 RD/RX EXERCISER
                                        .IDENT  /V02.1/
                                        .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     106                          .ASCII  /F/
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$SOFT
000022  022564'                 L$HPTP::.WORD   L$HW
000024  022614'                 L$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$MREV::
000050     003                          .BYTE   3
000051     003                          .BYTE   3
000052  000000                  L$EF::  .WORD   0
```

```
000054  000000                          .WORD   0
000056  000000                  L$SPC:: .WORD   0
000060  000000G                 L$DEVP::.WORD   L$DVTYP
000062  000000G                 L$REPP::.WORD   L$RPT
000064  000000                  L$EXP4::.WORD   0
000066  000000                  L$EXP5::.WORD   0
000070  000000G                 L$AUT:: .WORD   L$AU
000072  000000G                 L$DUT:: .WORD   L$DU
000074  000000                  L$LUN:: .WORD   0
000076  000000G                 L$DESP::.WORD   L$DESC
000100  104035                  L$LOAD::.WORD   -73743
000102  000126'                 L$ETP:: .WORD   L$ERRTBL
000104  000000G                 L$ICP:: .WORD   L$INIT
000106  000000G                 L$CCP:: .WORD   L$CLEAN
000110  000000G                 L$ACP:: .WORD   L$AUTO
000112  022676'                 L$PRT:: .WORD   L$PROT
000114  000000                  L$TEST::.WORD   0
000116  000000                  L$DLY:: .WORD   0
000120  000000                  L$HIME::.WORD   0
000122  000001                  D$PCNT::.WORD   1
000124  000000G                 L$DISPATCH::
                                        .WORD   T1
000126                          ERRTYP::.BLKW   1
000130                          ERRNBR::.BLKW   1
000132                          ERRMSG::.BLKW   1
000134                          ERRBLK::.BLKW   1
000136  111     120     040     P.AAA:  .ASCII  /IP /
000141  141     144     144             .ASCII  /add/
000144  162     145     163             .ASCII  /res/
000147  163     000     000             .ASCII  /s/<00><00>
000152  126     145     143     P.AAB:  .ASCII  /Vec/
000155  164     157     162             .ASCII  /tor/
000160  000     000                     .ASCII  <00><00>
000162  102     122     040     P.AAC:  .ASCII  /BR /
000165  114     145     166             .ASCII  /Lev/
000170  145     154     040             .ASCII  /el /
000173  133     165     163             .ASCII  /[us/
000176  165     141     154             .ASCII  /ual/
000201  154     171     040             .ASCII  /ly /
000204  064     055     122             .ASCII  /4-R/
000207  121     104     130             .ASCII  /QDX/
000212  061     040     065             .ASCII  /1 5/
000215  055     122     125             .ASCII  /-RU/
000220  130     065     060             .ASCII  /X50/
000223  135     000     000             .ASCII  /]/<00><00>
000226  104     162     151     P.AAD:  .ASCII  /Dri/
000231  166     145     040             .ASCII  /ve /
000234  156     165     155             .ASCII  /num/
000237  142     145     162             .ASCII  /ber/
000242  000     000                     .ASCII  <00><00>
000244  124     145     163     P.AAE:  .ASCII  /Tes/
000247  164     040     145             .ASCII  /t e/
000252  156     164     151             .ASCII  /nti/
```

L5


```
000255  162  145  040            .ASCII  /re /
000260  143  165  163            .ASCII  /cus/
000263  164  157  155            .ASCII  /tom/
000266  145  162  040            .ASCII  /er /
000271  141  162  145            .ASCII  /are/
000274  141  040  157            .ASCII  /a o/
000277  146  040  164            .ASCII  /f t/
000302  150  151  163            .ASCII  /his/
000305  040  144  151            .ASCII  / di/
000310  163  153  000            .ASCII  /sk/<00>
000313  000                      .ASCII  <00>
000314  114  157  167  P.AAF:    .ASCII  /Low/
000317  145  162  040            .ASCII  /er /
000322  157  143  164            .ASCII  /oct/
000325  141  154  040            .ASCII  /al /
000330  167  157  162            .ASCII  /wor/
000333  144  040  157            .ASCII  /d o/
000336  146  040  142            .ASCII  /f b/
000341  145  147  151            .ASCII  /egi/
000344  156  156  151            .ASCII  /nni/
000347  156  147  040            .ASCII  /ng /
000352  114  102  116            .ASCII  /LBN/
000355  040  141  144            .ASCII  / ad/
000360  144  162  145            .ASCII  /dre/
000363  163  163  000            .ASCII  /ss/<00>
000366  110  151  147  P.AAG:    .ASCII  /Hig/
000371  150  145  162            .ASCII  /her/
000374  040  157  143            .ASCII  / oc/
000377  164  141  154            .ASCII  /tal/
000402  040  167  157            .ASCII  / wo/
000405  162  144  040            .ASCII  /rd /
000410  157  146  040            .ASCII  /of /
000413  142  145  147            .ASCII  /beg/
000416  151  156  156            .ASCII  /inn/
000421  151  156  147            .ASCII  /ing/
000424  040  114  102            .ASCII  / LB/
000427  116  040  141            .ASCII  /N a/
000432  144  144  162            .ASCII  /dd~/
000435  145  163  163            .ASCII  /e%s/
000440  000  000                 .ASCII  <()><00>
000442  114  157  167  P.AAH:    .ASCII  /Low/
000445  145  162  040            .ASCII  /er /
000450  157  143  164            .ASCII  /oct/
000453  141  154  040            .ASCII  /al /
000456  167  157  162            .ASCII  /wor/
000461  144  040  157            .ASCII  /d o/
000464  146  040  145            .ASCII  /f e/
000467  156  144  151            .ASCII  /ndi/
000472  156  147  040            .ASCII  /ng /
000475  114  102  116            .ASCII  /LBN/
000500  040  141  144            .ASCII  / ad/
000503  144  162  145            .ASCII  /dre/
000506  163  163  000            .ASCII  /ss/<00>
```

M5


```
000511  000                             .ASCII  <00>
000512  110     151     147     P.AAI:  .ASCII  /Hig/
000515  150     145     162             .ASCII  /her/
000520  040     157     143             .ASCII  / oc/
000523  164     141     154             .ASCII  /tal/
000526  040     167     157             .ASCII  / wo/
000531  162     144     040             .ASCII  /rd /
000534  157     146     040             .ASCII  /of /
000537  145     156     144             .ASCII  /end/
000542  151     156     147             .ASCII  /ing/
000545  040     114     102             .ASCII  / LB/
000550  116     040     141             .ASCII  /N a/
000553  144     144     162             .ASCII  /ddr/
000556  145     163     163             .ASCII  /ess/
000561  000                             .ASCII  <00>
000562  127     162     '51     P.AAJ:  .ASCII  /Wri/
000565  164     145     040             .ASCII  /te /
000570  157     156     040             .ASCII  /on /
000573  143     165     163             .ASCII  /cus/
000576  164     157     155             .ASCII  /tom/
000601  145     162     040             .ASCII  /er /
000604  144     141     164             .ASCII  /dat/
000607  141     040     141             .ASCII  /a a/
000612  162     145     141             .ASCII  /rea/
000615  040     157     146             .ASCII  / of/
000620  040     164     150             .ASCII  / th/
000623  151     163     040             .ASCII  /is /
000626  144     151     163             .ASCII  /dis/
000631  153     040     165             .ASCII  /k u/
000634  156     151     164             .ASCII  /nit/
000637  000                             .ASCII  <00>
000640  052     052     040     P.AAK:  .ASCII  /** /
000643  127     101     122             .ASCII  /WAR/
000646  116     111     116             .ASCII  /NIN/
000651  107     040     055             .ASCII  /G -/
000654  040     103     125             .ASCII  / CU/
000657  123     124     117             .ASCII  /STO/
000662  115     105     122             .ASCII  /MER/
000665  040     104     101             .ASCII  / DA/
000670  124     101     040             .ASCII  /TA /
000673  101     122     105             .ASCII  /ARE/
000676  101     040     115             .ASCII  /A M/
000701  101     131     040             .ASCII  /AY /
000704  102     105     040             .ASCII  'BE /
000707  117     126     105             .ASCII  /OVE/
000712  122     127     122             .ASCII  /RWR/
000715  111     124     124             .ASCII  /ITT/
000720  105     116     041             .ASCII  /EN!/
000723  040     056     056             .ASCII  / ../
000726  056     040     103             .ASCII  /. C/
000731  117     116     106             .ASCII  /ONF/
000734  111     122     115             .ASCII  /IRM/
000737  000                             .ASCII  <00>
```

N5


```
000740  101  154  163        P.AAL:  .ASCII  /Als/
000743  157  040  162                .ASCII  /o r/
000746  165  156  040                .ASCII  /un /
000751  104  125  120                .ASCII  /DUP/
000754  040  145  170                .ASCII  / ex/
000757  145  162  143                .ASCII  /erc/
000762  151  163  145                .ASCII  /ise/
000765  162  000  000                .ASCII  /r/<00><00>
000770  127  162  151        P.AAM:  .ASCII  /Wri/
000773  164  145  040                .ASCII  /te /
000776  157  156  040                .ASCII  /on /
001001  144  151  141                .ASCII  /dia/
001004  147  156  157                .ASCII  /gno/
001007  163  164  151                .ASCII  /sti/
001012  143  040  141                .ASCII  /c a/
001015  162  145  141                .ASCII  /rea/
001020  000  000                     .ASCII  <00><00>
001022  110  141  162        P.AAN:  .ASCII  /Har/
001025  144  040  145                .ASCII  /d e/
001030  162  162  157                .ASCII  /rro/
001033  162  040  154                .ASCII  /r l/
001036  151  155  151                .ASCII  /imi/
001041  164  000  000                .ASCII  /t/<00><00>
001044  124  162  141        P.AAO:  .ASCII  /Tra/
001047  156  163  146                .ASCII  /nsf/
001052  145  162  040                .ASCII  /er /
001055  154  151  155                .ASCII  /lim/
001060  151  164  040                .ASCII  /it /
001063  151  156  040                .ASCII  /in /
001066  155  145  147                .ASCII  /meg/
001071  141  142  171                .ASCII  /aby/
001074  164  145  163                .ASCII  /tes/
001077  040  050  060                .ASCII  / (0/
001102  040  146  157                .ASCII  / fo/
001105  162  040  161                .ASCII  /r q/
001110  165  151  143                .ASCII  /uic/
001113  153  040  160                .ASCII  /k p/
001116  141  163  163                .ASCII  /ass/
001121  051  000  000                .ASCII  /)/<00><00>
001124  122  141  156        P.AAP:  .ASCII  /Ran/
001127  144  157  155                .ASCII  /dom/
001132  040  163  145                .ASCII  / se/
001135  145  153  040                .ASCII  /ek /
001140  155  157  144                .ASCII  /mod/
001143  145  000  000                .ASCII  /e/<00><00>
001146  122  145  141        P.AAQ:  .ASCII  /Rea/
001151  144  055  143                .ASCII  /d-c/
001154  157  155  160                .ASCII  /omp/
001157  141  162  145                .ASCII  /are/
001162  163  040  160                .ASCII  /s p/
001165  145  162  146                .ASCII  /erf/
001170  157  162  155                .ASCII  /orm/
001173  145  144  040                .ASCII  /ed /
```

```
001176  141  164  040                .ASCII  /at /
001201  164  150  145                .ASCII  /the/
001204  040  143  157                .ASCII  / co/
001207  156  164  162                .ASCII  /ntr/
001212  157  154  154                .ASCII  /oll/
001215  145  162  000                .ASCII  /er/<00>
001220  127  162  151  P.AAR:        .ASCII  /Wri/
001223  164  145  055                .ASCII  /te /
001226  143  157  155                .ASCII  /com/
001231  160  141  162                .ASCII  /per/
001234  145  163  040                .ASCII  /es /
001237  160  145  162                .ASCII  /per/
001242  146  157  162                .ASCII  /for/
001245  155  145  144                .ASCII  /med/
001250  040  141  164                .ASCII  / at/
001253  040  164  150                .ASCII  / th/
001256  145  040  143                .ASCII  /e c/
001261  157   56  164                .ASCII  /ont/
001264  162  157  154                .ASCII  /rol/
001267  154  145  162                .ASCII  /ler/
001272  000  000                     .ASCII  <00><00>
001274  103  150  145  P.AAS:        .ASCII  /Che/
001277  143  153  040                .ASCII  /ck /
001302  141  154  154                .ASCII  /all/
001305  040  167  162                .ASCII  / wr/
001310  151  164  145                .ASCII  /ite/
001313  163  040  141                .ASCII  /s a/
001316  164  040  150                .ASCII  /t h/
001321  157  163  164                .ASCII  /ost/
001324  040  142  171                .ASCII  / by/
001327  040  162  145                .ASCII  / re/
001332  141  144  151                .ASCII  /adi/
001335  156  147  000                .ASCII  /ng/<00>
001340  125  163  145  P.AAT:        .ASCII  /Use/
001343  162  055  144                .ASCII  /r-d/
001346  145  146  151                .ASCII  /efi/
001351  156  145  144                .ASCII  /ned/
001354  040  144  141                .ASCII  / da/
001357  164  141  040                .ASCII  /ta /
001362  160  141  164                .ASCII  /pat/
001365  164  145  162                .ASCII  /ter/
001370  156  000                     .ASCII  /n/<00>
001372  123  145  154  P.AAU:        .ASCII  /Sel/
001375  145  143  164                .ASCII  /ect/
001400  040  160  162                .ASCII  / pr/
001403  145  055  144                .ASCII  /e-d/
001406  145  146  151                .ASCII  /efi/
001411  156  145  144                .ASCII  /ned/
001414  040  144  141                .ASCII  / da/
001417  164  141  040                .ASCII  /ta /
001422  160  141  164                .ASCII  /pat/
001425  164  145  162                .ASCII  /ter/
001430  156  040  050                .ASCII  /n (/
```

```
001433  060  040  146                .ASCII  /0 f/
001436  157  162  040                .ASCII  /or /
001441  163  145  161                .ASCII  /seq/
001444  165  145  156                .ASCII  /uen/
001447  164  151  141                .ASCII  /tia/
001452  154  040  163                .ASCII  /l s/
001455  145  154  145                .ASCII  /ele/
001460  143  164  151                .ASCII  /cti/
001463  157  156  051                .ASCII  /on)/
001466  000  000                     .ASCII  <00><00>
001470  116  165  155     P.AAV:     .ASCII  /Num/
001473  142  145  162                .ASCII  /ber/
001476  040  157  146                .ASCII  / of/
001501  040  167  157                .ASCII  / wo/
001504  162  144  163                .ASCII  /rds/
001507  040  151  156                .ASCII  / in/
001512  040  144  141                .ASCII  / da/
001515  164  141  040                .ASCII  /ta /
001520  160  141  164                .ASCII  /pat/
001523  164  145  162                .ASCII  /ter/
001526  156  040  050                .ASCII  /n (/
001531  061  066  040                .ASCII  /16 /
001534  155  141  170                .ASCII  /max/
001537  151  155  165                .ASCII  /imu/
001542  155  051  000                .ASCII  /m)/<00>
001545  000                         .ASCII  <00>
001546  120  141  164     P.AAW:     .ASCII  /Pat/
001551  164  145  162                .ASCII  /ter/
001554  156  040  166                .ASCII  /n v/
001557  141  154  165                .ASCII  /alu/
001562  145  040  050                .ASCII  /e (/
001565  156  157  040                .ASCII  /no /
001570  154  145  141                .ASCII  /lea/
001573  144  151  156                .ASCII  /din/
001576  147  040  172                .ASCII  /g z/
001601  145  162  157                .ASCII  /ero/
001604  163  040  141                .ASCII  /s a/
001607  154  154  157                .ASCII  /llo/
001612  167  145  144                .ASCII  /wed/
001615  051  000  000                .ASCII  /)/<00><00>
001620  103  154  145     P.AAX:     .ASCII  /Cle/
001623  141  162  040                .ASCII  /ar /
001626  163  164  141                .ASCII  /sta/
001631  164  151  163                .ASCII  /tis/
001634  164  151  143                .ASCII  /tic/
001637  141  154  040                .ASCII  /al /
001642  164  141  142                .ASCII  /tab/
001645  154  145  163                .ASCII  /les/
001650  040  141  146                .ASCII  / af/
001653  164  145  162                .ASCII  /ter/
001656  040  160  162                .ASCII  / pr/
001661  151  156  164                .ASCII  /int/
001664  151  156  147                .ASCII  /ing/
```

D6


```
001667  000                             .ASCII  <00>
001670  120     145     162     P.AAY:  .ASCII  /Per/
001673  143     145     156             .ASCII  /cen/
001676  164     141     147             .ASCII  /tag/
001701  145     040     157             .ASCII  /e o/
001704  146     040     042             .ASCII  /f "/
001707  106     151     170             .ASCII  /Fix/
001712  145     144     040             .ASCII  /ed /
001715  104     151     163             .ASCII  /Dis/
001720  153     042     040             .ASCII  /k" /
001723  157     160     145             .ASCII  /ope/
001726  162     141     164             .ASCII  /rat/
001731  151     157     156             .ASCII  /ion/
001734  163     040     157             .ASCII  /s o/
001737  165     164     040             .ASCII  /ut /
001742  157     146     040             .ASCII  /of /
001745  164     157     164             .ASCII  /tot/
001750  141     154     040             .ASCII  /al /
001753  157     160     145             .ASCII  /ope/
001756  162     141     164             .ASCII  /rat/
001761  151     157     156             .ASCII  /ion/
001764  163     000                     .ASCII  /s/<00>
001766  125     156     151     P.AAZ:  .ASCII  /Uni/
001771  164     163     040             .ASCII  /ts /
001774  164     157     040             .ASCII  /to /
001777  142     145     040             .ASCII  /be /
002002  163     145     154             .ASCII  /sel/
002005  145     143     164             .ASCII  /ect/
002010  145     144     040             .ASCII  /ed /
002013  141     164     040             .ASCII  /at /
002016  162     141     156             .ASCII  /ran/
002021  144     157     155             .ASCII  /dom/
002024  040     050     116             .ASCII  / (N/
002027  157     054     040             .ASCII  /o, /
002032  151     155     160             .ASCII  /imp/
002035  154     151     145             .ASCII  /lie/
002040  163     040     163             .ASCII  /s s/
002043  145     161     165             .ASCII  /equ/
002046  145     156     164             .ASCII  /ent/
002051  151     141     154             .ASCII  /ial/
002054  051     000                     .ASCII  /)/<00>
002056  122     145     167     P.ABA:  .ASCII  /Rew/
002061  162     151     164             .ASCII  /rit/
002064  145     040     142             .ASCII  /e b/
002067  154     157     143             .ASCII  /loc/
002072  153     163     040             .ASCII  /ks /
002075  167     150     145             .ASCII  /whe/
002100  156     040     042             .ASCII  /n "/
002103  106     157     162             .ASCII  /For/
002106  143     145     144             .ASCII  /ced/
002111  040     105     162             .ASCII  / Er/
002114  162     157     162             .ASCII  /ror/
002117  042     040     144             .ASCII  /" d/
```

```
002122  145  164  145            .ASCII  /ete/
002125  143  164  145            .ASCII  /cte/
002130  144  040  157            .ASCII  /d o/
002133  156  040  162            .ASCII  /n r/
002136  145  141  144            .ASCII  /ead/
002141  163  000  000            .ASCII  /s/<00><00>
002144  110  141  154  P.ABB:    .ASCII  /Hal/
002147  164  040  157            .ASCII  /t o/
002152  156  040  157            .ASCII  /n o/
002155  164  150  145            .ASCII  /the/
002160  162  040  150            .ASCII  /r h/
002163  141  162  144            .ASCII  /ard/
002166  040  145  162            .ASCII  / er/
002171  162  157  162            .ASCII  /ror/
002174  163  040  050            .ASCII  /s (/
002177  043  163  040            .ASCII  /#s /
002202  063  061  055            .ASCII  /31-/
002205  063  064  054            .ASCII  /34,/
002210  040  063  066            .ASCII  / 36/
002213  055  063  067            .ASCII  /-37/
002216  054  040  063            .ASCII  /, 3/
002221  071  055  064            .ASCII  /9-4/
002224  065  051  000            .ASCII  /5)/<00>
002227  000                      .ASCII  <00>
002230  110  141  154  P.ABC:    .ASCII  /Hal/
002233  164  040  157            .ASCII  /t o/
002236  156  040  163            .ASCII  /n s/
002241  157  146  164            .ASCII  /oft/
002244  040  145  162            .ASCII  / er/
002247  162  157  162            .ASCII  /ror/
002252  163  040  050            .ASCII  /s (/
002255  043  163  040            .ASCII  /#s /
002260  065  060  055            .ASCII  /50-/
002263  065  064  051            .ASCII  /54)/
002266  000  000                  .ASCII  <00><00>
002270  110  141  154  P.ABD:    .ASCII  /Hal/
002273  164  040  157            .ASCII  /t o/
002276  156  040  142            .ASCII  /n b/
002301  141  144  055            .ASCII  /ad-/
002304  142  154  157            .ASCII  /blo/
002307  143  153  040            .ASCII  /ck /
002312  150  141  162            .ASCII  /har/
002315  144  040  145            .ASCII  /d e/
002320  162  162  157            .ASCII  /rro/
002323  162  163  040            .ASCII  /rs /
002326  050  043  163            .ASCII  /(#s/
002331  040  063  065            .ASCII  / 35/
002334  054  040  063            .ASCII  /, 3/
002337  070  051  000            .ASCII  /8)/<00>
002342  105  156  164  P.ABE:    .ASCII  /Ent/
002345  145  162  040            .ASCII  /er /
002350  164  151  155            .ASCII  /tim/
002353  145  040  141            .ASCII  /e a/
```

```
002356  163  040  110                .ASCII  /s H/
002361  110  115  115                .ASCII  /HMM/
002364  040  050  145                .ASCII  / (e/
002367  170  141  155                .ASCII  /xam/
002372  160  154  145                .ASCII  /ple/
002375  072  040  061                .ASCII  /: 1/
002400  063  060  065                .ASCII  /305/
002403  051  000  000                .ASCII  /)/<00><00>
002406  103  157  165  P.ABF:        .ASCII  /Cou/
002411  156  164  040                .ASCII  /nt /
002414  145  141  143                .ASCII  /eac/
002417  150  040  162                .ASCII  /h r/
002422  145  164  162                .ASCII  /etr/
002425  171  040  141                .ASCII  /y a/
002430  163  040  141                .ASCII  /s a/
002433  040  163  145                .ASCII  / se/
002436  160  145  162                .ASCII  /per/
002441  141  164  145                .ASCII  /ate/
002444  040  163  157                .ASCII  / so/
002447  146  164  040                .ASCII  /ft /
002452  145  162  162                .ASCII  /err/
002455  157  162  000                .ASCII  /or/<00>
002460  122  165  156  P.ABG:        .ASCII  /Run/
002463  156  151  156                .ASCII  'nin/
002466  147  040  165                .ASCII  /g u/
002471  156  144  145                .ASCII  /nde/
002474  162  040  164                .ASCII  /r t/
002477  150  145  040                .ASCII  /he /
002502  101  056  120                .ASCII  /A.P/
002505  056  124  056                .ASCII  /.T./
002510  040  115  157                .ASCII  / Mo/
002513  156  151  164                .ASCII  /nit/
002516  157  162  000                .ASCII  /or/<00>
002521  000                        P.ABH:  .ASCII  <00>
002522  124  150  145                .ASCII  /The/
002525  040  162  145                .ASCII  / re/
002530  155  141  151                 ASCII  /mai/
002533  156  151  156                .ASCII  /nin/
002536  147  040  161                .ASCII  /g a/
002541  165  145  163                .ASCII  /ues/
002544  164  151  157                .ASCII  /tio/
002547  156  163  040                .ASCII  /ns /
002552  157  156  154                .ASCII  /onl/
002555  171  040  141                .ASCII  /y a/
002560  160  160  154                .ASCII  /ppl/
002563  171  040  164                .ASCII  /y t/
002566  157  040  165                .ASCII  /o u/
002571  156  160  162                .ASCII  /npr/
002574  157  164  145                .ASCII  /ote/
002577  143  164  145                .ASCII  /cte/
002602  144  040  144                .ASCII  /d d/
002605  151  163  153                .ASCII  /isk/
002610  040  165  156                .ASCII  / un/
```

```
002613  151  164  163                .ASCII  /its/
002616  000  000                     .ASCII  <00><00>
002620  000  000             P.ABI:  .ASCII  <00><00>
002622  045  116  045         P.ABJ:  .ASCII  /%N%/
002625  101  052  052                 .ASCII  /A**/
002630  040  104  162                 .ASCII  / Dr/
002633  157  160  040                 .ASCII  /op /
002636  165  156  151                 .ASCII  /uni/
002641  164  040  045                 .ASCII  /t %/
002644  104  062  000                 .ASCII  /D2/<00>
002647  000                          .ASCII  <00>
002650  045  116  045         P.ABK:  .ASCII  /%N%/
002653  101  052  052                 .ASCII  /A**/
002656  040  120  122                 .ASCII  / PR/
002661  117  103  137                 .ASCII  /OC_/
002664  122  105  124                 .ASCII  /RET/
002667  120  113  124                 .ASCII  /PKT/
002672  072  040  103                 .ASCII  /: C/
002675  157  156  156                 .ASCII  /onn/
002700  040  111  104                 .ASCII  / ID/
002703  040  045  117                 .ASCII  / %O/
002706  066  045  101                 .ASCII  /6%A/
002711  040  162  145                 .ASCII  / re/
002714  143  145  151                 .ASCII  /cei/
002717  166  145  144                 .ASCII  /ved/
002722  000  000                     .ASCII  <00><00>
002724  045  116  045         P.ABL:  .ASCII  /%N%/
002727  101  052  052                 .ASCII  /A**/
002732  040  115  165                 .ASCII  / Mu/
002735  154  164  151                 .ASCII  /lti/
002740  055  144  162                 .ASCII  /-dr/
002743  151  166  145                 .ASCII  /ive/
002746  040  164  145                 .ASCII  / te/
002751  163  164  000                 .ASCII  /st/<00>
002754  045  116  045         P.ABM:  .ASCII  /%N%/
002757  101  052  052                 .ASCII  /A**/
002762  040  106  101                 .ASCII  / FA/
002765  124  101  114                 .ASCII  /TAL/
002770  137  105  122                 .ASCII  /_ER/
002773  122  117  122                 .ASCII  /ROR/
002776  072  040  122                 .ASCII  /: R/
003001  105  124  120                 .ASCII  /ETP/
003004  113  124  040                 .ASCII  /KT /
003007  156  157  164                 .ASCII  /not/
003012  040  141  166                 .ASCII  / av/
003015  141  151  154                 .ASCII  /ail/
003020  141  142  154                 .ASCII  /abl/
003023  145  000  000                 .ASCII  /e/<00><00>
003026  045  116  045         P.ABN:  .ASCII  /%N%/
003031  101  052  052                 .ASCII  /A**/
003034  040  106  123                 .ASCII  / FS/
003037  105  124  137                 .ASCII  /ET_/
003042  125  120  101                 .ASCII  /UPA/
```

```
003045  122  072  040          .ASCII  /R: /
003050  103  141  156          .ASCII  /Con/
003053  047  164  040          .ASCII  /'t /
003056  146  151  156          .ASCII  /fin/
003061  144  040  144          .ASCII  /d d/
003064  151  163  153          .ASCII  /isk/
003067  040  045  104          .ASCII  / %D/
003072  063  045  101          .ASCII  /3%A/
003075  040  151  156          .ASCII  / in/
003100  040  103  123          .ASCII  / CS/
003103  124  040  045          .ASCII  /T %/
003106  104  061  000          .ASCII  /D1/<00>
003111  000                    .ASCII  <00>
003112  045  116  045  P.ABO:   .ASCII  /%N%/
003115  101  052  052          .ASCII  /A**/
003120  040  102  141          .ASCII  / Ba/
003123  144  040  143          .ASCII  /d c/
003126  157  156  156          .ASCII  /onn/
003131  040  111  104          .ASCII  / ID/
003134  040  045  117          .ASCII  / %O/
003137  066  045  101          .ASCII  /6%A/
003142  040  162  145          .ASCII  / re/
003145  143  145  151          .ASCII  /cei/
003150  166  145  144          .ASCII  /ved/
003153  040  146  162          .ASCII  / fr/
003156  157  155  040          .ASCII  /om /
003161  045  117  066          .ASCII  /%O6/
003164  000  000                .ASCII  <00><00>
003166  045  116  045  P.ABP:   .ASCII  /%N%/
003171  101  052  052          .ASCII  /A**/
003174  040  115  145          .ASCII  / Me/
003177  163  163  141          .ASCII  /ssa/
003202  147  145  040          .ASCII  /ge /
003205  164  171  160          .ASCII  /typ/
003210  145  040  045          .ASCII  /e %/
003213  117  062  045          .ASCII  /02%/
003216  101  040  162          .ASCII  /A r/
003221  145  143  145          .ASCII  /ece/
003224  151  166  145          .ASCII  /ive/
003227  144  040  151          .ASCII  /d i/
003232  156  040  115          .ASCII  /n M/
003235  123  103  120          .ASCII  /SCP/
003240  040  160  141          .ASCII  / pa/
003243  143  153  145          .ASCII  /cke/
003246  164  000                .ASCII  /t/<00>
003250  045  116  045  P.ABQ:   .ASCII  /%N%/
003253  101  052  052          .ASCII  /A**/
003256  040  123  105          .ASCII  / SE/
003261  121  125  105          .ASCII  /QUE/
003264  116  072  040          .ASCII  /N: /
003267  122  105  124          .ASCII  /RET/
003272  120  113  124          .ASCII  /PKT/
003275  040  156  157          .ASCII  / no/
```

```
003300  164  040  141                .ASCII  /t a/
003303  166  141  151                .ASCII  /vai/
003306  154  141  142                .ASCII  /lab/
003311  154  145  000                .ASCII  /le/<00>
003314  045  116  045  P.ABR:        .ASCII  /%N%/
003317  101  052  052                .ASCII  /A**/
003322  040  105  162                .ASCII  / Er/
003325  162  157  162                .ASCII  /ror/
003330  040  151  156                .ASCII  / in/
003333  040  123  105                .ASCII  / SE/
003336  124  137  103                .ASCII  /T_C/
003341  124  114  122                .ASCII  /TLR/
003344  137  103  110                .ASCII  /_CH/
003347  101  122  000                .ASCII  /AR/<00>
003352  045  116  045  P.ABS:        .ASCII  /%N%/
003355  101  052  052                .ASCII  /A**/
003360  040  103  164                .ASCII  / Ct/
003363  154  162  040                .ASCII  /lr /
003366  164  151  155                .ASCII  /tim/
003371  145  157  165                .ASCII  /eou/
003374  164  040  075                .ASCII  /t =/
003377  040  045  104                .ASCII  / %D/
003402  063  045  101                .ASCII  /3%A/
003405  056  040  163                .ASCII  /. s/
003410  145  143  157                .ASCII  /eco/
003413  156  144  163                .ASCII  /nds/
003416  000  000                     .ASCII  <00><00>
003420  045  116  045  P.ABT:        .ASCII  /%N%/
003423  101  052  052                .ASCII  /A**/
003426  040  105  162                .ASCII  / Er/
003431  162  157  162                .ASCII  /ror/
003434  040  151  156                .ASCII  / in/
003437  040  125  116                .ASCII  / UN/
003442  111  124  137                .ASCII  /IT_/
003445  111  116  111                .ASCII  /INI/
003450  124  000                     .ASCII  /T/<00>
003452  045  116  045  P.ABU:        .ASCII  /%N%/
003455  101  052  052                .ASCII  /A**/
003460  040  125  116                .ASCII  / UN/
003463  111  124  137                .ASCII  /IT_/
003466  111  116  111                .ASCII  /INI/
003471  124  072  040                .ASCII  /T: /
003474  122  105  124                .ASCII  /RET/
003477  120  113  124                .ASCII  /PKT/
003502  040  150  141                .ASCII  / ha/
003505  163  040  142                .ASCII  /s b/
003510  141  144  040                .ASCII  /ad /
003513  105  116  104                .ASCII  /END/
003516  103  117  104                .ASCII  /COD/
003521  105  000  000                .ASCII  /E/<00><00>
003524  045  116  045  P.ABV:        .ASCII  /%N%/
003527  101  052  052                .ASCII  /A**/
003532  040  125  156                .ASCII  / Un/
```

```
003535  151  164  040                .ASCII  /it /
003540  163  151  172                .ASCII  /siz/
003543  145  040  050                .ASCII  /e (/
003546  114  157  051                .ASCII  /Lo)/
003551  040  075  040                .ASCII  / = /
003554  045  104  065                .ASCII  /%D5/
003557  045  101  056                .ASCII  /%A./
003562  000  000                     .ASCII  <00><00>
003564  045  116  045       P.ABW:   .ASCII  /%N%/
003567  101  052  052                .ASCII  /A**/
003572  040  125  156                .ASCII  / Un/
003575  151  164  040                .ASCII  /it /
003600  163  151  172                .ASCII  /siz/
003603  145  040  050                .ASCII  /e (/
003606  110  151  051                .ASCII  /Hi)/
003611  040  075  040                .ASCII  / = /
003614  045  104  065                .ASCII  /%D5/
003617  045  101  056                .ASCII  /%A./
003622  000  000                     .ASCII  <00><00>
003624  045  116  045       P.ABX:   .ASCII  /%N%/
003627  101  052  052                .ASCII  /A**/
003632  040  101  103                .ASCII  / AC/
003635  103  105  123                .ASCII  /CES/
003640  123  072  040                .ASCII  /S: /
003643  122  105  124                .ASCII  /RET/
003646  120  113  124                .ASCII  /PKT/
003651  040  150  141                .ASCII  / ha/
003654  163  040  142                .ASCII  /s b/
003657  141  144  040                .ASCII  /ad /
003662  105  116  104                .ASCII  /END/
003665  103  117  104                .ASCII  /COD/
003670  105  000                     .ASCII  /E/<00>
003672  045  116  045       P.ABY:   .ASCII  /%N%/
003675  101  052  052                .ASCII  /A**/
003700  040  121  111                .ASCII  / QI/
003703  117  137  125                .ASCII  /O_U/
003706  116  111  124                .ASCII  /NIT/
003711  072  040  103                .ASCII  /: C/
003714  123  124  040                .ASCII  /ST /
003717  045  104  061                .ASCII  /%D1/
003722  045  101  040                .ASCII  /%A /
003725  156  157  040                .ASCII  /no /
003730  165  156  151                .ASCII  /uni/
003733  164  040  163                .ASCII  /t s/
003736  145  154  145                .ASCII  /ele/
003741  143  164  145                .ASCII  /cte/
003744  144  000                     .ASCII  /d/<00>
003746  045  116  045       P.ABZ:   .ASCII  /%N%/
003751  101  052  052                .ASCII  /A**/
003754  040  125  156                .ASCII  / Un/
003757  151  164  040                .ASCII  /it /
003762  043  040  151                .ASCII  /# i/
003765  163  072  040                .ASCII  /s: /
```

```
003770  045  117  066            .ASCII  /%06/
003773  000                      .ASCII  <00>
003774  045  116  045  P.ACA:    .ASCII  /%N%/
003777  101  052  052            .ASCII  /A**/
004002  040  122  145            .ASCII  / Re/
004005  155  157  166            .ASCII  /mov/
004010  141  142  154            .ASCII  /abl/
004013  145  040  144            .ASCII  /e d/
004016  151  163  153            .ASCII  /isk/
004021  040  151  163            .ASCII  / is/
004024  040  163  145            .ASCII  / se/
004027  154  145  143            .ASCII  /lec/
004032  164  145  144            .ASCII  /ted/
004035  000                      .ASCII  <00>
004036  045  116  045  P.ACB:    .ASCII  /%N%/
004041  101  052  052            .ASCII  /A**/
004044  040  106  151            .ASCII  / Fi/
004047  170  145  144            .ASCII  /xed/
004052  040  144  151            .ASCII  / di/
004055  163  153  040            .ASCII  /sk /
004060  151  163  040            .ASCII  /is /
004063  163  145  154            .ASCII  /sel/
004066  145  143  164            .ASCII  /ect/
004071  145  144  000            .ASCII  /ed/<00>
004074  045  116  045  P.ACC:    .ASCII  /%N%/
004077  101  052  052            .ASCII  /A**/
004102  040  111  154            .ASCII  / Il/
004105  154  145  147            .ASCII  /leg/
004110  141  154  040            .ASCII  /al /
004113  146  165  156            .ASCII  /fun/
004116  143  164  151            .ASCII  /cti/
004121  157  156  072            .ASCII  /on:/
004124  040  045  117            .ASCII  / %0/
004127  066  000  000            .ASCII  /6/<00><00>
004132  045  116  045  P.ACD:    .ASCII  /%N%/
004135  101  052  052            .ASCII  /A**/
004140  040  103  157            .ASCII  / Co/
004143  155  155  141            .ASCII  /mma/
004146  156  144  040            .ASCII  /nd /
004151  162  145  146            .ASCII  /ref/
004154  040  043  040            .ASCII  / # /
004157  045  117  066            .ASCII  /%06/
004162  045  101  057            .ASCII  /%A/<57>
004165  045  117  066            .ASCII  /%06/
004170  045  101  040            .ASCII  /%A /
004173  050  117  143            .ASCII  /(Oc/
004176  164  051  040            .ASCII  /t) /
004201  156  157  164            .ASCII  /not/
004204  040  163  145            .ASCII  / se/
004207  156  164  040            .ASCII  /nt /
004212  142  171  040            .ASCII  /by /
004215  110  157  163            .ASCII  /Hos/
004220  164  000                 .ASCII  /t/<00>
```

```
004222  045  116  045  P.ACE:  .ASCII  /%N%/
004225  101  052  052          .ASCII  /A**/
004230  040  125  156          .ASCII  / Un/
004233  153  156  157          .ASCII  /kno/
004236  167  156  040          .ASCII  /wn /
004241  105  162  162          .ASCII  /Err/
004244  157  162  040          .ASCII  /or /
004247  114  157  147          .ASCII  /Log/
004252  040  146  157          .ASCII  / fo/
004255  162  155  141          .ASCII  /rma/
004260  164  040  045          .ASCII  /t %/
004263  117  063  045          .ASCII  /03%/
004266  101  040  162          .ASCII  /A r/
004271  145  143  145          .ASCII  /ece/
004274  151  166  145          .ASCII  /ive/
004277  144  000  000          .ASCII  /d/<00><00>
004302  045  116  045  P.ACF:  .ASCII  /%N%/
004305  101  052  052          .ASCII  /A**/
004310  040  117  160          .ASCII  / Op/
004313  055  143  157          .ASCII  /-co/
004316  144  145  040          .ASCII  /de /
004321  045  117  063          .ASCII  /%03/
004324  045  101  054          .ASCII  /%P,/
004327  040  105  156          .ASCII  / En/
004332  144  055  143          .ASCII  /d-c/
004335  157  144  145          .ASCII  /ode/
004340  040  045  117          .ASCII  / %0/
004343  063  045  101          .ASCII  /3%A/
004346  040  146  157          .ASCII  / fo/
004351  162  040  162          .ASCII  /r r/
004354  145  146  040          .ASCII  /ef /
004357  043  040  045          .ASCII  /# %/
004362  117  066  045          .ASCII  /06%/
004365  101  057  045          .ASCII  /A/<57>/%/
004370  117  066  045          .ASCII  /06%/
004373  101  040  050          .ASCII  /A (/
004376  070  051  000          .ASCII  /8)/<00>
004401  000                    .ASCII  <00>
004402  045  116  045  P.ACG:  .ASCII  /%N%/
004405  101  052  052          .ASCII  /A**/
004410  040  103  155          .ASCII  / Cm/
004413  144  055  142          .ASCII  /d-b/
004416  143  040  045          .ASCII  /c %/
004421  117  066  045          .ASCII  /06%/
004424  101  057  045          .ASCII  /A/<57>/%/
004427  117  066  045          .ASCII  /06%/
004432  101  040  122          .ASCII  /A R/
004435  163  160  055          .ASCII  /sp-/
004440  142  143  040          .ASCII  /bc /
004443  045  117  066          .ASCII  /%06/
004446  045  101  057          .ASCII  /%A/<57>
004451  045  117  066          .ASCII  /%06/
004454  045  101  040          .ASCII  /%A /
```

M6


```
004457  146  157  162                .ASCII  /for/
004462  040  045  117                .ASCII  / %O/
004465  066  045  101                .ASCII  /6%A/
004470  057  045  117                .ASCII  <57>/%O/
004473  066  045  101                .ASCII  /6%A/
004476  040  050  070                .ASCII  / (8/
004501  051  000  000                .ASCII  /)/<00><00>
004504  045  116  045  P.ACH:        .ASCII  /%N%/
004507  101  052  052                .ASCII  /A**/
004512  040  122  145                .ASCII  / Re/
004515  163  160  157                .ASCII  /spo/
004520  156  163  145                .ASCII  /nse/
004523  040  141  154                .ASCII  / al/
004526  162  145  141                .ASCII  /rea/
004531  144  171  040                .ASCII  /dy /
004534  162  145  143                .ASCII  /rec/
004537  145  151  166                .ASCII  /eiv/
004542  145  144  040                .ASCII  /ed /
004545  146  157  162                .ASCII  /for/
004550  040  143  155                .ASCII  / cm/
004553  144  040  045                .ASCII  /d %/
004556  117  066  045                .ASCII  /06%/
004561  101  057  045                .ASCII  /A/<57>/%/
004564  117  066  045                .ASCII  /06%/
004567  101  040  050                .ASCII  /A (/
004572  070  051  000                .ASCII  /8)/<00>
004575  000                         .ASCII  <00>
004576  045  116  045  P.ACI:        .ASCII  /%N%/
004601  101  052  052                .ASCII  /A**/
004604  040  106  141                .ASCII  / Fa/
004607  151  154  165                .ASCII  /ilu/
004612  162  145  040                .ASCII  /re /
004615  164  157  040                .ASCII  /to /
004620  163  145  156                .ASCII  /sen/
004623  144  040  143                .ASCII  /d c/
004626  157  155  155                .ASCII  /omm/
004631  141  156  144                .ASCII  /and/
004634  040  141  146                .ASCII  / af/
004637  164  145  162                .ASCII  /ter/
004642  040  043  040                .ASCII  / # /
004645  045  117  066                .ASCII  /%06/
004650  045  101  057                .ASCII  /%A/<57>
004653  045  117  066                .ASCII  /%06/
004656  045  101  040                .ASCII  /%A /
004661  050  070  051                .ASCII  /(8)/
004664  000  000                     .ASCII  <00><00>
004666  045  116  045  P.ACJ:        .ASCII  /%N%/
004671  101  125  116                .ASCII  /AUN/
004674  111  124  045                .ASCII  /IT%/
004677  104  062  045                .ASCII  /D2%/
004702  101  040  104                .ASCII  /A D/
004705  122  117  120                .ASCII  /ROP/
004710  120  105  104                .ASCII  /PED/
```

N6


```
004713  040     055     040             .ASCII  /  /
004716  000     000                     .ASCII  <00><00>
004720  045     101     125     P.ACL:  .ASCII  /%AU/
004723  123     105     122             .ASCII  /SER/
004726  040     103     117             .ASCII  / CO/
004731  115     115     101             .ASCII  /MMA/
004734  116     104     045             .ASCII  /ND%/
004737  116     000     000             .ASCII  /N/<00><00>
004742  045     101     103     P.ACM:  .ASCII  /%AC/
004745  117     116     106             .ASCII  /ONF/
004750  111     107     125             .ASCII  /IGU/
004753  122     101     124             .ASCII  /RAT/
004756  111     117     116             .ASCII  /ION/
004761  040     105     122             .ASCII  / ER/
004764  122     117     122             .ASCII  /ROR/
004767  045     116     000             .ASCII  /%N/<00>
004772  045     101     111     P.ACN:  .ASCII  /%AI/
004775  116     111     124             .ASCII  /NIT/
005000  040     105     122             .ASCII  / ER/
005003  122     117     122             .ASCII  /ROR/
005006  045     116     000             .ASCII  /%N/<00>
005011  000                             .ASCII  <00>
005012  045     101     124     P.ACO:  .ASCII  /%AT/
005015  122     101     116             .ASCII  /RAN/
005020  123     106     105             .ASCII  /SFE/
005023  122     040     114             .ASCII  /R L/
005026  111     115     111             .ASCII  /IMI/
005031  124     040     122             .ASCII  /T R/
005034  105     101     103             .ASCII  /EAC/
005037  110     105     104             .ASCII  /HED/
005042  045     116     000             .ASCII  /%N/<00>
005045  000                             .ASCII  <00>
005046  045     101     105     P.ACP:  .ASCII  /%AE/
005051  122     122     117             .ASCII  /RRO/
005054  122     040     114             .ASCII  /R L/
005057  111     115     111             .ASCII  /IMI/
005062  124     040     122             .ASCII  /T R/
005065  105     101     103             .ASCII  /EAC/
005070  110     105     104             .ASCII  /HED/
005073  045     116     000             .ASCII  /%N/<00>
005076  045     101     125     P.ACQ:  .ASCII  /%AU/
005101  116     122     105             .ASCII  /NRE/
005104  103     117     126             .ASCII  /COV/
005107  105     122     101             .ASCII  /ERA/
005112  102     114     105             .ASCII  /BLE/
005115  040     104     122             .ASCII  / DR/
005120  111     126     105             .ASCII  /IVE/
005123  040     105     122             .ASCII  / ER/
005126  122     117     122             .ASCII  /ROR/
005131  045     116     000             .ASCII  /%N/<00>
005134  045     101     125     P.ACR:  .ASCII  /%AU/
005137  116     122     105             .ASCII  /NRE/
005142  103     117     126             .ASCII  /COV/
```

```
005145  105  122  101          .ASCII  /ERA
005150  102  114  105          .ASCII  /BLE/
005153  040  103  117          .ASCII  / CO/
005156  116  124  122          .ASCII  /NTR/
005161  117  114  114          .ASCII  /OLL/
005164  105  122  040          .ASCII  /ER /
005167  105  122  122          .ASCII  /ERR/
005172  117  122  045          .ASCII  /OR-/
005175  116  000  000          .ASCII  /N/<00><00>
005200  045  101  106  P.ACS:  .ASCII  /-AF/
005203  101  111  114          .ASCII  /AIL/
005206  105  104  040          .ASCII  /ED /
005211  124  117  040          .ASCII  /TO /
005214  103  117  115          .ASCII  /COM/
005217  105  040  117          .ASCII  /E O/
005222  116  114  111          .ASCII  /NLI/
005225  116  105  045          .ASCII  /NE-/
005230  116  000               .ASCII  /N/<00>
005232  045  101  106  P.ACT:  .ASCII  /-AF/
005235  101  111  114          .ASCII  /AIL/
005240  105  104  040          .ASCII  /ED /
005243  124  117  040          .ASCII  /TO /
005246  101  103  103          .ASCII  /ACC/
005251  105  123  123          .ASCII  /ESS/
005254  040  105  111          .ASCII  / EI/
005257  124  110  105          .ASCII  /THE/
005262  122  040  106          .ASCII  /R F/
005265  111  122  123          .ASCII  /IRS/
005270  124  040  117          .ASCII  /T O/
005273  122  040  114          .ASCII  /R L/
005276  101  123  124          .ASCII  /AST/
005301  040  124  122          .ASCII  / TR/
005304  101  103  113          .ASCII  /ACK/
005307  040  104  125          .ASCII  / DU/
005312  122  111  116          .ASCII  /RIN/
005315  107  040  111          .ASCII  /G I/
005320  116  111  124          .ASCII  /NIT/
005323  045  116  000          .ASCII  /-N/<00>
005326  045  101  104  P.ACU:  .ASCII  /-AD/
005331  111  123  113          .ASCII  /ISK/
005334  040  127  122          .ASCII  / WR/
005337  111  124  105          .ASCII  /ITE/
005342  040  120  122          .ASCII  / PR/
005345  117  124  105          .ASCII  /OTE/
005350  103  124  105          .ASCII  /CTE/
005353  104  045  116          .ASCII  /D-N/
005356  000  000               .ASCII  <00><00>
005360  045  101  103  P.ACV:  .ASCII  /-AC/
005363  117  115  115          .ASCII  /OMM/
005366  101  116  104          .ASCII  /AND/
005371  040  124  111          .ASCII  / TI/
005374  115  105  040          .ASCII  /ME /
005377  117  125  124          .ASCII  /OUT/
```

```
005402    045    116    000                .ASCII  /%N/<00>
005405    000                               .ASCII  <00>
005406  004720                P.ACK:        .WORD   P.ACL
005410  004742'                             .WORD   P.ACM
005412  004772'                             .WORD   P.ACN
005414  005012                              .WORD   P.ACO
005416  005046'                             .WORD   P.ACP
005420  005076'                             .WORD   P.ACQ
005422  005134'                             .WORD   P.ACR
005424  005200'                             .WORD   P.ACS
005426  005232                              .WORD   P.ACT
005430  005326                              .WORD   P.ACU
005432  005360'                             .WORD   P.ACV
005434    045    116    045    P.ACW:        .ASCII  /%N%/
005437    101    120    117                .ASCII  /APO/
005442    127    105    122                .ASCII  /WER/
005445    040    104    105                .ASCII  / DE/
005450    114    101    131                .ASCII  /LAY/
005453    040    055    040                .ASCII  / - /
005456    127    101    111                .ASCII  /WAI/
005461    124    111    116                .ASCII  /TIN/
005464    107    000                       .ASCII  /G/<00>
005466    045    116    045    P.ACX:        .ASCII  /%N%/
005471    101    106    125                .ASCII  /AFU/
005474    116    103    124                .ASCII  /NCT/
005477    111    117    116                .ASCII  /ION/
005502    101    114    040                .ASCII  /AL /
005505    124    105    123                .ASCII  /TES/
005510    124    040    123                .ASCII  /T S/
005513    124    101    122                .ASCII  /TAR/
005516    124    105    104                .ASCII  /TED/
005521    000                               .ASCII  <00>
005522    045    116    045    P.ACY:        .ASCII  /%N%/
005525    116    045    101                .ASCII  /N%A/
005530    105    130    105                .ASCII  /EXE/
005533    122    103    111                .ASCII  /RCI/
005536    123    105    122                .ASCII  /SER/
005541    040    123    124                .ASCII  / ST/
005544    101    122    124                .ASCII  /ART/
005547    105    104    045                .ASCII  /ED%/
005552    116    000                       .ASCII  /N/<00>
005554    045    116    045    P.ACZ:        .ASCII  /%N%/
005557    116    045    101                .ASCII  /N%A/
005562    125    116    124                .ASCII  /UNT/
005565    040    104    123                .ASCII  / DS/
005570    113    045    123                .ASCII  /K%S/
005573    070    045    101                .ASCII  /8%A/
005576    043    040    117                .ASCII  /# O/
005601    106    040    040                .ASCII  /F  /
005604    040    043    040                .ASCII  / # /
005607    102    131    124                .ASCII  /BYT/
005612    105    123    040                .ASCII  /ES /
005615    040    040    043                .ASCII  /  #/
```

```
005620  040  117  106          .ASCII  / OF/
005623  040  040  040          .ASCII  /   /
005626  040  043  040          .ASCII  / # /
005631  102  131  124          .ASCII  /BYT/
005634  105  123  000          .ASCII  /ES/<00>
005637  000                    .ASCII  <00>
005640  045  101  040  P.ADA:  .ASCII  /%A /
005643  040  055  055          .ASCII  / --/
005646  110  101  122          .ASCII  /HAR/
005651  104  040  105          .ASCII  /D E/
005654  122  122  117          .ASCII  /RRO/
005657  122  123  055          .ASCII  /RS-/
005662  055  040  055          .ASCII  /- -/
005665  055  123  117          .ASCII  /-SO/
005670  106  124  040          .ASCII  /FT /
005673  105  122  122          .ASCII  /ERR/
005676  117  122  123          .ASCII  /ORS/
005701  055  055  000          .ASCII  /--/<00>
005704  045  116  045  P.ADB:  .ASCII  /%N%/
005707  101  040  043          .ASCII  /A #/
005712  040  040  040          .ASCII  /   /
005715  043  040  040          .ASCII  /#  /
005720  124  131  120          .ASCII  /TYP/
005723  105  040  040          .ASCII  /E  /
005726  122  105  101          .ASCII  /REA/
005731  104  123  040          .ASCII  /DS /
005734  040  040  040          .ASCII  /   /
005737  040  122  105          .ASCII  / RE/
005742  101  104  040          .ASCII  /AD /
005745  040  040  127          .ASCII  /  W/
005750  122  111  124          .ASCII  /RIT/
005753  105  123  040          .ASCII  /ES /
005756  040  040  127          .ASCII  /  W/
005761  122  111  124          .ASCII  /RIT/
005764  124  105  116          .ASCII  /TEN/
005767  000                    .ASCII  <00>
005770  045  101  040  P.ADC:  .ASCII  /%A /
005773  040  123  105          .ASCII  / SE/
005776  113  040  104          .ASCII  /K D/
006001  101  124  040          .ASCII  /AT /
006004  104  122  126          .ASCII  /DRV/
006007  040  110  123          .ASCII  / HS/
006012  124  040  123          .ASCII  /T S/
006015  105  113  040          .ASCII  /EK /
006020  104  101  124          .ASCII  /DAT/
006023  040  104  122          .ASCII  / DR/
006026  126  040  110          .ASCII  /V H/
006031  123  124  000          .ASCII  /ST/<00>
006034  045  116  045  P.ADD:  .ASCII  /%N%/
006037  101  055  055          .ASCII  /A--/
006042  055  040  055          .ASCII  /- -/
006045  055  055  040          .ASCII  /-- /
006050  055  055  055          .ASCII  /---/
```

```
006053  055  040  040          .ASCII  /-  /
006056  055  055  055          .ASCII  / --/
006061  055  055  040          .ASCII  /   /
006064  040  055  055          .ASCII  / --/
006067  055  055  055          .ASCII  /---/
006072  055  055  055          .ASCII  /---/
006075  055  040  055          .ASCII  /- -/
006100  055  055  055          .ASCII  /---/
006103  055  055  040          .ASCII  /--  /
006106  040  055  055          .ASCII  / --/
006111  055  055  055          .ASCII  /---/
006114  055  055  055          .ASCII  /---/
006117  055  000  000          .ASCII  /-/<00><00>
006122  045  101  040  P.ADE:   .ASCII  /%A /
006125  055  055  055          .ASCII  /---/
006130  040  055  055          .ASCII  / --/
006133  055  040  055          .ASCII  /- -/
006136  055  055  040          .ASCII  /--  /
006141  055  055  055          .ASCII  /---/
006144  040  055  055          .ASCII  / --/
006147  055  040  055          .ASCII  /- -/
006152  055  055  040          .ASCII  /--  /
006155  055  055  055          .ASCII  /---/
006160  040  055  055          .ASCII  / --/
006163  055  000  000          .ASCII  /-/<00><00>
006166  045  116  045  P.ADF:   .ASCII  /%N%/
006171  104  062  045          .ASCII  /D2%/
006174  104  064  045          .ASCII  /D4%/
006177  123  062  045          .ASCII  /S2%/
006202  124  000               .ASCII  /T/<00>
006204  045  104  064  P.ADG:   .ASCII  /%D4/
006207  045  132  063          .ASCII  /%Z3/
006212  045  104  063          .ASCII  /%D3/
006215  045  101  054          .ASCII  /%A,/
006220  045  132  063          .ASCII  /%Z3/
006223  045  101  054          .ASCII  /%A,/
006226  045  132  063          .ASCII  /%Z3/
006231  000                    .ASCII  <00>
006232  045  104  064  P.ADH:   .ASCII  /%D4/
006235  045  104  064          .ASCII  /%D4/
006240  045  104  064          .ASCII  /%D4/
006243  045  104  064          .ASCII  /%D4/
006246  045  104  064          .ASCII  /%D4/
006251  045  104  064          .ASCII  /%D4/
006254  045  104  064          .ASCII  /%D4/
006257  045  104  064          .ASCII  /%D4/
006262  000  000               .ASCII  <00><00>
006264  045  116  045  P.ADI:   .ASCII  /%N%/
006267  101  040  056          .ASCII  /A ./
006272  040  040  040          .ASCII  /   /
006275  056  040  040          .ASCII  /.  /
006300  103  116  124          .ASCII  /CNT/
006303  122  040  040          .ASCII  /R  /
```

```
006306  040  040  040          .ASCII  /   /
006311  040  056  040          .ASCII  / . /
006314  040  056  056          .ASCII  / ../
006317  056  056  056          .ASCII  /.../
006322  056  056  056          .ASCII  /.../
006325  056  040  040          .ASCII  /.  /
006330  040  040  040          .ASCII  /   /
006333  040  056  040          .ASCII  / . /
006336  040  056  056          .ASCII  / ../
006341  056  056  056          .ASCII  /.../
006344  056  056  056          .ASCII  /.../
006347  056  000  000          .ASCII  /./<00><00>
006352  045  101  040  P.ADJ:  .ASCII  /%A /
006355  040  040  056          .ASCII  /  ./
006360  040  040  040          .ASCII  /   /
006363  056  045  104          .ASCII  /.%D/
006366  064  045  101          .ASCII  /4%A/
006371  040  040  040          .ASCII  /   /
006374  056  040  040          .ASCII  /.  /
006377  040  056  040          .ASCII  / . /
006402  040  040  056          .ASCII  /  ./
006405  045  104  064          .ASCII  /%D4/
006410  045  101  040          .ASCII  /%A /
006413  040  040  056          .ASCII  /  ./
006416  000  000               .ASCII  <00><00>
006420  045  101  040  P.ADK:  .ASCII  /%A /
006423  040  040  056          .ASCII  /  ./
006426  040  040  040          .ASCII  /   /
006431  056  045  104          .ASCII  /.%D/
006434  064  045  101          .ASCII  /4%A/
006437  040  040  040          .ASCII  /   /
006442  056  040  040          .ASCII  /.  /
006445  040  056  040          .ASCII  / . /
006450  040  040  056          .ASCII  /  ./
006453  045  104  064          .ASCII  /%D4/
006456  045  101  040          .ASCII  /%A /
006461  040  040  056          .ASCII  /  ./
006464  000  000               .ASCII  <00><00>
006466  045  116  045  P.ADL:  .ASCII  /%N%/
006471  116  045  101          .ASCII  /N%A/
006474  125  116  111          .ASCII  /UNI/
006477  124  040  040          .ASCII  /T  /
006502  104  111  123          .ASCII  /DIS/
006505  113  040  040          .ASCII  /K  /
006510  040  040  040          .ASCII  /   /
006513  040  040  040          .ASCII  /   /
006516  040  040  040          .ASCII  /   /
006521  040  043  040          .ASCII  / # /
006524  117  106  040          .ASCII  /OF /
006527  040  040  043          .ASCII  /  #/
006532  040  102  114          .ASCII  / BL/
006535  113  123  040          .ASCII  /KS /
006540  040  040  040          .ASCII  /   /
```

```
006543  040  040  040            .ASCII  /   /
006546  043  040  117            .ASCII  /# O/
006551  106  040  040            .ASCII  /F  /
006554  040  040  043            .ASCII  /  #/
006557  040  102  114            .ASCII  / BL/
006562  113  123  040            .ASCII  /KS /
006565  000                      .ASCII  <00>
006566  045  116  045  P.ADM:    .ASCII  /%N%/
006571  101  040  040            .ASCII  /A  /
006574  043  040  040            .ASCII  /#  /
006577  040  040  040            .ASCII  /   /
006602  043  040  040            .ASCII  /#  /
006605  040  040  040            .ASCII  /   /
006610  124  131  120            .ASCII  /TYP/
006613  105  040  040            .ASCII  /E  /
006616  040  122  105            .ASCII  / RE/
006621  101  104  123            .ASCII  /ADS/
006624  040  040  040            .ASCII  /   /
006627  040  040  122            .ASCII  /  R/
006632  105  101  104            .ASCII  /EAD/
006635  040  040  040            .ASCII  /   /
006640  040  040  040            .ASCII  /   /
006643  127  122  111            .ASCII  /WRI/
006646  124  105  123            .ASCII  /TES/
006651  040  040  127            .ASCII  /  W/
006654  122  111  124            .ASCII  /RIT/
006657  124  105  116            .ASCII  /TEN/
006662  040  000                 .ASCII  / /<00>
006664  045  116  045  P.ADN:    .ASCII  /%N%/
006667  101  055  055            .ASCII  /A--/
006672  055  055  040            .ASCII  /-- /
006675  040  055  055            .ASCII  / --/
006700  055  055  040            .ASCII  /-- /
006703  040  055  055            .ASCII  / --/
006706  055  055  055            .ASCII  /---/
006711  055  055  040            .ASCII  /-- /
006714  040  055  055            .ASCII  / --/
006717  055  055  055            .ASCII  /---/
006722  055  040  040            .ASCII  /-  /
006725  040  055  055            .ASCII  / --/
006730  055  055  055            .ASCII  /---/
006733  055  040  040            .ASCII  /-  /
006736  040  040  040            .ASCII  /   /
006741  055  055  055            .ASCII  /---/
006744  055  055  055            .ASCII  /---/
006747  040  040  040            .ASCII  /   /
006752  055  055  055            .ASCII  /---/
006755  055  055  055            .ASCII  /---/
006760  040  040  000            .ASCII  /  /<00>
006763  000                      .ASCII  <00>
006764  045  116  045  P.ADO:    .ASCII  /%N%/
006767  123  061  045            .ASCII  /S1%/
006772  104  062  045            .ASCII  /D2%/
```

```
006775  123  064  045                .ASCII  /S4%/
007000  104  062  045                .ASCII  /D2%/
007003  101  040  040                .ASCII  /A  /
007006  040  104  102                .ASCII  / DB/
007011  116  040  111                .ASCII  /N I/
007014  057  117  040                .ASCII  <57>/O /
007017  040  045  104                .ASCII  / %D/
007022  066  045  123                .ASCII  /6%S/
007025  063  045  104                .ASCII  /3%D/
007030  066  045  123                .ASCII  /6%S/
007033  065  045  104                .ASCII  /5%D/
007036  066  045  123                .ASCII  /6%S/
007041  063  045  104                .ASCII  /3%D/
007044  066  000                     .ASCII  /6/<00>
007046  124  117  117     P.ADP:     .ASCII  /TOO/
007051  040  115  101                .ASCII  / MA/
007054  116  131  040                .ASCII  /NY /
007057  125  116  111                .ASCII  /UNI/
007062  124  123  000                .ASCII  /TS/<00>
007065  000                         .ASCII  <00>
007066  116  117  124     P.ADQ:     .ASCII  /NOT/
007071  040  105  116                .ASCII  / EN/
007074  117  125  107                .ASCII  /OUG/
007077  110  040  106                .ASCII  /H F/
007102  122  105  105                .ASCII  /REE/
007105  040  115  105                .ASCII  / ME/
007110  115  117  122                .ASCII  /MOR/
007113  131  040  106                .ASCII  /Y F/
007116  117  122  040                .ASCII  /OR /
007121  101  114  114                .ASCII  /ALL/
007124  117  103  101                .ASCII  /OCA/
007127  124  111  116                .ASCII  /TIN/
007132  107  040  122                .ASCII  /G R/
007135  105  101  104                .ASCII  /EAD/
007140  057  127  122                .ASCII  <57>/WR/
007143  111  124  105                .ASCII  /ITE/
007146  040  102  125                .ASCII  / BU/
007151  106  106  105                .ASCII  /FFE/
007154  122  123  000                .ASCII  /RS/<00>
007157  000                         .ASCII  <00>
007160  122  105  107     P.ADR:     .ASCII  /REG/
007163  111  123  124                .ASCII  /IST/
007166  105  122  040                .ASCII  /ER /
007171  105  130  111                .ASCII  /EXI/
007174  123  124  105                .ASCII  /STE/
007177  116  103  105                .ASCII  /NCE/
007202  040  124  105                .ASCII  / TE/
007205  123  124  040                .ASCII  /ST /
007210  106  101  111                .ASCII  /FAI/
007213  114  105  104                .ASCII  /LED/
007216  000  000                     .ASCII  <00><00>
007220  126  105  103     P.ADS:     .ASCII  /VEC/
007223  124  117  122                .ASCII  /TOR/
```

```
007226  040  124  105                    .ASCII  / TE/
007231  123  124  040                    .ASCII  /ST /
007234  106  101  111                    .ASCII  /FAI/
007237  114  105  104                    .ASCII  /LED/
007242  000  000                         .ASCII  <00><00>
007244  102  122  040          P.ADT:    .ASCII  /BR /
007247  114  105  126                    .ASCII  /LEV/
007252  105  114  040                    .ASCII  /EL /
007255  124  105  123                    .ASCII  /TES/
007260  124  040  106                    .ASCII  /T F/
007263  101  111  114                    .ASCII  /AIL/
007266  105  104  000                    .ASCII  /ED/<00>
007271  000                            .ASCII  <00>
007272  111  116  111          P.ADU:    .ASCII  /INI/
007275  124  040  123                    .ASCII  /T S/
007300  105  121  125                    .ASCII  /EQU/
007303  105  116  103                    .ASCII  /ENC/
007306  105  040  106                    .ASCII  /E F/
007311  101  111  114                    .ASCII  /AIL/
007314  105  104  000                    .ASCII  /ED/<00>
007317  000                            .ASCII  <00>
007320  106  101  124          P.ADV:    .ASCII  /FAT/
007323  101  114  040                    .ASCII  /AL /
007326  103  117  116                    .ASCII  /CON/
007331  124  122  117                    .ASCII  /TRO/
007334  114  114  105                    .ASCII  /LLE/
007337  122  040  105                    .ASCII  /R E/
007342  122  122  117                    .ASCII  /RRO/
007345  122  000  000                    .ASCII  /R/<00><00>
007350  117  116  114          P.ADW:    .ASCII  /ONL/
007353  111  116  105                    .ASCII  /INE/
007356  040  106  101                    .ASCII  / FA/
007361  111  114  105                    .ASCII  /ILE/
007364  104  000                         .ASCII  /D/<00>
007366  127  122  111          P.ADX:    .ASCII  /WRI/
007371  124  105  055                    .ASCII  /TE-/
007374  120  122  117                    .ASCII  /PRO/
007377  124  105  103                    .ASCII  /TEC/
007402  124  040  103                    .ASCII  /T C/
007405  117  116  106                    .ASCII  /ONF/
007410  114  111  103                    .ASCII  /LIC/
007413  124  000  000                    .ASCII  /T/<00><00>
007416  101  103  103          P.ADY:    .ASCII  /ACC/
007421  105  123  123                    .ASCII  /ESS/
007424  040  106  101                    .ASCII  / FA/
007427  111  114  105                    .ASCII  /ILE/
007432  104  000                         .ASCII  /D/<00>
007434  106  101  124          P.ADZ:    .ASCII  /FAT/
007437  101  114  040                    .ASCII  /AL /
007442  111  057  117                    .ASCII  /I/<57>/O/
007445  040  105  122                    .ASCII  / ER/
007450  122  117  122                    .ASCII  /ROR/
007453  000                            .ASCII  <00>
```

```
007454  104  111  123  P.AEA:  .ASCII  /DIS/
007457  113  040  124         .ASCII  /K T/
007462  131  120  105         .ASCII  /YPE/
007465  040  125  116         .ASCII  / UN/
007470  113  116  117         .ASCII  /KNO/
007473  127  116  040         .ASCII  /WN /
007476  124  117  040         .ASCII  /TO /
007501  105  130  105         .ASCII  /EXE/
007504  122  103  111         .ASCII  /RCI/
007507  123  105  122         .ASCII  /SER/
007512  000  000              .ASCII  <00><00>
007514  106  101  111  P.AEB:  .ASCII  /FAI/
007517  114  105  104         .ASCII  /LED/
007522  040  124  117         .ASCII  / TO/
007525  040  123  105         .ASCII  / SE/
007530  116  104  040         .ASCII  /ND /
007533  123  105  124         .ASCII  /SET/
007536  055  103  117         .ASCII  /-CO/
007541  116  124  122         .ASCII  /NTR/
007544  117  114  114         .ASCII  /OLL/
007547  105  122  055         .ASCII  /ER-/
007552  103  110  101         .ASCII  /CHA/
007555  122  101  103         .ASCII  /RAC/
007560  124  105  122         .ASCII  /TER/
007563  111  123  124         .ASCII  /IST/
007566  111  103  123         .ASCII  /ICS/
007571  040  103  117         .ASCII  / CO/
007574  115  115  101         .ASCII  /MMA/
007577  116  104  000         .ASCII  /ND/<00>
007602  123  105  124  P.AEC:  .ASCII  /SET/
007605  055  103  117         .ASCII  /-CO/
007610  116  124  122         .ASCII  /NTR/
007613  117  114  114         .ASCII  /OLL/
007616  105  122  055         .ASCII  /ER-/
007621  103  110  101         .ASCII  /CHA/
007624  122  101  103         .ASCII  /RAC/
007627  124  105  122         .ASCII  /TER/
007632  111  123  124         .ASCII  /IST/
007635  111  103  123         .ASCII  /ICS/
007640  040  122  105         .ASCII  / RE/
007643  123  120  117         .ASCII  /SPO/
007646  116  123  105         .ASCII  /NSE/
007651  040  110  101         .ASCII  / HA/
007654  123  040  102         .ASCII  /S B/
007657  101  104  040         .ASCII  /AD /
007662  105  116  104         .ASCII  /END/
007665  103  117  104         .ASCII  /COD/
007670  105  040  117         .ASCII  /E O/
007673  122  040  106         .ASCII  /R F/
007676  114  101  107         .ASCII  /LAG/
007701  123  040  111         .ASCII  /S I/
007704  116  040  105         .ASCII  /N E/
007707  122  122  117         .ASCII  /RRO/
```

```
007712  122  000            .ASCII  /R/<00>
007714  106  101  111  P.AED: .ASCII  /FAI/
007717  114  105  104       .ASCII  /LED/
007722  040  124  117       .ASCII  / TO/
007725  040  123  105       .ASCII  / SE/
007730  116  104  040       .ASCII  /ND /
007733  117  116  055       .ASCII  /ON-/
007736  114  111  116       .ASCII  /LIN/
007741  105  040  103       .ASCII  /E C/
007744  117  115  115       .ASCII  /OMM/
007747  101  116  104       .ASCII  /AND/
007752  000  000            .ASCII  <00><00>
007754  117  116  055  P.AEE: .ASCII  /ON-/
007757  114  111  116       .ASCII  /LIN/
007762  105  040  122       .ASCII  /E R/
007765  105  123  120       .ASCII  /ESP/
007770  117  116  123       .ASCII  /ONS/
007773  105  040  110       .ASCII  /E H/
007776  101  123  040       .ASCII  /AS /
010001  102  101  104       .ASCII  /BAD/
010004  040  105  116       .ASCII  / EN/
010007  104  103  117       .ASCII  /DCO/
010012  104  105  000       .ASCII  /DE/<00>
010015  000                 .ASCII  <00>
010016  117  116  055  P.AEF: .ASCII  /ON-/
010021  114  111  116       .ASCII  /LIN/
010024  105  040  122       .ASCII  /E R/
010027  105  123  120       .ASCII  /ESP/
010032  117  116  123       .ASCII  /ONS/
010035  105  040  110       .ASCII  /E H/
010040  101  123  040       .ASCII  /AS /
010043  125  116  113       .ASCII  /UNK/
010046  116  117  127       .ASCII  /NOW/
010051  116  040  104       .ASCII  /N D/
010054  105  126  111       .ASCII  /EVI/
010057  103  105  000       .ASCII  /CE/<00>
010062  111  057  117  P.AEG: .ASCII  /I/<57>/O/
010065  040  122  105       .ASCII  / RE/
010070  121  125  105       .ASCII  /QUE/
010073  123  124  040       .ASCII  /ST /
010076  106  101  111       .ASCII  /FAI/
010101  114  105  104       .ASCII  /LED/
010104  000  000            .ASCII  <00><00>
010106  045  101  115  P.AEH: .ASCII  /≠AM/
010111  117  122  105       .ASCII  /ORE/
010114  040  124  110       .ASCII  / TH/
010117  101  116  040       .ASCII  /AN /
010122  045  104  062       .ASCII  /≠D2/
010125  045  101  040       .ASCII  /≠A /
010130  125  116  111       .ASCII  /UNI/
010133  124  123  040       .ASCII  /TS /
010136  123  120  105       .ASCII  /SPE/
010141  103  111  106       .ASCII  /CIF/
```

```
010144  111  105  104            .ASCII  /IED/
010147  000                      .ASCII  <00>
010150  045  101  052  P.AEI:    .ASCII  /%A*/
010153  040  116  117            .ASCII  / NO/
010156  040  122  105            .ASCII  / RE/
010161  123  120  117            .ASCII  /SPO/
010164  116  123  105            .ASCII  /NSE/
010167  040  101  124            .ASCII  / AT/
010172  040  101  104            .ASCII  / AD/
010175  104  122  105            .ASCII  /DRE/
010200  123  123  040            .ASCII  /SS /
010203  045  117  066            .ASCII  /%O6/
010206  000  000                .ASCII  <00><00>
010210  045  101  052  P.AEJ:    .ASCII  /%A*/
010213  040  111  116            .ASCII  / IN/
010216  103  117  122            .ASCII  /COR/
010221  122  105  103            .ASCII  /REC/
010224  124  040  102            .ASCII  /T B/
010227  122  040  114            .ASCII  /R L/
010232  105  126  105            .ASCII  /EVE/
010235  114  040  106            .ASCII  /L F/
010240  117  122  040            .ASCII  /OR /
010243  104  122  111            .ASCII  /DRI/
010246  126  105  040            .ASCII  /VE /
010251  045  117  066            .ASCII  /%O6/
010254  000  000                .ASCII  <00><00>
010256  045  101  052  P.AEK:    .ASCII  /%A*/
010261  040  123  124            .ASCII  / ST/
010264  105  120  040            .ASCII  /EP /
010267  045  104  061            .ASCII  /%D1/
010272  045  101  040            .ASCII  /%A /
010275  122  105  101            .ASCII  /REA/
010300  104  040  105            .ASCII  /D E/
010303  122  122  117            .ASCII  /RRO/
010306  122  000                .ASCII  /R/<00>
010310  045  101  052  P.AEL:    .ASCII  /%A*/
010313  040  102  101            .ASCII  / BA/
010316  104  040  123            .ASCII  /D S/
010321  101  040  103            .ASCII  /A C/
010324  117  104  105            .ASCII  /ODE/
010327  040  106  122            .ASCII  / FR/
010332  117  115  040            .ASCII  /OM /
010335  104  122  111            .ASCII  /DRI/
010340  126  105  040            .ASCII  /VE /
010343  045  117  066            .ASCII  /%O6/
010346  000  000                .ASCII  <00><00>
010350  045  101  052  P.AEM:    .ASCII  /%A*/
010353  040  104  111            .ASCII  / DI/
010356  123  113  045            .ASCII  /SK%/
010361  104  062  045            .ASCII  /D2%/
010364  101  040  127            .ASCII  /A W/
010367  105  116  124            .ASCII  /ENT/
010372  040  117  106            .ASCII  / OF/
```

```
010375  106  114  111              .ASCII  /FLI/
010400  116  105  000              .ASCII  /NE/<00>
010403  000                       .ASCII  <00>
010404  045  101  052  P.AEN:     .ASCII  /%A*/
010407  040  104  122              .ASCII  / DR/
010412  111  126  105              .ASCII  /IVE/
010415  040  045  117              .ASCII  / %O/
010420  066  045  101              .ASCII  /6%A/
010423  040  116  117              .ASCII  / NO/
010426  124  040  120              .ASCII  /T P/
010431  122  117  103              .ASCII  /ROC/
010434  105  123  123              .ASCII  /ESS/
010437  111  116  107              .ASCII  /ING/
010442  040  103  117              .ASCII  / CO/
010445  115  115  101              .ASCII  /MMA/
010450  116  104  040              .ASCII  /ND /
010453  120  101  103              .ASCII  /PAC/
010456  113  105  124              .ASCII  /KET/
010461  123  000  000              .ASCII  /S/<00><00>
010464  045  101  052  P.AEO:     .ASCII  /%A*/
010467  040  104  111              .ASCII  / DI/
010472  123  113  045              .ASCII  /SK%/
010475  104  062  045              .ASCII  /D2%/
010500  101  040  127              .ASCII  /A W/
010503  105  116  124              .ASCII  /ENT/
010506  040  124  117              .ASCII  / TO/
010511  040  124  110              .ASCII  / TH/
010514  105  040  042              .ASCII  /E "/
010517  101  126  101              .ASCII  /AVA/
010522  111  114  101              .ASCII  /ILA/
010525  102  114  105              .ASCII  /BLE/
010530  042  040  123              .ASCII  /" S/
010533  124  101  124              .ASCII  /TAT/
010536  105  000                   .ASCII  /E/<00>
010540  040  055  040  P.AEP:     .ASCII  / - /
010543  125  116  122              .ASCII  /UNR/
010546  105  103  117              .ASCII  /ECO/
010551  107  116  111              .ASCII  /GNI/
010554  132  105  104              .ASCII  /ZED/
010557  040  115  105              .ASCII  / ME/
010562  123  123  101              .ASCII  /SSA/
010565  107  105  040              .ASCII  /GE /
010570  124  131  120              .ASCII  /TYP/
010573  105  000  000              .ASCII  /E/<00><00>
010576  040  055  040  P.AEQ:     .ASCII  / - /
010601  125  116  122              .ASCII  /UNR/
010604  105  103  117              .ASCII  /ECO/
010607  107  116  111              .ASCII  /GNI/
010612  132  105  104              .ASCII  /ZED/
010615  040  103  117              .ASCII  / CO/
010620  116  116  105              .ASCII  /NNE/
010623  103  124  111              .ASCII  /CTI/
010626  117  116  040              .ASCII  /ON /
```

```
010631  111     104     000             .ASCII  /ID/<00>
010634  040     055     040     P.AER:  .ASCII  / - /
010637  125     116     122             .ASCII  /UNR/
010642  105     103     117             .ASCII  /ECO/
010645  107     116     111             .ASCII  /GNI/
010650  132     105     104             .ASCII  /ZED/
010653  040     122     105             .ASCII  / RE/
010656  124     125     122             .ASCII  /TUR/
010661  116     040     115             .ASCII  /N M/
010664  105     123     123             .ASCII  /ESS/
010667  101     107     105             .ASCII  /AGE/
010672  000     000                     .ASCII  <00><00>
010674  040     055     040     P.AES:  .ASCII  / - /
010677  125     116     122             .ASCII  /UNR/
010702  105     103     117             .ASCII  /ECO/
010705  107     116     111             .ASCII  /GNI/
010710  132     105     104             .ASCII  /ZED/
010713  040     122     105             .ASCII  / RE/
010716  124     125     122             .ASCII  /TUR/
010721  116     040     120             .ASCII  /N P/
010724  101     103     113             .ASCII  /ACK/
010727  105     124     000             .ASCII  /ET/<00>
010732  040     055     040     P.AET:  .ASCII  / - /
010735  125     116     122             .ASCII  /UNR/
010740  105     103     117             .ASCII  /ECO/
010743  107     116     111             .ASCII  /GNI/
010746  132     105     104             .ASCII  /ZED/
010751  040     103     122             .ASCII  / CR/
010754  116     000                     .ASCII  /N/<00>
010756  040     055     040     P.AEU:  .ASCII  / - /
010761  125     116     122             .ASCII  /UNR/
010764  105     103     117             .ASCII  /ECO/
010767  107     116     111             .ASCII  /GNI/
010772  132     105     104             .ASCII  /ZED/
010775  040     117     120             .ASCII  / OP/
011000  103     117     104             .ASCII  /COD/
011003  105     000     000             .ASCII  /E/<00><00>
011006  040     055     040     P.AEV:  .ASCII  / - /
011011  115     123     103             .ASCII  /MSC/
011014  120     040     123             .ASCII  /P S/
011017  124     101     124             .ASCII  /TAT/
011022  125     123     040             .ASCII  /US /
011025  103     117     104             .ASCII  /COD/
011030  105     040     105             .ASCII  /E E/
011033  122     122     000             .ASCII  /RR/<00>
011036  040     055     040     P.AEW:  .ASCII  / - /
011041  104     125     120             .ASCII  /DUP/
011044  040     123     124             .ASCII  / ST/
011047  101     124     125             .ASCII  /ATU/
011052  123     040     103             .ASCII  /S C/
011055  117     104     105             .ASCII  /ODE/
011060  040     105     122             .ASCII  / ER/
011063  122     000     000             .ASCII  /R/<00><00>
```

```
011066  040     055     040     P.AEX:  .ASCII  /
011071  125     116     122             .ASCII  /UNR/
011074  105     103     117             .ASCII  /ECO/
011077  107     116     111             .ASCII  /GNI/
011102  132     105     104             .ASCII  /ZED/
011105  040     123     124             .ASCII  / ST/
011110  101     124     125             .ASCII  /ATU/
011113  123     040     103             .ASCII  /S C/
011116  117     104     105             .ASCII  /ODE/
011121  000                             .ASCII  <00>
011122  040     055     040     P.AEY:  .ASCII  / - /
011125  114     102     116             .ASCII  /LBN/
011130  040     110     117             .ASCII  / HO/
011133  123     124     040             .ASCII  /ST /
011136  103     117     115             .ASCII  /COM/
011141  120     101     122             .ASCII  /PAR/
011144  105     040     105             .ASCII  /E E/
011147  122     122     000             .ASCII  /RR/<00>
011152  040     055     040     P.AEZ:  .ASCII  / - /
011155  104     102     116             .ASCII  /DBN/
011160  040     110     117             .ASCII  / HO/
011163  123     124     040             .ASCII  /ST /
011166  103     117     115             .ASCII  /COM/
011171  120     101     122             .ASCII  /PAR/
011174  105     040     105             .ASCII  /E E/
011177  122     122     000             .ASCII  /RR/<00>
011202  040     055     040     P.AFA:  .ASCII  / - /
011205  125     116     101             .ASCII  /UNA/
011210  102     114     105             .ASCII  /BLE/
011213  040     124     117             .ASCII  / TO/
011216  040     114     117             .ASCII  / LO/
011221  101     104     040             .ASCII  /AD /
011224  104     125     120             .ASCII  /DUP/
011227  040     115     105             .ASCII  / ME/
011232  104     111     101             .ASCII  /DIA/
011235  000                             .ASCII  <00>
011236  040     055     040     P.AFB:  .ASCII  / - /
011241  105     122     122             .ASCII  /ERR/
011244  040     111     116             .ASCII  / IN/
011247  040     104     125             .ASCII  / OU/
011252  120     040     120             .ASCII  /P P/
011255  113     124     040             .ASCII  /KT /
011260  127     110     105             .ASCII  /WHE/
011263  116     040     125             .ASCII  /N U/
011266  123     111     116             .ASCII  /SIN/
011271  107     040     103             .ASCII  /G C/
011274  124     114     122             .ASCII  /TLR/
011277  040     114     103             .ASCII  / LC/
011302  040     120     122             .ASCII  / PR/
011305  117     107     000             .ASCII  /OG/<00>
011310  045     101     052     P.AFC:  .ASCII  /%A*/
011313  040     104     111             .ASCII  / DI/
011316  123     113     045             .ASCII  /SK%/
```

```
011321  104  062  000                .ASCII  /D2/<00>
011324  045  101  111      P.AFE:    .ASCII  /%AI/
011327  116  126  101                .ASCII  /NVA/
011332  114  111  104                .ASCII  /LID/
011335  040  103  117                .ASCII  / CO/
011340  115  115  101                .ASCII  /MMA/
011343  116  104  000                .ASCII  /ND/<00>
011346  045  101  103      P.AFF:    .ASCII  /%AC/
011351  117  115  115                .ASCII  /OMM/
011354  101  116  104                .ASCII  /AND/
011357  040  101  102                .ASCII  / AB/
011362  117  122  124                .ASCII  /ORT/
011365  105  104  000                .ASCII  /ED/<00>
011370  045  101  125      P.AFG:    .ASCII  /%AU/
011373  116  111  124                .ASCII  /NIT/
011376  040  117  106                .ASCII  / OF/
011401  106  114  111                .ASCII  /FLI/
011404  116  105  000                .ASCII  /NE/<00>
011407  000                         .ASCII  <00>
011410  045  101  124      P.AFH:    .ASCII  /%AT/
011413  122  101  116                .ASCII  /RAN/
011416  123  111  124                .ASCII  /SIT/
011421  111  117  116                .ASCII  /ION/
011424  040  124  117                .ASCII  / TO/
011427  040  101  126                .ASCII  / AV/
011432  101  111  114                .ASCII  /AIL/
011435  101  102  114                .ASCII  /ABL/
011440  105  040  123                .ASCII  /E S/
011443  124  101  124                .ASCII  /TAT/
011446  105  000                    .ASCII  /E/<00>
011450  045  101  115      P.AFI:    .ASCII  /%AM/
011453  105  104  111                .ASCII  /EDI/
011456  101  040  106                .ASCII  /A F/
011461  117  122  115                .ASCII  /ORM/
011464  101  124  040                .ASCII  /AT /
011467  105  122  122                .ASCII  /ERR/
011472  117  122  000                .ASCII  /OR/<00>
011475  000                         .ASCII  <00>
011476  045  101  127      P.AFJ:    .ASCII  /%AW/
011501  122  111  124                .ASCII  /RIT/
011504  105  055  120                .ASCII  /E-P/
011507  122  117  124                .ASCII  /ROT/
011512  105  103  124                .ASCII  'ECT/
011515  105  104  000                .ASCII  /ED/<00>
011520  045  101  104      P.AFK:    .ASCII  /%AD/
011523  105  126  111                .ASCII  /EVI/
011526  103  105  040                .ASCII  /CE /
011531  103  117  115                .ASCII  /COM/
011534  120  101  122                .ASCII  /PAR/
011537  105  040  105                .ASCII  /E E/
011542  122  122  117                .ASCII  /RRO/
011545  122  000  000                .ASCII  /R/<00><00>
011550  045  101  104      P.AFL:    .ASCII  /%AD/
```

```
011553  101  124  101                .ASCII  /ATA/
011556  040  105  122                .ASCII  / ER/
011561  122  117  122                .ASCII  /ROR/
011564  000  000                     .ASCII  <00><00>
011566  045  101  110     P.AFM:     .ASCII  /%AH/
011571  117  123  124                .ASCII  /OST/
011574  040  102  125                .ASCII  / BU/
011577  106  106  105                .ASCII  /FFE/
011602  122  040  101                .ASCII  /R A/
011605  103  103  105                .ASCII  /CCE/
011610  123  123  040                .ASCII  /SS /
011613  105  122  122                .ASCII  /ERR/
011616  117  122  000                .ASCII  /OR/<00>
011621  000                         .ASCII  <00>
011622  045  101  103     P.AFN:     .ASCII  /%AC/
011625  117  116  124                .ASCII  /ONT/
011630  122  117  114                .ASCII  /ROL/
011633  114  105  122                .ASCII  /LER/
011636  040  105  122                .ASCII  / ER/
011641  122  117  122                .ASCII  /ROR/
011644  000  000                     .ASCII  <00><00>
011646  045  101  104     P.AFO:     .ASCII  /%AD/
011651  122  111  126                .ASCII  /RIV/
011654  105  040  105                .ASCII  /E E/
011657  122  122  117                .ASCII  /RRO/
011662  122  000                     .ASCII  /R/<00>
011664  045  101  115     P.AFP:     .ASCII  /%AM/
011667  105  123  123                .ASCII  /ESS/
011672  101  107  105                .ASCII  /AGE/
011675  040  106  122                .ASCII  / FR/
011700  117  115  040                .ASCII  /OM /
011703  111  116  124                .ASCII  /INT/
011706  105  122  116                .ASCII  /ERN/
011711  101  114  040                .ASCII  /AL /
011714  104  111  101                .ASCII  /DIA/
011717  107  116  117                .ASCII  /GNO/
011722  123  124  11:                .ASCII  /STI/
011725  103  123  000                .ASCII  /CS/<00>
011730  045  101  110     P.AFQ:     .ASCII  /%AH/
011733  117  123  124                .ASCII  /OST/
011736  040  103  117                .ASCII  / CO/
011741  115  120  101                .ASCII  /MPA/
011744  122  105  040                .ASCII  /RE /
011747  105  122  122                .ASCII  /ERR/
011752  117  122  000                .ASCII  /OR/<00>
011755  000                         .ASCII  <00>
011756  045  101  103     P.AFR:     .ASCII  /%AC/
011761  117  115  115                .ASCII  /OMM/
011764  101  116  104                .ASCII  /AND/
011767  040  124  111                .ASCII  / TI/
011772  115  105  117                .ASCII  /MEO/
011775  125  124  000                .ASCII  /UT/<00>
012000  011324'              P.AFD:  .WORD   P.AFE
```

```
012002  011346'                         .WORD   P.AFF
012004  011370                          .WORD   P.AFG
012006  011410'                         .WORD   P.AFH
012010  011450'                         .WORD   P.AFI
012012  011476'                         .WORD   P.AFJ
012014  011520                          .WORD   P.AFK
012016  011550'                         .WORD   P.AFL
012020  011566                          .WORD   P.AFM
012022  011622'                         .WORD   P.AFN
012024  011646'                         .WORD   P.AFO
012026  011664                          .WORD   P.AFP
012030  011730'                         .WORD   P.AFQ
012032  011756'                         .WORD   P.AFR
012034     045     101     105  P.AFS:  .ASCII  /%AE/
012037     122     122     117          .ASCII  /RRO/
012042     122     040     114          .ASCII  /R L/
012045     117     107     040          .ASCII  /OG /
012050     115     105     123          .ASCII  /MES/
012053     123     101     107          .ASCII  /SAG/
012056     105     040     122          .ASCII  /E R/
012061     105     103     105          .ASCII  /ECE/
012064     111     126     105          .ASCII  /IVE/
012067     104     072     045          .ASCII  /D:%/
012072     116     000                  .ASCII  /N/<00>
012074     045     101     052  P.AFU:  .ASCII  /%A*/
012077     040     103     117          .ASCII  / CO/
012102     116     124     122          .ASCII  /NTR/
012105     117     114     114          .ASCII  /OLL/
012110     105     122     040          .ASCII  /ER /
012113     105     122     122          .ASCII  /ERR/
012116     117     122     045          .ASCII  /OR%/
012121     116     000     000          .ASCII  /N/<00><00>
012124     045     101     052  P.AFV:  .ASCII  /%A*/
012127     040     110     117          .ASCII  / HO/
012132     123     124     040          .ASCII  /ST /
012135     115     105     115          .ASCII  /MEM/
012140     117     122     131          .ASCII  /ORY/
012143     040     101     103          .ASCII  / AC/
012146     103     105     123          .ASCII  /CES/
012151     123     040     105          .ASCII  /S E/
012154     122     122     117          .ASCII  /RRO/
012157     122     045     116          .ASCII  /R%N/
012162     000     000                  .ASCII  <00><00>
012164     045     101     052  P.AFW:  .ASCII  /%A*/
012167     040     104     111          .ASCII  / DI/
012172     123     113     045          .ASCII  /SK%/
012175     104     062     045          .ASCII  /D2%/
012200     101     040     055          .ASCII  /A -/
012203     040     104     111          .ASCII  / DI/
012206     123     113     040          .ASCII  /SK /
012211     124     122     101          .ASCII  /TRA/
012214     116     123     106          .ASCII  /NSF/
012217     105     122     040          .ASCII  /ER /
```

```
012222  105     122     122             .ASCII  /ERR/
012225  117     122     045             .ASCII  /OR%/
012230  116     000                     .ASCII  /N/<00>
012232  045     101     052     P.AFX:  .ASCII  /%A*/
012235  040     104     111             .ASCII  / DI/
012240  123     113     045             .ASCII  /SK%/
012243  104     062     045             .ASCII  /D2%/
012246  101     040     055             .ASCII  /A  /
012251  040     042     123             .ASCII  / "S/
012254  124     101     116             .ASCII  /TAN/
012257  104     101     122             .ASCII  /DAR/
012262  104     040     104             .ASCII  /D D/
012265  111     123     113             .ASCII  /ISK/
012270  040     111     116             .ASCII  / IN/
012273  124     105     122             .ASCII  /TER/
012276  103     117     116             .ASCII  /CON/
012301  116     105     103             .ASCII  /NEC/
012304  124     042     040             .ASCII  /T" /
012307  105     122     122             .ASCII  /ERR/
012312  117     122     045             .ASCII  /OR%/
012315  116     000     000             .ASCII  /N/<00><00>
012320  045     101     052     P.AFY:  .ASCII  /%A*/
012323  040     104     111             .ASCII  / DI/
012326  123     113     045             .ASCII  /SK%/
012331  104     062     045             .ASCII  /D2%/
012334  101     040     055             .ASCII  /A -/
012337  040     042     123             .ASCII  / "S/
012342  115     101     114             .ASCII  /MAL/
012345  114     040     104             .ASCII  /L D/
012350  111     123     113             .ASCII  /ISK/
012353  042     040     105             .ASCII  /" E/
012356  122     122     117             .ASCII  /RRO/
012361  122     045     116             .ASCII  /R%N/
012364  000     000                     .ASCII  <00><00>
012366  012074                  P.AFT:  .WORD   P.AFU
012370  012124'                         .WORD   P.AFV
012372  012164'                         .WORD   P.AFW
012374  012232'                         .WORD   P.AFX
012376  012320'                         .WORD   P.AFY
012400  045     116     045     P.AFZ:  .ASCII  /%N%/
012403  101     052     040             .ASCII  /A* /
012406  123     101     072             .ASCII  /SA:/
012411  040     045     117             .ASCII  / %O/
012414  066     000                     .ASCII  /6/<00>
012416  045     116     045     P.AGA:  .ASCII  /%N%/
012421  101     052     040             .ASCII  /A* /
012424  123     124     101             .ASCII  /STA/
012427  124     125     123             .ASCII  /TUS/
012432  040     103     117             .ASCII  / CO/
012435  104     105     072             .ASCII  /DE:/
012440  040     045     117             .ASCII  / %O/
012443  062     000     000             .ASCII  /2/<00><00>
012446  045     117     064     P.AGB:  .ASCII  /%04/
```

```
012451  000                             .ASCII  <00>
012452  045     116     045     P.AGC:  .ASCII  /%N%/
012455  101     052     040             .ASCII  /A* /
012460  123     125     102             .ASCII  /SUB/
012463  137     103     117             .ASCII  /_CO/
012466  104     105     072             .ASCII  /DE:/
012471  040     000     000             .ASCII  / /<00><00>
012474  045     116     045     P.AGD:  .ASCII  /%N%/
012477  101     052     040             .ASCII  /A* /
012502  103     117     115             .ASCII  /COM/
012505  115     101     116             .ASCII  /MAN/
012510  104     072     040             .ASCII  /D: /
012513  000                             .ASCII  <00>
012514  045     101     122     P.AGE:  .ASCII  /%AR/
012517  105     101     104             .ASCII  /EAD/
012522  000     000                     .ASCII  <00><00>
012524  045     101     127     P.AGF:  .ASCII  /%AW/
012527  122     111     124             .ASCII  /RIT/
012532  105     000                     .ASCII  /E/<00>
012534  045     101     055     P.AGG:  .ASCII  /%A-/
012537  103     117     115             .ASCII  /COM/
012542  120     101     122             .ASCII  /PAR/
012545  105     000     000             .ASCII  /E/<00><00>
012550  045     101     117     P.AGH:  .ASCII  /%AO/
012553  116     114     111             .ASCII  /NLI/
012556  116     105     000             .ASCII  /NE/<00>
012561  000                             .ASCII  <00>
012562  045     101     101     P.AGI:  .ASCII  /%AA/
012565  103     103     105             .ASCII  /CCE/
012570  123     123     000             .ASCII  /SS/<00>
012573  000                             .ASCII  <00>
012574  045     117     063     P.AGJ:  .ASCII  /%03/
012577  000                             .ASCII  <00>
012600  045     116     045     P.AGK:  .ASCII  /%N%/
012603  101     052     040             .ASCII  /A* /
012606  102     101     104             .ASCII  /BAD/
012611  040     102     114             .ASCII  / BL/
012614  117     103     113             .ASCII  /OCK/
012617  040     050     110             .ASCII  / (H/
012622  157     163     164             .ASCII  /ost/
012625  040     162     145             .ASCII  / re/
012630  160     154     141             .ASCII  /pla/
012633  143     145     141             .ASCII  /cea/
012636  142     154     145             .ASCII  /ble/
012641  051     072     040             .ASCII  /): /
012644  045     104     065             .ASCII  /%D5/
012647  045     101     056             .ASCII  /%A./
012652  040     050     117             .ASCII  / (O/
012655  103     124     040             .ASCII  /CT /
012660  045     117     066             .ASCII  /%O6/
012663  045     101     051             .ASCII  /%A)/
012666  000     000                     .ASCII  <00><00>
012670  045     116     045     P.AGL:  .ASCII  /%N%/
```

```
012673  101     052     040             .ASCII  /A* /
012676  061     163     164             .ASCII  /1st/
012701  040     102     101             .ASCII  / BA/
012704  104     040     102             .ASCII  /D B/
012707  114     117     103             .ASCII  /LOC/
012712  113     040     050             .ASCII  /K (/
012715  110     157     163             .ASCII  /Hos/
012720  164     040     162             .ASCII  /t r/
012723  145     160     154             .ASCII  /epl/
012726  141     143     145             .ASCII  /ace/
012731  141     142     154             .ASCII  /abl/
012734  145     051     072             .ASCII  /e):/
012737  040     045     104             .ASCII  / %D/
012742  065     045     101             .ASCII  /5%A/
012745  056     040     050             .ASCII  /. (/
012750  117     103     124             .ASCII  /OCT/
012753  040     045     117             .ASCII  / %O/
012756  066     045     101             .ASCII  /6%A/
012761  051     000     000             .ASCII  /)/<00><00>
012764  045     116     045     P.AGM:  .ASCII  /%N%/
012767  101     052     040             .ASCII  /A* /
012772  102     101     104             .ASCII  /BAD/
012775  040     102     114             .ASCII  / BL/
013000  117     103     113             .ASCII  /OCK/
013003  040     122     105             .ASCII  / RE/
013006  120     117     122             .ASCII  /POR/
013011  124     105     104             .ASCII  /TED/
013014  040     050     122             .ASCII  / (R/
013017  145     160     154             .ASCII  /epl/
013022  141     143     145             .ASCII  /ace/
013025  144     051     072             .ASCII  /d):/
013030  040     045     104             .ASCII  / %D/
013033  045     101     056             .ASCII  /%A./
013036  040     050     117             .ASCII  / (O/
013041  103     124     040             .ASCII  /CT /
013044  045     117     066             .ASCII  /%O6/
013047  045     101     051             .ASCII  /%A)/
013052  000     000                     .ASCII  <00><00>
013054  045     116     045     P.AGN:  .ASCII  /%N%/
013057  101     052     040             .ASCII  /A* /
013062  114     102     116             .ASCII  /LBN/
013065  072     040     045             .ASCII  /: %/
013070  104     065     045             .ASCII  /D5%/
013073  101     056     040             .ASCII  /A. /
013076  050     117     103             .ASCII  /(OC/
013101  124     040     045             .ASCII  /T %/
013104  117     066     045             .ASCII  /O6%/
013107  101     051     000             .ASCII  /A)/<00>
013112  045     116     045     P.AGO:  .ASCII  /%N%/
013115  101     052     040             .ASCII  /A* /
013120  120     102     116             .ASCII  /PBN/
013123  072     040     045             .ASCII  /: %/
013126  104     065     045             .ASCII  /D5%/
```

```
013131  101  056  040            .ASCII  /A. /
013134  050  117  103            .ASCII  /(OC/
013137  124  040  045            .ASCII  /T %/
013142  117  066  045            .ASCII  /O6%/
013145  101  051  000            .ASCII  /A)/<00>
013150  045  116  045  P.AGP:    .ASCII  /%N%/
013153  101  052  040            .ASCII  /A* /
013156  114  102  116            .ASCII  /LBN/
013161  072  040  050            .ASCII  /: (/
013164  122  105  101            .ASCII  /REA/
013167  104  051  040            .ASCII  /D) /
013172  045  104  065            .ASCII  /%D5/
013175  045  101  056            .ASCII  /%A./
013200  040  050  117            .ASCII  / (O/
013203  103  124  040            .ASCII  /CT /
013206  045  117  066            .ASCII  /%O6/
013211  045  101  051            .ASCII  /%A)/
013214  000  000                 .ASCII  <00><00>
013216  045  116  045  P.AGQ:    .ASCII  /%N%/
013221  101  052  040            .ASCII  /A* /
013224  114  102  116            .ASCII  /LBN/
013227  072  040  050            .ASCII  /: (/
013232  127  122  111            .ASCII  /WRI/
013235  124  105  051            .ASCII  /TE)/
013240  040  045  104            .ASCII  / %D/
013243  065  045  101            .ASCII  /5%A/
013246  056  040  050            .ASCII  /. (/
013251  117  103  124            .ASCII  /OCT/
013254  040  045  117            .ASCII  / %O/
013257  066  045  101            .ASCII  /6%A/
013262  051  000                 .ASCII  /)/<00>
013264  045  116  045  P.AGR:    .ASCII  /%N%/
013267  101  052  040            .ASCII  /A* /
013272  122  105  120            .ASCII  /REP/
013275  114  101  103            .ASCII  /LAC/
013300  105  115  105            .ASCII  /EME/
013303  116  124  040            .ASCII  /NT /
013306  102  114  117            .ASCII  /BLO/
013311  103  113  040            .ASCII  /CK /
013314  116  117  056            .ASCII  /NO./
013317  040  045  104            .ASCII  / %D/
013322  065  045  101            .ASCII  /5%A/
013325  056  040  050            .ASCII  /. (/
013330  117  103  124            .ASCII  /OCT/
013333  040  045  117            .ASCII  / %O/
013336  066  045  101            .ASCII  /6%A/
013341  051  000  000            .ASCII  /)/<00><00>
013344  045  116  045  P.AGS:    .ASCII  /%N%/
013347  101  052  040            .ASCII  /A* /
013352  102  131  124            .ASCII  /BYT/
013355  105  040  103            .ASCII  /E C/
013360  117  125  116            .ASCII  /OUN/
013363  124  040  111            .ASCII  /T I/
```

```
013366  116  040  103                .ASCII  /N C/
013371  117  115  115                .ASCII  /OMM/
013374  101  116  104                .ASCII  /AND/
013377  072  040  045                .ASCII  /: %/
013402  104  065  045                .ASCII  /D5%/
013405  101  056  000                .ASCII  /A./<00>
013410  045  116  045  P.AGT:        .ASCII  /%N%/
013413  101  052  040                .ASCII  /A* /
013416  102  131  124                .ASCII  /BYT/
013421  105  040  103                .ASCII  /E C/
013424  117  125  116                .ASCII  /OUN/
013427  124  040  111                .ASCII  /T I/
013432  116  040  122                .ASCII  /N R/
013435  105  101  104                .ASCII  /EAD/
013440  040  103  117                .ASCII  / CO/
013443  115  115  101                .ASCII  /MMA/
013446  116  104  072                .ASCII  /ND:/
013451  040  045  104                .ASCII  / %D/
013454  065  045  101                .ASCII  /5%A/
013457  056  000  000                .ASCII  /./<00><00>
013462  045  116  045  P.AGU:        .ASCII  /%N%/
013465  101  052  040                .ASCII  /A* /
013470  102  131  124                .ASCII  /BYT/
013473  105  040  103                .ASCII  /E C/
013476  117  125  116                .ASCII  /OUN/
013501  124  040  111                .ASCII  /T I/
013504  116  040  127                .ASCII  /N W/
013507  122  111  124                .ASCII  /RIT/
013512  105  040  103                .ASCII  /E C/
013515  117  115  115                .ASCII  /OMM/
013520  101  116  104                .ASCII  /AND/
013523  072  040  045                .ASCII  /: %/
013526  104  065  045                .ASCII  /D5%/
013531  101  056  000                .ASCII  /A./<00>
013534  045  116  045  P.AGV:        .ASCII  /%N%/
013537  101  052  040                .ASCII  /A* /
013542  101  103  124                .ASCII  /ACT/
013545  125  101  114                .ASCII  /UAL/
013550  040  043  040                .ASCII  / # /
013553  117  106  040                .ASCII  /OF /
013556  102  131  124                .ASCII  /BYT/
013561  105  123  040                .ASCII  /ES /
013564  124  122  101                .ASCII  /TRA/
013567  116  123  106                .ASCII  /NSF/
013572  105  122  122                .ASCII  /ERR/
013575  105  104  072                .ASCII  /ED:/
013600  040  045  104                .ASCII  / %D/
013603  065  045  101                .ASCII  /5%A/
013606  056  000                     .ASCII  /./<00>
013610  045  116  045  P.AGW:        .ASCII  /%N%/
013613  101  052  040                .ASCII  /A* /
013616  111  057  117                .ASCII  /I/<57>/O/
013621  040  102  125                .ASCII  / BU/
```

```
013624  106  106  105                .ASCII  /FFE/
013627  122  040  101                .ASCII  /R A/
013632  104  104  122                .ASCII  /DDR/
013635  105  123  123                .ASCII  /ESS/
013640  040  050  063                .ASCII  / (3/
013643  062  040  142                .ASCII  /2 b/
013646  151  164  163                .ASCII  /its/
013651  051  072  040                .ASCII  /): /
013654  045  117  066                .ASCII  /%06/
013657  045  101  040                .ASCII  /%A /
013662  045  117  066                .ASCII  /%06/
013665  000                        .ASCII  <00>
013666  045  116  045  P.AGX:       .ASCII  /%N%/
013671  101  052  040                .ASCII  /A* /
013674  111  057  117                .ASCII  /I/<57>/O/
013677  040  102  125                .ASCII  / BU/
013702  106  106  105                .ASCII  /FFE/
013705  122  040  101                .ASCII  /R A/
013710  104  104  122                .ASCII  /DDR/
013713  105  123  123                .ASCII  /ESS/
013716  040  106  117                .ASCII  / FO/
013721  122  040  122                .ASCII  /R R/
013724  105  101  104                .ASCII  /EAD/
013727  040  050  063                .ASCII  / (3/
013732  062  040  142                .ASCII  /2 b/
013735  151  164  163                .ASCII  /its/
013740  051  072  040                .ASCII  /): /
013743  045  117  066                .ASCII  /%06/
013746  045  101  040                .ASCII  /%A /
013751  045  117  066                .ASCII  /%06/
013754  000  000                    .ASCII  <00><00>
013756  045  116  045  P.AGY:       .ASCII  /%N%/
013761  101  052  040                .ASCII  /A* /
013764  111  057  117                .ASCII  /I/<57>/O/
013767  040  102  125                .ASCII  / BU/
013772  106  106  105                .ASCII  /FFE/
013775  122  040  101                .ASCII  /R A/
014000  104  104  122                .ASCII  /DDR/
014003  105  123  123                .ASCII  /ESS/
014006  040  106  117                .ASCII  / FO/
014011  122  040  127                .ASCII  /R W/
014014  122  111  124                .ASCII  /RIT/
014017  105  040  050                .ASCII  /E (/
014022  063  062  040                .ASCII  /32 /
014025  142  151  164                .ASCII  /bit/
014030  163  051  072                .ASCII  /s):/
014033  040  045  117                .ASCII  / %O/
014036  066  045  101                .ASCII  /6%A/
014041  040  045  117                .ASCII  / %O/
014044  066  000                    .ASCII  /6/<00>
014046  045  116  045  P.AGZ:       .ASCII  /%N%/
014051  101  103  117                .ASCII  /ACO/
014054  116  124  105                .ASCII  /NTE/
```

```
014057  116     124     123             .ASCII  /NTS/
014062  040     117     106             .ASCII  / OF/
014065  040     103     117             .ASCII  / CO/
014070  115     115     101             .ASCII  /MMA/
014073  116     104     057             .ASCII  /ND/<57>
014076  122     105     123             .ASCII  /RES/
014101  120     117     116             .ASCII  /PON/
014104  123     105     040             .ASCII  /SE /
014107  120     101     103             .ASCII  /PAC/
014112  113     105     124             .ASCII  /KET/
014115  040     123     101             .ASCII  / SA/
014120  126     105     040             .ASCII  /VE /
014123  101     122     105             .ASCII  /ARE/
014126  101     072     045             .ASCII  /A:%/
014131  116     000     000             .ASCII  /N/<00><00>
014134  045     101     040     P.AHA:  .ASCII  /%A /
014137  045     117     066             .ASCII  /%06/
014142  000     000                     .ASCII  <00><00>
014144  045     116     045     P.AHB:  .ASCII  /%N%/
014147  101     052     040             .ASCII  /A* /
014152  124     111     115             .ASCII  /TIM/
014155  105     072     040             .ASCII  /E: /
014160  045     132     062             .ASCII  /%Z2/
014163  045     101     072             .ASCII  /%A:/
014166  045     132     062             .ASCII  /%Z2/
014171  045     101     040             .ASCII  /%A /
014174  110     117     125             .ASCII  /HOU/
014177  122     123     045             .ASCII  /RS%/
014202  116     000                     .ASCII  /N/<00>
014204  045     116     045     P.AHC:  .ASCII  /%N%/
014207  101     040     052             .ASCII  /A */
014212  040     104     111             .ASCII  / DI/
014215  123     113     040             .ASCII  /SK /
014220  072     040     045             .ASCII  /: %/
014223  104     062     000             .ASCII  /D2/<00>
014226  045     116     045     P.AHD:  .ASCII  /%N%/
014231  101     104     102             .ASCII  /ADB/
014234  116     072     040             .ASCII  /N: /
014237  045     104     065             .ASCII  /%D5/
014242  045     101     056             .ASCII  /%A./
014245  040     050     117             .ASCII  / (O/
014250  103     124     040             .ASCII  /CT /
014253  045     117     066             .ASCII  /%06/
014256  045     101     051             .ASCII  /%A)/
014261  000                             .ASCII  <00>
014262  045     116     045     P.AHE:  .ASCII  /%N%/
014265  101     102     131             .ASCII  /ABY/
014270  124     105     040             .ASCII  /TE /
014273  116     125     115             .ASCII  /NUM/
014276  102     105     122             .ASCII  /BER/
014301  072     040     045             .ASCII  /: %/
014304  104     063     000             .ASCII  /D3/<00>
014307  000                             .ASCII  <00>
```

```
014310  045  116  045  P.AHF:  .ASCII  /%N%/
014313  101  122  101          .ASCII  /ARA/
014316  116  104  117          .ASCII  /NDO/
014321  115  040  127          .ASCII  /M W/
014324  122  111  124          .ASCII  /RIT/
014327  124  105  116          .ASCII  /TEN/
014332  040  127  117          .ASCII  / WO/
014335  122  104  040          .ASCII  /RD /
014340  072  045  102          .ASCII  /:%B/
014343  061  066  000          .ASCII  /16/<00>
014346  045  116  045  P.AHG:  .ASCII  /%N%/
014351  101  122  101          .ASCII  /ARA/
014354  116  104  117          .ASCII  /NDO/
014357  115  040  122          .ASCII  /M R/
014362  105  101  104          .ASCII  /EAD/
014365  040  127  117          .ASCII  / WO/
014370  122  104  040          .ASCII  /RD /
014373  142  151  156          .ASCII  /bin/
014376  072  045  102          .ASCII  /:%B/
014401  061  066  045          .ASCII  /16%/
014404  101  040  157          .ASCII  /A o/
014407  143  164  072          .ASCII  /ct:/
014412  045  117  066          .ASCII  /%06/
014415  000                   .ASCII  <00>
014416  045  116  045  P.AHH:  .ASCII  /%N%/
014421  101  104  125          .ASCII  /ADU/
014424  120  114  111          .ASCII  /PLI/
014427  103  101  124          .ASCII  /CAT/
014432  105  040  125          .ASCII  /E U/
014435  116  111  124          .ASCII  /NIT/
014440  072  045  104          .ASCII  /:%D/
014443  062  045  101          .ASCII  /2%A/
014446  040  101  124          .ASCII  / AT/
014451  040  111  120          .ASCII  / IP/
014454  072  040  045          .ASCII  /: %/
014457  117  066  000          .ASCII  /06/<00>
014462  045  116  045  P.AHI:  .ASCII  /%N%/
014465  101  115  117          .ASCII  /AMO/
014470  122  105  040          .ASCII  /RE /
014473  124  110  101          .ASCII  /THA/
014476  116  040  045          .ASCII  /N %/
014501  104  061  045          .ASCII  /D1%/
014504  101  040  104          .ASCII  /A D/
014507  111  106  106          .ASCII  /IFF/
014512  105  122  105          .ASCII  /ERE/
014515  116  124  040          .ASCII  /NT /
014520  111  120  040          .ASCII  /IP /
014523  101  104  104          .ASCII  /ADD/
014526  122  105  123          .ASCII  /RES/
014531  123  105  123          .ASCII  /SES/
014534  000  000               .ASCII  <00><00>
014536  045  101  123  P.AHJ:  .ASCII  /%AS/
014541  120  111  116          .ASCII  /PIN/
```

```
014544  055  104  117          .ASCII  / DO/
014547  127  116  040          .ASCII  /WN /
014552  111  107  116          .ASCII  /IGN/
014555  117  122  105          .ASCII  /ORE/
014560  104  000               .ASCII  /D/<00>
014562  045  101  123  P.AHK:   .ASCII  /%AS/
014565  124  111  114          .ASCII  /TIL/
014570  114  040  103          .ASCII  /L C/
014573  117  116  116          .ASCII  /ONN/
014576  105  103  124          .ASCII  /ECT/
014601  105  104  000          .ASCII  /ED/<00>
014604  045  101  104  P.AHL:   .ASCII  /%AD/
014607  125  120  114          .ASCII  /UPL/
014612  111  103  101          .ASCII  /ICA/
014615  124  105  040          .ASCII  /TE /
014620  125  116  111          .ASCII  /UNI/
014623  124  040  116          .ASCII  /T N/
014626  125  115  102          .ASCII  /UMB/
014631  105  122  000          .ASCII  /ER/<00>
014634  045  101  101  P.AHM:   .ASCII  /%AA/
014637  114  122  105          .ASCII  /LRE/
014642  101  104  131          .ASCII  /ADY/
014645  040  117  116          .ASCII  / ON/
014650  114  111  116          .ASCII  /LIN/
014653  105  000  000          .ASCII  /E/<00><00>
014656  045  101  123  P.AHN:   .ASCII  /%AS/
014661  124  111  114          .ASCII  /TIL/
014664  114  040  117          .ASCII  /L O/
014667  116  114  111          .ASCII  /NLI/
014672  116  105  000          .ASCII  /NE/<00>
014675  000                    .ASCII  <00>
014676  045  101  125  P.AHO:   .ASCII  /%AU/
014701  116  111  124          .ASCII  /NIT/
014704  040  125  116          .ASCII  / UN/
014707  113  116  117          .ASCII  /KNO/
014712  127  116  040          .ASCII  /WN /
014715  117  122  040          .ASCII  /OR /
014720  117  116  114          .ASCII  /ONL/
014723  111  116  105          .ASCII  /INE/
014726  040  124  117          .ASCII  / TO/
014731  040  101  116          .ASCII  / AN/
014734  117  124  110          .ASCII  /OTH/
014737  105  122  040          .ASCII  /ER /
014742  103  117  116          .ASCII  /CON/
014745  124  122  117          .ASCII  /TRO/
014750  114  114  105          .ASCII  /LLE/
014753  122  000  000          .ASCII  /R/<00><00>
014756  045  101  116  P.AHP:   .ASCII  /%AN/
014761  117  040  126          .ASCII  /O V/
014764  117  114  125          .ASCII  /OLU/
014767  115  105  040          .ASCII  /ME /
014772  115  117  125          .ASCII  /MOU/
014775  116  124  105          .ASCII  /NTE/
```

```
015000  104  040  117          .ASCII  /D O/
015003  122  040  104          .ASCII  /R D/
015006  122  111  126          .ASCII  /RIV/
015011  105  040  104          .ASCII  /E D/
015014  111  123  101          .ASCII  /ISA/
015017  102  114  105          .ASCII  /BLE/
015022  104  040  102          .ASCII  /D B/
015025  131  040  123          .ASCII  /Y S/
015030  127  111  124          .ASCII  /WIT/
015033  103  110  000          .ASCII  /CH/<00>
015036  045  101  125  P.AHQ:   .ASCII  /%AU/
015041  116  111  124          .ASCII  /NIT/
015044  040  111  116          .ASCII  / IN/
015047  117  120  105          .ASCII  /OPE/
015052  122  101  124          .ASCII  /RAT/
015055  111  126  105          .ASCII  /IVE/
015060  040  050  122          .ASCII  / (R/
015063  104  065  061          .ASCII  /D51/
015066  057  065  062          .ASCII  <57>/52/
015071  040  167  162          .ASCII  / wr/
015074  151  164  145          .ASCII  /ite/
015077  040  146  141          .ASCII  / fa/
015102  165  154  164          .ASCII  /ult/
015105  051  000  000          .ASCII  /)/<00><00>
015110  045  101  125  P.AHR:   .ASCII  /%AU/
015113  116  111  124          .ASCII  /NIT/
015116  040  104  111          .ASCII  / DI/
015121  123  101  102          .ASCII  /SAB/
015124  114  105  104          .ASCII  /LED/
015127  040  102  131          .ASCII  / BY/
015132  040  106  111          .ASCII  / FI/
015135  105  114  104          .ASCII  /ELD/
015140  040  123  105          .ASCII  / SE/
015143  122  126  111          .ASCII  /RVI/
015146  103  105  040          .ASCII  /CE /
015151  117  122  040          .ASCII  /OR /
015154  111  116  124          .ASCII  /INT/
015157  105  122  116          .ASCII  /ERN/
015162  101  114  040          .ASCII  /AL /
015165  104  111  101          .ASCII  /DIA/
015170  107  116  117          .ASCII  /GNO/
015173  123  124  111          .ASCII  /STI/
015176  103  123  000          .ASCII  /CS/<00>
015201  000                    .ASCII  <00>
015202  045  101  042  P.AHS:   .ASCII  /%A"/
015205  106  117  122          .ASCII  /FOR/
015210  103  105  104          .ASCII  /CED/
015213  040  105  122          .ASCII  / ER/
015216  122  117  122          .ASCII  /ROR/
015221  042  040  104          .ASCII  /" D/
015224  105  124  105          .ASCII  /ETE/
015227  103  124  105          .ASCII  /CTE/
015232  104  040  127          .ASCII  /D W/
```

```
015235  110  111  114          .ASCII  /HIL/
015240  105  040  101          .ASCII  /E A/
015243  103  103  105          .ASCII  /CCE/
015246  123  123  111          .ASCII  /SSI/
015251  116  107  040          .ASCII  /NG /
015254  106  103  124          .ASCII  /FCT/
015257  040  117  122          .ASCII  / OR/
015262  040  122  103          .ASCII  / RC/
015265  124  000  000          .ASCII  /T/<00><00>
015270  045  101  123  P.AHT:  .ASCII  /%AS/
015273  105  103  124          .ASCII  /ECT/
015276  117  122  040          .ASCII  /OR /
015301  110  101  104          .ASCII  /HAD/
015304  040  102  105          .ASCII  / BE/
015307  105  116  040          .ASCII  /EN /
015312  127  122  111          .ASCII  /WRI/
015315  124  124  105          .ASCII  /TTE/
015320  116  040  127          .ASCII  /N W/
015323  111  124  110          .ASCII  /ITH/
015326  040  042  106          .ASCII  / "F/
015331  117  122  103          .ASCII  /ORC/
015334  105  104  040          .ASCII  /ED /
015337  105  122  122          .ASCII  /ERR/
015342  117  122  042          .ASCII  /OR"/
015345  040  115  117          .ASCII  / MO/
015350  104  111  106          .ASCII  /DIF/
015353  111  105  122          .ASCII  /IER/
015356  000  000               .ASCII  <00><00>
015360  045  101  106  P.AHU:  .ASCII  /%AF/
015363  103  124  040          .ASCII  /CT /
015366  117  122  040          .ASCII  /OR /
015371  122  103  124          .ASCII  /RCT/
015374  040  125  116          .ASCII  / UN/
015377  122  105  101          .ASCII  /REA/
015402  104  101  102          .ASCII  /DAB/
015405  114  105  040          .ASCII  /LE /
015410  055  040  111          .ASCII  /- I/
015413  116  126  101          .ASCII  /NVA/
015416  114  111  104          .ASCII  /LID/
015421  040  123  105          .ASCII  / SE/
015424  103  124  117          .ASCII  /CTO/
015427  122  040  110          .ASCII  /R H/
015432  105  101  104          .ASCII  /EAD/
015435  105  122  000          .ASCII  /ER/<00>
015440  045  101  110  P.AHV:  .ASCII  /%AH/
015443  105  101  104          .ASCII  /EAD/
015446  105  122  040          .ASCII  /ER /
015451  103  117  115          .ASCII  /COM/
015454  120  101  122          .ASCII  /PAR/
015457  105  040  105          .ASCII  /E E/
015462  122  122  117          .ASCII  /RRO/
015465  122  040  050          .ASCII  /R (/
015470  126  141  154          .ASCII  /Val/
```

```
015473  151  144  040                .ASCII  /id /
015476  150  145  141                .ASCII  /hea/
015501  144  145  162                .ASCII  /der/
015504  040  156  157                .ASCII  / no/
015507  164  040  146                .ASCII  /t f/
015512  157  165  156                .ASCII  /oun/
015515  144  051  000                .ASCII  /d)/<00>
015520  045  101  106  P.AHW:        .ASCII  /%AF/
015523  103  124  040                .ASCII  /CT /
015526  117  122  040                .ASCII  /OR /
015531  122  103  124                .ASCII  /RCT/
015534  040  125  116                .ASCII  / UN/
015537  122  105  101                .ASCII  /REA/
015542  104  101  102                .ASCII  /DAB/
015545  114  105  040                .ASCII  /LE /
015550  055  040  104                .ASCII  /- D/
015553  101  124  101                .ASCII  /ATA/
015556  040  123  131                .ASCII  / SY/
015561  116  103  040                .ASCII  /NC /
015564  124  111  115                .ASCII  /TIM/
015567  105  117  125                .ASCII  /EOU/
015572  124  000                     .ASCII  /T/<00>
015574  045  101  104  P.AHX:        .ASCII  /%AD/
015577  101  124  101                .ASCII  /ATA/
015602  040  123  131                .ASCII  / SY/
015605  116  103  040                .ASCII  /NC /
015610  116  117  124                .ASCII  /NOT/
015613  040  106  117                .ASCII  / FO/
015616  125  116  104                .ASCII  /UND/
015621  040  050  104                .ASCII  / (D/
015624  141  164  141                .ASCII  /ata/
015627  040  163  171                .ASCII  / sy/
015632  156  143  040                .ASCII  /nc /
015635  164  151  155                .ASCII  /tim/
015640  145  157  165                .ASCII  /eou/
015643  164  051  000                .ASCII  /t)/<00>
015646  045  101  106  P.AHY:        .ASCII  /%AF/
015651  103  124  040                .ASCII  /CT /
015654  117  122  040                .ASCII  /OR /
015657  122  103  124                .ASCII  /RCT/
015662  040  125  116                .ASCII  / UN/
015665  122  105  101                .ASCII  /REA/
015670  104  101  102                .ASCII  /DAB/
015673  114  105  040                .ASCII  /LE /
015676  055  040  125                .ASCII  /- U/
015701  116  103  117                .ASCII  /NCO/
015704  122  122  105                .ASCII  /RRE/
015707  103  124  101                .ASCII  /CTA/
015712  102  114  105                .ASCII  /BLE/
015715  040  105  103                .ASCII  / EC/
015720  103  040  105                .ASCII  /C E/
015723  122  122  117                .ASCII  /RRO/
015726  122  000                     .ASCII  /R/<00>
```

```
015730  045  101  125  P.AHZ:  .ASCII  /%AU/
015733  116  103  117         .ASCII  /NCO/
015736  122  122  105         .ASCII  /RRE/
015741  103  124  101         .ASCII  /CTA/
015744  102  114  105         .ASCII  /BLE/
015747  040  105  103         .ASCII  / EC/
015752  103  040  105         .ASCII  /C E/
015755  122  122  117         .ASCII  /RRO/
015760  122  000              .ASCII  /R/<00>
015762  045  101  122  P.AIA:  .ASCII  /%AR/
015765  103  124  040         .ASCII  /CT /
015770  103  117  122         .ASCII  /COR/
015773  122  125  120         .ASCII  /RUP/
015776  124  105  104         .ASCII  /TED/
016001  000                   .ASCII  <00>
016002  045  101  116  P.AIB:  .ASCII  /%AN/
016005  117  040  122         .ASCII  /O R/
016010  105  120  114         .ASCII  /EPL/
016013  101  103  105         .ASCII  /ACE/
016016  115  105  116         .ASCII  /MEN/
016021  124  040  102         .ASCII  /T B/
016024  114  117  103         .ASCII  /LOC/
016027  113  040  101         .ASCII  /K A/
016032  126  101  111         .ASCII  /VAI/
016035  114  101  102         .ASCII  /LAB/
016040  114  105  040         .ASCII  /LE /
016043  050  122  103         .ASCII  /(RC/
016046  124  040  146         .ASCII  /T f/
016051  165  154  154         .ASCII  /ull/
016054  051  000              .ASCII  /)/<00>
016056  045  101  104  P.AIC:  .ASCII  /%AD/
016061  111  123  113         .ASCII  /ISK/
016064  040  116  117         .ASCII  / NO/
016067  124  040  106         .ASCII  /T F/
016072  117  122  115         .ASCII  /ORM/
016075  101  124  124         .ASCII  /ATT/
016100  105  104  040         .ASCII  /ED /
016103  127  111  124         .ASCII  /WIT/
016106  110  040  065         .ASCII  /H 5/
016111  061  062  040         .ASCII  /12 /
016114  102  131  124         .ASCII  /BYT/
016117  105  040  123         .ASCII  /E S/
016122  105  103  124         .ASCII  /ECT/
016125  117  122  123         .ASCII  /ORS/
016130  000  000              .ASCII  <00><00>
016132  045  101  104  P.AID:  .ASCII  /%AD/
016135  111  123  113         .ASCII  /ISK/
016140  040  116  117         .ASCII  / NO/
016143  124  040  106         .ASCII  /T F/
016146  117  122  115         .ASCII  /ORM/
016151  101  124  124         .ASCII  /ATT/
016154  105  104  040         .ASCII  /ED /
016157  117  122  040         .ASCII  /OR /
```

```
016162  106  103  124                .ASCII  /FCT/
016165  040  103  117                .ASCII  / CO/
016170  122  122  125                .ASCII  /RRU/
016173  120  124  105                .ASCII  /PTE/
016176  104  000                     .ASCII  /D/<00>
016200  045  101  117     P.AIE:     .ASCII  /%AO/
016203  116  105  040                .ASCII  /NE /
016206  123  131  115                .ASCII  /SYM/
016211  102  117  114                .ASCII  /BOL/
016214  040  105  103                .ASCII  / EC/
016217  103  040  105                .ASCII  /C E/
016222  122  122  117                .ASCII  /RRO/
016225  122  000  000                .ASCII  /R/<00><00>
016230  045  101  124     P.AIF:     .ASCII  /%AT/
016233  127  117  040                .ASCII  /WO /
016236  123  131  115                .ASCII  /SYM/
016241  102  117  114                .ASCII  /BOL/
016244  040  105  103                .ASCII  / EC/
016247  103  040  105                .ASCII  /C E/
016252  122  122  117                .ASCII  /RRO/
016255  122  000  000                .ASCII  /R/<00><00>
016260  045  101  124     P.AIG:     .ASCII  /%AT/
016263  110  122  105                .ASCII  /HRE/
016266  105  040  123                .ASCII  /E S/
016271  131  115  102                .ASCII  /YMB/
016274  117  114  040                .ASCII  /OL /
016277  105  103  103                .ASCII  /ECC/
016302  040  105  122                .ASCII  / ER/
016305  122  117  122                .ASCII  /ROR/
016310  000  000                     .ASCII  <00><00>
016312  045  101  106     P.AIH:     .ASCII  /%AF/
016315  117  125  122                .ASCII  /OUR/
016320  040  123  131                .ASCII  / SY/
016323  115  102  117                .ASCII  /MBO/
016326  114  040  105                .ASCII  /L E/
016331  103  103  040                .ASCII  /CC /
016334  105  122  122                .ASCII  /ERR/
016337  117  122  000                .ASCII  /OR/<00>
016342  045  101  106     P.AII:     .ASCII  /%AF/
016345  111  126  105                .ASCII  /IVE/
016350  040  123  131                .ASCII  / SY/
016353  115  102  117                .ASCII  /MBO/
016356  114  040  105                .ASCII  /L E/
016361  103  103  040                .ASCII  /CC /
016364  105  122  122                .ASCII  /ERR/
016367  117  122  000                .ASCII  /OR/<00>
016372  045  101  123     P.AIJ:     .ASCII  /%AS/
016375  111  130  040                .ASCII  /IX /
016400  123  131  115                .ASCII  /SYM/
016403  102  117  114                .ASCII  /BOL/
016406  040  105  103                .ASCII  / EC/
016411  103  040  105                .ASCII  /C E/
016414  122  122  117                .ASCII  /RRO/
```

```
016417  122  000  000          .ASCII  /R/<00><00>
016422  045  101  123  P.AIK:   .ASCII  /%AS/
016425  105  126  105          .ASCII  /EVE/
016430  116  040  123          .ASCII  /N S/
016433  131  115  102          .ASCII  /YMB/
016436  117  114  040          .ASCII  /OL /
016441  105  103  103          .ASCII  /ECC/
016444  040  105  122          .ASCII  / ER/
016447  122  117  122          .ASCII  /ROR/
016452  000  000               .ASCII  <00><00>
016454  045  101  105  P.AIL:   .ASCII  /%AE/
016457  111  107  110          .ASCII  /IGH/
016462  124  040  123          .ASCII  /T S/
016465  131  115  102          .ASCII  /YMB/
016470  117  114  040          .ASCII  /OL /
016473  105  103  103          .ASCII  /ECC/
016476  040  105  122          .ASCII  / ER/
016501  122  117  122          .ASCII  /ROR/
016504  000  000               .ASCII  <00><00>
016506  045  101  103  P.AIM:   .ASCII  /%AC/
016511  117  122  122          .ASCII  /ORR/
016514  105  103  124          .ASCII  /ECT/
016517  101  102  114          .ASCII  /ABL/
016522  105  040  105          .ASCII  /E E/
016525  122  122  117          .ASCII  /RRO/
016530  122  040  111          .ASCII  /R I/
016533  116  040  105          .ASCII  /N E/
016536  103  103  040          .ASCII  /CC /
016541  106  111  105          .ASCII  /FIE/
016544  114  104  000          .ASCII  /LD/<00>
016547  000                    .ASCII  <00>
016550  045  101  125  P.AIN:   .ASCII  /%AU/
016553  116  111  124          .ASCII  /NIT/
016556  040  123  117          .ASCII  / SO/
016561  106  124  127          .ASCII  /FTW/
016564  101  122  105          .ASCII  /ARE/
016567  040  127  122          .ASCII  / WR/
016572  111  124  105          .ASCII  /ITE/
016575  040  120  122          .ASCII  / PR/
016600  117  124  105          .ASCII  /OTE/
016603  103  124  105          .ASCII  /CTE/
016606  104  000               .ASCII  /D/<00>
016610  045  101  125  P.AIO:   .ASCII  /%AU/
016613  116  111  124          .ASCII  /NIT/
016616  040  110  101          .ASCII  / HA/
016621  122  104  127          .ASCII  /RDW/
016624  101  122  105          .ASCII  /ARE/
016627  040  127  122          .ASCII  / WR/
016632  111  124  105          .ASCII  /ITE/
016635  040  120  122          .ASCII  / PR/
016640  117  124  105          .ASCII  /OTE/
016643  103  124  105          .ASCII  /CTE/
016646  104  000               .ASCII  /D/<00>
```

```
016650  045  101  117  P.AIP:  .ASCII  /%AO/
016653  104  104  040          .ASCII  /DD /
016656  124  122  101          .ASCII  /TRA/
016661  116  123  106          .ASCII  /NSF/
016664  105  122  040          .ASCII  /ER /
016667  101  104  104          .ASCII  /ADD/
016672  122  105  123          .ASCII  /RES/
016675  123  000  000          .ASCII  /S/<00><00>
016700  045  101  117  P.AIQ:  .ASCII  /%AO/
016703  104  104  040          .ASCII  /DD /
016706  102  131  124          .ASCII  /BYT/
016711  105  040  103          .ASCII  /E C/
016714  117  125  116          .ASCII  /OUN/
016717  124  000  000          .ASCII  /T/<00><00>
016722  045  101  116  P.AIR:  .ASCII  /%AN/
016725  117  116  055          .ASCII  /ON-/
016730  105  130  111          .ASCII  /EXI/
016733  123  124  105          .ASCII  /STE/
016736  116  124  040          .ASCII  /NT /
016741  110  117  123          .ASCII  /HOS/
016744  124  040  115          .ASCII  /T M/
016747  105  115  117          .ASCII  /EMO/
016752  122  131  000          .ASCII  /RY/<00>
016755  000                   .ASCII  <00>
016756  045  101  110  P.AIS:  .ASCII  /%AH/
016761  117  123  124          .ASCII  /OST/
016764  040  115  105          .ASCII  / ME/
016767  115  117  122          .ASCII  /MOR/
016772  131  040  120          .ASCII  /Y P/
016775  101  122  111          .ASCII  /ARI/
017000  124  131  040          .ASCII  /TY /
017003  105  122  122          .ASCII  /ERR/
017006  117  122  000          .ASCII  /OR/<00>
017011  000                   .ASCII  <00>
017012  045  101  103  P.AIT:  .ASCII  /%AC/
017015  117  115  115          .ASCII  /OMM/
017020  101  116  104          .ASCII  /AND/
017023  040  124  111          .ASCII  / TI/
017026  115  117  125          .ASCII  /MOU/
017031  124  040  117          .ASCII  /T O/
017034  122  040  122          .ASCII  /R R/
017037  105  124  122          .ASCII  /ETR/
017042  131  040  114          .ASCII  /Y L/
017045  111  115  111          .ASCII  /IMI/
017050  124  040  105          .ASCII  /T E/
017053  130  103  105           ASCII  /XCE/
017056  105  104  105          .ASCII  /EDE/
017061  104  000  000          .ASCII  /D/<00><00>
017064  045  101  123  P.AIU:  .ASCII  /%AS/
017067  105  122  111          .ASCII  /ERI/
017072  101  114  111          .ASCII  /ALI/
017075  132  105  122          .ASCII  /ZER/
017100  057  104  105          .ASCII  <57>/DE/
```

```
017103  123  105  122                .ASCII  /SER/
017106  111  101  114                .ASCII  /IAL/
017111  111  132  105                .ASCII  /IZE/
017114  122  040  117                .ASCII  /R O/
017117  126  105  122                .ASCII  /VER/
017122  122  125  116                .ASCII  /RUN/
017125  040  117  122                .ASCII  / OR/
017130  040  125  116                .ASCII  / UN/
017133  104  105  122                .ASCII  /DER/
017136  122  125  116                .ASCII  /RUN/
017141  000                        .ASCII  <00>
017142  045  101  042     P.AIV:     .ASCII  /%A"/
017145  105  122  122                .ASCII  /ERR/
017150  117  122  040                .ASCII  /OR /
017153  104  105  124                .ASCII  /DET/
017156  105  103  124                .ASCII  /ECT/
017161  111  117  116                .ASCII  /ION/
017164  040  103  117                .ASCII  / CO/
017167  104  105  042                .ASCII  /DE"/
017172  040  105  122                .ASCII  / ER/
017175  122  117  122                .ASCII  /ROR/
017200  000  000                    .ASCII  <00><00>
017202  045  101  111     P.AIW:     .ASCII  /%AI/
017205  116  103  117                .ASCII  /NCO/
017210  116  123  111                .ASCII  /NSI/
017213  123  124  105                .ASCII  /STE/
017216  116  124  040                .ASCII  /NT /
017221  111  116  124                .ASCII  /INT/
017224  105  122  116                .ASCII  /ERN/
017227  101  114  040                .ASCII  /AL /
017232  104  101  124                .ASCII  /DAT/
017235  101  040  123                .ASCII  /A S/
017240  124  122  125                .ASCII  /TRU/
017243  103  124  125                .ASCII  /CTU/
017246  122  105  000                .ASCII  /RE/<00>
017251  000                        .ASCII  <00>
017252  045  101  104     P.AIX:     .ASCII  /%AD/
017255  122  111  126                .ASCII  /RIV/
017260  105  040  103                .ASCII  /E C/
017263  117  115  115                .ASCII  /OMM/
017266  101  116  104                .ASCII  /AND/
017271  040  124  111                .ASCII  / TI/
017274  115  105  117                .ASCII  /MEO/
017277  125  124  040                .ASCII  /UT /
017302  050  116  157                .ASCII  /(No/
017305  040  162  145                .ASCII  / re/
017310  163  160  157                .ASCII  /spo/
017313  156  163  145                .ASCII  /nse/
017316  040  157  162                .ASCII  / or/
017321  040  163  145                .ASCII  / se/
017324  145  153  040                .ASCII  /ek /
017327  151  156  143                .ASCII  /inc/
017332  157  155  160                .ASCII  /omp/
```

```
017335  154  145  164              .ASCII  /let/
017340  145  051  000              .ASCII  /e)/<00>
017343  000                       .ASCII  <00>
017344  045  101  103  P.AIY:     .ASCII  /%AC/
017347  117  116  124              .ASCII  /ONT/
017352  122  117  114              .ASCII  /ROL/
017355  114  105  122              .ASCII  /LER/
017360  040  104  105              .ASCII  / DE/
017363  124  105  103              .ASCII  /TEC/
017366  124  105  104              .ASCII  /TED/
017371  040  124  122              .ASCII  / TR/
017374  101  116  123              .ASCII  /ANS/
017377  115  111  123              .ASCII  /MIS/
017402  123  111  117              .ASCII  /SIO/
017405  116  040  117              .ASCII  /N O/
017410  122  040  120              .ASCII  /R P/
017413  122  117  124              .ASCII  /ROT/
017416  117  103  117              .ASCII  /OCO/
017421  114  040  105              .ASCII  /L E/
017424  122  122  117              .ASCII  /RRO/
017427  122  000  000              .ASCII  /R/<00><00>
017432  045  101  120  P.AIZ:     .ASCII  /%AP/
017435  117  123  111              .ASCII  /OSI/
017440  124  111  117              .ASCII  /TIO/
017443  116  040  105              .ASCII  /N E/
017446  122  122  117              .ASCII  /RRO/
017451  122  040  050              .ASCII  /R (/
017454  115  151  163              .ASCII  /Mis/
017457  055  163  145              .ASCII  /-se/
017462  145  153  051              .ASCII  /ek)/
017465  000                       .ASCII  <00>
017466  045  101  114  P.AJA:     .ASCII  /%AL/
017471  117  123  124              .ASCII  /OST/
017474  040  122  105              .ASCII  / RE/
017477  101  104  057              .ASCII  /AD/<57>
017502  127  122  111              .ASCII  /WRI/
017505  124  105  040              .ASCII  /TE /
017510  122  105  101              .ASCII  /REA/
017513  104  131  040              .ASCII  /DY /
017516  104  125  122              .ASCII  /DUR/
017521  111  116  107              .ASCII  /ING/
017524  057  102  105              .ASCII  <57>/BE/
017527  124  127  105              .ASCII  /TWE/
017532  105  116  040              .ASCII  /EN /
017535  124  122  101              .ASCII  /TRA/
017540  116  123  106              .ASCII  /NSF/
017543  105  122  123              .ASCII  /ERS/
017546  000  000                   .ASCII  <00><00>
017550  045  101  104  P.AJB:     .ASCII  /%AD/
017553  122  111  126              .ASCII  /RIV/
017556  105  040  103              .ASCII  /E C/
017561  114  117  103              .ASCII  /LOC/
017564  113  040  104              .ASCII  /K D/
```

```
017567  122     117     120             .ASCII  /ROP/
017572  117     125     124             .ASCII  /OUT/
017575  000                             .ASCII  <00>
017576  045     101     114     P.AJC:  .ASCII  /%AL/
017601  117     123     124             .ASCII  /OST/
017604  040     122     105             .ASCII  / RE/
017607  103     105     111             .ASCII  /CEI/
017612  126     105     122             .ASCII  /VER/
017615  040     122     105             .ASCII  / RE/
017620  101     104     131             .ASCII  /ADY/
017623  040     102     105             .ASCII  / BE/
017626  124     127     105             .ASCII  /TWE/
017631  105     116     040             .ASCII  /EN /
017634  123     105     103             .ASCII  /SEC/
017637  124     117     122             .ASCII  /TOR/
017642  123     000                     .ASCII  /S/<00>
017644  045     101     104     P.AJD:  .ASCII  /%AD/
017647  122     111     126             .ASCII  /RIV/
017652  105     040     104             .ASCII  /E D/
017655  105     124     105             .ASCII  /ETE/
017660  103     124     105             .ASCII  /CTE/
017663  104     040     105             .ASCII  /D E/
017666  122     122     117             .ASCII  /RRO/
017671  122     000     000             .ASCII  /R/<00><00>
017674  045     101     103     P.AJE:  .ASCII  /%AC/
017677  117     116     124             .ASCII  /ONT/
017702  122     117     114             .ASCII  /ROL/
017705  114     105     122             .ASCII  /LER/
017710  040     104     105             .ASCII  / DE/
017713  124     105     103             .ASCII  /TEC/
017716  124     105     104             .ASCII  /TED/
017721  040     120     125             .ASCII  / PU/
017724  114     123     105             .ASCII  /LSE/
017727  040     117     122             .ASCII  / OR/
017732  040     123     124             .ASCII  / ST/
017735  101     124     105             .ASCII  /ATE/
017740  040     120     101             .ASCII  / PA/
017743  122     111     124             .ASCII  /RIT/
017746  131     040     105             .ASCII  /Y E/
017751  122     122     117             .ASCII  /RRO/
017754  122     000                     .ASCII  /R/<00>
017756  045     101     103     P.AJG:  .ASCII  /%AC/
017761  117     116     124             .ASCII  /ONT/
017764  122     117     114             .ASCII  /ROL/
017767  114     105     122             .ASCII  /LER/
017772  040     124     111             .ASCII  / TI/
017775  115     105     117             .ASCII  /MEO/
020000  125     124     000             .ASCII  /UT/<00>
020003  000                             .ASCII  <00>
020004  045     101     105     P.AJH:  .ASCII  /%AE/
020007  116     126     105             .ASCII  /NVE
020012  114     117     120             .ASCII  /LCP
020015  105     057     120             .ASCII  /E/    >/P/
```

```
020020  101  103  113                .ASCII  /ACK/
020023  105  124  040                .ASCII  /ET /
020026  122  105  101                .ASCII  /REA/
020031  104  040  105                .ASCII  /D E/
020034  122  122  117                .ASCII  /RRO/
020037  122  040  050                .ASCII  /R (/
020042  120  141  162                .ASCII  /Par/
020045  151  164  171                .ASCII  /ity/
020050  040  157  162                .ASCII  / or/
020053  040  164  151                .ASCII  / ti/
020056  155  145  157                .ASCII  /meo/
020061  165  164  051                .ASCII  /ut)/
020064  000  000                     .ASCII  <00><00>
020066  045  101  105          P.AJI:  .ASCII  /%AE/
020071  116  126  105                .ASCII  /NVE/
020074  114  117  120                .ASCII  /LOP/
020077  105  057  120                .ASCII  /E/<57>/P/
020102  101  103  113                .ASCII  /ACK/
020105  105  124  040                .ASCII  /ET /
020110  127  122  111                .ASCII  /WRI/
020113  124  105  040                .ASCII  /TE /
020116  105  122  122                .ASCII  /ERR/
020121  117  122  040                .ASCII  /OR /
020124  050  120  141                .ASCII  /(Pa/
020127  162  151  164                .ASCII  /rit/
020132  171  040  157                .ASCII  /y o/
020135  162  040  164                .ASCII  /r t/
020140  151  155  145                .ASCII  /ime/
020143  157  165  164                .ASCII  /out/
020146  051  000                     .ASCII  /)/<00>
020150  045  101  103          P.AJJ:  .ASCII  /%AC/
020153  117  116  124                .ASCII  /ONT/
020156  122  117  114                .ASCII  /ROL/
020161  114  105  122                .ASCII  /LER/
020164  040  122  117                .ASCII  / RO/
020167  115  040  101                .ASCII  /M A/
020172  116  104  040                .ASCII  /ND /
020175  122  101  115                .ASCII  /RAM/
020200  040  120  101                .ASCII  / PA/
020203  122  111  124                .ASCII  /RIT/
020206  131  040  105                .ASCII  /Y E/
020211  122  122  117                .ASCII  /RRO/
020214  122  000                     .ASCII  /R/<00>
020216  045  101  103          P.AJK:  .ASCII  /%AC/
020221  117  116  124                .ASCII  /ONT/
020224  122  117  114                .ASCII  /ROL/
020227  114  105  122                .ASCII  /LER/
020232  040  122  101                .ASCII  / RA/
020235  115  040  120                .ASCII  /M P/
020240  101  122  111                .ASCII  /ARI/
020243  124  131  040                .ASCII  /TY /
020246  105  122  122                .ASCII  /ERR/
020251  117  122  000                .ASCII  /OR/<00>
```

```
020254  045     101     103     P.AJL:  .ASCII  /%AC/
020257  117     116     124             .ASCII  /ONT/
020262  122     117     114             .ASCII  /ROL/
020265  114     105     122             .ASCII  /LER/
020270  040     122     117             .ASCII  / RO/
020273  115     040     120             .ASCII  /M P/
020276  101     122     111             .ASCII  /ARI/
020301  124     131     040             .ASCII  /TY /
020304  105     122     122             .ASCII  /ERR/
020307  117     122     000             .ASCII  /OR/<00>
020312  045     101     122     P.AJM:  .ASCII  /%AR/
020315  111     116     107             .ASCII  /ING/
020320  040     122     105             .ASCII  / RE/
020323  101     104     040             .ASCII  /AD /
020326  105     122     122             .ASCII  /ERR/
020331  117     122     040             .ASCII  /OR /
020334  050     120     141             .ASCII  /(Pa/
020337  162     151     164             .ASCII  /rit/
020342  171     040     157             .ASCII  /y o/
020345  162     040     164             .ASCII  /r t/
020350  151     155     145             .ASCII  /ime/
020353  157     165     164             .ASCII  /out/
020356  051     000                     .ASCII  /)/<00>
020360  045     101     122     P.AJN:  .ASCII  /%AR/
020363  111     116     107             .ASCII  /ING/
020366  040     127     122             .ASCII  / WR/
020371  111     124     105             .ASCII  /ITE/
020374  040     105     122             .ASCII  / ER/
020377  122     117     122             .ASCII  /ROR/
020402  040     050     120             .ASCII  / (P/
020405  141     162     151             .ASCII  /ari/
020410  164     171     040             .ASCII  /ty /
020413  157     162     040             .ASCII  /or /
020416  164     151     155             .ASCII  /tim/
020421  145     157     165             .ASCII  /eou/
020424  164     051     000             .ASCII  /t)/<00>
020427  000                             .ASCII  <00>
020430  111     116     124     P.AJO:  .ASCII  /INT/
020433  105     122     122             .ASCII  /ERR/
020436  125     120     124             .ASCII  /UPT/
020441  040     115     101             .ASCII  / MA/
020444  123     124     105             .ASCII  /STE/
020447  122     040     106             .ASCII  /R F/
020452  101     111     114             .ASCII  /AIL/
020455  125     122     105             .ASCII  /URE/
020460  000     000                     .ASCII  <00><00>
020462  045     101     110     P.AJP:  .ASCII  /%AH/
020465  117     123     124             .ASCII  /OST/
020470  040     101     103             .ASCII  / AC/
020473  103     105     123             .ASCII  /CES/
020476  123     040     124             .ASCII  /S T/
020501  111     115     105             .ASCII  /IME/
020504  117     125     124             .ASCII  /OUT/
```

```
020507  040     050     110             .ASCII  / (H/
020512  151     147     150             .ASCII  /igh/
020515  145     162     040             .ASCII  /er /
020520  154     145     166             .ASCII  /lev/
020523  145     154     040             .ASCII  /el /
020526  160     162     157             .ASCII  /pro/
020531  164     157     143             .ASCII  /toc/
020534  157     154     040             .ASCII  /ol /
020537  144     145     160             .ASCII  /dep/
020542  145     156     144             .ASCII  /end/
020545  145     156     164             .ASCII  /ent/
020550  051     000                     .ASCII  /)/<00>
020552  045     101     103     P.AJQ:  .ASCII  /%AC/
020555  122     105     104             .ASCII  /RED/
020560  111     124     040             .ASCII  /IT /
020563  114     111     115             .ASCII  /LIM/
020566  111     124     040             .ASCII  /IT /
020571  105     130     103             .ASCII  /EXC/
020574  105     105     104             .ASCII  /EED/
020577  105     104     000             .ASCII  /ED/<00>
020602  045     101     121     P.AJR:  .ASCII  /%AQ/
020605  055     102     125             .ASCII  /-BU/
020610  123     040     115             .ASCII  /S M/
020613  101     123     124             .ASCII  /AST/
020616  105     122     040             .ASCII  /ER /
020621  105     122     122             .ASCII  /ERR/
020624  117     122     000             .ASCII  /OR/<00>
020627  000                             .ASCII  <00>
020630  045     101     103     P.AJS:  .ASCII  /%AC/
020633  117     116     124             .ASCII  /ONT/
020636  122     117     114             .ASCII  /ROL/
020641  114     105     122             .ASCII  /LER/
020644  040     106     101             .ASCII  / FA/
020647  124     101     114             .ASCII  /TAL/
020652  040     105     122             .ASCII  / ER/
020655  122     117     122             .ASCII  /ROR/
020660  000     000                     .ASCII  <00><00>
020662  045     101     111     P.AJT:  .ASCII  /%AI/
020665  116     123     124             .ASCII  /NST/
020670  122     125     103             .ASCII  /RUC/
020673  124     111     117             .ASCII  /TIO/
020676  116     040     114             .ASCII  /N L/
020701  117     117     120             .ASCII  /OOP/
020704  040     124     111             .ASCII  / TI/
020707  115     105     117             .ASCII  /MEO/
020712  125     124     000             .ASCII  /UT/<00>
020715  000                             .ASCII  <00>
020716  045     101     111     P.AJU:  .ASCII  /%AI/
020721  114     114     105             .ASCII  /LLE/
020724  107     101     114             .ASCII  /GAL/
020727  040     126     111             .ASCII  / VI/
020732  122     124     125             .ASCII  /RTU/
020735  101     114     040             .ASCII  /AL /
```

```
020740  103  111  122                .ASCII  /CIR/
020743  103  125  111                .ASCII  /CUI/
020746  124  040  111                .ASCII  /T I/
020751  101  000  000                .ASCII  /D/<00><00>
020754  045  101  111  P.AJV:        .ASCII  /%AI/
020757  116  124  105                .ASCII  /NTE/
020762  122  122  125                .ASCII  /RRU/
020765  120  124  040                .ASCII  /PT /
020770  126  105  103                .ASCII  /VEC/
020773  124  117  122                .ASCII  /TOR/
020776  040  111  114                .ASCII  / IL/
021001  114  105  107                .ASCII  /LEG/
021004  101  114  000                .ASCII  /AL/<00>
021007  000                         .ASCII  <00>
021010  045  101  115  P.AJW:        .ASCII  /%AM/
021013  101  111  116                .ASCII  /AIN/
021016  124  105  116                .ASCII  /TEN/
021021  101  116  103                .ASCII  /ANC/
021024  105  040  122                .ASCII  /E R/
021027  105  101  104                .ASCII  /EAD/
021032  057  127  122                .ASCII  <57>/WR/
021035  111  124  105                .ASCII  /ITE/
021040  040  111  116                .ASCII  / IN/
021043  126  101  114                .ASCII  /VAL/
021046  111  104  040                .ASCII  /ID /
021051  122  105  107                .ASCII  /REG/
021054  111  117  116                .ASCII  /ION/
021057  040  111  104                .ASCII  / ID/
021062  105  116  124                .ASCII  /ENT/
021065  111  106  111                .ASCII  /IFI/
021070  105  122  000                .ASCII  /ER/<00>
021073  000                         .ASCII  <00>
021074  045  101  115  P.AJX:        .ASCII  /%AM/
021077  101  111  116                .ASCII  /AIN/
021102  124  105  116                .ASCII  /TEN/
021105  101  116  103                .ASCII  /ANC/
021110  105  040  127                .ASCII  /E W/
021113  122  111  124                .ASCII  /RIT/
021116  105  040  114                .ASCII  /E L/
021121  117  101  104                .ASCII  /OAD/
021124  040  124  117                .ASCII  / TO/
021127  040  116  117                .ASCII  / NO/
021132  116  055  114                .ASCII  /N-L/
021135  117  101  104                .ASCII  /OAD/
021140  101  102  114                .ASCII  /ABL/
021143  105  040  103                .ASCII  /E C/
021146  117  116  124                .ASCII  /ONT/
021151  122  117  114                .ASCII  /ROL/
021154  114  105  122                .ASCII  /LER/
021157  000                         .ASCII  <00>
021160  045  101  103  P.AJY:        .ASCII  /%AC/
021163  117  116  124                .ASCII  /ONT/
021166  122  117  114                .ASCII  /ROL/
```

```
021171  114  105  122                .ASCII  /LER/
021174  040  122  101                .ASCII  / RA/
021177  115  040  105                .ASCII  /M E/
021202  122  122  117                .ASCII  /RRO/
021205  122  040  050                .ASCII  /R (/
021210  116  157  156                .ASCII  /Non/
021213  055  160  141                .ASCII  /-pa/
021216  162  151  164                .ASCII  /rit/
021221  171  051  000                .ASCII  /y)/<00>
021224  045  101  111     P.AJZ:     .ASCII  /%AI/
021227  116  111  124                .ASCII  /NIT/
021232  040  123  105                .ASCII  / SE/
021235  121  125  105                .ASCII  /QUE/
021240  116  103  105                .ASCII  /NCE/
021243  040  105  122                .ASCII  / ER/
021246  122  117  122                .ASCII  /ROR/
021251  000                         .ASCII  <00>
021252  045  101  110     P.AKA:     .ASCII  /%AH/
021255  111  107  110                .ASCII  /IGH/
021260  105  122  040                .ASCII  /ER /
021263  114  105  126                .ASCII  /LEV/
021266  105  114  040                .ASCII  /EL /
021271  120  122  117                .ASCII  /PRO/
021274  124  117  103                .ASCII  /TOC/
021277  117  114  040                .ASCII  /OL /
021302  111  116  103                .ASCII  /INC/
021305  117  115  120                .ASCII  /OMP/
021310  101  124  111                .ASCII  /ATI/
021313  102  111  114                .ASCII  /BIL/
021316  111  124  131                .ASCII  /ITY/
021321  040  105  122                .ASCII  / ER.
021324  122  117  122                .ASCII  /ROR/
021327  000                         .ASCII  <00>
021330  045  101  120     P.AKB:     .ASCII  /%AP/
021333  125  122  107                .ASCII  /URG/
021336  105  057  120                .ASCII  /E/<57>/P/
021341  117  114  114                .ASCII  /OLL/
021344  040  110  101                .ASCII  / HA/
021347  122  104  127                .ASCII  /RDW/
021352  101  122  105                .ASCII  /ARE/
021355  040  106  101                .ASCII  / FA/
021360  111  114  125                .ASCII  /ILU/
021363  122  105  000                .ASCII  /RE/<00>
021366  045  101  115     P.AKC:     .ASCII  /%AM/
021371  101  120  120                .ASCII  /APP/
021374  111  116  107                .ASCII  /ING/
021377  040  122  105                .ASCII  / RE/
021402  107  111  123                .ASCII  /GIS/
021405  124  105  122                .ASCII  /TER/
021410  040  122  105                .ASCII  / RE/
021413  101  104  040                .ASCII  /AD /
021416  106  101  111                .ASCII  /FAI/
021421  114  125  122                .ASCII  /LUR/
```

```
021424      105      040      050                .ASCII  /E (/
021427      120      141      162                .ASCII  /Par/
021432      151      164      171                .ASCII  /ity/
021435      040      157      162                .ASCII  / or/
021440      040      164      151                .ASCII  / ti/
021443      155      145      157                .ASCII  /meo/
021446      165      164      051                .ASCII  /ut)/
021451      000                                  .ASCII  <00>
021452   017756'                        P.AJF:   .WORD   P.AJG
021454   020004'                                 .WORD   P.AJH
021456   020066'                                 .WORD   P.AJI
021460   020150'                                 .WORD   P.AJJ
021462   020216'                                 .WORD   P.AJK
021464   020254'                                 .WORD   P.AJL
021466   020312'                                 .WORD   P.AJM
021470   020360'                                 .WORD   P.AJN
021472   020430'                                 .WORD   P.AJO
021474   020462'                                 .WORD   P.AJP
021476   020552'                                 .WORD   P.AJQ
021500   020602'                                 .WORD   P.AJR
021502   020630'                                 .WORD   P.AJS
021504   020662'                                 .WORD   P.AJT
021506   020716'                                 .WORD   P.AJU
021510   020754'                                 .WORD   P.AJV
021512   021010'                                 .WORD   P.AJW
021514   021074'                                 .WORD   P.AJX
021516   021160'                                 .WORD   P.AJY
021520   021224'                                 .WORD   P.AJZ
021522   021252'                                 .WORD   P.AKA
021524   021330'                                 .WORD   P.AKB
021526   021366'                                 .WORD   P.AKC
021530      045      101      124        P.AKE:   .ASCII  /%AT/
021533      061      061      040                .ASCII  /11 /
021536      103      120      125                .ASCII  /CPU/
021541      040      106      101                .ASCII  / FA/
021544      111      114      125                .ASCII  /ILU/
021547      122      105      000                .ASCII  /RE/<00>
021552      045      101      116        P.AKF:   .ASCII  /%AN/
021555      117      116      055                .ASCII  /ON-/
021560      120      101      122                .ASCII  /PAR/
021563      111      124      131                .ASCII  /ITY/
021566      040      122      101                .ASCII  / RA/
021571      115      040      105                .ASCII  /M E/
021574      122      122      117                .ASCII  /RRO/
021577      122      000      000                .ASCII  /R/<00><00>
021602      045      101      123        P.AKG:   .ASCII  /%AS/
021605      124      101      124                .ASCII  /TAT/
021610      105      040      115                .ASCII  /E M/
021613      101      103      110                .ASCII  /ACH/
021616      111      116      105                .ASCII  /INE/
021621      040      106      101                .ASCII  / FA/
021624      111      114      125                .ASCII  /ILU/
021627      122      105      040                .ASCII  /RE /
```

```
021632  055  040  124                .ASCII  /- T/
021635  061  061  040                .ASCII  /11 /
021640  101  104  104                .ASCII  /ADD/
021643  122  105  123                .ASCII  /RES/
021646  123  040  122                .ASCII  /S R/
021651  105  107  111                .ASCII  /EGI/
021654  123  124  105                .ASCII  /STE/
021657  122  000  000                .ASCII  /R/<00><00>
021662  045  101  123  P.AKH:        .ASCII  /%AS/
021665  124  101  124                .ASCII  /TAT/
021670  105  040  115                .ASCII  /E M/
021673  101  103  110                .ASCII  /ACH/
021676  111  116  105                .ASCII  /INE/
021701  040  106  101                .ASCII  / FA/
021704  111  114  125                .ASCII  /ILU/
021707  122  105  040                .ASCII  /RE /
021712  055  040  121                .ASCII  /- Q/
021715  055  102  125                .ASCII  /-BU/
021720  123  040  101                .ASCII  /S A/
021723  104  104  122                .ASCII  /DDR/
021726  105  123  123                .ASCII  /ESS/
021731  040  122  105                .ASCII  / RE/
021734  107  111  123                .ASCII  /GIS/
021737  124  105  122                .ASCII  /TER/
021742  000  000                     .ASCII  <00><00>
021744  045  101  123  P.AKI:        .ASCII  /%AS/
021747  124  101  124                .ASCII  /TAT/
021752  105  040  115                .ASCII  /E M/
021755  101  103  110                .ASCII  /ACH/
021760  111  116  105                .ASCII  /INE/
021763  040  106  101                .ASCII  / FA/
021766  111  114  125                .ASCII  /ILU/
021771  122  105  040                .ASCII  /RE /
021774  055  040  103                .ASCII  /- C/
021777  122  103  040                .ASCII  /RC /
022002  122  105  107                .ASCII  /REG/
022005  111  123  124                .ASCII  /IST/
022010  105  122  000                .ASCII  /ER/<00>
022013  000                         .ASCII  <00>
022014  045  101  123  P.AKJ:        .ASCII  /%AS/
022017  124  101  124                .ASCII  /TAT/
022022  105  040  115                .ASCII  /E M/
022025  101  103  110                .ASCII  /ACH/
022030  111  116  105                .ASCII  /INE/
022033  040  106  101                .ASCII  / FA/
022036  111  114  125                .ASCII  /ILU/
022041  122  105  040                .ASCII  /RE /
022044  055  040  123                .ASCII  /- S/
022047  105  122  111                .ASCII  /ERI/
022052  101  114  111                .ASCII  /ALI/
022055  132  105  122                .ASCII  /ZER/
022060  057  104  105                .ASCII  <57>/DE/
022063  123  105  122                .ASCII  /SER/
```

```
022066      111     101     114             .ASCII  /IAL/
022071      111     132     105             .ASCII  /IZE/
022074      122     040     122             .ASCII  /R R/
022077      105     107     111             .ASCII  /EGI/
022102      123     124     105             .ASCII  /STE/
022105      122     000     000             .ASCII  /R/<00><00>
022110      045     101     123  P.AKK:     .ASCII  /%AS/
022113      124     101     124             .ASCII  /TAT/
022116      105     040     115             .ASCII  /E M/
022121      101     103     110             .ASCII  /ACH/
022124      111     116     105             .ASCII  /INE/
022127      040     106     101             .ASCII  / FA/
022132      111     114     125             .ASCII  /ILU/
022135      122     105     040             .ASCII  /RE /
022140      055     040     127             .ASCII  /- W/
022143      122     117     116             .ASCII  /RON/
022146      107     040     110             .ASCII  /G H/
022151      101     122     104             .ASCII  /ARD/
022154      127     101     122             .ASCII  /WAR/
022157      105     040     126             .ASCII  /E V/
022162      105     122     123             .ASCII  /ERS/
022165      111     117     116             .ASCII  /ION/
022170      000     000                     .ASCII  <00><00>
022172  021530'                 P.AKD:      .WORD   P.AKE
022174  021552'                             .WORD   P.AKF
022176  021602'                             .WORD   P.AKG
022200  021662'                             .WORD   P.AKH
022202  021744'                             .WORD   P.AKI
022204  022014'                             .WORD   P.AKJ
022206  022110'                             .WORD   P.AKK
022210      045     116     045  P.AKL:     .ASCII  /%N%/
022213      101     132     122             .ASCII  /AZR/
022216      121     101     040             .ASCII  /QA /
022221      104     105     126             .ASCII  /DEV/
022224      040     106     124             .ASCII  / FT/
022227      114     040     040             .ASCII  /L  /
022232      045     132     065             .ASCII  /%Z5/
022235      045     101     040             .ASCII  /%A /
022240      117     116     040             .ASCII  /ON /
022243      125     116     111             .ASCII  /UNI/
022246      124     040     045             .ASCII  /T %/
022251      132     062     045             .ASCII  /Z2%/
022254      101     040     124             .ASCII  /A T/
022257      123     124     040             .ASCII  /ST /
022262      060     060     061             .ASCII  /001/
022265      040     123     125             .ASCII  / SU/
022270      102     040     060             .ASCII  /B 0/
022273      060     060     040             .ASCII  /00 /
022276      120     103     072             .ASCII  /PC:/
022301      040     045     117             .ASCII  / %O/
022304      066     000                     .ASCII  /6/<00>
022306      045     116     045  P.AKM:     .ASCII  /%N%/
022311      101     132     122             .ASCII  /AZR/
```

```
022314  121  101  040            .ASCII  /QA /
022317  110  122  104            .ASCII  /HRD/
022322  040  105  122            .ASCII  / ER/
022325  122  040  040            .ASCII  /R  /
022330  045  132  065            .ASCII  /%Z5/
022333  045  101  040            .ASCII  /%A /
022336  117  116  040            .ASCII  /ON /
022341  125  116  111            .ASCII  /UNI/
022344  124  040  045            .ASCII  /T %/
022347  132  062  045            .ASCII  /Z2%/
022352  101  040  124            .ASCII  /A T/
022355  123  124  040            .ASCII  /ST /
022360  060  060  061            .ASCII  /001/
022363  040  123  125            .ASCII  / SU/
022366  102  040  060            .ASCII  /B 0/
022371  060  060  040            .ASCII  /00 /
022374  120  103  072            .ASCII  /PC:/
022377  040  045  117            .ASCII  / %O/
022402  066  000                  .ASCII  /6/<00>
022404  045  116  045  P.AKN:    .ASCII  /%N%/
022407  101  132  122            .ASCII  /AZR/
022412  121  101  040            .ASCII  /QA /
022415  123  106  124            .ASCII  /SFT/
022420  040  105  122            .ASCII  / ER/
022423  122  040  040            .ASCII  /R  /
022426  045  132  065            .ASCII  /%Z5/
022431  045  101  040            .ASCII  /%A /
022434  117  116  040            .ASCII  /ON /
022437  125  116  111            .ASCII  /UNI/
022442  124  040  045            .ASCII  /T %/
022445  132  062  045            .ASCII  /Z2%/
022450  101  040  124            .ASCII  /A T/
022453  123  124  040            .ASCII  /ST /
022456  060  060  061            .ASCII  /001/
022461  040  123  125            .ASCII  / SU/
022464  102  040  060            .ASCII  /B 0/
022467  060  060  040            .ASCII  /00 /
022472  120  103  072            .ASCII  /PC:/
022475  040  045  117            .ASCII  / %O/
022500  066  045  116            .ASCII  /6%N/
022503  000                      .ASCII  <00>
022504  045  116  045  P.AKO:    .ASCII  /%N%/
022507  101  111  057            .ASCII  /AI/<57>
022512  117  040  122            .ASCII  /O R/
022515  105  121  125            .ASCII  /EQU/
022520  105  123  124            .ASCII  /EST/
022523  040  106  101            .ASCII  / FA/
022526  111  114  105            .ASCII  /ILE/
022531  104  045  116            .ASCII  /D%N/
022534  000  000                  .ASCII  <00><00>
022536  045  123  064  P.AKP:    .ASCII  /%S4/
022541  000                      .ASCII  <00>
022542  045  116  000  P.AKQ:    .ASCII  /%N/<00>
```

```
022545     000
022546     045      101      040          .ASCII  <00>
022551     055      040      000  P.AKR:  .ASCII  /%A /
022554     045      101      052          .ASCII  /- /<00>
022557     040      000      000  P.AKS:  .ASCII  /%A*/
022562  000000C                           .ASCII  / /<00><00>
                                 L$HWLEN::
                                         .WORD   <<L$NDHW-L$HWLEN>/2>
022564  172150                   HWPT.IP.ADDR::
                                         .WORD   -5630
022566  000154                   HWPT.VECTOR::
                                         .WORD   154
022570  000004                   HWPT.BR.LEVEL::
                                         .WORD   4
022572  000200                   HWPT.DISK::
                                         .WORD   200
022574  000000                   HWPTS0.LBN::
                                         .WORD   0
022576  000000                   HWPTS1.LBN::
                                         .WORD   0
022600  177777                   HWPTE0.LBN::
                                         .WORD   -1
022602  000000                   HWPTE1.LBN::
                                         .WORD   0
022604  020040                   NAME.LO::
                                         .WORD   20040
022606  020040                   NAME.HI::
                                         .WORD   20040
022610                           L$NDHW::.BLKW   1
022612  000000C                  L$SWLEN::
                                         .WORD   <<L$NDSW-L$SWLEN>/2>
022614  000040                   SWP.ERROR::
                                         .WORD   40
022616  000000                   SWP.XFER::
                                         .WORD   0
022620  054046                   SWP.FLAGS::
                                         .WORD   54046
022622  000000                   SWP.DPAT::
                                         .WORD   0
022624  000143                   SWP.RAT::
                                         .WORD   143
022626  000000                   SWP.TIME::
                                         .WORD   0
022630  000013                   DUPROUND::
                                         .WORD   13
022632  000020                   SWP.UCNT::
                                         .WORD   20
022634                           SWP.UDPAT::
                                         .BLKW   20
022674                           L$NDSW::.BLKW   1
022676  000000                   L$PROT::.WORD   0
022700  177777                           .WORD   -1
022702  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                  CLK.PRESENT::
                        .BLKB   1
005771                  HOE.FLAG::
                        .BLKB   1
005772                  S.PATTERN::
                        .BLKW   1
005774                  S.DUPPKT::
                        .BLKW   1
005776                  P.INDEX::
                        .BLKW   1
006000  000000          RD.COUNT::
                        .WORD   0
```

```
006002                          BRLEVEL::
                                        .BLKW   1
006004                          D.FAIL::.BLKB   1
006005                          FORCED.ERROR::
                                        .BLKB   1
006006                          FER.LBN::
                                        .BLKW   1
006010                          FER.BC::.BLKW   1
006012                          INIT.OCCURED::
006012  000                             .BYTE   0
006013                          ADDR.VECT.OK::
006013  000                             .BYTE   0


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


100000                          BIT15==         -100000
040000                          BIT14==         40000
020000                          BIT13==         20000
010000                          BIT12==         10000
004000                          BIT11==         4000
002000                          BIT10==         2000
001000                          BIT09==         1000
000400                          BIT08==         400
000200                          BIT07==         200
000100                          BIT06==         100
000040                          BIT05==         40
000020                          BIT04==         20
000010                          BIT03==         10
000004                          BIT02==         4
000002                          BIT01==         2
000001                          BIT00==         1
001000                          BIT9==          1000
000400                          BIT8==          400
000200                          BIT7==          200
000100                          BIT6==          100
000040                          BIT5==          40
000020                          BIT4==          20
000010                          BIT3==          10
000004                          BIT2==          4
000002                          BIT1==          2
000001                          BIT0==          1
000040                          EF.START==      40
000037                          EF.RESTART==    37
000036                          EF.CONTINUE==   36
000035                          EF.NEW==        35
000034                          EF.PWR==        34
000340                          PRI07==         340
000300                          PRI06==         300
000240                          PRI05==         240
```

```
000200                  PRI04==         200
000140                  PRI03==         140
000100                  PRI02==         100
000040                  PRI01==         40
000000                  PRI00==         0
000004                  EVL==           4
000010                  LOT==           10
000020                  ADR==           20
000040                  IDU==           40
000100                  ISR==           100
000200                  UAM==           200
000400                  BOE==           400
001000                  PNY==           1000
002000                  PRI==           2000
004000                  IXE==           4000
010000                  IBE==           10000
020000                  IER==           20000
040000                  LOE==           40000
100000                  HOE==           -100000
000126'                 L$ERRTBL==      ERRTYP
022614'                 L$SW==          L$SWLEN*2
022564'                 L$HW==          L$HWLEN*2
000011'                 L$DEPO==        L$REV*1
000136'                 HWQ1==          P.AAA
000152'                 HWQ2==          P.AAB
000162'                 HWQ3==          P.AAC
000226'                 HWQ4==          P.AAD
000244'                 HWQ5==          P.AAE
000314'                 HWQ6A==         P.AAF
000366'                 HWQ6B==         P.AAG
000442'                 HWQ7A==         P.AAH
000512'                 HWQ7B==         P.AAI
000562'                 HWQ8==          P.AAJ
000640'                 HWQ9==          P.AAK
000740'                 HWQ10==         P.AAL
000770'                 HWQ11==         P.AAM
001022'                 SWQ1==          P.AAN
001044'                 SWQ2==          P.AAO
001124'                 SWQ4==          P.AAP
001146'                 SWQ7==          P.AAQ
001220'                 SWQ9==          P.AAR
001274'                 SWQ10==         P.AAS
001340'                 SWQ11==         P.AAT
001372'                 SWQ12==         P.AAU
001470'                 SWQ13==         P.AAV
001546'                 SWQ14==         P.AAW
001620'                 SWQ15==         P.AAX
001670'                 SWQ17==         P.AAY
001766'                 SWQ19==         P.AAZ
002056'                 SWQ20==         P.ABA
002144'                 SWQ21==         P.ABB
002230'                 SWQ22==         P.ABC
002270'                 SWQ23==         P.ABD
```

```
002342          SWQ24**         P.ABE
002406          SWQ25**         P.ABF
002460'         SWQ26**         P.ABG
002522'         SWM1**          P.ABH
002620'         NULL**          P.ABI
002622'         DBM5**          P.ABJ
002650'         DBM12**         P.ABK
002724'         DBM15**         P.ABL
002754'         DBM18**         P.ABM
003026'         DBM19**         P.ABN
003112'         DBM20**         P.ABO
003166'         DBM21**         P.ABP
003250'         DBM22**         P.ABQ
003314          DBM23**         P.ABR
003352          DBM25**         P.ABS
003420'         DBM26**         P.ABT
003452'         DBM27**         P.ABU
003524'         DBM28A**        P.ABV
003564'         DBM28B**        P.ABW
003624'         DBM29**         P.ABX
003672'         DBM32**         P.ABY
003746'         DBM101**        P.ABZ
003774'         DBM104**        P.ACA
004036'         DBM105**        P.ACB
004074'         DBM107**        P.ACC
004132'         DBM108**        P.ACD
004222'         DBM109**        P.ACE
004302'         DBM111**        P.ACF
004402'         DBM112**        P.ACG
004504'         DBM120**        P.ACH
004576'         DBM121**        P.ACI
004666'         DU.MSG**        P.ACJ
005406'         DU.RSN**        P.ACK
005434'         MSG.01**        P.ACW
005466'         MSG.02**        P.ACX
005522'         MSG.03**        P.ACY
005554          RPT1**          P.ACZ
005640'         RPT2**          P.ADA
005704'         RPT3**          P.ADB
005770'         RPT4**          P.ADC
006034'         RPT5**          P.ADD
006122'         RPT6**          P.ADE
006166'         RPT7**          P.ADF
006204'         RPT8**          P.ADG
006232'         RPT9**          P.ADH
006264'         RPT10**         P.ADI
006352'         RPT11**         P.ADJ
006420'         RPT12**         P.ADK
006466'         RPT13**         P.ADL
006566'         RPT14**         P.ADM
006664'         RPT15**         P.ADN
006764'         RPT16**         P.ADO
007046'         EGS.01**        P.ADP
```

```
007066'  EGS.02**   P.ADQ
007160'  EGD.10**   P.ADR
007220   EGD.11**   P.ADS
007244'  EGD.12**   P.ADT
007272'  EGD.13**   P.ADU
007320'  EGD.14**   P.ADV
007350'  EGD.15**   P.ADW
007366'  EGD.16**   P.ADX
007416'  EGD.17**   P.ADY
007434'  EGD.18**   P.ADZ
007454'  EGD.19**   P.AEA
007514'  EGD.20**   P.AEB
007602'  EGD.21**   P.AEC
007714'  EGD.22**   P.AED
007754'  EGD.23**   P.AEE
010016'  EGD.24**   P.AEF
010062'  EGH.30**   P.AEG
010106'  EBS.01**   P.AEH
010150'  EBD.10**   P.AEI
010210'  EBD.12**   P.AEJ
010256'  EBD.13**   P.AEK
010310'  FBD.14**   P.AEL
010350'  EBD.18**   P.AEM
010404'  EBD.19**   P.AEN
010464'  EBD.24**   P.AEO
010540'  EH.0**     P.AEP
010576'  EH.1**     P.AEQ
010634'  EH.2**     P.AER
010674'  EH.3**     P.AES
010732'  EH.4**     P.AET
010756'  EH.5**     P.AEU
011006'  EH.6**     P.AEV
011036'  EH.7**     P.AEW
011066'  EH.8**     P.AEX
011122'  EH.9**     P.AEY
011152'  EH.10**    P.AEZ
011202'  EH.12**    P.AFA
011236'  EH.13**    P.AFB
011310'  ERR.OO**   P.AFC
012000'  ERR.COD**  P.AFD
012034'  ELG.OO**   P.AFS
012366'  ELG.FMT**  P.AFT
012400'  EX.SA**    P.AFZ
012416'  EX.SC**    P.AGA
012446'  EX.SBO**   P.AGB
012452'  EX.SB**    P.AGC
012474'  EX.CMD**   P.AGD
012514'  EX.RD**    P.AGE
012524'  EX.WRT**   P.AGF
012534'  EX.CMP**   P.AGG
012550'  EX.ONL**   P.AGH
012562'  EX.ACC**   P.AGI
012574'  EX.OP**    P.AGJ
```

```
012600          EX.BB--         P.AGK
012670          EX.BB1-         P.AGL
012764          EX.BBU-         P.AGM
013054          EX.LBN-         P.AGN
013112          EX.PBN-         P.AGO
013150          EX.LBR-         P.AGP
013216          EX.LBW-         P.AGQ
013264          EX.RBN-         P.AGR
013344          EX.CBC-         P.AGS
013410          EX.CBR-         P.AGT
013462          EX.CBW-         P.AGU
013534          EX.BC--         P.AGV
013610          EX.BD--         P.AGW
013666          EX.BDR-         P.AGX
013756          EX.BDW-         P.AGY
014046          EX.RP--         P.AGZ
014134          EX.WRD-         P.AHA
014144          EX.TIM-         P.AHB
014204          XX13--          P.AHC
014226          XX23--          P.AHD
014262          XX32--          P.AHE
014310          XX33--          P.AHF
014346          XX34--          P.AHG
014416          CER.01-         P.AHH
014462          CER.02-         P.AHI
014536          SC.SDI-         P.AHJ
014562          SC.CON-         P.AHK
014604          SC.DUP-         P.AHL
014634          SC.ONL-         P.AHM
014656          SC.SON-         P.AHN
014676          SC.UNK-         P.AHO
014756          SC.VOL-         P.AHP
015036          SC.IOP-         P.AHQ
015110          SC.DIS-         P.AHR
015202          SC.FER-         P.AHS
015270          SC.FE2-         P.AHT
015360          SC.ISM-         P.AHU
015440          SC.IS2-         P.AHV
015520          SC.DST-         P.AHW
015574          SC.DS2-         P.AHX
015646          SC.ECC-         P.AHY
015730          SC.ECD-         P.AHZ
015762          SC.RCT-         P.AIA
016002          SC.FUL-         P.AIB
016056          SC.576-         P.AIC
016132          SC.FCT-         P.AID
016200          SC.EC1-         P.AIE
016230          SC.EC2-         P.AIF
016260          SC.EC3-         P.AIG
016312          SC.EC4-         P.AIH
016342          SC.EC5-         P.AII
016372          SC.EC6-         P.AIJ
016422          SC.EC7-         P.AIK
```

```
        016454          SC.EC8--        P.AIL
        016506          SC.EC9--        P.AIM
        016550          SC.SWP--        P.AIN
        016610          SC.MWP--        P.AIO
        016650          SC.ODA--        P.AIP
        016700          SC.ODB--        P.AIQ
        016722          SC.NXM--        P.AIR
        016756          SC.PAR--        P.AIS
        017012          SC.CTO--        P.AIT
        017064          SC.SDS--        P.AIU
        017142          SC.EDC--        P.AIV
        017202          SC.IDS--        P.AIW
        017252          SC.SRT--        P.AIX
        017344          SC.SRI--        P.AIY
        017432          SC.POE--        P.AIZ
        017466          SC.RDY--        P.AJA
        017550          SC.CLK--        P.AJB
        017576          SC.RSP--        P.AJC
        017644          SC.SUR--        P.AJD
        017674          SC.PSP--        P.AJE
        021452          CNTR.ERR--      P.AJF
        022172          RDRX.ERR--      P.AKD
        022210          DF.MSG--        P.AKL
        022306          HRD.MSG--       P.AKM
        022404          SFT.MSG--       P.AKN
        022504          HRD.SUB--       P.AKO
        022536          SPACE4--        P.AKP
        022542          CRLF--          P.AKQ
        022546          DASH--          P.AKR
        022554          ASTERISK--      P.AKS
        022564          DFPTBL--        L$HWLEN*2
        022614          SFPTBL--        L$SWLEN*2


;                               PSECT SUMMARY
;
;       Psect Name              Words     Attributes
;        $CODE$                  4834       RO  ,  I  ,  LCL,  REL,  CON
;        $FFF$                   1542       RW  ,  D  ,  GBL,  REL,  CON


;                           Library Statistics
;
;                                    -------- Symbols ----  -        Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQAF0.L16;11    406       181        44         21          00:00.1
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAF0.BL1/LIST=ZRQAF0.LS1/OBJECT=ZRQAF0.OB1/SOURCE=PAGE:53
```

```
;    3331  0     module ZRQAM2 (
;    3332  0
;    3333  0     %title 'RD/RX EXERCISER'
;    3334  0                  ident = 'V01.9',
;    3335  0                  addressing_mode (absolute),
;    3336  0                  environment (noeis)
;    3337  0                  ) =
;    3338  0
;    3339  1     begin
;    3340  1
;    3341  1     %sbttl 'DECLARATIONS'
;    3342  1
;    3343  1     library 'ZRQAF0.L16 ;                          ! RDRX EXERCISER GLOBAL LIBRARY
;    3344  1
;    3345  1     require 'BLSMAC.REQ';                          ! DIAGNOSTIC SUPERVISOR LIBRARY
;    4836  1
;    4837  1     forward routine
;    4838  1         NEX_TRAP : L%ISR novalue,
;    4839  1         EMS_01 : novalue,
;    4840  1         EMS_TIM : novalue,
;    4841  1         EMS_DBN : NOVALUE,                                          !ZZZ
;    4842  1         EMS_BLK : NOVALUE,                                          !ZZZ
;    4843  1         SET_CPAR : novalue,
;    4844  1         SET_UPAR : novalue;
;    4845  1
;    4846  1     external
;    4847  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    4848  1                           ! RUN-TIME CONTROLLER STATUS TABLES
;    4849  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    4850  1                           ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    4851  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    4852  1                           ! DRIVER CONTROLLER TABLES
;    4853  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    4854  1                           ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    4855  1         RDRX_ADDR : ref rdrx field (RC_REG),
;    4856  1                           ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    4857  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;    4858  1                           ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    4859  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                     !ZZZ
;    4860  1                           !CONTAINS LBNS (HI + LO FIELDS) FOR SEQUENTIAL  !ZZZ
;    4861  1                           !I/O TRANSFER FOR EACH UNIT.                    !ZZZ
;    4862  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    4863  1                           ! STATISTICS TABLES
;    4864  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    4865  1                           ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    4866  1         DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;    4867  1                           !INFO FROM RECEIVE AND SEND CMDS                 ZZZ
;    4868  1         TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TRACK DIRECTION ZZZ
;    4869  1         RDM_CNT : WORD,                          !NO OF RANDOM NOS   \\KEEP      ZZZ
;    4870  1         RANDOM : VECTOR [RDM_LEN, WORD],         !RANDOM NO TABLE    //TOGETHER  ZZZ
;    4871  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    4872  1                           ! STATISTICS TABLE FOR CONTROLLER ERRORS
;    4873  1         MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
```

```
;  4874  1                          ! MSCP PACKET POOL
;  4875  1      IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  4876  1                          ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;  4877  1      PKT_USE : vector [PKT_CNT, byte, signed],
;  4878  1                          ! MSCP PACKET POOL ALLOCATION TABLE
;  4879  1      RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;  4880  1                          ! RETURN PACKET POOL
;  4881  1      RP_USE : vector [RP_CNT, byte, signed],
;  4882  1                          ! RETURN PACKET POOL ALLOCATION TABLE
;  4883  1      RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
;  4884  1      RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
;  4885  1                          ! CURRENT RETURN PACKET ADDRESS
;  4886  1      ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;  4887  1                          ! ERROR-LOG PACKET SAVE AREA
;  4888  1      BUFF_ADDR : vector [MAX_BUF_CNT],                  ! TABLE OF I/O BUFFER DESCRIPTORS
;  4889  1      BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],     ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;  4890  1      IODQ : vector [IODQ_LEN, byte],                    ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;  4891  1      IODQ_IN : word,                                    ! I/O DONE QUEUE IN POINTER
;  4892  1      IODQ_OUT : word,                                   ! I/O DONE QUEUE OUT POINTER
;  4893  1      ENTRY_REASON : byte,                               ! CURRENT OPERATOR COMMAND
;  4894  1      EOP_FLAG : byte,                                   ! END-OF-PASS FLAG
;  4895  1      DUP_FLAGS : WORD,                                  !DUP FLAGS       ZZZ
;  4896  1      CCTLR : word,                                      ! NUMBER OF "CURRENT" CONTROLLER
;  4897  1      CDISK : word,                                      ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  4898  1      CUOFF : word,                                      ! CURRENT UNIT CST OFFSET
;  4899  1      CTLR_CNT : word,                                   ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;  4900  1      DUR : vector [MAX_UNITS, byte],                    ! DROP UNIT REASON
;  4901  1      QIO : vector [MAX_CTLR, byte],                     ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;  4902  1      FREE_MEM_ADDR,                                     ! START OF FREE MEMORY
;  4903  1      BYTS_PER_QIO : word,                               ! SIZE (BYTES) OF AN I/O BUFFER
;  4904  1      ST_CODE : word,                                    ! CURRENT STATUS CODE
;  4905  1      SB_CODE : word,                                    ! CURRENT SUB-CODE
;  4906  1      STEP : word,                                       ! CURRENT STEP IN HARD_INIT
;  4907  1      OF_RC : signed word,                               ! OFFSET (0 OR 2) TO READ IP OR SA
;  4908  1      SA_REG : word,                                     ! STORAGE FOR SA REGISTER READS AND WRITES
;  4909  1      CMD_TIME : word,                                   ! COMMAND TIMEOUT VALUE (IN SECONDS)
;  4910  1      NEX : word,                                        ! NON-EXISTENT MEMORY TRAP INDICATOR
;  4911  1      CRN_LOW : word,                                    ! COMMAND REF NUMBER OF LAST COMMAND SENT
;  4912  1      CRN_HIGH : word,                                   ! COMMAND REF NUMBER (HI ORDER)
;  4913  1      TEMP1 : WORD,                                      !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;  4914  1      TEMP2 : WORD,                                      !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;  4915  1      CREDIT_BAL : word,                                 ! CREDIT BALANCE
;  4916  1      NEXT_PKT_USE : byte,                               ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;  4917  1      HOURS: byte,                                       ! TIME OF DAY (HOURS)
;  4918  1      MINUTES : byte,                                    ! TIME OF DAY (MINUTES)
;  4919  1      CLK_TICKS : word,                                  ! TIME OF DAY (LINE-CLOCK TICKS)
;  4920  1      CLK_PRESENT : byte,                                ! FLAG INDICATES IF LINE-CLOCK PRESENT
;  4921  1      HOE_FLAG : byte,                                   ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;  4922  1      FORCED_ERROR : byte,                               ! "FORCED ERROR" DETECTED IN LAST READ
;  4923  1      FER_LBN : word,                                    ! LBN OF THE "FORCED ERROR" BLOCK
;  4924  1      FER_BC : word,                                     ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;  4925  1      INIT_OCCURED : byte,                               ! EXERCISER INITIALIZATION COMPLETE
;  4926  1      ADDR_VECT_OK : byte,                               ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
```

```
;  4927  1          DBM5,
;  4928  1          P_INDEX : SIGNED WORD,          !CURRENT MESSAGE PACKET INDEX      ZZZ
;  4929  1          S_PATTERN : WORD,               !PATTERN FOR DUP WRITES            ZZZ
;  4930  1          S_DUPPKT : WORD,                !DBN BYTE COUNTER                  ZZZ
;  4931  1          RD_COUNT : WORD,                ! NUMBER OF WINCHESTER UNITS       ZZZ
;  4932  1          BRLEVEL : WORD,                 !BUS REQUEST PRIORITY LEVEL        ZZZ
;  4933  1          D_FAIL : BYTE,                  !SIGNIFIES DUP TYPE ERROR          ZZZ
;  4934  1          DBM107,
;  4935  1          DU_MSG,
;  4936  1          DU_RSN : vector [11],
;  4937  1          RPT1,
;  4938  1          RPT2,
;  4939  1          RPT3,
;  4940  1          RPT4,
;  4941  1          RPT5,
;  4942  1          RPT6,
;  4943  1          RPT7,
;  4944  1          RPT8,
;  4945  1          RPT9,
;  4946  1          RPT10,
;  4947  1          RPT11,
;  4948  1          RPT12,
;  4949  1          RPT13,
;  4950  1          RPT14,
;  4951  1          RPT15,
;  4952  1          RPT16,
;  4953  1      !ZZZ    RPT17,
;  4954  1      !ZZZ    RPT18,
;  4955  1      !ZZZ    RPT19,
;  4956  1
;  4957  1          MSG_01,
;  4958  1          EGS_01,
;  4959  1          EBS_01,
;  4960  1          EBD_10,
;  4961  1          EBD_12,
;  4962  1          EBD_13,
;  4963  1          EBD_14,
;  4964  1          EBD_18,
;  4965  1          EBD_19,
;  4966  1          EBD_24,
;  4967  1          ERR_00,
;  4968  1          ERR_C00 : vector [14],
;  4969  1          ELG_00,
;  4970  1          ELG_FMT : vector [5],
;  4971  1          EX_TIM,
;  4972  1          XX13,                                                          !ZZZ
;  4973  1          XX23,                                                          !ZZZ
;  4974  1          XX32,                                                          !ZZZ
;  4975  1          XX33,                                                          !ZZZ
;  4976  1          XX34,                                                          !ZZZ
;  4977  1          EX_SA,
;  4978  1          EX_SC,
;  4979  1          EX_SBO,
```

```
;    4980  1          EX_SB.
;    4981  1          EX_RP.
;    4982  1          EX_WRD.
;    4983  1          EX_CMD.
;    4984  1          EX_RD.
;    4985  1          EX_WRT.
;    4986  1          EX_CMP.
;    4987  1          EX_ONL.
;    4988  1          EX_ACC.
;    4989  1          EX_OP.
;    4990  1          EX_BB.
;    4991  1          EX_BB1.
;    4992  1          EX_BBU.
;    4993  1          EX_LBN.
;    4994  1          EX_PBN.
;    4995  1          EX_LBR.
;    4996  1          EX_LBW.
;    4997  1          EX_RBN.
;    4998  1          EX_CBC.
;    4999  1          EX_CBR.
;    5000  1          EX_CBW.
;    5001  1          EX_BC.
;    5002  1          EX_BD.
;    5003  1          EX_BDR.
;    5004  1          EX_BDW.
;    5005  1          SC_SDI.
;    5006  1          SC_CON.
;    5007  1          SC_DUP.
;    5008  1          SC_ONL.
;    5009  1          SC_SDN.
;    5010  1          SC_UNK.
;    5011  1          SC_VOL.
;    5012  1          SC_IOP.
;    5013  1          SC_DIS.
;    5014  1          SC_FER.
;    5015  1          SC_FE2.
;    5016  1          SC_ISH.
;    5017  1          SC_IS2.
;    5018  1          SC_DST.
;    5019  1          SC_DS2.
;    5020  1          SC_ECC.
;    5021  1          SC_ECD.
;    5022  1          SC_RCT.
;    5023  1          SC_FUL.
;    5024  1          SC_576.
;    5025  1          SC_FCT.
;    5026  1          SC_SWP.
;    5027  1          SC_HWP.
;    5028  1          SC_EC1.
;    5029  1          SC_EC2.
;    5030  1          SC_EC3.
;    5031  1          SC_EC4.
;    5032  1          SC_EC5.
```

```
;    5033  1          SC_EC6,
;    5034  1          SC_EC7,
;    5035  1          SC_EC8,
;    5036  1          SC_EC9,
;    5037  1          SC_ODA,
;    5038  1          SC_OD8,
;    5039  1          SC_NXM,
;    5040  1          SC_PAR,
;    5041  1          SC_CTO,
;    5042  1          SC_SDS,
;    5043  1          SC_EDC,
;    5044  1          SC_IDS,
;    5045  1          SC_SRT,
;    5046  1          SC_SRI,
;    5047  1          SC_POE,
;    5048  1          SC_RDY,
;    5049  1          SC_CLK,
;    5050  1          SC_RSP,
;    5051  1          SC_SUR,
;    5052  1          SC_PSP,
;    5053  1          CER_01,
;    5054  1          CER_02,
;    5055  1          CNTR_ERR : vector [23],
;    5056  1          RDRX_ERR : vector [7],
;    5057  1          SPACE4,
;    5058  1          CRLF,
;    5059  1          DASH,
;    5060  1          ASTERISK,
;    5061  1          HWQ1,
;    5062  1          HWQ2,
;    5063  1          HWQ3,
;    5064  1          HWQ4,
;    5065  1          HWQ5,
;    5066  1          HWQ6A,
;    5067  1          HWQ6B,
;    5068  1          HWQ7A,
;    5069  1          HWQ7B,
;    5070  1          HWQ8,
;    5071  1          HWQ9,
;    5072  1          HWQ10,                                    !ZZZ
;    5073  1          HWQ11,                                    !ZZZ
;    5074  1          SWQ1,
;    5075  1          SWQ2,
;    5076  1          SWQ4,
;    5077  1          SWQ7,
;    5078  1          SWQ9,
;    5079  1          SWQ10,
;    5080  1          SWQ11,
;    5081  1          SWQ12,
;    5082  1          SWQ13,
;    5083  1          SWQ14,
;    5084  1          SWQ15,
;    5085  1          SWQ17,
```

```
;    5086  1          SWQ19,
;    5087  1          SWQ20,
;    5088  1          SWQ21,
;    5089  1          SWQ22,
;    5090  1          SWQ23,
;    5091  1          SWQ24,
;    5092  1          SWQ25,
;    5093  1          SWQ26,                                          !ZZZ
;    5094  1          EH_0,                                           !ZZZ
;    5095  1          EH_1,                                           !ZZZ
;    5096  1          EH_2,                                           !ZZZ
;    5097  1          EH_3,                                           !ZZZ
;    5098  1          EH_4,                                           !ZZZ
;    5099  1          EH_5,                                           !ZZZ
;    5100  1          EH_6,                                           !ZZZ
;    5101  1          EH_7,                                           !ZZZ
;    5102  1          EH_8,                                           !ZZZ
;    5103  1          EH_9,                                           !ZZZ
;    5104  1          EH_10,                                          !ZZZ
;    5105  1          EH_12,                                          !ZZZ
;    5106  1          EH_13,                                          !ZZZ
;    5107  1          SWM1,
;    5108  1          NULL,
;    5109  1          SWP_FLAGS : word,
;    5110  1          L$HIMEM,
;    5111  1          L$LUN,
;    5112  1          L$UNIT;
;    5113  1      !   0_BRK;
;    5114  1
;    5115  1      own
;    5116  1          TBL_SUC : vector [17] initial (NULL, SC_SDI, SC_CON, NULL, SC_DUP, NULL, NULL,
;    5117  1                    NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SON),
;    5118  1          TBL_OFL : vector [9] initial (SC_UNK, SC_VOL, SC_IOP, NULL, SC_DUP, NULL, NULL,
;    5119  1                    NULL, SC_DIS),
;    5120  1          TBL_MFE : vector [11] initial (SC_FER, NULL, SC_ISH, SC_DST, SC_EC9, SC 576,
;    5121  1                    SC_FCT, SC_ECC, SC_RCT, SC_FUL, SC_EC1),
;    5122  1          TBL_WPT : vector [3] initial (NULL, SC_SWP, SC_HWP),
;    5123  1          TBL_DAT : vector [16] initial (SC_FE2, NULL, SC_IS2, SC_DS2, SC_EC9, NULL, NULL,
;    5124  1                    SC_ECD, SC_EC1, SC_EC2, SC_EC3, SC_EC4, SC_EC5, SC_EC6, SC_EC7, SC_EC8),
;    5125  1          TBL_HST : vector [5] initial (NULL, SC_ODA, SC_ODB, SC_NXM, SC_PAR),
;    5126  1          TBL_CNT : vector [4] initial (SC_CTO, SC_SDS, SC_EDC, SC_IDS),
;    5127  1          TBL_DRV : vector [9] initial (NULL, SC_SRT, SC_SRI, SC_POE, SC_RDY, SC_CLK, SC_RSP,
;    5128  1                    SC_SUR, SC_PSP);
;    5129  1
;    5130  1
```

```
;    5131  1     %sbttl 'TYPE AND DESCRIPTION
;    5132  1
;    5133  1     EQUALS;
;    5134  1
;    5135  1     DEVTYP (%asciz'RQDX1 or RUX50');                    ! NAME OF DEVICE SUPPORTED BY PROGRAM
;    5136  1     DESCRIPT (%asciz'RD/RX EXERCISER');     ! TEST DESCRIPTION
```

```
;    5137  1     %sbttl 'HARDWARE PARAMETER CODING SECTION
;    5138  1
;    5139  1     !+
;    5140  1     ! THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5141  1     ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5142  1     ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5143  1     ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5144  1     ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5145  1     ! WITH THE OPERATOR.
;    5146  1     !-
;    5147  1
;    5148  1     BGNHRD;
;    5149  1
;    5150  1     GPRMA (HWQ1, 0, 0, %o'160000', %o'177777', YES, 1);                    ! IP ADDRESS
;    5151  1     GPRMA (HWQ2, 2, 0, %o'4', %o'774', YES, 1);                            ! VECTOR
;    5152  1     GPRMD (HWQ3, 4, 0, %o'377', %o'0', %o'7', YES, 1);                     ! BR LEVEL
;    5153  1     GPRMD (HWQ4, 6, 0, %o'17', %decimal'0', %decimal'15', YES, 1);         ! RDRX DRIVE NUMBER
;    5154  1     GPRML (HWQ10, 6, %o'000040', YES, 1);                                  !ALSO RUN DUP EXERCISER     ZZZ
;    5155  1     XFERF (NODU);                                                          !                           ZZZ
;    5156  1     GPRML (HWQ11, 6, %o'000100', YES, 1);                                  !WRITE DIAG AREA            ZZZ
;    5157  1     $L (NODU);                                                             !                           ZZZ
;    5158  1     GPRML (HWQ5, 6, %o'000200', YES, 1);                                   !TEST ENTIRE CUSTOMER AREA? ZZZ
;    5159  1     XFERT (TOQ8);                                                          !BR IF YES                  ZZZ
;    5160  1     GPRMD (HWQ6A, 8, 0, %o'177777', %decimal'0', %o'177777', YES, 1);      ! STARTING LBN LO           ZZZ
;    5161  1     GPRMD (HWQ6B, 10, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! STARTING LBN HI           ZZZ
;    5162  1     GPRMD (HWQ7A, 12, 0, %o'177777', GP$ATLO (8), %o'177777', YES, 1);     ! ENDING LBN LO             ZZZ
;    5163  1     GPRMD (HWQ7B, 14, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! ENDING LBN HI             ZZZ
;    5164  1     $L (TOQ8);                                                             !                           ZZZ
;    5165  1     GPRML (HWQ8, 6, %o'100000', NO, 0);                                    ! WRITE ON CUST DATA AREA
;    5166  1     XFERF (HWDONE);                                                        ! NO   DONE
;    5167  1     GPRML (HWQ9, 6, %o'100000', NO, 1);                                    ! ** WARNING / CONFIRM
;    5168  1     $L (HWDONE);
;    5169  1
;    5170  1     ENDHRD;
```

```
;  5171  1     %sbttl 'SOFTWARE PARAMETER CODING SECTION'
;  5172  1
;  5173  1     !+
;  5174  1     ! THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;  5175  1     ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;  5176  1     ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;  5177  1     ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;  5178  1     ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;  5179  1     ! WITH THE OPERATOR.
;  5180  1     !-
;  5181  1
;  5182  1     BGNSFT;
;  5183  1
;  5184  1     !GPRML (SWQ16, 4, SWF_TRC, YES, 1);                          ! ENABLE DIAGNOSTIC TRACE
;  5185  1     GPRMD (SWQ24, 10, D, %o'177777', 0, 2359, YES, 1);           ! START TIME
;  5186  1     GPRMD (SWQ1, 0, D, %o'177777', 0, 65535, YES, 1);            ! ERROR LIMIT
;  5187  1     GPRMD (SWQ2, 2, D, %o'177777', 0, 99, YES, 1);               ! TRANSFER LIMIT
;  5188  1     GPRMD (SWQ17, 8, D, %o'177777', 0, 100, YES, 1);             ! PERCENT OF RD OPERATIONS
;  5189  1     GPRML (SWQ15, 4, SWF_CST, YES, 1);                           ! CLEAR STATISTICAL TABLES ?
;  5190  1     GPRML (SWQ20, 4, SWF_FER, YES, 1);                           ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT SET?
;  5191  1     GPRML (SWQ23, 4, SWF_BLK, YES, 1);                           ! HALT ON BAD-BLOCK TYPE ERRORS WITH 'HOE' FLAG?
;  5192  1     GPRML (SWQ21, 4, SWF_HRD, YES, 1);                           ! HALT ON HARD ERRORS WITH 'HOE' FLAG SET?
;  5193  1     GPRML (SWQ22, 4, SWF_SFT, YES, 1);                           ! HALT ON SOFT ERRORS WITH 'HOE' FLAG SET?
;  5194  1     GPRML (SWQ25, 4, SWF_TRY, YES, 1);                           ! COUNT EACH RETRY AS ANOTHER SOFT-ERROR?
;  5195  1     GPRML (SWQ4, 4, SWF_RDM, YES, 1);                            ! RANDOM SEEK MODE ?
;  5196  1     XFERF (SW1);                                                 ! IF NO, DO NEXT QUESTION
;  5197  1     XFER (SW2);                                                  !
;  5198  1     $L (SW1);                                                    !
;  5199  1     GPRML (SWQ19, 4, SWF_SEQ, YES, 1);                           ! RANDOM OR SEQUENTIAL SELECTION OF DRIVES
;  5200  1     $L (SW2);                                                    !
;  5201  1     GPRML (SWQ7, 4, SWF_CRC, YES, 1);                            ! READ-COMPARES AT CONTROLLER ?
;  5202  1     GPRML (SWQ26, 4, SWF_APT, YES, 1);                           !RUNNING UNDER A.P.T. MONITOR?          ZZZ
;  5203  1     DISPLAY (SWM1);                                              ! REMAINING QUESTIONS ONLY APPLY ...
;  5204  1     GPRML (SWQ9, 4, SWF_CWC, YES, 1);                            ! WRITE-COMPARES AT CONTROLLER ?
;  5205  1     XFERF (SW3);                                                 ! IF NO, DO NEXT QUESTION
;  5206  1     XFER (SW4);                                                  !
;  5207  1     $L (SW3);                                                    !
;  5208  1     GPRML (SWQ10, 4, SWF_HWC, YES, 1);                           ! CHECK WRITES AT HOST BY READING ?
;  5209  1     $L (SW4);                                                    !
;  5210  1     GPRML (SWQ11, 4, SWF_UDP, YES, 1);                           ! USER-DEFINED DATA PATTERN ?
;  5211  1     XFERF (SW5);                                                 ! IF NO, DO NEXT QUESTION
;  5212  1     XFER (SW6);                                                  !
;  5213  1     $L (SW5);                                                    !
;  5214  1     GPRMD (SWQ12, 6, D, %o'177777', 0, DP_CNT, YES, 1);          ! SELECT PRE-DEFINED DATA PATTERN
;  5215  1     XFER (SW7);                                                  ! DONE
;  5216  1     $L (SW6);                                                    !
;  5217  1     GPRMD (SWQ13, 12, D, %o'177777', 1, MAX_UDP_CNT, YES, 1);    ! NO. OF WORDS IN USER DATA PATTERN
;  5218  1     GPRMD (SWQ14, 14, O, %o'177777', 0, %o'177777', NO, 12);     ! PATTERN VALUES
;  5219  1     $L (SW7);
;  5220  1     ENDSFT;
```

```
;    5221  1
;    5222  1
;    5223  1     %sbttl 'REPORT CODING SECTION
;    5224  1
;    5225  1
;    5226  1     !+
;    5227  1     ! THE REPORT CODING SECTION CONTAINS THE
;    5228  1     ! "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;    5229  1     !
;    5230  1
;    5231  1
;    5232  2     BGNRPT;
;    5233  2
;    5234  2     local
;    5235  2         CUR_PRIORITY : word;
;    5236  2
;    5237  2     GETPRI (CUR_PRIORITY);
;    5238  2     !ZZZ SETPRI (PRI04);                                    !ZZZ
;    5239  2     SETPRI (.BRLEVEL);                              !ZZZ
;    5240  2
;    5241  2     PRINTS (RPT1);
;    5242  2     PRINTS (RPT2);
;    5243  2     PRINTS (RPT3);
;    5244  2     PRINTS (RPT4);
;    5245  2     PRINTS (RPT5);
;    5246  2     PRINTS (RPT6);
;    5247  2
;    5248  2     incr CTLR from 0 to MAX_CTLR - 1 do
;    5249  2
;    5250  3         begin
;    5251  3         SET_CPAR (.CTLR);
;    5252  3
;    5253  3         incr DISK from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF UN) by UNIT SIZE do
;    5254  3
;    5255  4             begin
;    5256  4             SET_UPAR (.DISK);
;    5257  4
;    5258  4
;    5259  4             if .CST_ADDR [.DISK + OF_DATA, D_PRES] eql PRESENT
;    5260  4             then
;    5261  4
;    5262  5                 begin
;  P 5263  5                 PRINTS (RPT7,
;    5264  5                     .L$LUN, .CST_ADDR [.DISK + OF_DATA, D_DISK_NUM], CST [.CTLR, .DISK + OF_NAME_0, D_NAME_0]);
;  P 5265  5                 PRINTS (RPT8,
;  P 5266  5                     .T_ADDR [TOT_READS_HI], .T_ADDR [TOT_READS_LO],
;    5267  5                     .T_ADDR [MTOT_BYT_RED], .T_ADDR [TOT_BYT_RED_HI], .T_ADDR [TOT_BYT_RED_LO]);
;  P 5268  5                 PRINTS (RPT8,
;  P 5269  5                     .T_ADDR [TOT_WRITES_HI], .T_ADDR [TOT_WRITES_LO],
;    5270  5                     .T_ADDR [MTOT_BYT_WRT], .T_ADDR [TOT_BYT_WRT_HI], .T_ADDR [TOT_BYT_WRT LO]);
;  P 5271  5                 PRINTS (RPT9,
;  P 5272  5                     .T_ADDR [ERR_HRD_SEK], .T_ADDR [ERR_HRD_DAT], .T_ADDR [ERR_HRD_DRV], .T_ADDR [ERR_HRD_HST],
;    5273  5                     .T_ADDR [ERR_SFT_SEK], .T_ADDR [ERR_SFT_DAT], .T_ADDR [ERR_SFT_DRV], .T_ADDR [ERR_SFT_HST]);
```

```
;    5274  4                    end;
;    5275  3                end;
;    5276  3
;    5277  3
;    5278  3            if .CST [.CTLR, STATE] eql PRESENT
;    5279  3            then
;    5280  3
;    5281  4                begin
;    5282  4                PRINTS (RPT10);
;    5283  4                PRINTS (RPT11, .C_ERR_TBL [.CTLR, C_ERR_HRD], .C_ERR_TBL [.CTLR, C_ERR_SFT]);
;    5284  3                end;
;    5285  3
;    5286  3
;    5287  2            end;
;    5288  2
;    5289  2        SETPRI (.CUR_PRIORITY);
;    5290  2
;    5291  2        IF .RD_COUNT NEQ 0                                              !IF THERE IS A WINCHESTER       ZZZ
;    5292  2        THEN                                                            !THEN OUTPUT EXTRA LINES        ZZZ
;    5293  2
;    5294  3        begin                                                           ! PRINTS DUP DATA
;    5295  3        printc(crlf);
;    5296  3        PRINTS(RPT13);
;    5297  3        PRINTS(RPT14);
;    5298  3        PRINTS(RPT15);
;    5299  3        INCR CTLR FROM 0 TO MAX_CTLR-1 DO
;    5300  4                BEGIN
;    5301  4                SET_CPAR(.CTLR);
;    5302  4                INCR DISK FROM (0+OF_UN) TO (3+UNIT_SIZE+OF_UN) BY UNIT_SIZE DO
;    5303  5                    BEGIN
;    5304  5                    SET_UPAR(.DISK);
;    5305  5                    IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_51 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;    5306  5                    THEN
; P  5307  5                PRINTS (RPT16,
; P  5308  5                    .L0LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;    5309  5                    .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;    5310  5
;    5311  5    !ZZZ            IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_52 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;    5312  5    !ZZZ            THEN
;    5313  5    !ZZZ        PRINTS (RPT18,
;    5314  5    !ZZZ            .L0LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;    5315  5    !ZZZ            .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;    5316  4                    END;
;    5317  3                END;
;    5318  3                PRINTS (CRLF);
;    5319  2        END;
;    5320  2
.    5321  1        ENDRPT;


                                .TITLE  ZRQAM2 RD/RX EXERCISER
                                .IDENT  /V01.9/
                                .ENABL  AMA
```

```
000000                                          .PSECT  $CODE$,  RO
000000      122     121     104         L$DVTYP::
                                        .ASCII  /RQD/
000003      130     061     040                 .ASCII  /X1 /
000006      157     162     040                 .ASCII  /or /
000011      122     125     130                 .ASCII  /RUX/
000014      065     060     000                 .ASCII  /50/<00>
000017      000                                 .ASCII  <00>
000020      122     104     057         L$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          $LTOQB:         .WORD   1004
000214  003120          GP$12::         .WORD   3120
000216  000000G                         .WORD   HWQB
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.SDI
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.ODB
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, CLK.PRESENT
.GLOBL  HOE.FLAG, FORCED.ERROR, FER.LBN
.GLOBL  FER.BC, INIT.OCCURED, ADDR.VECT.OK
.GLOBL  DBM5, P.INDEX, S.PATTERN, S.DUPPKT
.GLOBL  RD.COUNT, BRLEVEL, D.FAIL, DBM107
.GLOBL  DU.MSG, DU.RSN, RPT1, RPT2, RPT3
.GLOBL  RPT4, RPT5, RPT6, RPT7, RPT8, RPT9
.GLOBL  RPT10, RPT11, RPT12, RPT13, RPT14
.GLOBL  RPT15, 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.OD8, 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

100000                  BIT15=          -100000
```

```
040000                  BIT14**         40000
020000                  BIT13**         20000
010000                  BIT12**         10000
004000                  BIT11**         4000
002000                  BIT10**         2000
001000                  BIT09**         1000
000400                  BIT08**         400
000200                  BIT07**         200
000100                  BIT06**         100
000040                  BIT05**         40
000020                  BIT04**         20
000010                  BIT03**         10
000004                  BIT02**         4
000002                  BIT01**         2
000001                  BIT00**         1
001000                  BIT9**          1000
000400                  BIT8**          400
000200                  BIT7**          200
000100                  BIT6**          100
000040                  BIT5**          40
000020                  BIT4**          20
000010                  BIT3**          10
000004                  BIT2**          4
000002                  BIT1**          2
000001                  BIT0**          1
000040                  EF.START**      40
000037                  EF.RESTART**    37
000036                  EF.CONTINUE**   36
000035                  EF.NEW**        35
000034                  EF.PWR**        34
000340                  PRI07**         340
000300                  PRI06**         300
000240                  PRI05**         240
000200                  PRI04**         200
000140                  PRI03**         140
000100                  PRI02**         100
000040                  PRI01**         40
000000                  PRI00**         0
000004                  EVL**           4
000010                  LOT**           10
000020                  ADR**           20
000040                  IDU**           40
000100                  ISR**           100
000200                  UAM**           200
000400                  BOE**           400
001000                  PNT**           1000
002000                  PRI**           2000
004000                  IXE**           4000
010000                  IBE**           10000
020000                  IER**           20000
040000                  LOE**           40000
100000                  HOE**           -100000
000044'                 L$HARD**        L$HRDLN+2
```

J12


```
        000240                          L$SOFT==        L$SFTLN+2


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

000000  004137  000000G        LRPT:   JSR     R1,$SAVE4                       ;                               5220
000004  104440                         TRAP    40                              ;                               5237
000006  010004                         MOV     R0,R4                           ; *,CUR.PRIORITY
000010  013700  000000G                MOV     BRLEVEL,R0                      ;                               5239
000014  104441                         TRAP    41
000016  012746  000000G                MOV     #RPT1, (SP)                     ;                               5241
000022  012746  000001                 MOV     #1,-(SP)
000026  010600                         MOV     SP,R0                           ; SP,*
000030  104416                         TRAP    16
000032  012716  000000G                MOV     #RPT2,(SP)                      ;                               5242
000036  012746  000001                 MOV     #1,-(SP)
000042  010600                         MOV     SP,R0                           ; SP,*
000044  104416                         TRAP    16
000046  012716  000000G                MOV     #RPT3,(SP)                      ;                               5243
000052  012746  000001                 MOV     #1,-(SP)
000056  010600                         MOV     SP,R0                           ; SP,*
000060  104416                         TRAP    16
000062  012716  000000G                MOV     #RPT4,(SP)                      ;                               5244
000066  012746  000001                 MOV     #1,-(SP)
000072  010600                         MOV     SP,R0                           ; SP,*
000074  104416                         TRAP    16
000076  012716  000000G                MOV     #RPT5,(SP)                      ;                               5245
000102  012746  000001                 MOV     #1,-(SP)
000106  010600                         MOV     SP,R0                           ; SP,*
000110  104416                         TRAP    16
000112  012716  000000G                MOV     #RPT6,(SP)                      ;                               5246
000116  012746  000001                 MOV     #1,-(SP)
000122  010600                         MOV     SP,R0                           ; SP,*
000124  104416                         TRAP    16
000126  005002                         CLR     R2                              ; CTLR                          5248
000130  010216          1$:            MOV     R2,(SP)                         ; CTLR,*                        5251
000132  004737  000000V                JSR     PC,SET.CPAR
000136  012703  000003                 MOV     #3,R3                           ; *,DISK                        5253
000142  010316          2$:            MOV     R3,(SP)                         ; DISK,*                        5256
000144  004737  000000V                JSR     PC,SET.UPAR
000150  010301                         MOV     R3,R1                           ; DISK,*                        5259
000152  006301                         ASL     R1
000154  063701  000000G                ADD     CST.ADDR,R1
000160  032711  040000                 BIT     #40000,(R1)
000164  001535                         BEQ     3$
000166  010216                         MOV     R2,(SP)                         ; CTLR,*                        5264
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               ;                       5267
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               ;                       5270
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               ;                       5273
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                  ;                       5262
000460  062703  000012  3$:     ADD     #12,R3                  ; *,DISK                5253
000464  020327  000041          CMP     R3,#41                  ; DISK,*
000470  003624                  BLE     2$
```

```
000472  010216                          MOV     R2,(SP)                         ; CTLR,*                        5278
000474  012746  000126                  MOV     #126.,-(SP)
000500  004737  000000G                 JSR     PC,BL$MUL
000504  005726                          TST     (SP)+
000506  005760  000002G                 TST     CST+2(R0)
000512  100026                          BPL     4$
000514  012716  000000G                 MOV     #RPT10,(SP)                     ;                               5282
000520  012746  000001                  MOV     #1,-(SP)
000524  010600                          MOV     SP,R0                           ; SP,*
000526  104416                          TRAP    16
000530  010200                          MOV     R2,R0                           ; CTLR,*                        5283
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                          ;                               5281
000570  005202                  4$:     INC     R2                              ; CTLR                          5248
000572  000243                          .WORD CLV!CLC

000574  003002                          BGT     5$
000576  000137  000660'                 JMP     1$
000602  010400                  5$:     MOV     R4,R0                           ; CUR.PRIORITY,*                5289
000604  104441                          TRAP    41
000606  005737  000000G                 TST     RD.COUNT                        ;                               5291
000612  001522                          BEQ     9$
000614  012716  000000G                 MOV     #CRLF,(SP)                      ;                               5295
000620  012746  000001                  MOV     #1,-(SP)
000624  010600                          MOV     SP,R0                           ; SP,*
000626  104416                          TRAP    16
000630  012716  000000G                 MOV     #RPT13,(SP)                     ;                               5296
000634  012746  000001                  MOV     #1,-(SP)
000640  010600                          MOV     SP,R0                           ; SP,*
000642  104416                          TRAP    16
000644  012716  000000G                 MOV     #RPT14,(SP)                     ;                               5297
000650  012746  000001                  MOV     #1,-(SP)
000654  010600                          MOV     SP,R0                           ; SP,*
000656  104416                          TRAP    16
000660  012716  000000G                 MOV     #RPT15,(SP)                     ;                               5298
000664  012746  000001                  MOV     #1,-(SP)
000670  010600                          MOV     SP,R0                           ; SP,*
000672  104416                          TRAP    16
000674  005003                          CLR     R3                              ; CTLR                          5299
000676  010316                  6$:     MOV     R3,(SP)                         ; CTLR,*                        5301
000700  004737  000000V                 JSR     PC,SET.CPAR
000704  012702  000003                  MOV     #3,R2                           ; *,DISK                        5302
000710  010216                  7$:     MOV     R2,(SP)                         ; DISK,*                        5304
000712  004737  000000V                 JSR     PC,SET.UPAR
000716  010201                          MOV     R2,R1                           ; DISK,*                        5305
```

M12


```
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                       ;                               5309
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                        5302
001024  020227  000041                  CMP     R2,#41                          ; DISK,*
001030  003727                          BLE     7$
001032  005203                          INC     R3                              ; CTLR                          5299
001034  000243                          .WORD CLV!CLC

001036  003717                          BLE     6$
001040  012716  000000G                 MOV     #CRLF,(SP)                      ;                               5318
001044  012746  000001                  MOV     #1,-(SP)
001050  010600                          MOV     SP,R0                           ; SP,*
001052  104416                          TRAP    16
001054  062706  000012                  ADD     #12,SP                          ;                               5294
001060  062706  000016          9$:     ADD     #16,SP                          ;                               5220
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                         ;                               5319
000004  104425                          TRAP    25
000006  000207                          RTS     PC

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


;   5322  1
```

N12


```
;    5323  1     %sbttl 'INITIALIZE SECTION
;    5324  1
;    5325  2     BGNINIT;
;    5326  2
;    5327  2     local
;    5328  2         DELAY_MULT : word,
;    5329  2         FLAG : byte,
;    5330  2         TEMP : word,
;    5331  2         HWPT_REF : ref block [HWPT_LEN, word] field (HWP FIELDS),
;    5332  2         CLEAR_TABLES : byte,
;    5333  2         SMALLEST_DRIVE : byte,
;    5334  2         BLANKS : WORD INITIAL (%o'020040 ),                          !ZZZ
;    5335  2         HWPT_ADDRESS : vector [MAX_UNITS, word];
;    5336  2
;    5337  2     SETPRI (PRIO7);                                                 ! NO INTERRUPTS ALLOWED DURING INIT
;    5338  2
;    5339  2     if READEF (EF_NEW)                                              ! IS THIS A NEW PASS?
;    5340  2     then
;    5341  3         begin
;    5342  3         ENTRY_REASON = NEW_PASS;
;    5343  3
;    5344  4         if not BIT_TST (SWP_FLAGS, SWF_CST)
;    5345  3         then
;    5346  3             CLEAR_TABLES = FALSE
;    5347  3         else
;    5348  3             CLEAR TABLES = TRUE;
;    5349  3
;    5350  2         end;
;    5351  2
;    5352  2     if READEF (EF_START)                                            ! IS THIS A START?
;    5353  2     then
;    5354  3         begin
;    5355  3         BRESET;
;    5356  3         ENTRY_REASON = START,
;    5357  3         CLEAR_TABLES = TRUE;
;    5358  3         ADDR_VECT_OK = FALSE;
;    5359  3         INIT_OCCURED = FALSE;
;    5360  2         end;
;    5361  2
;    5362  2     if READEF (EF_RESTART)                                          ! IS THIS A RESTART?
;    5363  2     then
;    5364  3         begin
;    5365  3         ENTRY_REASON = RESTART;
;    5366  3         CLEAR_TABLES = TRUE;
;    5367  2         end;
;    5368  2
;    5369  2     if READEF (EF_CONTINUE)                                         ! IS THIS A CONTINUE?
;    5370  2     then
;    5371  3         begin
;    5372  3         ENTRY_REASON = CONT;
;    5373  3
;    5374  4         if not BIT_TST (SWP FLAGS, SWF_CST)
;    5375  3         then
```

```
;    5376  3            CLEAR_TABLES = FALSE
;    5377  3        else
;    5378  3            CLEAR_TABLES = TRUE;
;    5379  3
;    5380  2        end;
;    5381  2
;    5382  2    if READEF (EF_PWR)                                          ! ARE WE HERE BECAUSE OF POWER FAIL
;    5383  2    then
;    5384  3        begin
;    5385  3        ENTRY_REASON = PWR_FAIL;
;    5386  3        ADDR_VECT_OK = FALSE;
;    5387  3        INIT_OCCURED = FALSE;
;    5388  3        CLEAR_TABLES = TRUE;
;    5389  3        PRINTF (MSG_01);                                        ! "POWER DELAY   WAITING"
;    5390  3
;    5391  3        incr COUNT from 0 to 60 do                              ! WAIT APPROX. 60 SECONDS
;    5392  4            begin
;    5393  4            DELAY_MULT = 333;
;    5394  4            DELAY (.DELAY_MULT);
;    5395  4            BREAK;                                              ! BREAK FOR AC'
;    5396  3            end;
;    5397  3
;    5398  2        end;
;    5399  2
;    5400  2    !SETVEC (0_TVEC, 0_BRK, PRI07);                             ! SET ODT TRAP VECTOR
;    5401  2
;    5402  2    !+
;    5403  2    !       MAKE SURE THAT NOT MORE THAN MAX_UNITS HAVE BEEN SPECIFIED
;    5404  2    !       IF THERE ARE TOO MANY, NOTIFY USER AND RETURN TO SUPERVISOR
;    5405  2    !       (DIAGNOSTIC IS ABORTED).
;    5406  2    !
;    5407  2
;    5408  2    if .L$UNIT gtru MAX_UNITS
;    5409  2    then
;    5410  3        begin
;    5411  3        ERRSF (1, EGS_01, EMS_01);
;    5412  3        DOCLN;
;    5413  2        end;
;    5414  2
;    5415  2    !+
;    5416  2    !       THE FOLLOWING CODE IS EXECUTED FOR ALL ENTRY REASONS EXCEPT NEW_PASS.
;    5417  2    !       ALL RUN-TIME CONTROLLER STATUS TABLES (CSTs) ARE CLEARED TO 0, THEN
;    5418  2    !       LOADED WITH CONFIGURATION DATA FROM THE HARDWARE P-TABLES.
;    5419  2    !
;    5420  2
;    5421  2    if .ENTRY_REASON neq NEW_PASS
;    5422  2    then
;    5423  3        begin
;    5424  3        SMALLEST_DRIVE = 255;                                   ! LARGEST DISK NO. ALLOWED BY MSCP
;    5425  3
;    5426  3        incr COUNT from 0 to ((MAX_CTLR * CST_LEN * 2)   2) by 2 do
;    5427  3            (CST + .COUNT) = 0;
;    5428  3
```

```
;  5429  3          incr UNIT from 0 to (.L$UNIT   1) do                                    ! LOOP THROUGH ALL UNITS
;  5430  3
;  5431  3              if (HWPT_ADDRESS [.UNIT] + GPHARD (.UNIT, HWPT REF)) neqa 0         ! IF HWP TABLE FOUND
;  5432  3              then
;  5433  3
;  5434  3                  if .HWPT_REF [HWP_DISK_NUM] lssu .SMALLEST_DRIVE                ! FIND OUT THE SMALLEST DISK NUMBER
;  5435  3                  then
;  5436  3                      SMALLEST_DRIVE = .HWPT_REF [HWP DISK NUM];
;  5437  3
;  5438  3          incr UNIT from 0 to (.L$UNIT - 1) do                                    ! LOOP THROUGH ALL UNITS
;  5439  3
;  5440  3              if .HWPT_ADDRESS [.UNIT] neqa 0                                     ! IF HWP TABLE FOUND
;  5441  3              then
;  5442  4                  begin
;  5443  4                  FLAG = NOT_FOUND;
;  5444  4                  HWPT_REF = .HWPT_ADDRESS [.UNIT];
;  5445  4
;  5446  4                  incr CTLR from 0 to (MAX_CTLR - 1) do                           ! LOOP THROUGH ALL CSTs
;  5447  4
;  5448  4                      if .CST [.CTLR, IP_ADDR] eqla .HWPT_REF [HWP_IP_ADDR]
;  5449  4                      then
;  5450  4
;  5451  4                          if .CST [.CTLR, (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE
;  5452  4                                  + OF_UN + OF_DATA, D_PRES] eql NOT_PRESENT
;  5453  4                          then
;  5454  5                              begin                                           ! IF EMPTY SLOT FOUND
;  5455  5                              TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;  5456  5                              CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;  5457  5                                                                              ! COPY DISK ADDR AND PROT BIT
;  5458  5                              CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;  5459  5                              CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;  5460  5                              CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;  5461  5
;  5462  5                              IF .HWPT_REF [HWP_ENTIRE] EQL TRUE              !ZZZ IF DEFAULT TEST RANGE,
;  5463  5                              THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;        !ZZZ MAKE HI ADDR ALL ONES
;  5464  5
;  5465  5                              CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =
;  5466  5                                      .HWPT_REF [HWP_BEG_TRK];                !ZZZ
;  5467  5                              CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =          !ZZZ
;  5468  5                                      .HWPT_REF [HWP_BEG_TRK1];               !ZZZ
;  5469  5                              CST [.CTLR, .TEMP + OF_END, D_END0] =           !ZZZ
;  5470  5                                      .HWPT_REF [HWP_END_TRK];                !ZZZ
;  5471  5                              CST [.CTLR, .TEMP + OF_END1, D_END1] =          !ZZZ
;  5472  5                                      .HWPT_REF [HWP_END_TRK1];               !ZZZ
;  5473  5
;  5474  5                              CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;  5475  5                              CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;  5476  5
;  5477  5
;  5478  5                              CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;    !ZZZ
;  5479  5                              CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =  !ZZZ
;  5480  5                                      NOT (.HWPT_REF [HWP_DISK_DUPEX]);       !ZZZ
;  5481  5
```

```
;    5482  5                        CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =    !ZZZ
;    5483  5                            (.HWPT_REF [HWP_DISK_DUPWT]);               !ZZZ
;    5484  5                        CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;     !ZZZ
;    5485  5                        FLAG = FOUND;
;    5486  5                        exitloop;
;    5487  5                        end
;    5488  4                    else
;    5489  5                        begin                                           ! DUPLICATE UNIT
;    5490  5                        PRINTF (CER_01, .HWPT_REF [HWP_DISK_NUM], .HWPT_REF [HWP_IP_ADDR]);
;    5491  5                                                                        ! "DUPLICATE UNIT: X AT IP: XXXXXX"
;    5492  5                        DUR [.UNIT] = DU_CONF;                          ! CONFIGURATION ERROR
;    5493  5                        DODU (.UNIT);                                   ! DROP UNIT
;    5494  5                        FLAG = FOUND;
;    5495  5                        exitloop;
;    5496  4                        end;
;    5497  4
;    5498  4                if .FLAG eql NOT_FOUND                                  ! IF NO IP MATCH TO EXISTING CST
;    5499  4                then
;    5500  5                    begin
;    5501  5
;    5502  5                    incr CTLR from 0 to (MAX_CTLR - 1) do               ! LOOP THROUGH EACH CST
;    5503  5
;    5504  5                        if .CST [.CTLR, IP_ADDR] eqla 0                 ! IF EMPTY CST FOUND
;    5505  5                        then
;    5506  6                            begin
;    5507  6                            CST [.CTLR, IP_ADDR] = .HWPT_REF [HWP_IP_ADDR];
;    5508  6                            CST [.CTLR, VEC_ADDR] = .HWPT_REF [HWP_VECTOR];
;    5509  6                            CST [.CTLR, BR_LEV] = .HWPT_REF [HWP_BR_LEVEL];
;    5510  6                            TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE * OF_UN;
;    5511  6                            CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;    5512  6                                                                        ! COPY DISK ADDR AND PROT BIT
;    5513  6                            CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;    5514  6                            CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;    5515  6                            CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;    5516  6
;    5517  6                            IF .HWPT_REF [HWP_ENTIRE] EQL TRUE          !ZZZ IF DEFAULT TEST RANGE,
;    5518  6                            THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;    !ZZZ MAKE NT ADDR ALL ONES
;    5519  6
;    5520  6                            CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =       !ZZZ
;    5521  6                                .HWPT_REF [HWP_BEG_TRK];                !ZZZ
;    5522  6                            CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =      !ZZZ
;    5523  6                                .HWPT_REF [HWP_BEG_TRK1];               !ZZZ
;    5524  6                            CST [.CTLR, .TEMP + OF_END, D_END0] =       !ZZZ
;    5525  6                                .HWPT_REF [HWP_END_TRK];                !ZZZ
;    5526  6                            CST [.CTLR, .TEMP + OF_END1, D_END1] =      !ZZZ
;    5527  6                                .HWPT_REF [HWP_END_TRK1];               !ZZZ
;    5528  6
;    5529  6                            CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;    5530  6                            CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;    5531  6
;    5532  6
;    5533  6                            CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;    !ZZZ
;    5534  6                            CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =  !ZZZ
```

```
;  5535  6                                    NOT (.HWPT_REF [HWP_DISK_DUPEX]);                    !ZZZ
;  5536  6                            CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =                 !ZZZ
;  5537  6                                (.HWPT_REF [HWP_DISK_DUPWT]);                            !ZZZ
;  5538  6                            CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;                  !ZZZ
;  5539  6                            FLAG = FOUND;
;  5540  6                            exitloop;
;  5541  5                            end;                                                         ! IF EMPTY CST FOUND
;  5542  5
;  5543  5                        if .FLAG eql NOT_FOUND                                           ! IF NO EMPTY CST FOUND
;  5544  5                        then
;  5545  6                            begin
;  5546  6                            PRINTF (CER_02, MAX_CTLR);                                   ! "MORE THAN X IP ADDRESSES.
;  5547  6                            DUR [.UNIT] = DU_CONF;                                       ! CONFIGURATION ERROR
;  5548  6                            DODU (.UNIT);                                                ! DROP UNIT
;  5549  5                            end;
;  5550  5
;  5551  4                        end;                                                             ! IF NO IP ADDR MATCH IN CST
;  5552  4
;  5553  3                    end;                                                                 ! IF GPHARD RETURNS A HWP TABLE
;  5554  3            !
;  5555  3            ! CONFIGURATON CHECK FOR LEGAL RDRX UNIT MIX BECAUSE WE HAVE DIFFERENT
;  5556  3            ! DRIVES :  THE RD51, RD52, AND RX50.
;  5557  3            ! (NEEDED?)
;  5558  3            !
;  5559  2            end;                                                                         ! END OF "NON NEW_PASS" INIT
;  5560  2
;  5561  2        if .ENTRY_REASON eql NEW_PASS
;  5562  2        then
;  5563  3            begin
;  5564  3
;  5565  3            incr UNIT from 0 to (.L$UNIT - 1) do
;  5566  3                GPHARD (.UNIT, HWPT_REF);                                                ! DUMMY GPHARDs FOR NEW PASS
;  5567  3
;  5568  3            incr CTLR from 0 to (MAX_CTLR - 1) do
;  5569  4                begin
;  5570  4                CST [.CTLR, U_CNT] = 0;                                                  ! REINITIALIZE UNIT COUNT
;  5571  4
;  5572  4                incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  5573  4                    CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;                    ! START EACH UNIT AS OFFLINE
;  5574  4
;  5575  3                end;
;  5576  3
;  5577  2            end;
;  5578  2
;  5579  2        if .ENTRY_REASON eql START
;  5580  2        then
;  5581  3            begin
;  5582  3            CTLR_CNT = 0;                                                                ! NUMBER OF CONFIGURED CONTROLLERS
;  5583  3
;  5584  3            incr CTLR from 0 to (MAX_CTLR - 1) do
;  5585  3
;  5586  3                if .CST [.CTLR, IP_ADDR] neqa 0                                          ! IF CONTROLLER IS PRESENT
;  5587  3                then
```

```
;    5588  3                CTLR_CNT = .CTLR_CNT + 1;                          ! INCREMENT CONTROLLER COUNT
;    5589  3
;    5590  3            MEMORY (FREE_MEM_ADDR);                                ! GET START OF FREE MEMORY
;    5591  3
;    5592  2            end;                                                   ! END OF "START" INITIALIZATION
;    5593  2
;    5594  2        !+
;    5595  2        !       CLEAR STATISTICS TABLES
;    5596  2        !
;    5597  2
;    5598  2        incr UNITS from 0 to MAX_UNITS - 1 do                      ! CLEAR CURRENT STATISTICS
;    5599  2            incr COUNT from 0 to TALLY_CLEAR - 1 do
;    5600  2                TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;
;    5601  2
;    5602  2        if .CLEAR_TABLES                                           ! IF CLEAR TABLES ON EVERY PASS
;    5603  2        then
;    5604  2            incr UNITS from 0 to MAX_UNITS - 1 do
;    5605  2                incr COUNT from TALLY_CLEAR to TALLY_LEN - 1 do    ! INITIALIZE TOTALS
;    5606  2                    TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;       !
;    5607  2
;    5608  2        if .CLEAR_TABLES
;    5609  2        then
;    5610  2            incr CTLR from 0 to MAX_CTLR - 1 do
;    5611  3                begin
;    5612  3                C_ERR_TBL [.CTLR, C_ERR_HRD] = 0;                  ! INITALIZE CONTROLLER ERRORS
;    5613  3                C_ERR_TBL [.CTLR, C_ERR_SFT] = 0;                  !
;    5614  2                end;
;    5615  2
;    5616  2        !+
;    5617  2        !       MISCELLANEOUS INITIALIZATON
;    5618  2        !-
;    5619  2
;    5620  2        incr CTLR from 0 to (MAX_CTLR - 1) do                      ! INIT NO. OF OUTSTANDING QIOs
;    5621  2            QIO [.CTLR] = 0;
;    5622  2
;    5623  2        incr COUNT from 0 to (RP_CNT - 1) do                       ! INITIALIZE RETURN PACKET POOL
;    5624  2            RP_USE [.COUNT] = -1;
;    5625  2
;    5626  2        if .CLK_PRESENT                                            ! STOP CLOCK IF PRESENT
;    5627  2        then
;    5628  2            LINE_CLOCK = 0;
;    5629  2
;    5630  2        IODQ_IN = IODQ_OUT = 0;                                    ! INIT I/O DONE QUEUE POINTERS
;    5631  2        CRN_LOW = CRN_HIGH = 0;                                    ! INIT COMMAND REFERENCE NUMBER
;    5632  2        SETPRI (PRI00);                                            ! SET PROGRAM PRIORITY TO 0
;    5633  2
;    5634  1        ENDINIT;

                                .GLOBL  L$DLY

                                .SBTTL  LINIT INITIALIZE SECTION
```

```
000000  004137  000000G             LINIT:  JSR     R1,$SAVE5                       ;                                       5321
000004  162706  000030                      SUB     #30,SP
000010  012746  020040                      MOV     #20040,-(SP)                    ; *,BLANKS
000014  012700  000340                      MOV     #340,R0                         ;                                       5337
000020  104441                              TRAP    41
000022  012700  000035                      MOV     #35,R0                          ;                                       5339
000026  104447                              TRAP    47
000030  103014                              BHIS    2$
000032  112737  000005  000000G             MOVB    #5,ENTRY.REASON                 ;                                       5342
000040  105737  000000G                     TSTB    SWP.FLAGS                       ;                                       5344
000044  100403                              BMI     1$
000046  105066  000012                      CLRB    12(SP)                          ; CLEAR.TABLES                          5346
000052  000403                              BR      2$                              ;                                       5344
000054  112766  000001  000012      1$:     MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5348
000062  012700  000040              2$:     MOV     #40,R0                          ;                                       5352
000066  104447                              TRAP    47
000070  103013                              BHIS    3$
000072  104433                              TRAP    33                              ;                                       5354
000074  112737  000001  000000G             MOVB    #1,ENTRY.REASON                 ;                                       5356
000102  112766  000001  000012              MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5357
000110  105037  000000G                     CLRB    ADDR.VECT.OK                    ;                                       5358
000114  105037  000000G                     CLRB    INIT.OCCURED                    ;                                       5359
000120  012700  000037              3$:     MOV     #37,R0                          ;                                       5362
000124  104447                              TRAP    47
000126  103006                              BHIS    4$
000130  112737  000002  000000G             MOVB    #2,ENTRY.REASON                 ;                                       5365
000136  112766  000001  000012              MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5366
000144  012700  000036              4$:     MOV     #36,R0                          ;                                       5369
000150  104447                              TRAP    47
000152  103014                              BHIS    6$
000154  112737  000003  000000G             MOVB    #3,ENTRY.REASON                 ;                                       5372
000162  105737  000000G                     TSTB    SWP.FLAGS                       ;                                       5374
000166  100403                              BMI     5$
000170  105066  000012                      CLRB    12(SP)                          ; CLEAR.TABLES                          5376
000174  000403                              BR      6$                              ;                                       5374
000176  112766  000001  000012      5$:     MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5378
000204  012700  000034              6$:     MOV     #34,R0                          ;                                       5382
000210  104447                              TRAP    47
000212  103043                              BHIS    12$
000214  112737  000004  000000G             MOVB    #4,ENTRY.REASON                 ;                                       5385
000222  105037  000000G                     CLRB    ADDR.VECT.OK                    ;                                       5386
000226  105037  000000G                     CLRB    INIT.OCCURED                    ;                                       5387
000232  112766  000001  000012              MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5388
000240  012746  000000G                     MOV     #MSG.01,-(SP)                   ;                                       5389
000244  012746  000001                      MOV     #1,-(SP)
000250  010600                              MOV     SP,R0                           ; SP,*
000252  104417                              TRAP    17
000254  012702  000075                      MOV     #75,R2                          ; *,COUNT                               5391
000260  012703  000515              7$:     MOV     #515,R3                         ; *,DELAY.MULT                          5393
000264  010301                              MOV     R3,R1                           ; DELAY.MULT,$$TMP2                     5394
000266  001411              8$:     BEQ     11$
000270  013700  000000G                     MOV     L$DLY,R0                        ; *,$$TMP1
000274  001404                              BEQ     10$
```

H13


```
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$:    TRAP    22
000314  005302                              DEC     R2                              ; COUNT                         5391
000316  001360                              BNE     7$
000320  022626                              CMP     (SP)+,(SP)+                     ;                               5384
000322  023727  000000G 000004      12$:    CMP     L$UNIT,#4                       ;                               5408
000330  101405                              BLOS    13$
000332  104454                              TRAP    54                              ;                               5411
000334  000001                              .WORD   1
000336  000000G                             .WORD   EGS.01
000340  000000V                             .WORD   EMS.01
000342  104444                              TRAP    44
000344  123727  000000G 000005      13$:    CMPB    ENTRY.REASON,#5                 ;                               5421
000352  001002                              BNE     14$
000354  000137  003726'                      JMP     43$
000360  112766  000377  000010      14$:    MOVB    #377,10(SP)                     ; *,SMALLEST.DRIVE              5424
000366  005000                              CLR     R0                              ; COUNT                         5426
000370  005060  000000G             15$:    CLR     CST(R0)                         ; *(COUNT)                      5427
000374  062700  000002                      ADD     #2,R0                           ; *,COUNT                       5426
000400  020027  000124                      CMP     R0,#124                         ; COUNT,*
000404  003771                              BLE     15$
000406  013704  000000G                     MOV     L$UNIT,R4                       ;                               5429
000412  005003                              CLR     R3                              ; UNIT
000414  000435                              BR      18$
000416  010302                      16$:    MOV     R3,R2                           ; UNIT,*                        5431
000420  006302                              ASL     R2
000422  012700  000022                      MOV     #22,R0
000426  060600                              ADD     SP,R0                           ; HWPT.ADDRESS,*
000430  060002                              ADD     R0,R2
000432  010300                              MOV     R3,R0                           ; UNIT,*
000434  104442                              TRAP    42
000436  010001                              MOV     R0,R1                           ; *,HWPT.REF
000440  010112                              MOV     R1,(R2)                         ; HWPT.REF,*
000442  001421                              BEQ     17$
000444  005002                              CLR     R2                              ;                               5434
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),*                 5436
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$:    INC     R3                              ; UNIT                          5429
000510  020304                      18$:    CMP     R3,R4                           ; UNIT,*
000512  002741                              BLT     16$
000514  013766  000000G 000016              MOV     L$UNIT,16(SP)                   ;                               5438
000522  005004                              CLR     R4                              ; UNIT
```

```
000524  000137  003704                  19$:    JMP     41$
000530  010400                                  MOV     R4,R0                   ; UNIT,*                5440
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$:    CLRB    6(SP)                   ; FLAG                  5443
000560  011001                                  MOV     (R0),R1                 ; *,HWPT.REF            5444
000562  005066  000002                          CLR     2(SP)                   ; CTLR                  5446
000566  016646  000002                  21$:    MOV     2(SP),-(SP)             ; CTLR,*                5448
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$:    MOV     #1,14(SP)               ;                       5485
000624  112766  000001  000006              MOVB    #1,6(SP)                ; *,FLAG
000632  012705  000006                          MOV     #6,R5                   ;                       5451
000636  060105                                  ADD     R1,R5                   ; HWPT.REF,*
000640  111546                                  MOVB    (R5),-(SP)
000642  042716  177760                          BIC     #177760,(SP)
000646  005000                                  CLR     R0
000650  156600  000012                          BISB    12(SP),R0               ; SMALLEST.DRIVE,*
000654  160016                                  SUB     R0,(SP)
000656  012746  000012                          MOV     #12.,-(SP)
000662  004737  000000G                         JSR     PC,BL$MUL
000666  010066  000010                          MOV     R0,10(SP)
000672  005726                                  TST     (SP)+
000674  016616  000004                          MOV     4(SP),(SP)              ; CTLR,*                5452
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                5455
000736  062702  000003                          ADD     #3,R2                   ; *,TEMP
000742  010300                                  MOV     R3,R0                   ;                       5456
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,*                5458
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)                     ;                               5459
001000  052710  040000                  BIS     #40000,(R0)                     ;                               5460
001004  105715                          TSTB    (R5)                            ;                               5462
001006  100003                          BPL     23$
001010  012761  177777  000016          MOV     #-1,16(R1)                      ; *,*(HWPT.REF)                 5463
001016  010300                  23$:    MOV     R3,R0                           ;                               5465
001020  060200                          ADD     R2,R0                           ; TEMP,*
001022  006300                          ASL     R0
001024  016160  000010  000002G         MOV     10(R1),CST+2(R0)                ; *(HWPT.REF),*
001032  010300                          MOV     R3,R0                           ;                               5467
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                           ;                               5469
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                           ;                               5471
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                           ;                               5474
001100  060200                          ADD     R2,R0                           ; TEMP,*
001102  006300                          ASL     R0
001104  011660  000012G                 MOV     (SP),CST+12(R0)                 ; BLANKS,*
001110  010300                          MOV     R3,R0                           ;                               5475
001112  060200                          ADD     R2,R0                           ; TEMP,*
001114  006300                          ASL     R0
001116  011660  000014G                 MOV     (SP),CST+14(R0)                 ; BLANKS,*
001122  010300                          MOV     R3,R0                           ;                               5478
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)                      ;                               5480
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$:    TST     (SP)+
001176  111516                          MOVB    (R5),(SP)                       ;                               5482
001200  042710  010000                  BIC     #10000,(R0)
001204  032726  000100                  BIT     #100,(SP)+
001210  001402                          BEQ     26$
001212  052710  010000                  BIS     #10000,(R0)
001216  010300                  26$:    MOV     R3,R0                           ;                               5484
001220  060200                          ADD     R2,R0                           ; TEMP,*
```

```
001222  006300                          ASL     R0
001224  005060  000022G                 CLR     CST+22(R0)
001230  000430                          BR      29$                             ;                               5454
001232  011146                  27$:    MOV     (R1),-(SP)                      ; HWPT.REF,*                    5490
001234  111546                          MOVB    (R5),-(SP)
001236  042716  177760                  BIC     #177760,(SP)
001242  012746  000000G                 MOV     #CER.01,-(SP)
001246  012746  000003                  MOV     #3,-(SP)
001252  010600                          MOV     SP,R0                           ; SP,*
001254  104417                          TRAP    17
001256  062706  000010                  ADD     #10,SP
001262  112764  000001  000000G         MOVB    #1,DUR(R4)                      ; *,*(UNIT)                     5492
001270  010400                          MOV     R4,R0                           ; UNIT,*                        5493
001272  104451                          TRAP    51
001274  000406                          BR      29$                             ;                               5489
001276  005266  000002          28$:    INC     2(SP)                           ; CTLR                          5446
001302  000243                          .WORD CLV!CLC

001304  003002                          BGT     29$
001306  000137  002414'                 JMP     21$
001312  105766  000006          29$:    TSTB    6(SP)                           ; FLAG                          5498
001316  001402                          BEQ     30$
001320  000137  003702'                 JMP     40$
001324  005066  000014          30$:    CLR     14(SP)                          ; CTLR                          5502
001330  016646  000014          31$:    MOV     14(SP),-(SP)                    ; CTLR,*                        5504
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$:    MOV     (R1),CST(R0)                    ; HWPT.REF,*                    5507
001364  016103  000002                  MOV     2(R1),R3                        ; *(HWPT.REF),*                 5508
001370  042703  177000                  BIC     #177000,R3
001374  042760  000777  000002G         BIC     #777,CST+2(R0)
001402  050360  000002G                 BIS     R3,CST+2(R0)
001406  116160  000004  000004G         MOVB    4(R1),CST+4(R0)                 ; *(HWPT.REF),*                 5509
001414  012705  000006                  MOV     #6,R5                           ;                               5510
001420  060105                          ADD     R1,R5                           ; HWPT.REF,*
001422  111546                          MOVB    (R5),-(SP)
001424  042716  177760                  BIC     #177760,(SP)
001430  005000                          CLR     R0
001432  156600  000012                  BISB    12(SP),R0                       ; SMALLEST.DRIVE,*
001436  160016                          SUB     R0,(SP)
001440  012746  000012                  MOV     #12,-(SP)
001444  004737  000000G                 JSR     PC,BL$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,*                        5511
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,*                5513
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)             ;                       5514
001534  052710  040000                  BIS     #40000,(R0)             ;                       5515
001540  105715                          TSTB    (R5)                    ;                       5517
001542  100003                          BPL     33$
001544  012761  177777  000016          MOV     #-1,16(R1)              ; *,*(HWPT.REF)         5518
001552  010300                  33$:    MOV     R3,R0                   ;                       5520
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                   ;                       5522
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                   ;                       5524
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                   ;                       5526
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                   ;                       5529
001634  060200                          ADD     R2,R0                   ; TEMP,*
001636  006300                          ASL     R0
001640  011660  000012G                 MOV     (SP),CST+12(R0)         ; BLANKS,*
001644  010300                          MOV     R3,R0                   ;                       5530
001646  060200                          ADD     R2,R0                   ; TEMP,*
001650  006300                          ASL     R0
001652  011660  000014G                 MOV     (SP),CST+14(R0)         ; BLANKS,*
001656  010300                          MOV     R3,R0                   ;                       5533
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)              ;                       5535
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)+
```

M13


```
001722  103002                          BCC     35$
001724  052710  100000                  BIS     #100000,(R0)
001730  005726                  35$:    TST     (SP)+
001732  111516                          MOVB    (R5),(SP)               ;                       5536
001734  042710  010000                  BIC     #10000,(R0)
001740  032726  000100                  BIT     #100,(SP)+
001744  001402                          BEQ     36$
001746  052710  010000                  BIS     #10000,(R0)
001752  010300                  36$:    MOV     R3,R0                   ;                       5538
001754  060200                          ADD     R2,R0                   ; TEMP,*
001756  006300                          ASL     R0
001760  005060  000022G                 CLR     CST+22(R0)
001764  112766  000001  000006          MOVB    #1,6(SP)                ; *,FLAG                5539
001772  000410                          BR      39$                     ;                       5506
001774  005266  000014          37$:    INC     14(SP)                  ; CTLR                  5502
002000  000243                          .WORD CLV!CLC

002002  003002                          BGT     38$
002004  000137  003156'                 JMP     31$
002010  105766  000006          38$:    TSTB    6(SP)                   ; FLAG                  5543
002014  001017                  39$:    BNE     40$
002016  012746  000001                  MOV     #1,-(SP)                ;                       5546
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)             5547
002044  010400                          MOV     R4,R0                   ; UNIT,*                5548
002046  104451                          TRAP    51
002050  062706  000006                  ADD     #6,SP                   ;                       5545
002054  005204                  40$:    INC     R4                      ; UNIT                  5438
002056  020466  000016          41$:    CMP     R4,16(SP)               ; UNIT,*
002062  002002                          BGE     42$
002064  000137  002356'                 JMP     19$
002070  123727  000000G 000005  42$:    CMPB    ENTRY.REASON,#5         ;                       5561
002076  001051                          BNE     48$
002100  013703  000000G         43$:    MOV     L$UNIT,R3               ;                       5565
002104  005004                          CLR     R4                      ; UNIT
002106  000404                          BR      45$
002110  010400                  44$:    MOV     R4,R0                   ; UNIT,*                5566
002112  104442                          TRAP    42
002114  010001                          MOV     R0,R1                   ; *,HWPT.REF
002116  005204                          INC     R4                      ; UNIT                  5565
002120  020403                  45$:    CMP     R4,R3                   ; UNIT,*
002122  002772                          BLT     44$
002124  005003                          CLR     R3                      ; CTLR                  5568
002126  010346                  46$:    MOV     R3,-(SP)                ; CTLR,*                5570
002130  012746  000126                  MOV     #126,-(SP)
002134  004737  000000G                 JSR     PC,BL$MUL
002140  105060  000005G                 CLRB    CST+5(R0)
002144  010316                          MOV     R3,(SP)                 ; CTLR,*                5573
002146  012746  000053                  MOV     #53, (SP)
002152  004737  000000G                 JSR     PC,BL$MUL
```

N13


```
002156  012701  000003                  MOV     #3,R1                   ; *,OFFSET              5572
002162  010002                  47$:    MOV     R0,R2                   ;                       5573
002164  060102                          ADD     R1,R2                   ; OFFSET,*
002166  006302                          ASL     R2
002170  042762  020000  000000G         BIC     #20000,CST(R2)
002176  062701  000012                  ADD     #12,R1                  ; *,OFFSET              5572
002202  020127  000041                  CMP     R1,#41                  ; OFFSET,*
002206  003765                          BLE     47$
002210  062706  000006                  ADD     #6,SP                   ;                       5569
002214  005203                          INC     R3                      ; CTLR                  5568
002216  000243                          .WORD CLV!CLC

002220  003742                          BLE     46$
002222  123727  000000G 000001  48$:    CMPB    ENTRY.REASON,#1         ;                       5579
002230  001017                          BNE     51$
002232  005037  000000G                 CLR     CTLR.CNT                ;                       5582
002236  005000                          CLR     R0                      ; CTLR                  5584
002240  005760  000000G         49$:    TST     CST(R0)                 ; *(CTLR)               5586
002244  001402                          BEQ     50$
002246  005237  000000G                 INC     CTLR.CNT                ;                       5588
002252  062700  000126          50$:    ADD     #126,R0                 ; *,CTLR                5584
002256  000243                          .WORD CLV!CLC

002260  003767                          BLE     49$
002262  104431                          TRAP    31                      ;                       5590
002264  010037  000000G                 MOV     R0,FREE.MEM.ADDR
002270  005001                  51$:    CLR     R1                      ; UNITS                 5598
002272  005003                  52$:    CLR     R3                      ; COUNT                 5599
002274  010300                  53$:    MOV     R3,R0                   ; COUNT,*               5600
002276  060100                          ADD     R1,R0                   ; UNITS,*
002300  006300                          ASL     R0
002302  005060  000000G                 CLR     TALLY(R0)
002306  005203                          INC     R3                      ; COUNT                 5599
002310  020327  000006                  CMP     R3,#6                   ; COUNT,*
002314  003767                          BLE     53$
002316  062701  000033                  ADD     #33,R1                  ; *,UNITS               5598
002322  020127  000121                  CMP     R1,#121                 ; UNITS,*
002326  003761                          BLE     52$
002330  032766  000001  000012          BIT     #1,12(SP)               ; *,CLEAR.TABLES        5602
002336  001436                          BEQ     57$
002340  005001                          CLR     R1                      ; UNITS                 5604
002342  012703  000007                  MOV     #7,R3           54$:    ; *,COUNT               5605
002346  010300                  55$:    MOV     R3,R0                   ; COUNT,*               5606
002350  060100                          ADD     R1,R0                   ; UNITS,*
002352  006300                          ASL     R0
002354  005060  000000G                 CLR     TALLY(R0)
002360  005203                          INC     R3                      ; COUNT                 5605
002362  020327  000032                  CMP     R3,#32                  ; COUNT,*
002366  003767                          BLE     55$
002370  062701  000033                  ADD     #33,R1                  ; *,UNITS               5604
002374  020127  000121                  CMP     R1,#121                 ; UNITS,*
002400  003760                          BLE     54$
002402  032766  000001  000012          BIT     #1,12(SP)               ; *,CLEAR.TABLES        5608
```

```
002410  001411                          BEQ     5718
002412  005000                          CLR     R0                      ; CTLR                  5610
002414  105060  000000G         5618:   CLRB    C.ERR_TBL(R0)           ; @(CTLR)               5612
002420  105060  000001G                 CLRB    C.ERR_TBL+1(R0)         ; @(CTLR)               5613
002424  062700  000002                  ADD     #2,R0                   ; @.CTLR                5610
002430  000243                          .WORD CLV!CLC

002432  003770                          BLE     5618
002434  005000                  5718:   CLR     R0                      ; CTLR                  5620
002436  105060  000000G         5818:   CLRB    @IO(R0)                 ; @(CTLR)               5621
002442  005200                          INC     R0                      ; CTLR                  5620
002444  000243                           .WORD CLV!CLC

002446  003773                          BLE     5818
002450  005000                          CLR     R0                      ; COUNT                 5623
002452  112760  000377  000000G 5918:   MOVB    #377,@P.USE(R0)         ; @.@(COUNT)            5624
002460  005200                          INC     R0                      ; COUNT                 5623
002462  020027  000007                  CMP     R0,#7                   ; COUNT,#
002466  003771                          BLE     5918
002470  132737  000001  000000G        BITB    #1,CLK.PRESENT          ;                       5626
002476  001402                          BEQ     6018
002500  005037  177546                  CLR     @#177546                ;                       5628
002504  005037  000000G         6018:   CLR     IODQ.OUT                ;                       5630
002510  005037  000000G                 CLR     IODQ.IN
002514  005037  000000G                 CLR     CRN.HIGH                ;                       5631
002520  005037  000000G                 CLR     CRN.LOW
002524  005000                          CLR     R0                      ;                       5632
002526  104441                          TRAP    41
002530  062706  000032                  ADD     #32,SP                  ;                       5321
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                ;                       5632
000004  104411                          TRAP    11
000006  000207                          RTS     PC

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

```
;    5635  1     %sbttl 'AUTODROP SECTION
;    5636  1
;    5637  1     !+
;    5638  1     ! THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;    5639  1     ! THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
;    5640  1     ! SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
;    5641  1     ! DROPPED FROM TESTING.
;    5642  1     !
;    5643  1
;    5644  2     BGNAUTO;
;    5645  2
;    5646  2     !if BIT_TST (SWP_FLAGS, SWF_TRC)
;    5647  2     !then
;    5648  2     !   PRINTF (DBM3);
;    5649  2
;    5650  2     return;
;    5651  2
;    5652  1     ENDAUTO;


                                        .SBTTL  LAUTO AUTODROP SECTION
000000  000207                  LAUTO:  RTS     PC                      ;                                       5634

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


                                        .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  004374'         L$AUTO::JSR     PC,LAUTO                ;                                       5650
000004  104461                          TRAP    61
000006  000207                          RTS     PC

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

```
;    5653  1      %sbttl 'CLEANUP CODING SECTION
;    5654  1
;    5655  1      !+
;    5656  1      ! THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
;    5657  1      ! AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
;    5658  1      !
;    5659  1
;    5660  2      BGNCLN;
;    5661  2
;    5662  2      LABEL                                                                                    !ZZZ
;    5663  2          LZ1;                                                                                 !ZZZ
;    5664  2
;    5665  2      DORPT;
;    5666  2
;    5667  2      !CLRVEC (O_TVEC);                                   ! RETURN ODT TRAP TO DIAGNOSTIC SUPERVISER
;    5668  2
;    5669  2      if .CLK_PRESENT
;    5670  2      then
;    5671  3          begin
;    5672  3          LINE_CLOCK = 0;                                 ! STOP THE LINE-CLOCK
;    5673  3      !   CLRVEC (%o'100');                               ! RETURN LINE-CLOCK'S VECTOR TO SUPERVISOR
;    5674  2          end;
;    5675  2
;    5676  2      incr CTLR from 0 to (MAX_CTLR   1) do               ! FOR EACH CONTROLLER
;    5677  2
;    5678  2          if (RDRX_ADDR = .CST [.CTLR, IP_ADDR]) neqa 0   ! IF CONTROLLER EXISTS
;    5679  2          then
;    5680  3              begin
;    5681  3
;    5682  3              if .ADDR_VECT_OK
;    5683  3              then
;    5684  4      LZ1:        begin                                                                        !ZZZ
;    5685  4
;    5686  4                  if .DCT [.CTLR, STAT] eql ONLINE        ! IF CONTROLLER ALIVE
;    5687  4                  then
;    5688  4
;    5689  4                      incr COUNT from 1 to 10000 do
;    5690  5                          begin
;    5691  5                          DELAY (1);
;    5692  5
;    5693  5                          if .DCT [.CTLR, CRING_CNT] eql 0    ! WAIT TILL OUTSTANDING COMMANDS FINISHED
;    5694  5                          then
;    5695  5
;    5696  5                               INCR Z  FROM 0 TO 3  DO                                         !ZZZ
;    5697  6                                  BEGIN                                                        !ZZZ
;    5698  6                                  TEMP1 =  (.DCT [.CTLR, RR_BEG]) + 4 * .Z;     !DESCRIPTOR ADDRESS !ZZZ
;    5699  6                                  TEMP2 = ..TEMP1;                              !PACKET ADDRESS     !ZZZ
;    5700  6                                  IF ..TEMP2  EQL CRN_LOW                       !CRN                !ZZZ
;    5701  6                                  THEN                                          !IF THE LAST CRN IS BACK, !ZZZ
;    5702  6                                     (WRT_RDRX (RCIP, RC_ALL, ALL_ONES); LEAVE LZ1); !THEN STOP WAITING !ZZZ
;    5703  5                                  END;                                                         !ZZZ
;    5704  4                          end;
;    5705  4
```

```
;    5706  4              WRT RDRX (RCIP, RC ALL, ALL_ONES);          ! WRITE IP TO STOP DEVICE
;    5707  3              end;
;    5708  3
;    5709  3          CLRVEC (.CST[ CTLR, VEC ADDR]);                 ! RETURN CONTROLLER S TRAP VECTOR TO SUPERVISOR
;    5710  2          end;
;    5711  2
;    5712  1     ENDCLN;


                                        .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  004137  000000G          LCLEAN: JSR     R1,$SAVE5                       ;                               5652
000004  005746                          TST     -(SP)
000006  104424                          TRAP    24                              ;                               5663
000010  132737  000001  000000G         BITB    #1,CLK.PRESENT                  ;                               5669
000016  001402                          BEQ     1$
000020  005037  177546                  CLR     @#177546                        ;                               5672
000024  005005                  1$:     CLR     R5                              ; CTLR                          5676
000026  010546                  2$:     MOV     R5,-(SP)                        ; CTLR,*                        5678
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                 ;                               5682
000062  001466                          BEQ     12$
000064  010546                          MOV     R5,-(SP)                        ; CTLR,*                        5686
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                       5689
000112  012702  000001          3$:     MOV     #1,R2                           ; *,$$TMP2                      5691
000116  001410                  4$:     BEQ     7$
00012C  013701  000000G                 MOV     L$DLY,R1                        ; *,$$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$:     TSTB    DCT(R0)                         ;                               5693
000144  001027                          BNE     10$
000146  005001                          CLR     R1                              ; Z                             5696
000150  016037  000004G 000000G  8$:     MOV     DCT+4(R0),TEMP1                 ;                               5698
000156  060137  000000G                 ADD     R1,TEMP1                        ; Z,*
000162  017737  000000G 000000G         MOV     @TEMP1,TEMP2                    ;                               5699
000170  027727  000000G 000000G         CMP     @TEMP2,#CRN.LOW                 ;                               5700
000176  001005                          BNE     9$
000200  012702  177777                  MOV     #-1,R2                          ; *,RC.REG                      5702
000204  010277  000000G                 MOV     R2,@RDRX.ADDR                   ; RC.REG,*
000210  000413                          BR      12$
```

```
000212  062701  000004          9$:     ADD     #4,R1                   ; *,Z                   5696
000216  020127  000014                  CMP     R1,#14                  ; Z,*
000222  003752                          BLE     8$
000224  005304                  10$:    DEC     R4                      ; COUNT                 5689
000226  001331                          BNE     3$
000230  012700  177777          11$:    MOV     #-1,R0                  ; *,RC.REG              5706
000234  010077  000000G                 MOV     R0,@RDRX.ADDR           ; RC.REG,*
000240  016300  000002G         12$:    MOV     CST+2(R3),R0            ;                       5709
000244  042700  177000                  BIC     #177000,R0
000250  104436                          TRAP    36
000252  005205                  13$:    INC     R5                      ; CTLR                  5676
000254  000243                          .WORD CLV!CLC

000256  003663                          BLE     2$
000260  005726                          TST     (SP)+                   ;                       5652
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               ;                       5710
000004  104412                          TRAP    12
000006  000207                          RTS     PC

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

```
;    5713  1     %sbttl  DROP UNIT SECTION
;    5714  1
;    5715  1     !+
;    5716  1     ! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;    5717  1     ! TO NO LONGER BE TESTED.
;    5718  1     !
;    5719  1
;    5720  2     BGNDU;
;    5721  2
;    5722  2     local
;    5723  2         UNIT : word,                                                    ! UNIT NUMBER
;    5724  2         PRINT : byte initial (byte (FALSE));                            ! NO PRINTING
;    5725  2
;    5726  2     label
;    5727  2         SEARCH;
;    5728  2
;    5729  3     begin
;    5730  3
;    5731  3     register
;    5732  3         INPUT = 0;                                                      ! R0 = UNIT NO.
;    5733  3
;    5734  3     UNIT = .INPUT;                                                      ! GET UNIT NUMBER
;    5735  2     end;                                                                ! UNDECLARE R0
;    5736  2
;    5737  2     !ZZZif BIT_TST (SWP_FLAGS, SWF_TRC)
;    5738  2     !ZZZthen
;    5739  2     !ZZZ    PRINTF (DBM5, .UNIT);
;    5740  2
;    5741  2     SEARCH :                                                            ! SEARCH BLOCK
;    5742  3     begin
;    5743  3
;    5744  3     incr CTLR from 0 to (MAX_CTLR - 1) do                               ! FOR EACH CNTR
;    5745  3
;    5746  3         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    5747  3
;    5748  3             if (.CST [.CTLR, .OFFSET + OF_DATA, D_UNIT] eql .UNIT) and  ! IF UNIT MATCHES
;    5749  4                (.CST [.CTLR, .OFFSET + OF_DATA, D_PRES] eql PRESENT)
;    5750  3             then
;    5751  4                 begin
;    5752  4
;    5753  4                 if (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE) or  ! IF UNIT ALIVE
;    5754  4                    (.DUR [.UNIT] eql DU_ONLINE) or
;    5755  5                    (.DUR [.UNIT] eql DU_PROTECT)
;    5756  4                 then
;    5757  5                     begin
;    5758  5                     PRINT = TRUE;                                      ! O.K. TO PRINT
;    5759  5
;    5760  5                     if (.CST [.CTLR, U_CNT] gtru 0) and
;    5761  6                        (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;    5762  5                     then
;    5763  5                         CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] - 1;  ! DECREMENT COUNT
;    5764  5
;    5765  5                     if (.CST [.CTLR, U_CNT] eql 0) and
```

```
;  5766  6                    (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;  5767  5                then
;  5768  5                    EOP_FLAG = TRUE;                                    ! ALL UNITS OFFLINE
;  5769  5
;  5770  5                CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;       ! MARK UNIT OFFLINE
;  5771  4                end;                                                    ! IF UNIT ALIVE
;  5772  4
;  5773  4             leave SEARCH;                                              ! EXIT SEARCH BLOCK
;  5774  3             end;                                                       ! IF UNIT FOUND
;  5775  3
;  5776  2     end;
;  5777  2
;  5778  2     if .PRINT or                                                       ! IF OK TO PRINT
;  5779  2        (.DUR [.UNIT] eql DU_CONF) or
;  5780  2        (.DUR [.UNIT] eql DU_INIT) or
;  5781  2        (.DUR [.UNIT] eql DU_ONLINE) or
;  5782  3        (.DUR [.UNIT] eql DU_PROTECT)
;  5783  2     then
;  5784  3         begin
;  5785  3         PRINTF (DU_MSG, .UNIT);                                        ! "UNIT XX DROPPED"
;  5786  3         PRINTF (.DU_RSN [.DUR [.UNIT]]);                               ! REASON
;  5787  2         end;
;  5788  2
;  5789  1     ENDDU;


                                .SBTTL  LDU DROP UNIT SECTION
000000  004137  000000G     LDU:  JSR     R1,$SAVE5                 ;                       5712
000004  024646                   CMP     -(SP),-(SP)
000006  105066  000002           CLRB    2(SP)                     ; PRINT
000012  010001                   MOV     R0,R1                     ; INPUT,UNIT            5734
000014  005005                   CLR     R5                        ; CTLR                  5744
000016  010546              1$:  MOV     R5,-(SP)                  ; CTLR,*                5748
000020  012746  000053           MOV     #53,-(SP)
000024  004737  000000G          JSR     PC,BL$MUL
000030  010066  000004           MOV     R0,4(SP)
000034  012703  000003           MOV     #3,R3                     ; *,OFFSET              5746
000040  010300              2$:  MOV     R3,R0                     ; OFFSET,*              5748
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)               ;                       5749
000100  001452                   BEQ     8$
000102  005004                   CLR     R4                        ;                       5753
000104  032712  020000           BIT     #20000,(R2)
000110  001402                   BEQ     3$
```

```
000112  005204                          INC     R4
000114  000410                          BR      4$
000116  126127  000000G 000007   3$:     CMPB    DUR(R1),#7              ; *(UNIT),*     5754
000124  001404                          BEQ     4$
000126  126127  000000G 000011           CMPB    DUR(R1),#11             ; *(UNIT),*     5755
000134  001032                          BNE     7$
000136  112766  000001  000006   4$:     MOVB    #1,6(SP)                ; *,PRINT       5758
000144  010516                          MOV     R5,(SP)                 ; CTLR,*        5760
000146  012746  000126                  MOV     #126,-(SP)
000152  004737  000000G                 JSR     PC,BL$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                      ;               5761
000174  105660  000001                  SBCB    1(R0)                   ;               5763
000200  001006                          BNE     6$                      ;               5765
000202  032712  020000   5$:     BIT     #20000,(R2)             ;               5766
000206  001403                          BEQ     6$
000210  112737  000001  000000G         MOVB    #1,EOP.FLAG             ;               5768
000216  042712  020000   6$:     BIC     #20000,(R2)             ;               5770
000222  022626           7$:     CMP     (SP)+,(SP)+             ;               5751
000224  000411                          BR      9$
000226  062703  000012   8$:     ADD     #12,R3                  ; *,OFFSET      5746
000232  020327  000041                  CMP     R3,#41                  ; OFFSET,*
000236  003700                          BLE     2$
000240  022626                          CMP     (SP)+,(SP)+
000242  005205                          INC     R5                      ; CTLR          5744
000244  000243                  .WORD CLV!CLC

000246  003663                          BLE     1$
000250  032766  000001  000002   9$:     BIT     #1,2(SP)                ; *,PRINT       5778
000256  001020                          BNE     10$
000260  126127  000000G 000001           CMPB    DUR(R1),#1              ; *(UNIT),*     5779
000266  001414                          BEQ     10$
000270  126127  000000G 000002           CMPB    DUR(R1),#2              ; *(UNIT),*     5780
000276  001410                          BEQ     10$
000300  126127  000000G 000007           CMPB    DUR(R1),#7              ; *(UNIT),*     5781
000306  001404                          BEQ     10$
000310  126127  000000G 000011           CMPB    DUR(R1),#11             ; *(UNIT),*     5782
000316  001024                          BNE     11$
000320  010146           10$:    MOV     R1,-(SP)                ; UNIT,*        5785
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),*     5786
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                          ;                       5784
000370  022626                          11$:    CMP     (SP)+,(SP)+                     ;                       5712
000372  000207                                  RTS     PC

; 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                          ;                       5787
000004  104453                                  TRAP    53
000006  000207                                  RTS     PC

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

```
;    5790  1     %sbttl 'ADD UNIT SECTION
;    5791  1
;    5792  1     !+
;    5793  1     ! THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;    5794  1     !  TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;    5795  1     !  TO THE TEST CYCLE.
;    5796  1     !
;    5797  1
;    5798  2     BGNAU;
;    5799  2
;    5800  2     local
;    5801  2         STINDX : word,
;    5802  2         ENDIDX : word;
;    5803  2
;    5804  2     register
;    5805  2         UNIT = 0;                                   ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
;    5806  2
;    5807  3     if BIT_TST (SWP_FLAGS, SWF_CST)
;    5808  2     then
;    5809  3         begin                                       ! IF CLEAR STAT. TABLES TRUE....
;    5810  3         STINDX = .UNIT * TALLY_LEN;                 ! ZERO OUT
;    5811  3         ENDIDX = .STINDX + TALLY_LEN   1;           !    ADDED
;    5812  3
;    5813  3         incr COUNT from .STINDX to .ENDIDX do       !    UNIT'S
;    5814  3             TALLY [.COUNT] = 0;                     !    STATISTICS
;    5815  3
;    5816  2         end;
;    5817  2
;    5818  1     ENDAU;


                                        .SBTTL  LAU ADD UNIT SECTION
000000  004137  000000G         LAU:    JSR     R1,$SAVE2               ;                               5789
000004  105737  000000G                 TSTB    SWP.FLAGS               ;                               5807
000010  100023                         BPL     3$
000012  010046                         MOV     R0,-(SP)                ; UNIT,*                        5810
000014  012746  000033                 MOV     #33,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  010002                         MOV     R0,R2                   ; STINDX,ENDIDX                 5811
000026  062702  000032                 ADD     #32,R2                  ; *,ENDIDX
000032  010001                         MOV     R0,R1                   ; STINDX,COUNT                  5813
000034  005301                         DEC     R1                      ; COUNT
000036  000404                         BR      2$
000040  010100                 1$:     MOV     R1,R0                   ; COUNT,*                       5814
000042  006300                         ASL     R0
000044  005060  000000G                 CLR     TALLY(R0)
000050  005201                 2$:     INC     R1                      ; COUNT                         5813
000052  020102                         CMP     R1,R2                   ; COUNT,ENDIDX
000054  003771                         BLE     1$
000056  022626                         CMP     (SP)+,(SP)+             ;                               5809
000060  000207                 3$:     RTS     PC                      ;                               5789

; 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                  ;                       5816
000004  104452                          TRAP    52
000006  000207                          RTS     PC

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

```
;    5819  1     %sbttl 'NON EXISTENT MEMORY TRAP HANDLER
;    5820  1
;    5821  1     !+
;    5822  1     !       THIS TRAP HANDLER IS VECTORED FROM LOCATION 4 FOR ALL UNIBUS TIMEOUT
;    5823  1     !       ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO REFERENCE A NON-EXISTENT
;    5824  1     !       MEMORY LOCATION. ITS MAIN PURPOSE IS TO SET A FLAG FOR THE RDRX
;    5825  1     !       REGISTER EXISTENCE TEST, INDICATING THE ABSENCE OF A DEVICE REGISTER.
;    5826  1     !
;    5827  1
;    5828  2     BGNSRV (NEX_TRAP);
;    5829  2
;    5830  2     NEX = TRUE;                                     ! NEX TRAP OCCURRED
;    5831  2
;    5832  1     ENDSRV;


                                        .SBTTL  NEX.TRAP NON-EXISTENT MEMORY TRAP HANDLER
000000  012737  000001  000000G  NEX.TRAP::
                                        MOV     #1,NEX                                  ;  5830
000006  000002                          RTI                                             ;  5828

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

```
;    5833  1     %sbttl 'TIME OF DAY
;    5834  1
;    5835  1     !+
;    5836  1     !       THIS INTERRUPT SERVICE ROUTINE KEEPS TRACK OF THE TIME-OF DAY
;    5837  1     !
;    5838  1
;    5839  2     BGNSRV (TIME);
;    5840  2
;    5841  2     CLK_TICKS = .CLK_TICKS + 1;             ! INCREMENT CLOCK-TICKS
;    5842  2
;    5843  2     if .CLK_TICKS gequ 3600
;    5844  2     then
;    5845  3         begin
;    5846  3         MINUTES = .MINUTES + 1;             ! UPDATE MINUTE COUNT
;    5847  3         CLK_TICKS = 0;
;    5848  2         end;
;    5849  2
;    5850  2     if .MINUTES gequ 60
;    5851  2     then
;    5852  3         begin
;    5853  3         HOURS = .HOURS + 1;                 ! UPDATE HOUR COUNT
;    5854  3         MINUTES = 0;
;    5855  2         end;
;    5856  2
;    5857  2     if .HOURS gequ 24
;    5858  2     then
;    5859  2         HOURS = 0;                          ! RATIONALIZE HOURS
;    5860  2
;    5861  1     ENDSRV;


                                        .SBTTL  TIME TIME OF DAY
000000  005237  000000G                TIME::  INC     CLK.TICKS                       ;  5841
000004  023727  000000G 007020                 CMP     CLK.TICKS,#7020                 ;  5843
000012  103404                                 BLO     1$
000014  105237  000000G                        INCB    MINUTES                         ;  5846
000020  005037  000000G                        CLR     CLK.TICKS                       ;  5847
000024  123727  000000G 000074         1$:     CMPB    MINUTES,#74                     ;  5850
000032  103404                                 BLO     2$
000034  105237  000000G                        INCB    HOURS                           ;  5853
000040  105037  000000G                        CLRB    MINUTES                         ;  5854
000044  123727  000000G 000030         2$:     CMPB    HOURS,#30                       ;  5857
000052  103402                                 BLO     3$
000054  105037  000000G                        CLRB    HOURS                           ;  5859
000060  000002                         3$:     RTI                                     ;  5839

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

```
;    5862  1     %sbttl  GLOBAL ROUTINES
;    5863  1
;    5864  1     global routine SET_CPAR (CTLR) : novalue =
;    5865  1
;    5866  1     !+
;    5867  1     !       THIS ROUTINE SETS UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    5868  1     !       FOR THE GIVEN CONTROLLER NUMBER.
;    5869  1     !
;    5870  1     !       INPUTS:
;    5871  1     !               CTLR   CONTROLLER NUMBER
;    5872  1     !
;    5873  1     !       IMPLICIT OUTPUTS:
;    5874  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    5875  1     !               CST_ADDR - ADDRESS OF CONTROLLER'S STATUS TABLE
;    5876  1     !               DCT_ADDR - ADDRESS OF CONTROLLER'S DRIVER TABLE
;    5877  1     !               RORX_ADDR   ADDRESS OF CONTROLLER'S IP REGISTER
;    5878  1     !-
;    5879  1
;    5880  2         begin
;    5881  2         CCTLR = .CTLR;                          ! SET CURRENT CONTROLLER NUMBER
;    5882  2         CST_ADDR = CST + (.CTLR * CST_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER S CS*
;    5883  2         DCT_ADDR = DCT + (.CTLR * DCT_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER S DC*
;    5884  2         RORX_ADDR = .CST_ADDR [IP_ADDR];            ! GET CONTROLLER'S DEVICE ADDRESS
;    5885  1         end.


                                        .SBTTL  SET.CPAR GLOBAL ROUTINES
000000  010146                     SET.CPAR::
                                        MOV     R1,-(SP)                                    .                5864
000002  016601  000004                  MOV     4(SP),R1                                    . [illegible]    5881
000006  010137  000000G                 MOV     R1,CCTLR
000012  010146                         MOV     R1,-(SP)                                    .                5882
000014  012746  000126                  MOV     #126,-(SP)
000020  004737  000000G                 JSR     PC,BLISMUL
000024  062700  000000G                 ADD     #CST,R0
000030  010037  000000G                 MOV     R0,CST.ADDR
000034  010116                         MOV     R1,(SP)                                     .                5883
000036  012746  000022                  MOV     #22,-(SP)
000042  004737  000000G                 JSR     PC,BLISMUL
000046  062700  000000G                 ADD     #DCT,R0
000052  010037  000000G                 MOV     R0,DCT.ADDR
000056  017737  000000G 000000G         MOV     @CST.ADDR,RORX.ADDR                         .                5884
000064  062706  000006                  ADD     #6,SP                                       .                5880
000070  012601                         MOV     (SP)+,R1                                    :                5864
000072  000207                         RTS     PC

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

```
:    5886  1     global routine SET_UPAR (OFFSET) : novalue =
:    5887  .     !+
:    5888  1     !       THIS ROUTINE SETS UP THE COMMONLY-USED UNIT-RELATED DATA ITEMS FOR
:    5889  1     !       THE CURRENT CONTROLLER AND GIVEN CST OFFSET
:    5890  1     !
:    5891  1     !       INPUTS:
:    5892  1     !               OFFSET   WORD OFFSET INTO CURRENT CONTROLLER S CST WHICH
:    5893  1     !                        DESCRIBES A UNIT
:    5894  1     !
:    5895  1     !       IMPLICIT INPUTS.
:    5896  1     !               CST ADDR   ADDRESS OF CURRENT CONTROLLER S CST
:    5897  1     !
:    5898  1     !       IMPLICIT OUTPUTS:
:    5899  1     !               CUOFF - CURRENT UNIT'S CST OFFSET
:    5900  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
:    5901  1     !               L$LUN - CURRENT UNIT NUMBER (DRS UNIT NUMBER)
:    5902  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
:    5903  1     !-
:    5904  2         begin
:    5905  2         CUOFF = .OFFSET;
:    5906  2         CDISK = .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM];
:    5907  2         L$LUN = .CST_ADDR [.OFFSET + OF_DATA, D_UNIT];
:    5908  2         T_ADDR = TALLY + (.L$LUN * TALLY_LEN * 2);
:    5909  1         end;


                                        .SBTTL  SET.UPAR GLOBAL ROUTINES
000000  010146                    SET.UPAR::
                                        MOV     R1,-(SP)                        ;                               5886
000002  016637  000004  000000G         MOV     4(SP),CUOFF                     ; OFFSET.*                      5905
000010  016600  000004                  MOV     4(SP),R0                        ; CUOFF.*                       5906
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                         ;                               5907
000036  000301                          SWAB    R1
000040  042701  177760                  BIC     #177760,R1
000044  010137  000000G                 MOV     R1,L$LUN
000050  010146                          MOV     R1,-(SP)                        ; L$LUN.*                       5908
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)+                     ;                               5904
000074  012601                          MOV     (SP)+,R1                        ;                               5886
000076  000207                          RTS     PC

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

```
;    5910  1
;    5911  1     global routine GET_PKT (CTLR) =
;    5912  1
;    5913  1     !+
;    5914  1     !       THIS ROUTINE SEARCHES THE MSCP PACKET POOL ALLOCATION TABLE (PKT_USE)
;    5915  1     !       FOR A FREE MSCP PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;    5916  1     !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED
;    5917  1     !       TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;    5918  1     !
;    5919  1     !       INPUTS:
;    5920  1     !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;    5921  1     !
;    5922  1
;    5923  2         begin
;    5924  2
;    5925  2         local
;    5926  2             index : signed word initial ( 1),
;    5927  2             RING_ADDR : word,
;    5928  2             PACKET_OWNED : byte,
;    5929  2             NEXT_PACKET : byte;
;    5930  2
;    5931  2
;    5932  2         NEXT_PACKET = .NEXT_PKT_USE;                    ! NEXT PACKET TO TRY
;    5933  2
;    5934  2         incr COUNT from 0 to (PKT_CNT - 1) do           ! FOR EACH ENTRY IN ALLOCATION TABLE
;    5935  3             begin
;    5936  3             PACKET_OWNED = FALSE;
;    5937  3
;    5938  3             if .PKT_USE [.NEXT_PACKET] lss 0            ! IF ENTRY INDICATES FREE PACKET
;    5939  3
;    5940  3             then
;    5941  4                 begin
;    5942  4                 RING_ADDR = .DCT_ADDR [RR_BEG];         ! FIRST RESPONSE PACKET'S ADDRESS
;    5943  4
;    5944  4                 incr I from 1 to (RRING_LEN + CRING_LEN) do ! FOR EACH PACKET ADDRESS
;    5945  4
;    5946  4                     if (..RING_ADDR eqla .MSCP_PKT [.NEXT_PACKET, PKT_LO]) and
;    5947  5                        (((.RING_ADDR + 2) and ED_OWN) eql ED_OWN)
;    5948  5
;    5949  4                     then
;    5950  5                         begin                           ! CHECK ADDRESS AND OWNERSHIP
;    5951  5                         PACKET_OWNED = TRUE;            ! PACKET OWNED BY CONTROLLER
;    5952  5                         exitloop;
;    5953  5                         end
;    5954  4                     else
;    5955  4                         RING_ADDR = .RING_ADDR + 4;     ! ADDRESS OF NEXT PACKET IN RING
;    5956  4
;    5957  4                 if not .PACKET_OWNED                    ! IF NOT ALREADY USED
;    5958  4
;    5959  4                 then
;    5960  5                     begin
;    5961  5                     PKT_USE [.NEXT_PACKET] = .CTLR;     ! ALLOCATE PACKET TO CONTROLLER
;    5962  5                     index = .NEXT_PACKET;
```

```
;    5963  5
;    5964  5                    incr J from 2 to (PKT_LEN   1) do        ! ZERO OUT PACKET
;    5965  5                        MSCP_PKT [.NEXT_PACKET, .J, 0, 16, 0] = 0;
;    5966  5
;    5967  5                    exitloop;                                ! DONE
;    5968  5
;    5969  4                    end;
;    5970  4
;    5971  3                end;
;    5972  3
;    5973  3            NEXT_PACKET = .NEXT_PACKET + 1;                  ! TRY NEXT PACKET IN RING
;    5974  3
;    5975  3            if .NEXT_PACKET gequ PKT_CNT
;    5976  3
;    5977  3            then
;    5978  3                NEXT_PACKET = 0;                             ! IF BEYOND ALL PACKETS, START AT THE TOP
;    5979  3
;    5980  2            end;
;    5981  2
;    5982  2        if (.index geq 0) and                                ! IF PACKET FOUND
;    5983  3           (.PKT_USE [.index] geq 0)
;    5984  2        then
;    5985  2
;    5986  3            begin
;    5987  3            MSCP_PKT [.index, MSGLEN] = SZ_GEN;              ! PACKET SIZE - ONLY ONLINE AND SCC CHANGE IT
;    5988  3            MSCP_PKT [.index, CREDITS] = 1;                  ! CREDIT SIZE
;    5989  3            NEXT_PKT_USE = .NEXT_PACKET + 1;                 ! NEXT PACKET TO ALLOCATE
;    5990  3
;    5991  3            if.NEXT_PKT_USE gequ PKT_CNT
;    5992  3            then
;    5993  3                NEXT_PKT_USE = 0;                            ! CYCLE BACK TO BEGINNING IF AT END
;    5994  3
;    5995  2            end;
;    5996  2
;    5997  2        return .index;
;    5998  2
;    5999  1        end;


                                .SBTTL  GET.PKT GLOBAL ROUTINES
000000  004137  000000G    GET.PKT::
                            JSR     R1,$SAVE5                       ;                                5911
000004  162706  000006             SUB     #6,SP
000010  012704  177777             MOV     #-1,R4                  ; *,INDEX                        5923
000014  113766  000000G 000004     MOVB    NEXT.PKT.USE,4(SP)      ; *,NEXT.PACKET                  5932
000022  012766  000014  000002     MOV     #14,2(SP)               ; *,COUNT                        5934
000030  105016             1$:     CLRB    (SP)                    ; PACKET.OWNED                   5936
000032  005001                     CLR     R1                      ;                                5938
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             ;                                5942
000052  016005  000004             MOV     4(R0),R5                ; *,RING.ADDR
```

```
000056  010146                          MOV     R1, (SP)                        ;                               5946
000060  012746  000106                  MOV     #106,-(SP)
000064  004737  000000G                 JSR     PC,BL$MUL
000070  012702  000010                  MOV     #10,R2                          ; *,I                           5944
000074  021560  000000G         2$:     CMP     (R5),MSCP.PKT(R0)               ; RING.ADDR,*                   5946
000100  001014                          BNE     3$
000102  012703  00000G                  MOV     #2,R3                           ;                               5947
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                5951
000130  000404                          BR      4$                              ;                               5950
000132  062705  000004          3$:     ADD     #4,R5                           ; *,RING.ADDR                   5955
000136  005302                          DEC     R2                              ; I                             5944
000140  001355                          BNE     2$
000142  032766  000001  000004  4$:     BIT     #1,4(SP)                        ; *,PACKET.OWNED                5957
000150  001027                          BNE     6$
000152  116661  000030  000000G         MOVB    30(SP),PKT.USE(R1)              ; CTLR,*                        5961
000160  010104                          MOV     R1,R4                           ; *,INDEX                       5962
000162  010116                          MOV     R1,(SP)                         ;                               5965
000164  012746  000043                  MOV     #43,-(SP)
000170  004737  000000G                 JSR     PC,BL$MUL
000174  005726                          TST     (SP)+
000176  012702  000002                  MOV     #2,R2                           ; *,J                           5964
000202  010003          5$:     MOV     R0,R3                           ;                               5965
000204  060203                          ADD     R2,R3                           ; J,*
000206  006303                          ASL     R3
000210  005063  000000G                 CLR     MSCP.PKT(R3)
000214  005202                          INC     R2                              ; J                             5964
000216  020227  000042                  CMP     R2,#42                          ; J,*
000222  003767                          BLE     5$
000224  022626                          CMP     (SP)+,(SP)+                     ;                               5960
000226  000414                          BR      9$
000230  022626          6$:     CMP     (SP)+,(SP)+                     ;                               5941
000232  105266  000004          7$:     INCB    4(SP)                           ; NEXT.PACKET                   5973
000236  126627  000004  000014          CMPB    4(SP),#14                       ; NEXT.PACKET,*                 5975
000244  103402                          BLO     8$
000246  105066  000004                  CLRB    4(SP)                           ; NEXT.PACKET                   5978
000252  005366  000002          8$:     DEC     2(SP)                           ; COUNT                         5934
000256  001264                          BNE     1$
000260  005704          9$:     TST     R4                              ; INDEX                         5982
000262  002435                          BLT     11$
000264  105764  000000G                 TSTB    PKT.USE(R4)                     ; *(INDEX)                      5983
000270  002432                          BLT     11$
000272  010446                          MOV     R4,-(SP)                        ; INDEX,*                       5987
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)             ;                               5988
000320  152760  000001  000010G         BISB    #1,MSCP.PKT+10(R0)
000326  005000                          CLR     R0                              ;                               5989
000330  156600  000010                  BISB    10(SP),R0                       ; NEXT.PACKET,*
```

```
000334  005200                          INC     R0
000336  110037  000000G                 MOVB    R0,NEXT.PKT.USE
000342  120027  000014                  CMPB    R0,#14                  ; NEXT.PKT.USE,*        5991
000346  103402                          BLO     10$
000350  105037  000000G                 CLRB    NEXT.PKT.USE            ;                       5993
000354  022626                  10$:    CMP     (SP)+,(SP)+             ;                       5986
000356  010400                  11$:    MOV     R4,R0                   ; INDEX,*               5923
000360  062706  0C0006                  ADD     #6,SP                   ;                       5911
000364  000207                          RTS     PC

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


;   6000  1
;   6001  1
```

```
;    6002  1
;    6003  1
;    6004  1     global routine PUT_PKT (index) : novalue =
;    6005  1
;    6006  1     !+
;    6007  1     !       THE MSCP PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;    6008  1     !       ROUTINE.
;    6009  1     !
;    6010  1
;    6011  1
;    6012  2         begin
;    6013  2
;    6014  2
;    6015  2         local
;    6016  2             RING_ADDR : word,
;    6017  2             OWNER : word;
;    6018  2
;    6019  2         RING_ADDR = .DCT_ADDR [RR_BEG];                             ! ADDRESS IN FIRST RESPONSE RING
;    6020  2
;    6021  2         incr COUNT from 1 to (RRING_LEN + CRING_LEN) do             ! FOR EACH ADDRESS IN THE RINGS
;    6022  3             begin
;    6023  3
;    6024  3             if .MSCP_PKT [.index, PKT_LO] eqla ..RING_ADDR          ! IF ADDRESS MATCHES
;    6025  3
;    6026  3             then
;    6027  4                 begin
;    6028  4                 OWNER = .RING_ADDR + 2;                             ! ADDRESS OF OWNERSHIP WORD
;    6029  4                 .OWNER = ..OWNER and (not (ED_OWN)) and (not (ED_FLAG)) ;   ! GIVE OWNERSHIP TO HOST
;    6030  3                 end;
;    6031  3
;    6032  3
;    6033  3             RING_ADDR = .RING_ADDR + 4;                             ! LOOK AT NEXT PACKET ADDRESS IN RING
;    6034  2             end;
;    6035  2
;    6036  2
;    6037  2         PKT_USE [.index] = -1;
;    6038  2
;    6039  1         end;


                                        .SBTTL  PUT.PKT GLOBAL ROUTINES
000000  004137  000000G          PUT.PKT::
                                        JSR     R1,$SAVE4                       ;                                   6004
000004  013700  000000G                 MOV     DCT.ADDR,R0                     ;                                   6019
000010  016001  000004                  MOV     4(R0),R1                        ; *,RING.ADDR
000014  016602  000014                  MOV     14(SP),R2                       ; INDEX,*                           6024
000020  010246                          MOV     R2,-(SP)
000022  012746  000106                  MOV     #106,-(SP)
000026  004737  000000G                 JSR     PC,BL$MUL
000032  012704  000010                  MOV     #10,R4                          ; *,COUNT                           6021
000036  026011  000000G          1$:     CMP     MSCP.PKT(R0),(R1)               ; *,RING.ADDR                       6024
000042  001005                          BNE     2$
000044  012703  000002                  MOV     #2,R3                           ; *,OWNER                           6028
```

```
000050  060103                                  ADD     R1,R3                           ; RING.ADDR,OWNER
000052  042713  140000                          BIC     #140000,(R3)                    ; *,OWNER                       6029
000056  062701  000004                  2$:     ADD     #4,R1                           ; *,RING.ADDR                   6033
000062  005304                                  DEC     R4                              ; COUNT                         6021
000064  001364                                  BNE     1$                                                              6037
000066  112762  000377  000000G                MOVB    #377,PKT.USE(R2)                ;                               6012
000074  022626                                  CMP     (SP)+,(SP)+                     ;                               6004
000076  000207                                  RTS     PC                              ;

; Routine Size:  32 words,      Routine Base:  $CODE$ + 625$
; Maximum stack depth per invocation:  8 words


;   6040  1
;   6041  1
```

```
;  6042  1     routine PUTA_PKT (CTLR) : novalue =
;  6043  1
;  6044  1     !+
;  6045  1     !       THIS ROUTINE DEALLOCATES ALL MSCP PACKETS WHICH HAVE BEEN ALLOCATED
;  6046  1     !       TO A PARTICULAR CONTROLLER.
;  6047  1     !
;  6048  1     !       INPUTS:
;  6049  1     !               CTLR   CONTROLLER NUMBER
;  6050  1     !
;  6051  1
;  6052  1         incr COUNT from 0 to (PKT_CNT   1) do       ! FOR EACH ENTRY IN ALLOCATION TABLE
;  6053  1
;  6054  1             if .PKT_USE [.COUNT] eql .CTLR          ! IF PACKET IS ALLOCATED TO GIVEN CONTROLLER
;  6055  1             then
;  6056  1                 PKT_USE [.COUNT] = -1;              ! DEALLOCATE IT


                                        .SBTTL  PUTA.PKT GLOBAL ROUTINES
000000  010146                 PUTA.PKT:
                                        MOV     R1,-(SP)                ;                               6042
000002  005000                         CLR     R0                      ; COUNT                         6052
000004  116001  000000G        1$:     MOVB    PKT.USE(R0),R1          ; *(COUNT),*                    6054
000010  020166  000004                  CMP     R1,4(SP)                ; *,CTLR
000014  001003                          BNE     2$
000016  112760  000377  000000G        MOVB    #377,PKT.USE(R0)        ; *,*(COUNT)                    6056
000024  005200                 2$:     INC     R0                      ; COUNT                         6052
000026  020027  000013                  CMP     R0,#13                  ; COUNT,*
000032  003764                          BLE     1$
000034  012601                          MOV     (SP)+,R1                ;                               6042
000036  000207                          RTS     PC

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

```
;  6057  1     global routine GET_RETPKT (CTLR) =
;  6058  1
;  6059  1     !+
;  6060  1     !       THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
;  6061  1     !       FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;  6062  1     !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
;  6063  1     !       THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;  6064  1     !
;  6065  1     !       INPUTS:
;  6066  1     !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;  6067  1     !
;  6068  1
;  6069  2         begin
;  6070  2
;  6071  2         local
;  6072  2             index : signed word initial ( 1);       ! ASSUME NONE AVAILABLE
;  6073  2
;  6074  2         incr COUNT from 0 to (RP_CNT - 1) do        ! FOR EACH ENTRY IN TABLE
;  6075  2
;  6076  2             if .RP_USE [.COUNT] lss 0               ! IF FREE RETPKT IS FOUND
;  6077  2             then
;  6078  3                 begin
;  6079  3                 RP_USE [.COUNT] = .CTLR;            ! ALLOCATE RETURN PACKET TO CONTROLLER
;  6080  3                 index = .COUNT;
;  6081  3
;  6082  3                 incr J from 0 to (RP_LEN   1) do    ! ZERO OUT RETPKT
;  6083  3                     RETPKT [.COUNT, .J, 0, 16, 0] = 0;
;  6084  3
;  6085  3                 exitloop;                           ! DONE
;  6086  2                 end;
;  6087  2
;  6088  2             return .index;                          ! RETURN PACKET INDEX (OR  1) TO CALLER
;  6089  1             end;


                                        .SBTTL  GET.RETPKT GLOBAL ROUTINES
000000  004137  000000G         GET.RETPKT::
                                        JSR     R1,$SAVE4                       ;                       6057
000004  012703  177777                  MOV     #-1,R3                          ; *,INDEX               6069
000010  005001                          CLR     R1                              ; COUNT                 6074
000012  105761  000000G         1$:     TSTB    RP.USE(R1)                      ; *(COUNT)              6076
000016  002025                          BGE     3$
000020  116661  000014  000000G        MOVB    14(SP),RP.USE(R1)               ; CTLR,*(COUNT)         6079
000026  010103                          MOV     R1,R3                           ; COUNT,INDEX           6080
000030  010146                          MOV     R1,-(SP)                        ; COUNT,*               6083
000032  012746  000026                  MOV     #26, (SP)
000036  004737  000000G                 JSR     PC,BL$MUL
000042  022626                          CMP     (SP)+,(SP)+
000044  005002                          CLR     R2                              ; J                     6082
000046  010004                  2$:     MOV     R0,R4                           ;                       6083
000050  060204                          ADD     R2,R4                           ; J,*
000052  006304                          ASL     R4
000054  005064  000000G                 CLR     RETPKT(R4)
```

```
000060  005202                          INC     R2                      ; J                     6082
000062  020227  000025                  CMP     R2,#25                  ; J,*
000066  003767                          BLE     2$
000070  000404                          BR      4$                      ;                       6078
000072  005201                  3$:     INC     R1                      ; COUNT                 6074
000074  020127  000007                  CMP     R1,#7                   ; COUNT,*
000100  003744                          BLE     1$
000102  010300                  4$:     MOV     R3,R0                   ; INDEX,*               6069
000104  000207                          RTS     PC                      ;                       6057

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

```
;   6090  1     global routine PUT_RETPKT (index) : novalue =
;   6091  1
;   6092  1     !*
;   6093  1     !       THE RETURN PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;   6094  1     !       ROUTINE.
;   6095  1     !
;   6096  1
;   6097  1         RP USE [.index] =  1;


                                        .SBTTL  PUT.RETPKT GLOBAL ROUTINES
000000  016600  000002          PUT.RETPKT::
                                MOV     2(SP),R0                        ; INDEX,*                       6097
000004  112760  000377  000000G        MOVB    #377,RP.USE(R0)
000012  000207                  RTS     PC                              ;                               6090

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

```
;   6098  1
;   6099  1
;   6100  1     global routine GET_IO BUFF (ADDR) : novalue =
;   6101  1
;   6102  1
;   6103  1     !+
;   6104  1     !       THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
;   6105  1     !       POOL.
;   6106  1     !
;   6107  1     !       INPUTS:
;   6108  1     !               ADDR   ADDRESS TO STORE THE 2 WORD BUFFER DESCRIPTOR
;   6109  1     !
;   6110  1     !       IMPLICIT INPUTS:
;   6111  1     !               CCTLR   CURRENT CONTROLLER NUMBER
;   6112  1     !
;   6113  1     !       OUTPUTS:
;   6114  1     !               THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
;   6115  1     !               WORDS AT "ADDR" AND "ADDR + 2". OTHERWISE, A ZERO IS RETURNED
;   6116  1     !               AT "ADDR" IF NO BUFFERS ARE AVAILABLE.
;   6117  1     !
;   6118  1
;   6119  1
;   6120  1
;   6121  2         begin
;   6122  2         .ADDR = 0;                                          ! ASSUME FAILURE
;   6123  2
;   6124  2         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR   1) do   ! FOR EACH ENTRY IN BUFFER TABLE
;   6125  2
;   6126  2             if .BUFF_OWN [.COUNT] lss 0                     ! IF BUFFER IS FREE
;   6127  2
;   6128  2             then
;   6129  2
;   6130  3                 begin
;   6131  3                 BUFF_OWN [.COUNT] = .CCTLR;                 ! ALLOCATE BUFFER TO CONTROLLER
;   6132  3                 .ADDR = .BUFF_ADDR [.COUNT];                ! RETURN BUFFER DESCRIPTOR
;   6133  3                 exitloop;                                   ! DONE
;   6134  2                 end;
;   6135  2
;   6136  2
;   6137  1         end;                                                ! ROUTINE GET IO_BUFF


                                        .SBTTL  GET.IO.BUFF GLOBAL ROUTINES
000000  010146                    GET.IO.BUFF::
                                        MOV     R1,-(SP)                ;                       6100
000002  005076  000004                  CLR     @4(SP)                  ; ADDR                  6122
000006  005001                          CLR     R1                      ; COUNT                 6124
000010  105761  000000G         1$:     TSTB    BUFF.OWN(R1)            ; *(COUNT)              6126
000014  002011                          BGE     2$
000016  113761  000000G 000000G       MOVB    CCTLR,BUFF.OWN(R1)      ; *,*(COUNT)            6131
000024  010100                          MOV     R1,R0                   ; COUNT,*               6132
000026  006300                          ASL     R0
000030  016076  000000G 000004          MOV     BUFF.ADDR(R0),@4(SP)    ; *,ADDR
```

```
000036  000404                          BR      3$                      ;                               6130
000040  005201                  2$:     INC     R1                      ; COUNT                         6124
000042  020127 00000'                    CMP     R1,#7                   ; COUNT,#
000046  003760                          BLE     1$
000050  012601                  3$:     MOV     (SP)+,R1                ;                               6100
000052  000207                          RTS     PC

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


;   6138  1
;   6139  1
```

```
;    6140  1     global routine PUT IO BUFF (ADDR  : novalue =
;    6141  1
;    6142  1     !+
;    6143  1     !       THIS ROUTINE HANDLES THE DEALLOCATION OF AN I/O BUFFER, RETURNING IT
;    6144  1     !       TO THE BUFFER POOL
;    6145  1     !
;    6146  1     !       INPUTS:
;    6147  1     !               ADDR   ADDRESS OF THE 2-WORD BUFFER DESCRIPTOR TO BE
;    6148  1     !                      DEALLOCATED
;    6149  1     !
;    6150  1
;    6151  1         incr COUNT from 0 to (QIO_PER CTLR * MAX CTLR  1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;    6152  1
;    6153  1             if .BUFF_ADDR [.COUNT] eqla ..ADDR                     ! IF THIS IS THE BUFFER'S ENTRY
;    6154  1             then
;    6155  2                 begin
;    6156  2                 BUFF_OWN [.COUNT] =  1;                            ! DEALLOCATE BUFFER
;    6157  2                 exitloop;                                          ! DONE
;    6158  1                 end;


                                        .SBTTL  PUT.IO.BUFF GLOBAL ROUTINES
000000  010146                 PUT.IO.BUFF::
                                        MOV     R1,-(SP)                ;                                       6140
000002  005001                          CLR     R1                      ; COUNT                                 6151
000004  010100                 1$:     MOV     R1,R0                   ; COUNT,*                               6153
000006  006300                          ASL     R0
000010  026076  000000G 000004          CMP     BUFF.ADDR(R0),@4(SP)    ; *,ADDR
000016  001004                          BNE     2$
000020  112761  000377  000000C         MOVB    #377,BUFF.OWN(R1)       ; *,*(COUNT)                            6156
000026  000404                          BR      3$                      ;                                       6155
000030  005201                 2$:     INC     R1                      ; COUNT                                 6151
000032  020127  000007                  CMP     R1,#7                   ; COUNT,*
000036  003762                          BLE     1$
000040  012601                 3$:     MOV     (SP)+,R1                ;                                       6140
000042  000207                          RTS     PC

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

```
;  6159  1     global routine PUTA_BUFF : novalue =
;  6160  1
;  6161  1     !+
;  6162  1     !       THIS ROUTINE DEALLOCATES ALL I/O BUFFERS WHICH HAVE BEEN ALLOCATED TO
;  6163  1     !       THE CURRENT CONTROLLER (CCTLR).
;  6164  1     !-
;  6165  1
;  6166  1         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;  6167  1
;  6168  1             if .BUFF_OWN [.COUNT] eql .CCTLR                        ! IF THIS BUFFER ALLOCATED TO CURRENT CONTROLLER
;  6169  1             then
;  6170  1                 BUFF_OWN [.COUNT] = -1;                             ! DEALLOCATE IT


                                        .SBTTL  PUTA.BUFF GLOBAL ROUTINES
000000  010146                 PUTA.BUFF::
                                        MOV     R1,-(SP)                    ;                                       6159
000002  005000                         CLR     R0                          ; COUNT                                 6166
000004  116001  000000G        1$:     MOVB    BUFF.OWN(R0),R1             ; *(COUNT),*                            6168
000010  020137  000000G                CMP     R1,CCTLR
000014  001003                         BNE     2$
000016  112760  000377  000000G        MOVB    #377,BUFF.OWN(R0)           ; *,*(COUNT)                            6170
000024  005200                 2$:     INC     R0                          ; COUNT                                 6166
000026  020027  000007                 CMP     R0,#7                       ; COUNT,*
000032  003764                         BLE     1$
000034  012601                         MOV     (SP)+,R1                    ;                                       6159
000036  000207                         RTS     PC

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

```
;   6171  1     global routine OUT_IODQ =
;   6172  1
;   6173  1     !+
;   6174  1     !       THIS ROUTINE RETURNS TO THE CALLER THE NEXT RETPKT INDEX TO BE
;   6175  1     !       PROCESSED FROM THE I/O DONE QUEUE (IODQ). THE "OUT" POINTER TO THE
;   6176  1     !       QUEUE IS ALSO UPDATED.
;   6177  1     !
;   6178  1     !       INPUTS:
;   6179  1     !               NONE
;   6180  1     !
;   6181  1     !       OUTPUTS:
;   6182  1     !               THE INDEX OF THE NEXT RETPKT TO BE PROCESSED.
;   6183  1     !
;   6184  1
;   6185  2         begin
;   6186  2
;   6187  2         local
;   6188  2             index : word;
;   6189  2
;   6190  2         index = .IODQ [.IODQ_OUT];              ! GET NEXT RETPKT INDEX
;   6191  2         IODQ_OUT = .IODQ_OUT + 1;               ! ADVANCE "OUT" POINTER
;   6192  2
;   6193  2         if .IODQ_OUT gequ IODQ_LEN              ! IF BEYOND END OF QUEUE
;   6194  2         then
;   6195  2             IODQ_OUT = 0;                       ! SET POINTER TO BEGINNING OF QUEUE
;   6196  2
;   6197  2         return .index;                          ! RETURN INDEX TO CALLER
;   6198  1         end;


                                        .SBTTL  OUT.IODQ GLOBAL ROUTINES
000000  013700  000000G           OUT.IODQ::
                                        MOV     IODQ.OUT,R0             ;                       6190
000004  116000  000000G                   MOVB    IODQ(R0),R0             ; *,INDEX
000010  042700  177400                    BIC     #177400,R0              ; *,INDEX
000014  005237  000000G                   INC     IODQ.OUT                ;                       6191
000020  023727  000000G 000010            CMP     IODQ.OUT,#10            ;                       6193
000026  103402                            BLO     1$
000030  005037  000000G                   CLR     IODQ.OUT                ;                       6195
000034  000207                    1$:     RTS     PC                      ;                       6171

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

```
;   6199  1     global routine IN_IODQ (index) : novalue =
;   6200  1
;   6201  1     !+
;   6202  1     !       THIS ROUTINE INSERTS A RETURN PACKET INDEX INTO THE I/O DONE QUEUE, AND
;   6203  1     !       UPDATES THE IODQ_IN POINTER.
;   6204  1     !
;   6205  1
;   6206  1         if ((.IODQ_IN + 1) eql .IODQ_OUT) or
;   6207  2            (.IODQ_IN - (IODQ_LEN   1) eql .IODQ_OUT)
;   6208  1         then
;   6209  1             return
;   6210  1         else
;   6211  2             begin
;   6212  2             IODQ [.IODQ_IN] = .index;              ! LOAD INDEX INTO QUEUE
;   6213  2             IODQ_IN = .IODQ_IN + 1;                ! ADVANCE "IN" POINTER
;   6214  2
;   6215  2             if .IODQ_IN gequ IODQ_LEN              ! IF BEYOND END OF QUEUE
;   6216  2             then
;   6217  2                 IODQ_IN = 0;                       ! CYCLE BACK TO BEGINNING OF QUEUE
;   6218  2
;   6219  1             end;                                   ! IF IODQ IS NOT FULL

                                .SBTTL  IN.IODQ GLOBAL ROUTINES
000000  010146                IN.IODQ::
                                MOV     R1,-(SP)                        ;                       6199
000002  013701  000000G         MOV     IODQ.IN,R1                      ;                       6206
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                           ;                       6207
000022  162700  000007          SUB     #7,R0
000026  020037  000000G         CMP     R0,IODQ.OUT
000032  001413                  BEQ     1$                              ;                       6209
000034  116661  000004 000000G  MOVB    4(SP),IODQ(R1)                  ; INDEX,*               6212
000042  005237  000000G         INC     IODQ.IN                         ;                       6213
000046  023727  000000G 000010  CMP     IODQ.IN,#10                     ;                       6215
000054  103402                  BLO     1$
000056  005037  000000G         CLR     IODQ.IN                         ;                       6217
000062  012601                1$:       MOV     (SP)+,R1                ;                       6199
000064  000207                  RTS     PC

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

```
;    6220  1
;    6221  1
;    6222  1     global routine DROP CTLR (CTLR, REASON) : novalue =
;    6223  1
;    6224  1
;    6225  1
;    6226  1     !+
;    6227  1     !       THIS ROUTINE DROPS ALL UNITS ASSOCIATED WITH THE CONTROLLER DESIGNATED
;    6228  1     !       BY "CTLR". THE REASON FOR DROPPING THE DEVICE IS LOADED INTO THE DUR
;    6229  1     !       VECTOR FOR EACH ATTACHED UNIT. THIS DATA IS THEN USED BY THE DROP UNIT
;    6230  1     !       SECTION.
;    6231  1     !
;    6232  1
;    6233  1
;    6234  2         begin
;    6235  2
;    6236  2         local
;    6237  2             UNIT;
;    6238  2
;    6239  2         incr N from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT SIZE * OF UN) by UNIT SIZE do        ! FOR EACH UNIT
;    6240  2
;    6241  2             if .CST [.CTLR, .N + OF_DATA, D_PRES] eql PRESENT                                        ! IF CONFIGURED
;    6242  2             then
;    6243  3                 begin
;    6244  3                 UNIT = .CST [.CTLR, .N + OF_DATA, D UNIT];                                           ! DRS UNIT NUMBER
;    6245  3                 DUR [.UNIT] = .REASON;                                                               ! DROP REASON
;    6246  3                 DODU (.UNIT);                                                                        ! DROP UNIT
;    6247  2                 end;
;    6248  2
;    6249  1         end;


                                .SBTTL  DROP.CTLR GLOBAL ROUTINES
(00000  004137  000000G         DROP.CTLR::
                                JSR     R1,$SAVE3                       ;                                    6222
000004  016646  000014          MOV     14(SP),-(SP)                    ; CTLR,*                             6241
000010  012746  000053          MOV     #53,-(SP)
000014  004737  000000G         JSR     PC,BL$MUL
000020  010003                  MOV     R0,R3
000022  012702  000003          MOV     #3,R2                           ; *,N                                6239
000026  010300          1$:     MOV     R3,R0                           ;                                    6241
000030  060200                  ADD     R2,R0                           ; N,*
000032  006300                  ASL     R0
000034  032760  040000  000000G BIT     #40000,CST(R0)
000042  001412                  BEQ     2$
000044  016001  000000G         MOV     CST(R0),R1                      ; *,UNIT                             6244
000050  000301                  SWAB    R1                              ; UNIT
000052  042701  177760          BIC     #177760,R1                      ; *,UNIT
000056  116661  000016  000000G MOVB    16(SP),DUR(R1)                  ; REASON,*(UNIT)                     6245
000064  010100                  MOV     R1,R0                           ; UNIT,*                             6246
000066  104451                  TRAP    51
000070  062702  000012  2$:     ADD     #12,R2                          ; *,N                                6239
000074  020227  000041          CMP     R2,#41                          ; N,*
```

```
000100  003752                          BLE     1$
000102  022626                          CMP     (SP)+,(SP)+                     ;                       6234
000104  000207                          RTS     PC                              ;                       6222

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


;   6250  1
;   6251  1
```

```
;  6252  1     global routine DRV_CTLERR (CTLR) : novalue =
;  6253  1
;  6254  1     !+
;  6255  1     !       THIS ROUTINE IS CALLED BY DRV_TIMCHK AND FATAL_ERROR WHENEVER AN
;  6256  1     !       UNRECOVERABLE CONTROLLER ERROR HAS BEEN DETECTED. ITS PURPOSE IS TO
;  6257  1     !       CLEAN UP ALL CONTROLLER-RELATED DATA IN THE "DRIVER" PORTION OF THE
;  6258  1     !       PROGRAM. THIS INCLUDES MARKING THE CONTROLLER OFFLINE, CLEARING THE
;  6259  1     !       C-RING COUNT, AND DEALLOCATING MSCP PACKETS DESCRIBED IN THE RESPONSE
;  6260  1     !       RING.
;  6261  1     !
;  6262  1     !       INPUTS:
;  6263  1     !               CTLR    DYING CONTROLLER NUMBER
;  6264  1     !
;  6265  1
;  6266  2         begin
;  6267  2
;  6268  2         local
;  6269  2             D_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS);  ! CONTROLLER'S DCT ADDRESS
;  6270  2
;  6271  2         D_ADDR = DCT + (.CTLR * DCT_LEN * 2);                       ! GET CONTROLLER'S DCT ADDR
;  6272  2         D_ADDR [WORD0] = OFFLINE;                                   ! MARK DCT OFFLINE AND CLEAR CRING_CNT
;  6273  2         PUTA_PKT (.CTLR);                                           ! RELEASE ALL PACKETS ALLOCATED TO CONTROLLER
;  6274  2         DROP_CTLR (.CTLR, DU_CFATAL);                               ! DROP ALL UNITS ON THE CONTROLLER
;  6275  1         end;                                                        ! ROUTINE DRV_CTLERR


                                .SBTTL  DRV.CTLERR GLOBAL ROUTINES
000000  010146            DRV.CTLERR::
                                MOV     R1,-(SP)                    ;                       6252
000002  016601  000004          MOV     4(SP),R1                    ; CTLR,*                6271
000006  010146                  MOV     R1,-(SP)
000010  012746  000022          MOV     #22,-(SP)
000014  004737  0000000G        JSR     PC,BL$MUL
000020  062700  0000000G        ADD     #DCT,R0
000024  005010                  CLR     (R0)                        ; D.ADDR                6272
000026  010116                  MOV     R1,(SP)                     ;                       6273
000030  004737  006354'          JSR     PC,PUTA.PKT
000034  010116                  MOV     R1,(SP)                     ;                       6274
000036  012746  000006          MOV     #6,-(SP)
000042  004737  007042'          JSR     PC,DROP.CTLR
000046  062706  000006          ADD     #6,SP                       ;                       6266
000052  012601                  MOV     (SP)+,R1                    ;                       6252
000054  000207                  RTS     PC

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

```
;    6276  1     global routine SEND (index) =
;    6277  1
;    6278  1     !+
;    6279  1     !       IF THE CURRENT RDRX IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
;    6280  1     !       ROUTINE "SENDS" A COMMAND TO THE RDRX BY LOADING THE PACKET
;    6281  1     !       DESCRIPTOR OF AN MSCP PACKET INTO THE COMMAND RING AND READING THE
;    6282  1     !       DEVICE'S IP REGISTER.  IF THE
;    6283  1     !       CURRENT RDRX IN NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
;    6284  1     !       THE CALLER, AND NO ACTION IS TAKEN.
;    6285  1     !
;    6286  1     !       INPUTS:
;    6287  1     !               INDEX - INDEX OF MSCP PACKET CONTAINING THE COMMAND TO
;    6288  1     !                       BE SENT
;    6289  1     !
;    6290  1     !       IMPLICIT INPUTS:
;    6291  1     !               CCTLR   CURRENT CONTROLLER NUMBER
;    6292  1     !               DCT ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;    6293  1     !-
;    6294  1
;    6295  2         begin
;    6296  2
;    6297  2         local
;    6298  2             SLOT_ADDR,
;    6299  2             TEMP : word,
;    6300  2             CUR_PRIORITY : word;
;    6301  2
;    6302  2         if (.DCT_ADDR [CRING_CNT] lssu CRING_LEN) and                   ! IF CRING IS NOT FULL AND
;    6303  3            ((.DCT_ADDR [STAT] eql ONLINE) or                            !   IF DEVICE IS ONLINE OR
;    6304  3             (.MSCP_PKT [.index, OPCODE] eql OP_SCC))                    !   IT IS A SET-CTRL-CHAR COMMAND
;    6305  2         then
;    6306  2
;    6307  4             if (not ((.MSCP_PKT [.index, OPCODE] eql OP_ACC) or (.MSCP_PKT [.index, OPCODE] eql OP_ONL) or
;    6308  4                     (.MSCP_PKT [.index, OPCODE] eql OP_RD) or (.MSCP_PKT [.index, OPCODE] eql OP_SCC) or
;    6309  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_SDD) OR                  !ZZZ
;    6310  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_RCD) OR                  !ZZZ
;    6311  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_GDS) OR                  !ZZZ
;    6312  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ELP) OR                  !ZZZ
;    6313  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ABT) OR                  !ZZZ
;    6314  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ESP) OR                  !ZZZ
;    6315  3                     (.MSCP_PKT [.index, OPCODE] eql OP_WRT)))
;    6316  2             then
;    6317  3                 begin
;    6318  3                 PRINTF (D8M107, .MSCP_PKT [.index, OPCODE]);
;    6319  3                 return FAILURE;
;    6320  3                 end
;    6321  2             else
;    6322  3                 begin
;    6323  3
;    6324  3                 do
;    6325  3                     BREAK                                               ! LOOP TILL CREDIT BALANCE POSITIVE
;    6326  4                 until ((.MSCP_PKT [.index, CMD_TYPE] eql IMM_CMD) and
;    6327  3                        (.CREDIT_BAL gequ 1)) or
;    6328  3                       (.CREDIT_BAL gtru 1);
```

K16


```
;  6329  3
;  6330  3                MSCP_PKT [.index, CRN_LO] = (CRN_LOW = .CRN_LOW + 1);       ! ASSIGN CMD REF NUM
;  6331  3
;  6332  3                if .CRN_LOW eql 0
;  6333  3                then
;  6334  3                    CRN_HIGH = .CRN_HIGH + 1;                               ! CMD REF NUM (HIGH ORDER)
;  6335  3
;  6336  3                MSCP_PKT [.index, CRN_HI] = .CRN_HIGH;                      !
;  6337  3                SLOT_ADDR = .DCT_ADDR [CR_NEXT];                            ! ADDR OF NEXT COMMAND SLOT
;  6338  3
;  6339  3                do                                                          ! WAIT TILL NEXT SLOT HOST OWNED
;  6340  3                    BREAK                                                   !
;  6341  3                until ((.(.SLOT_ADDR + 2) and ED_OWN) eql 0);               !
;  6342  3
;  6343  3                GETPRI (CUR_PRIORITY);                                      ! NO INTERRUPTS WHILE POINTERS UPDATED
;  6344  3    !ZZZ        SETPRI (PRI04);                                             !
;  6345  3                SETPRI (.BRLEVEL);                                          !          ZZZ
;  6346  3
;  6347  3                .SLOT_ADDR = .MSCP_PKT [.index, PKT_LO];                    ! LOAD BUFF DESC (LO) INTO COMMAND SLOT
;  6348  3                SLOT_ADDR = .SLOT_ADDR + 2;                                 ! ADVANCE TO NEXT WORD
;  6349  3                .SLOT_ADDR = .MSCP_PKT [.index, PKT_HI];                    ! LOAD BUFF DESC (HI) INTO COMMAND SLOT
;  6350  3                .SLOT_ADDR = ..SLOT_ADDR and (not (ED_FLAG));               ! CLEAR INTERRUPT FLAG IN CASE SET
;  6351  3                .SLOT_ADDR = ..SLOT_ADDR or ED_OWN;                         ! GIVE OWNERSHIP TO CONTROLLER
;  6352  3                SLOT_ADDR = .SLOT_ADDR + 2;                                 ! ADVANCE TO NEXT COMMAND SLOT
;  6353  3
;  6354  3                if .SLOT_ADDR gtra .DCT_ADDR [CR_END]                       ! IF BEYOND END OF CRING
;  6355  3                then
;  6356  3                    SLOT_ADDR = .DCT_ADDR [CR_BEG];                         ! CYCLE BACK TO BEGINNING
;  6357  3
;  6358  3                DCT_ADDR [CR_NEXT] = .SLOT_ADDR;                            ! RESTORE CR_NEXT POINTER IN DCT
;  6359  3                DCT_ADDR [CRING_CNT] = .DCT_ADDR [CRING_CNT] + 1;           ! INCR # OF COMMANDS IN CRING
;  6360  4                IF (.MSCP_PKT [.INDEX, CONNID] EQL CID_MSCP)                !IF MSCP COMMAND                ZZZ
;  6361  3                THEN (CREDIT_BAL = .CREDIT_BAL - 1);                        !DECR CREDIT BALANCE            ZZZ
;  6362  3                TEMP = .RDRX_ADDR [RCIP, RC_ALL];                           ! READ IP TO FORCE PORT TO POLL
;  6363  3                SETPRI (.CUR_PRIORITY);                                     ! LOWER PRIORITY
;  6364  3                return SUCCESS;
;  6365  3                end
;  6366  3
;  6367  2            else
;  6368  2                return FAILURE;                                             ! IF DEVICE IS NOT ONLINE
;  6369  2
;  6370  1            end;                                                            ! ROUTINE END


                                        .SBTTL  SEND GLOBAL ROUTINES
000000  004137  000000G           SEND::  JSR     R1,$SAVE3                   ;                                   6276
000004  005746                          TST     -(SP)
000006  127727  000000G 000004           CMPB    @DCT.ADDR,#4                ;                                   6302
000014  103100                          BHIS    2$
000016  005777  000000G                 TST     @DCT.ADDR                   ;                                   6303
000022  100413                          BMI     1$
000024  016646  000014                  MOV     14(SP),-(SP)                ; INDEX,*                           6304
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$:         MOV     14(SP),-(SP)            ; INDEX,+                       6307
000056  012746  000106                          MOV     #106,-(SP)
000062  004737  000000G                        JSR     PC,BL$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                  ;                               6308
000120  001437                                  BEQ     3$
000122  020027  000004                          CMP     R0,#4
000126  001434                                  BEQ     3$
000130  020027  000005                          CMP     R0,#5                   ;                               6310
000134  001431                                  BEQ     3$
000136  020027  000001                          CMP     R0,#1                   ;                               6311
000142  001426                                  BEQ     3$
000144  020027  000003                          CMP     R0,#3                   ;                               6312
000150  001423                                  BEQ     3$
000152  020027  000006                          CMP     R0,#6                   ;                               6313
000156  001420                                  BEQ     3$
000160  020027  000002                          CMP     R0,#2                   ;                               6314
000164  001415                                  BEQ     3$
000166  020027  000042                          CMP     R0,#42                  ;                               6315
000172  001412                                  BEQ     3$
000174  010046                                  MOV     R0,-(SP)                ;                               6318
000176  012746  000000G                        MOV     #UBM107,-(SP)
000202  012746  000002                          MOV     #2,-(SP)
000206  010600                                  MOV     SP,R0                   ; SP,+
000210  104417                                  TRAP    17
000212  062706  000006                          ADD     #6,SP                   ;                               6317
000216  000504                      2$:         BR      10$                     ;                               6307
000220  104422                      3$:         TRAP    22                      ;                               6324
000222  105762  000004G                        TSTB    MSCP.PKT+4(R2)          ;                               6326
000226  001003                                  BNE     4$
000230  005737  000000G                        TST     CREDIT.BAL              ;                               6327
000234  001004                                  BNE     5$
000236  023727  000000G 000001      4$:         CMP     CREDIT.BAL,#1           ;                               6328
000244  101765                                  BLOS    3$
000246  013700  000000G              5$:         MOV     CRN.LOW,R0              ;                               6330
000252  005200                                  INC     R0
000254  010037  000000G                        MOV     R0,CRN.LOW
000260  010062  000012G                        MOV     R0,MSCP.PKT+12(R2)
000264  001002                                  BNE     6$                      ;                               6332
000266  005237  000000G                        INC     CRN.HIGH                ;                               6334
000272  013762  000000G 000014G      6$:         MOV     CRN.HIGH,MSCP.PKT+14(R2)  ;                             6336
000300  01370C  000000G                        MOV     DCT.ADDR,R0             ;                               6337
```

```
000304  016001  000020                          MOV     20(R0),R1                       ; *,SLOT.ADDR
000310  104422                          7$:     TRAP    22                              ;                       6339
000312  032761  100000  000002                  BIT     #-100000,2(R1)                  ; *,*(SLOT.ADDR)        6341
000320  001373                                  BNE     7$
000322  104440                                  TRAP    40                              ;                       6343
000324  010003                                  MOV     R0,R3                           ; *,CUR.PRIORITY
000326  013700  000000G                         MOV     BRLEVEL,R0                      ;                       6345
000332  104441                                  TRAP    41
000334  016221  000000G                         MOV     MSCP.PKT(R2),(R1)+              ; *,SLOT.ADDR           6347
000340  016211  000002G                         MOV     MSCP.PKT+2(R2),(R1)             ; *,SLOT.ADDR           6349
000344  042711  040000                          BIC     #40000,(R1)                     ; *,SLOT.ADDR           6350
000350  052721  100000                          BIS     #100000,(R1)+                   ; *,SLOT.ADDR           6351
000354  013700  000000G                         MOV     DCT.ADDR,R0                     ;                       6354
000360  020160  000012                          CMP     R1,12(R0)                       ; SLOT.ADDR,*
000364  101402                                  BLOS    8$
000366  016001  000010                          MOV     10(R0),R1                       ; *,SLOT.ADDR           6356
000372  010160  000020                  8$:     MOV     R1,20(R0)                       ; SLOT.ADDR,*           6358
000376  105210                                  INCB    (R0)                            ;                       6359
000400  105762  000011G                         TSTB    MSCP.PKT+11(R2)                 ;                       6360
000404  001002                                  BNE     9$
000406  005337  000000G                         DEC     CREDIT.BAL                      ;                       5361
000412  017716  000000G                  9$:     MOV     @RDRX.ADDR,(SP)                 ; *,RC.REG              6362
000416  010300                                  MOV     R3,R0                           ; CUR.PRIORITY,*        6363
000420  104441                                  TRAP    41
000422  012700  000001                          MOV     #1,R0                           ;                       6307
000426  000401                                  BR      11$                             ;                       6368
000430  005000                          10$:    CLR     R0
000432  005726                          11$:    TST     (SP)+                           ;                       6276
000434  000207                                  RTS     PC

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

```
;  6371  1     global routine WAIT : novalue =

;  6372  1
;  6373  1     !+
;  6374  1     !       THE PURPOSE OF THIS ROUTINE IS TO KILL TIME UNTIL AN RDRX INTERRUPT
;  6375  1     !       RESULTS IN A RETURN PACKET INDEX BEING DEPOSITED INTO THE I/O DONE
;  6376  1     !       QUEUE (IODQ).
;  6377  1     !
;  6378  1
;  6379  1         do
;  6380  1             BREAK                                   ! BREAK FOR ACT
;  6381  1         until .IODQ_IN neq .IODQ_OUT;


                                        .SBTTL  WAIT GLOBAL ROUTINES
000000  104422                         WAIT::
000000                                  1$:     TRAP    22                      ;                       6379
000002  023737  000000G 000000G                 CMP     IODQ.IN,IODQ.OUT        ;                       6381
000010  001773                                 BEQ     1$
000012  000207                                 RTS     PC                      ;                       6371

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


;  6382  1
```

```
;    6383  1
;    6384  1      GLOBAL ROUTINE MODULAS (LO_LIMIT, HI LIMIT) =                                   !ZZZ
;    6385  1                                                                                     !ZZZ
;    6386  1      !+ THE PURPOSE OF THIS ROUTINE IS TO GET A RANDOM NUMBER BETWEEN               !ZZZ
;    6387  1      !  THE LOW AND HIGH LIMITS. THIS SHOULD WORK FOR A 16 BIT WORD.               !ZZZ
;    6388  1      !  THE "MOD" FUNC ONLY WORKS ON 15 BITS.                                      !ZZZ
;    6389  1                                                                                     !ZZZ
;    6390  2      BEGIN                                                                          !ZZZ
;    6391  2      OWN     X : WORD;                   !VARIABLE FOR RANDOM WD TABLE              !ZZZ
;    6392  2      LOCAL   ANSWER : UNSIGNED WORD,     !FINAL ANSWER                              !ZZZ
;    6393  2              SAVESZ : UNSIGNED WORD,     !SAVES SIZE OF WINDOW                      !ZZZ
;    6394  2              SIZE : UNSIGNED WORD;       !SIZE OF WINDOW                            !ZZZ
;    6395  2                                                                                     !ZZZ
;    6396  2                                                                                     !ZZZ
;    6397  2      X = .X + 1;                                                                    !ZZZ
;    6398  2      IF .X GEQ RDM_LEN                                                              !ZZZ
;    6399  2      THEN X = 0;                 !KEEP ROTATING RANDOM NUMBERS USED                 !ZZZ
;    6400  2
                          !ZZZ
;    6401  2      SIZE = .HI_LIMIT - .LO_LIMIT;                                                  !ZZZ
;    6402  2      SAVESZ = .HI_LIMIT   .LO_LIMIT;                                                !ZZZ
;    6403  3      IF (.SIZE LEQU %O'077777')      !IF BIT 15 NOT SET                             !ZZZ
;    6404  3      THEN ANSWER = ((.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1))                  !ZZZ
;    6405  3                                      !ONLY 15 BIT WD, SO TAKE RANDOM SAMPLE         !ZZZ
;    6406  2      ELSE                            !16 BIT WD                                     !ZZZ
;    6407  3              BEGIN                                                                  !ZZZ
;    6408  3              SIZE = .SIZE ^ -1;      !MAKES SIZE A 15 BIT LENGTH, OR DIV BY 2       !ZZZ
;    6409  3              ANSWER = (.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1);                !ZZZ
;    6410  3                                      !GIVES 15 BIT RANDOM NUMBER                    !ZZZ
;    6411  3              ANSWER = .ANSWER ^ 1;   !BUILD UP TO REGULAR SIZE                      !ZZZ
;    6412  3              ANSWER = .ANSWER + (.RANDOM [.X + 1] AND 1);                           !ZZZ
;    6413  3                                      !RANDOMLY FILL BIT 0                           !ZZZ
;    6414  4              IF (.ANSWER GTRU SAVESZ) !ITS POSSIBLE TO BE 1 LARGER THAN SIZE        !ZZZ
;    6415  3              THEN ANSWER = .SAVESZ;  !SO CHECK.                                     !ZZZ
;    6416  2              END;                                                                   !ZZZ
;    6417  2      RETURN .ANSWER;
;    6418  1      END;                            !END MODULAS ROUTINE                           !ZZZ


007700                                  X:      .BLKW   1


                                        .SBTTL  MODULAS GLOBAL ROUTINES
000000  004137  000000G                 MODULAS::
                                        JSR     R1,$SAVE2                       ;                       6384
000004  005746                          TST     -(SP)
000006  005237  007700'                 INC     X                               ;                       6397
000012  023727  007700'  000020         CMP     X,#20                           ;                       6398
000020  002402                          BLT     1$
000022  005037  007700'                 CLR     X                               ;                       6399
000026  016600  000012                  1$:     MOV     12(SP),R0               ; HI.LIMIT,*            6401
000032  166600  000014                  SUB     14(SP),R0               ; LO.LIMIT,*
000036  010001                          MOV     R0,R1                   ; *,SIZE
000040  010016                          MOV     R0,(SP)                 ; *,SAVESZ                      6402
```

```
000042  013700  007700                    MOV     X,R0                            ;                               6404
000046  006300                            ASL     R0
000050  020127  077777                    CMP     R1,#77777                       ; SIZE,*                        6403
000054  101011                            BHI     2$
000056  016046  000000G                   MOV     RANDOM(R0),-(SP)                ;                               6404
000062  042716  100000                    BIC     #100000,(SP)
000066  010146                            MOV     R1,-(SP)                        ; SIZE,*
000070  005216                            INC     (SP)
000072  004737  000000G                   JSR     PC,BL$MOD
000076  000431                            BR      3$                              ;                               6403
000100  006201                    2$:     ASR     R1                              ; SIZE                          6408
000102  016046  000000G                   MOV     RANDOM(R0),-(SP)                ;                               6409
000106  042716  100000                    BIC     #100000,(SP)
000112  010146                            MOV     R1,-(SP)                        ; SIZE,*
000114  005216                            INC     (SP)
000116  004737  000000G                   JSR     PC,BL$MOD
000122  006300                            ASL     R0                              ; ANSWER                        6411
000124  013701  007700                    MOV     X,R1                            ;                               6412
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                           ;                               6414
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                 6415
000162  062706  000006            3$:     ADD     #6,SP                           ;                               6384
000166  000207                            RTS     PC

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

```
;    6419  1     %sbttl 'ERROR MESSAGE SUBROUTINES
;    6420  1
;    6421  1     routine EMS_SA : novalue =
;    6422  1
;    6423  1     !+
;    6424  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SA_REG" WHICH CONTAINS
;    6425  1     !       THE CONTENTS OF THE SA REGISTER.
;    6426  1     !-
;    6427  1
;    6428  2         begin
;    6429  2
;    6430  2         if .SA_REG eql %o'177777                                    ! IF CONTROLLER TIME-OUT
;    6431  2         then
;    6432  3             begin
;    6433  3             PRINTX (CRLF);
;    6434  3             PRINTX (ASTERISK);
;    6435  3             PRINTX (.CNTR_ERR [0]);
;    6436  3             end
;    6437  2         else
;    6438  2
;    6439  2             if (.SA_REG and %o'003777') lequ 22                     ! IF GENERIC CONTROLLER ERROR
;    6440  2             then
;    6441  3                 begin
;    6442  3                 PRINTX (CRLF);
;    6443  3                 PRINTX (ASTERISK);
;    6444  3                 PRINTX (.CNTR_ERR [.SA_REG and %o'003777']);
;    6445  3                 end
;    6446  2             else
;    6447  2
;    6448  2                 if ((.SA_REG and %o'003777') - 400) lequ 6          ! IF RDRX SPECIFIC CONTROLLER ERROR
;    6449  2                 then
;    6450  3                     begin
;    6451  3                     PRINTX (CRLF);
;    6452  3                     PRINTX (ASTERISK);
;    6453  3                     PRINTX (.RDRX_ERR [(.SA_REG and %o'003777') - 400]);
;    6454  3                     end
;    6455  2                 else
;    6456  2                     PRINTX (EX_SA, .SA_REG);                       ! JUST PRINT CONTENTS OF SA
;    6457  2
;    6458  2         EMS_TIM ();                                                 ! TIME
;    6459  1         end;


                                        .SBTTL  EMS.SA ERROR MESSAGE SUBROUTINES
000000  010146                  EMS.SA: MOV     R1,-(SP)                        ;                                       6421
000002  013701  000000G                 MOV     SA.REG,R1                       ;                                       6430
000006  020127  1777 /                  CMP     R1,#-1
000012  001023                          BNE     1$
000014  012746  000000G                 MOV     #CRLF,-(SP)                     ;                                       6433
000020  012746  000001                  MOV     #1,-(SP)
000024  010600                          MOV     SP,R0                           ; SP,*
000026  104415                          TRAP    15
000030  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                                       6434
```

```
000034  012746  000001                        MOV     #1, (SP)
000040  010600                                MOV     SP,R0                   ; SP,*
000042  104415                                TRAP    15
000044  013716  000000G                       MOV     CNTR.ERR,(SP)           ;                       6435
000050  012746  000001                        MOV     #1,-(SP)
000054  010600                                MOV     SP,R0                   ; SP,*
000056  104415                                TRAP    15
000060  000475                                BR      3$                      ;                       6432
000062  010100                        1$:     MOV     R1,R0                   ;                       6439
000064  042700  174000                        BIC     #174000,R0
000070  020027  000026                        CMP     R0,#26
000074  101030                                BHI     2$
000076  012746  000000G                       MOV     #CRLF, (SP)             ;                       6442
000102  012746  000001                        MOV     #1, (SP)
000106  010600                                MOV     SP,R0                   ; SP,*
000110  104415                                TRAP    15
000112  012716  000000G                       MOV     #ASTERISK,(SP)          ;                       6443
000116  012746  000001                        MOV     #1,-(SP)
000122  010600                                MOV     SP,R0                   ; SP,*
000124  104415                                TRAP    15
000126  013700  000000G                       MOV     SA.REG,R0               ;                       5444
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$                      ;                       6441
000156  010100                        2$:     MOV     R1,R0                   ;                       6448
000160  042700  174000                        BIC     #174000,R0
000164  162700  000620                        SUB     #620,R0
000170  020027  000006                        CMP     R0,#6
000174  101031                                BHI     4$
000176  012746  000000G                       MOV     #CRLF,-(SP)             ;                       6451
000202  012746  000001                        MOV     #1,-(SP)
000206  010600                                MOV     SP,R0                   ; SP,*
000210  104415                                TRAP    15
000212  012716  000000G                       MOV     #ASTERISK,(SP)          ;                       6452
000216  012746  000001                        MOV     #1,-(SP)
000222  010600                                MOV     SP,R0                   ; SP,*
000224  104415                                TRAP    15
000226  013700  000000G                       MOV     SA.REG,R0               ;                       6453
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$:     TST     (SP)+                   ;                       6450
000256  000407                                BR      5$                      ;                       6448
000260  010146                        4$:     MOV     R1,-(SP)                ;                       6456
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              ;                       6458
000302  062706  000006                  ADD     #6,SP                   ;                       6428
000306  012601                          MOV     (SP)+,R1                ;                       6421
000310  000207                          RTS     PC

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

```
;  6460  1     routine EMS_SBC : novalue =
;  6461  1
;  6462  1     !+
;  6463  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SB_CODE" (SUB CODE) IF
;  6464  1     !       EITHER THE STATUS CODE (ST_CODE) OR THE SUB-CODE IS NON-ZERO. (A
;  6465  1     !       NON-ZERO SUB-CODE ALWAYS HAS SIGNIFICANCE, WHEREAS A ZERO SUB CODE ONLY
;  6466  1     !       HAS MEANING WITH A NON-ZERO STATUS CODE).
;  6467  1     !
;  6468  1
;  6469  2         begin
;  6470  2
;  6471  2         if (.ST_CODE or .SB_CODE) neq 0                                 ! PRINT SUB CODE ONLY ON ERROR
;  6472  2         then
;  6473  3             begin
;  6474  3             PRINTX (EX_SB);                                             ! SUB-CODE :
;  6475  3
;  6476  3             case .ST_CODE from ST_SUC to ST_DRV of
;  6477  3                 set
;  6478  3
;  6479  3                 [ST_SUC]:           if .SB_CODE lequ 16                 ! SUCCESS SUB-CODES
;  6480  3                                     then
;  6481  3                                         PRINTX (.TBL_SUC [.SB CODE]);
;  6482  3
;  6483  3                 [ST CMD]:           PRINTX (EX_SBO, .SB_CODE / 8);      ! INVALID COMMAND
;  6484  3
;  6485  3                 [ST_ABO]:           ;                                   ! COMMAND ABORTED
;  6486  3
;  6487  3                 [ST_OFL]:           if .SB_CODE lequ 8                  ! UNIT OFFLINE
;  6488  3                                     then
;  6489  3                                         PRINTX (.TBL_OFL [.SB_CODE]);
;  6490  3
;  6491  3                 [ST AVL]:           ;                                   ! UNIT AVAILABLE
;  6492  3
;  6493  3                 [ST_MFE]:           if .SB_CODE lequ 10                 ! MEDIA FORMAT ERROR
;  6494  3                                     then
;  6495  3                                         PRINTX (.TBL_MFE [.SB_CODE]);
;  6496  3
;  6497  3                 [ST_WPT]:           if (.SB_CODE / 128) lequ 2          ! WRITE PROTECTED
;  6498  3                                     then
;  6499  3                                         PRINTX (.TBL_WPT [(.SB_CODE / 128)]);
;  6500  3
;  6501  3                 [ST_CMP]:           ;                                   ! COMPARE ERROR
;  6502  3
;  6503  3                 [ST_DAT]:           if .SB_CODE lequ 15                 ! DATA ERROR
;  6504  3                                     then
;  6505  3                                         PRINTX (.TBL_DAT [.SB_CODE]);
;  6506  3
;  6507  3                 [ST HST]:           if .SB_CODE lequ 4                  ! HOST ACCESS ERROR
;  6508  3                                     then
;  6509  3                                         PRINTX (.TBL_HST [.SB_CODE]);
;  6510  3
;  6511  3                 [ST_CNT]:           if .SB_CODE lequ 3                  ! CONTROLLER ERROR
;  6512  3                                     then
```

```
;  6513  3                                    PRINTX (.TBL_CNT [.SB CODE]);
;  6514  3
;  6515  3                [ST_DRV]:           if .SB CODE lequ 8                  ! DRIVE ERROR
;  6516  3                                    then
;  6517  3                                        PRINTX (.TBL DRV [.SB_CODE]);
;  6518  3
;  6519  3                [outrange]:         PRINTX (EX_SBO, .SB CODE);          ! JUST PRINT SUB-CODE IF NO MATCH
;  6520  3                tes;
;  6521  3
;  6522  2            end;
;  6523  2
;  6524  1        end;


                                    .SBTTL  EMS.SBC ERROR MESSAGE SUBROUTINES
000000  013700  000000G      EMS.SBC:MOV     ST.CODE,R0                          ;                                   6471
000004  053700  000000G              BIS     SB.CODE,R0
000010  001001                       BNE     1$
000012  000207                       RTS     PC
000014  012746  000000G      1$:     MOV     #EX.SB,-(SP)                        ;                                   6474
000020  012746  000001               MOV     #1,-(SP)
000024  010600                       MOV     SP,R0                               ; SP,*
000026  104415                       TRAP    15
000030  013700  000000G              MOV     ST.CODE,R0                          ;                                   6476
000034  020027  000013               CMP     R0,#13
000040  101003                       BHI     3$
000042  006300                       ASL     R0
000044  066007  000000'              ADD     P.AAA(R0),PC                        ; Case dispatch
000050  013716  000000G      3$:     MOV     SB.CODE,(SP)                        ;                                   6519
000054  012746  000000G              MOV     #EX.SBO,-(SP)
000060  012746  000002               MOV     #2,-(SP)
000064  010600                       MOV     SP,R0                               ; SP,*
000066  104415                       TRAP    15
000070  022626                       CMP     (SP)+,(SP)+
000072  000435                       BR      6$                                  ;                                   6476
000074  023727  000000G 000020 4$:  CMP     SB.CODE,#20                         ;                                   6479
000102  101165                       BHI     14$
000104  013700  000000G              MOV     SB.CODE,R0                          ;                                   6481
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$:     MOV     SB.CODE,(SP)                        ;                                   6483
000134  012746  000010               MOV     #10,-(SP)
000140  004737  000000G              JSR     PC,BL$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$:     BR      16$                             ;                               6476
000170  023727  000000G 000010        7$:     CMP     SB.CODE,#10                     ;                               6487
000176  101142                                  BHI     16$
000200  013700  000000G                        MOV     SB.CODE,R0                      ;                               6489
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$:     CMP     SB.CODE,#12                     ;                               6493
000232  101124                                  BHI     16$
000234  013700  000000G                        MOV     SB.CODE,R0                      ;                               6495
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$:     MOV     SB.CODE,(SP)                    ;                               6497
000264  012746  000200                        MOV     #200,-(SP)
000270  004737  000000G                        JSR     PC,BL$DIV
000274  005726                                  TST     (SP)+
000276  020027  000002                        CMP     R0,#2
000302  101100                                  BHI     16$
000304  006300                                  ASL     R0                              ;                               6499
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$:    CMP     SB.CODE,#17                     ;                               6503
000332  101064                                  BHI     16$
000334  013700  000000G                        MOV     SB.CODE,R0                      ;                               6505
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$:    CMP     SB.CODE,#4                      ;                               6507
000366  101046                                  BHI     16$
000370  013700  000000G                        MOV     SB.CODE,R0                      ;                               6509
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$:    CMP     SB.CODE,#3                      ;                               6511
000422  101030                                  BHI     16$
000424  013700  000000G                        MOV     SB.CODE,R0                      ;                               6513
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$:    CMP     SB.CODE,#10             ;                       6515
000456  101012                        14$:    BHI     16$
000460  013700  000000G                       MOV     SB.CODE,R0              ;                       6517
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)+             ;                       6473
000506  000207                                RTS     PC                      ;                       6460

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


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:                                          ; CASE Table for EMS.SBC+0044   6476
000000  000024                 2$:     .WORD   24                              ; [4$]
000002  000060                         .WORD   60                              ; [5$]
000004  000434                         .WORD   434                             ; [16$]
000006  000120                         .WORD   120                             ; [7$]
000010  000434                         .WORD   434                             ; [16$]
000012  000154                         .WORD   154                             ; [8$]
000014  000210                         .WORD   210                             ; [9$]
000016  000434                         .WORD   434                             ; [16$]
000020  000254                         .WORD   254                             ; [10$]
000022  000310                         .WORD   310                             ; [11$]
000024  000344                         .WORD   344                             ; [12$]
000026  000400                         .WORD   400                             ; [13$]
```

```
;   6525  1     routine EMS_CMD : novalue =
;   6526  1
;   6527  1     !+
;   6528  1     !       THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
;   6529  1     !       PRESENT) OF THE CURRENT RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
;   6530  1     !       INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
;   6531  1     !
;   6532  1     !       IMPLICIT INPUTS:
;   6533  1     !               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
;   6534  1     !
;   6535  1
;   6536  2         begin
;   6537  2         PRINTX (EX_CMD);                                        ! "COMMAND: '
;   6538  2
;   6539  2         selectoneu (.RP_ADDR [ENDCOD] and OP_MSK) of
;   6540  2             set
;   6541  2
;   6542  2             [OP_ONL]:       PRINTX (EX_ONL);                    ! ONLINE
;   6543  2
;   6544  2             [OP_ACC]:       PRINTX (EX_ACC);                    ! ACCESS
;   6545  2
;   6546  3             [OP_RD]:        begin
;   6547  3                             PRINTX (EX_RD);                     ! READ
;   6548  3
;   6549  3                             if .RP_ADDR [CMDMOD] neq 0
;   6550  3                             then
;   6551  3                                 PRINTX (EX_CMP);                ! COMPARE
;   6552  3
;   6553  2                             end;
;   6554  2
;   6555  3             [OP_WRT]:       begin
;   6556  3                             PRINTX (EX_WRT);                    ! WRITE
;   6557  3
;   6558  3                             if .RP_ADDR [CMDMOD] neq 0
;   6559  3                             then
;   6560  3                                 PRINTX (EX_CMP);                ! COMPARE
;   6561  3
;   6562  2                             end;
;   6563  2
;   6564  2             [otherwise]:    PRINTX (EX_OP, .RP_ADDR [ENDCOD]);  ! ENDCODE VALUE IF NO MATCH
;   6565  2             tes;
;   6566  2
;   6567  1         end;                                                    ! ROUTINE EMS_CMD


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

000000  004137  000000G  EMS.CMD:JSR     R1,$SAVE2                       ;                       6525
000004  012746  000000G          MOV     #EX.CMD,-(SP)                   ;                       6537
000010  012746  000001           MOV     #1,-(SP)
000014  010600                   MOV     SP,R0                           ; SP,*
000016  104415                   TRAP    15
```

```
000020  013702  000000G             MOV     RP.ADDR,R2                      ;                       6539
000024  116201  000014              MOVB    14(R2),R1
000030  042701  177600              BIC     #177600,R1
000034  020127  000011              CMP     R1,#11                          ;                       6542
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$:     CMP     R1,#20                          ;                       6544
000064  001007                      BNE     2$
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$:     CMP     R1,#41                          ;                       6546
000110  001022                      BNE     3$
000112  012716  000000G             MOV     #EX.RD,(SP)                     ;                       6547
000116  012746  000001              MOV     #1,-(SP)
000122  010600                      MOV     SP,R0                           ; SP,*
000124  104415                      TRAP    15
000126  013700  000000G             MOV     RP.ADDR,R0                      ;                       6549
000132  005760  000012              TST     12(R0)
000136  001434                      BEQ     5$
000140  012716  000000G             MOV     #EX.CMP,(SP)                    ;                       6551
000144  012746  000001              MOV     #1,-(SP)
000150  010600                      MOV     SP,R0                           ; SP,*
000152  104415                      TRAP    15
000154  000424                      BR      4$
000156  020127  000042      3$:     CMP     R1,#42                          ;                       6555
000162  001024                      BNE     6$
000164  012716  000000G             MOV     #EX.WRT,(SP)                    ;                       6556
000170  012746  000001              MOV     #1,-(SP)
000174  010600                      MOV     SP,R0                           ; SP,*
000176  104415                      TRAP    15
000200  013700  000000G             MOV     RP.ADDR,R0                      .                       6558
000204  005760  000012              TST     12(R0)
000210  001407                      BEQ     5$
000212  012716  000000G             MOV     #EX.CMP,(SP)                    ;                       6560
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)+                           ;                       6555
000232  000412                      BR      7$                              .                       6539
000234  005016              6$:     CLR     (SP)                            ;                       6564
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)+                     ;                       6536
000262  000207                          RTS     PC                              ;                       6525

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

```
;   6568  1     GLOBAL ROUTINE EMS_DBN : NOVALUE =                                          !ZZZ
;   6569  1     !+                                                                         !ZZZ
;   6570  1     !       THIS ROUTINE PRINTS THE PRESENT DBN                                !ZZZ
;   6571  1     !                                                                          !ZZZ
;   6572  1     !       IMPLICIT INPUTS:                                                   !ZZZ
;   6573  1     !               CST_ADDR - ADDRESS OF CONTROLLER STATUS TABLE              !ZZZ
;   6574  1     !                                                                          !ZZZ
;   6575  1                                                                                !ZZZ
;   6576  2     BEGIN                                                                      !ZZZ
;   6577  2     PRINTB (XX13, .CDISK);                          !"DISK XXX"                !ZZZ
; P 6578  2     PRINTB (XX23, .CST_ADDR [.CUOFF * OF_DBN, D_DBN], .CST_ADDR                !ZZZ
;   6579  2                            [.CUOFF * OF_DBN, D_DBN]);      !"DBN: XXXXXX."  !ZZZ
;   6580  2     PRINTB (XX32, .S_DUPPKT - 2);                   !PRINT BYTE COUNT          !ZZZ
;   6581  2     PRINTB (XX33, .S_PATTERN);                      !PRINT THE PATTERN         !ZZZ
;   6582  2     PRINTB (XX34, .(DUPPKT + .S_DUPPKT), .(DUPPKT + .S_DUPPKT));    !PRINT THE WORD READ    !ZZZ
;   6583  2     EMS_BLK (DUPPKT+2, 256);                        !PRINT WHOLE BLOCK READ    !ZZZ
;   6584  1     END;                                            !IN OCTAL                  !ZZZ


                                        .SBTTL  EMS.DBN ERROR MESSAGE SUBROUTINES
000000  013746  000000G          EMS.DBN::
                                        MOV     CDISK,-(SP)                     ;                                       6577
000004  012746  000000G                 MOV     #XX13, (SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,R0
000016  104414                          TRAP    14
000020  013700  000000G                 MOV     CUOFF,R0                        ;                                       6579
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,R0
000060  104414                          TRAP    14
000062  013716  000000G                 MOV     S.DUPPKT,(SP)                   ;                                       6580
000066  162716  000002                  SUB     #2,(SP)
000072  012746  000000G                 MOV     #XX32,-(SP)
000076  012746  000002                  MOV     #2,-(SP)
000102  010600                          MOV     SP,R0                           ; SP,R0
000104  104414                          TRAP    14
000106  013716  000000G                 MOV     S.PATTERN,(SP)                  ;                                       6581
000112  012746  000000G                 MOV     #XX33,-(SP)
000116  012746  000002                  MOV     #2,-(SP)
000122  010600                          MOV     SP,R0                           ; SP,R0
000124  104414                          TRAP    14
000126  013700  000000G                 MOV     S.DUPPKT,R0                     ;                                       6582
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,R0
000152  104414                          TRAP    14
000154  012716  000002G                 MOV     #OUPPKT+2,(SP)                  ;                               6583
000160  012746  000400                  MOV     #400,-(SP)
000164  004737  000000V                 JSR     PC,EMS.BLK
000170  062706  000034                  ADD     #34,SP                          ;                               6576
000174  000207                          RTS     PC                              ;                               6568

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


;   6585  1
;   6586  1
```

```
;    6587  1
;    6588  1      GLOBAL ROUTINE EMS_BLK (ADDR, LENGTH) : NOVALUE =                        !ZZZ
;    6589  1                                                                              !ZZZ
;    6590  1      !+                                                                      !ZZZ
;    6591  1      !       THIS ROUTINE WILL PRINTX A BLOCK OF MEMORY, WHICH IS 'LENGTH    !ZZZ
;    6592  1      !       WORDS LONG STARTING AT ADDRESS 'ADDR'. PRINTING IS DONE IN OCTAL!ZZZ
;    6593  1      !       8 WDS TO A LINE.                                                !ZZZ
;    6594  1      !-                                                                      !ZZZ
;    6595  1                                                                              !ZZZ
;    6596  2      BEGIN                                                                   !ZZZ
;    6597  2      LITERAL                                                                 !ZZZ
;    6598  2              MASK = %o'7';                                                   !ZZZ
;    6599  2                                                                              !ZZZ
;    6600  2      PRINTX (CRLF);                                                          !ZZZ
;    6601  2              INCR COUNT FROM 1 TO .LENGTH DO          !FOR EACH WD TO PRINT  !ZZZ
;    6602  3                      BEGIN                                                   !ZZZ
;    6603  3                      IF ((.COUNT - 1) AND MASK) EQL 0 !IF START OF NEW LINE  !ZZZ
;    6604  3                      THEN                                                    !ZZZ
;    6605  3                              PRINTX (SPACE4);          !PRINT 4 BLANKS       !ZZZ
;    6606  3                                                                              !ZZZ
;    6607  3                      PRINTX (EX_WRD, ..ADDR);          !PRINTX A WORD        !ZZZ
;    6608  3                      ADDR = .ADDR +2;                  !TO NEXT ADDRESS      !ZZZ
;    6609  3                                                                              !ZZZ
;    6610  4                      IF (((.COUNT AND MASK) EQL 0) OR !END OF LINE OR        !ZZZ
;    6611  4                          (.COUNT EQL .LENGTH))        !WHEN DONE            !ZZZ
;    6612  3                      THEN                                                    !ZZZ
;    6613  3                              PRINTX (CRLF);            !PRINT CR LF          !ZZZ
;    6614  2                      END;                                                    !ZZZ
;    6615  1              END;                                                            !ZZZ


                                                .SBTTL  EMS.BLK ERROR MESSAGE SUBROUTINES
000000  010146                        EMS.BLK::
                                                MOV     R1,-(SP)                        ;                       6588
000002  012746  000000G                         MOV     #CRLF,-(SP)                     ;                       6600
000006  012746  000001                          MOV     #1,-(SP)
000012  010600                                  MOV     SP,R0                           ; SP,*
000014  104415                                  TRAP    15
000016  005001                                  CLR     R1                              ; COUNT                 6601
000020  000445                                  BR      5$
000022  010100                        1$:       MOV     R1,R0                           ; COUNT,*               6603
000024  005300                                  DEC     R0
000026  032700  000007                          BIT     #7,R0
000032  001007                                  BNE     2$
000034  012716  000000G                         MOV     #SPACE4,(SP)                    ;                       6605
000040  012746  000001                          MOV     #1,-(SP)
000044  010600                                  MOV     SP,R0                           ; SP,*
000046  104415                                  TRAP    15
000050  005726                                  TST     (SP)+
000052  017616  000C12                2$:       MOV     @12(SP),(SP)                    ; ADDR,*                6607
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                6608
000100  032701  000007                          BIT     #7,R1                   ; *,COUNT               6610
000104  001403                                  BEQ     3$
000106  020166  000014                          CMP     R1,14(SP)               ; COUNT,LENGTH          6611
000112  001007                                  BNE     4$
000114  012716  000000G                 3$:     MOV     #CRLF,(SP)              ;                       6613
000120  012746  000001                          MOV     #1,-(SP)
000124  010600                                  MOV     SP,R0                   ; SP,*
000126  104415                                  TRAP    15
000130  005726                                  TST     (SP)+
000132  022626                          4$:     CMP     (SP)+,(SP)+             ;                       6602
000134  005201                          5$:     INC     R1                      ; COUNT                 6601
000136  020166  000010                          CMP     R1,10(SP)               ; COUNT,LENGTH
000142  003727                                  BLE     1$
000144  022626                                  CMP     (SP)+,(SP)+             ;                       6596
000146  012601                                  MOV     (SP)+,R1                ;                       6588
000150  000207                                  RTS     PC

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


;   6616  1
;   6617  1
```

```
;   6618  1     routine EMS_LBN : novalue =
;   6619  1
;   6620  1     !+
;   6621  1     !       THIS ROUTINE PRINTS (EXTENDED) ONE OF TWO BLOCK NUMBERS APPEARING IN
;   6622  1     !       THE CURRENT RETURN PACKET. NORMALLY, THE LBN FIELD IS PRINTED; THIS
;   6623  1     !       FIELD WAS COPIED INTO THE RETURN PACKET FROM THE ASSOCIATED COMMAND
;   6624  1     !       PACKET. HOWEVER, IF THE "FLAGS" FIELD OF THE CURRENT RETURN PACKET
;   6625  1     !       INDICATES "BAD BLOCK REPORTED", THEN THE "FIRST BAD BLOCK" FIELD IS
;   6626  1     !       PRINTED.
;   6627  1     !
;   6628  1     !       IMPLICIT INPUTS:
;   6629  1     !               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
;   6630  1     !
;   6631  1
;   6632  2         begin
;   6633  2
;   6634  2         if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and              ! IF NO BAD BLOCK FOUND
;   6635  3            (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;   6636  2         then
;   6637  2             PRINTX (EX_LBN, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);
;   6638  2
;   6639  2         if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and              ! IF BAD BLOCKS FOUND AND REPLACED
;   6640  3            (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;   6641  2         then
;   6642  2             PRINTX (EX_BBU, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;   6643  2
;   6644  2         if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                  ! IF HOST REPLACEABLE BAD BLOCK FOUND
;   6645  3            (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;   6646  2         then
;   6647  2             PRINTX (EX_BB, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;   6648  2
;   6649  2         if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                  ! IF MORE THAN 1 HOST REPLACEABLE BAD BLOCK FOUND
;   6650  3            (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;   6651  2         then
;   6652  2             PRINTX (EX BB1, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;   6653  1         end;


                                        .SBTTL  EMS.LBN ERROR MESSAGE SUBROUTINES
000000  013700  000000G          EMS.LBN:MOV     RP.ADDR,R0                  ;                                   6634
000004  105760  000015                  TSTB    15(R0)
000010  100417                          BMI     1$
000012  132760  000100  000015          BITB    #100,15(R0)                 ;                                   6635
000020  001013                          BNE     1$
000022  016046  000050                  MOV     50(R0),-(SP)                ;                                   6637
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                  ;                                   6639
000054  105760  000015                  TSTB    15(R0)
```

```
000060  100417                          BMI     2$
000062  132760  000100  000015          BITB    #100,15(R0)                     ;                       6640
000070  001413                          BEQ     2$
000072  016046  000040                  MOV     40(R0),-(SP)                    ;                       6642
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$:    MOV     RP.ADDR,R0                      ;                       6644
000124  105760  000015                  TSTB    15(R0)
000130  100017                          BPL     3$
000132  132760  000100  000015          BITB    #100,15(R0)                     ;                       6645
000140  001013                          BNE     3$
000142  016046  000040                  MOV     40(R0),-(SP)                    ;                       6647
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$:    MOV     RP.ADDR,R0                      ;                       6649
000174  105760  000015                  TSTB    15(R0)
000200  100017                          BPL     4$
000202  132760  000100  000015          BITB    #100,15(R0)                     ;                       6650
000210  001413                          BEQ     4$
000212  016046  000040                  MOV     40(R0),-(SP)                    ;                       6652
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                              ;                       6618

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

```
;   6654  1     routine EMS_BC : novalue =
;   6655  1
;   6656  1     !+
;   6657  1     !       THIS ROUTINE PRINTS (EXTENDED) BOTH BYTE COUNT FIELDS OF THE CURRENT
;   6658  1     !       RETURN PACKET: THE BYTE COUNT FROM THE COMMAND PACKET AND THE
;   6659  1     !       ACTUAL NUMBER OF BYTES TRANSFERRED (FROM THE RESPONSE PACKET).
;   6660  1     !
;   6661  1     !       IMPLICIT INPUTS:
;   6662  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;   6663  1     !
;   6664  1
;   6665  2         begin
;   6666  2         PRINTX (EX_CBC, .RP_ADDR [CBCNT_LO]);       ! "BYTE COUNT IN COMMAND: XXXXX."
;   6667  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);         ! "ACTUAL # OF BYTES TRANSFERRED: XXXXX."
;   6668  1         end;                                        ! ROUTINE EMS_BC

                                        .SBTTL  EMS.BC ERROR MESSAGE SUBROUTINES
000000  013700  000000G        EMS.BC: MOV     RP.ADDR,R0                      ;                       6666
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                      ;                       6667
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                          ;                       6665
000054  000207                          RTS     PC                              ;                       6654

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

```
;   6669  1     routine EMS_BD : novalue =
;   6670  1
;   6671  1     !+
;   6672  1     !       THIS ROUTINE PRINTS (EXTENDED) THE TWO-WORD I/O BUFFER DESCRIPTOR
;   6673  1     !       APPEARING IN THE CURRENT RETURN PACKET.
;   6674  1     !
;   6675  1     !       IMPLICIT INPUTS:
;   6676  1     !               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
;   6677  1     !
;   6678  1
;   6679  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                      ;                                       6679
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                              ;                                       6669

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

```
;    6680  1
;    6681  1
;    6682  1     routine EMS_RP : novalue
;    6683  1
;    6684  1
;    6685  1     !+
;    6686  1     !       THIS ROUTINE IS RESPONSIBLE FOR PRINTING (EXTENDED) THE RELEVANT FIELDS
;    6687  1     !       OF THE CURRENT RETURN PACKET.
;    6688  1     !
;    6689  1
;    6690  1
;    6691  2         begin
;    6692  2         EMS_SBC ();                                     ! SUB-CODE
;    6693  2         EMS_CMD ();                                     ! COMMAND (AND MODIFIER)
;    6694  2
;    6695  2         if (.RP_ADDR [ENDCOD] and OP_MSK) neq OP_ONL
;    6696  2
;    6697  2         then
;    6698  2             EMS_LBN ();                                 ! LBN OR BAD BLOCK NUMBER
;    6699  2
;    6700  2         if ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD) or
;    6701  3            ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_WRT)
;    6702  3
;    6703  2         then
;    6704  3             begin
;    6705  3             EMS_BC ();                                  ! BYTE COUNTS
;    6706  3             EMS_BD ();                                  ! I/O BUFFER DESCRIPTOR
;    6707  2             end;
;    6708  2
;    6709  2         EMS_TIM ();                                     ! TIME
;    6710  1         end;                                            ! ROUTINE EMS_RP

                        .SBTTL  EMS.RP ERROR MESSAGE SUBROUTINES
000000  010146          EMS.RP: MOV     R1,-(SP)                    ;                   6682
000002  004737  010404'         JSR     PC,EMS_SBC                  ;                   6692
000006  004737  011114'         JSR     PC,EMS.CMD                  ;                   6693
000012  013700  000000G         MOV     RP.ADDR,R0                  ;                   6695
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                  ;                   6698
000040  013700  000000G     1$: MOV     RP.ADDR,R0                  ;                   6700
000044  116001  000014          MOVB    14(R0),R1
000050  042701  177600          BIC     #177600,R1
000054  020127  000041          CMP     R1,#41
000060  001407                  BEQ     2$
000062  116000  000014          MOVB    14(R0),R0                   ;                   6701
000066  042700  177600          BIC     #177600,R0
000072  020027  000042          CMP     R0,#42
000076  001004                  BNE     3$
000100  004737  012212      2$: JSR     PC,EMS.BC                   ;                   6705
```

```
000104  004737  012270                          JSR     PC,EMS.BD                       ;                       6706
000110  004737  000000V                 3$:     JSR     PC,EMS.TIM                      ;                       6709
000114  012601                                  MOV     (SP)+,R1                        ;                       6682
000116  000207                                  RTS     PC

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


;   6711  1
;   6712  1
```

```
;   6713  1     global routine EMS_RP1 : novalue *
;   6714  1
;   6715  1     !*
;   6716  1     !       THIS  ROUTINE  IS  CALLED  TO  PRINT  THE  ENTIRE CONTENTS OF THE
;   6717  1     !       RETURN PACKET DESIGNATED BY THE GLOBAL DATUM "RP_ADDR". HOWEVER, THE
;   6718  1     !       PRINTING WILL ONLY OCCUR IF EXTENDED ERROR PRINTING IS ENABLED.
;   6719  1     !
;   6720  1
;   6721  2         begin
;   6722  2         PRINTX (EX_RP);                         ! "CONTENTS OF RETURN PACKET:
;   6723  2         EMS_BLK (.RP ADDR, RP LEN);             ! PRINT BLOCK OF WORDS
;   6724  1         end;


                                        .SBTTL  EMS.RP1 ERROR MESSAGE SUBROUTINES
000000  012746  000000G        EMS.RP1::
                                MOV     @EX.RP,-(SP)                    ;                       6722
000004  012746  000001          MOV     #1,-(SP)
000010  010600                  MOV     SP,R0                           ; SP,*
000012  104415                  TRAP    15
000014  013716  000000G         MOV     RP.ADDR,(SP)                    ;                       6723
000020  012746  000026          MOV     #26,-(SP)
000024  004737  011576'         JSR     PC,EMS.BLK
000030  062706  000006          ADD     #6,SP                           ;                       6721
000034  000207                  RTS     PC                              ;                       6713

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

```
;  6725  1     global routine EMS_EL (index) : novalue =
;  6726  1
;  6727  1     !+
;  6728  1     !       THIS ROUTINE IS CALLED FROM 'SEQUEN' AND 'DATAGM' AND PRINTS THE CONTENTS OF THE
;  6729  1     !       ERROR-LOG PACKET
;  6730  1     !
;  6731  1
;  6732  2         begin
;  6733  2
;  6734  2         local
;  6735  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;  6736  2             REASON : word,
;  6737  2             DISK_NUM : byte,
;  6738  2             ELOG_CODE : byte,
;  6739  2             ELOG_SUB : word;
;  6740  2
;  6741  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                  ! ERROR LOG PACKET S ADDRESS
;  6742  2         REASON = .ELOG_ADDR [EL_FORMAT];                               ! FORMAT
;  6743  2         DISK_NUM = .ELOG_ADDR [EL_DK_NUM];                             ! DISK NUMBER
;  6744  2         ELOG_CODE = .ELOG_ADDR [EL_CODE];                              ! CODE
;  6745  2         ELOG_SUB = .ELOG_ADDR [EL_SUBCODE];                            ! SUBCODE
;  6746  2         PRINTB (ELG_00);                                               ! ERROR-LOG MESSAGE RECEIVED
;  6747  2
;  6748  2         if (.REASON eql FORMAT_CNTR) or
;  6749  3            (.REASON eql FORMAT_HOST)
;  6750  2         then
;  6751  3             PRINTB (.ELG_FMT [.REASON])                                ! PRINT BASIC REASON
;  6752  2         else
;  6753  2             PRINTB ( ELG_FMT [.REASON], .DISK_NUM);                    ! PRINT BASIC REASON WITH DISK NUMBER
;  6754  2
;  6755  2         if (.ELOG_CODE gtru 0) and
;  6756  3            (.ELOG_CODE lequ 11)
;  6757  2         then
;  6758  3             begin
;  6759  3             PRINTX (ASTERISK);
;  6760  3             PRINTX (.ERR_COD [.ELOG_CODE   1]);                        ! CODE
;  6761  3             end
;  6762  2         else
;  6763  2
;  6764  2             if .ELOG_CODE eql ST_DIA                                   ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  6765  2             then
;  6766  3                 begin
;  6767  3                 PRINTX (ASTERISK);
;  6768  3                 PRINTX (.ERR_COD [12]);
;  6769  2                 end;
;  6770  2
;  6771  2         if (.ELOG_CODE eql ST_MFE) and
;  6772  3            (.ELOG_SUB lequ 10)
;  6773  2         then
;  6774  3             begin
;  6775  3             PRINTX (CRLF);
;  6776  3             PRINTX (ASTERISK);
;  6777  3             PRINTX (.TBL_MFE [.ELOG_SUB]);                             ! MEDIA FORMAT ERROR
```

N2


```
;  6778  2              end;
;  6779  2
;  6780  2          if (.ELOG_CODE eql ST_DAT) and
;  6781  3             (.ELOG_SUB lequ 15)
;  6782  2          then
;  6783  3              begin
;  6784  3              PRINTX (CRLF);
;  6785  3              PRINTX (ASTERISK);
;  6786  3              PRINTX (.TBL_DAT [.ELOG_SUB]);                    ! DATA ERROR
;  6787  2              end;
;  6788  2
;  6789  2          if ( ELOG_CODE eql ST_HST) and
;  6790  3             (.ELOG_SUB lequ 4)
;  6791  2          then
;  6792  3              begin
;  6793  3              PRINTX (CRLF);
;  6794  3              PRINTX (ASTERISK);
;  6795  3              PRINTX (.TBL_HST [.ELOG SUB]);                    ! HOST ACCESS ERROR
;  6796  2              end;
;  6797  2
;  6798  2          if (.ELOG_CODE eql ST CNT) and
;  6799  3             (.ELOG_SUB lequ 3)
;  6800  2          then
;  6801  3              begin
;  6802  3              PRINTX (CRLF);
;  6803  3              PRINTX (ASTERISK);
;  6804  3              PRINTX (.TBL_CNT [.ELOG_SUB]);                    ! CONTROLLER ERROR
;  6805  2              end;
;  6806  2
;  6807  2          if (.ELOG_CODE eql ST_DRV) and
;  6808  3             (.ELOG_SUB lequ 8)
;  6809  2          then
;  6810  3              begin
;  6811  3              PRINTX (CRLF);
;  6812  3              PRINTX (ASTERISK);
;  6813  3              PRINTX (.TBL_DRV [.ELOG_SUB]);                    ! DRIVE ERROR
;  6814  2              end;
;  6815  2
;  6816  2          if .REASON eql FORMAT_XFER                            ! IF DISK XFER INVOLVED
;  6817  2          then
;  6818  2
;  6819  2              if .ELOG ADDR [EL_BLOCK_TYPE] eql TYPE LBN        ! PRINT PBN OR RBN
;  6820  2              then
;  6821  3                  PRINTX (EX_PBN, .ELOG ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK])
;  6822  2              else
;  6823  2                  PRINTX (EX_RBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL BLOCK]);
;  6824  2
;  6825  2          EMS_TIM ();                                           ! TIME
;  6826  2          EMS_BLK ((.ELOG_ADDR + 2), ((.ELOG ADDR [EL MSGLEN] + 1) / 2) + 2); ! PRINTX CONTENTS OF PACKE
;  6827  2          ELOG_ADDR [EL_CONTENTS] = EMPTY;                      ! DECLARE SAVE AREA FREE
;  6828  2
;  6829  1          end;
```

```
                                        .SBTTL  EMS.EL ERROR MESSAGE SUBROUTINES
000000  004137  0000000G          EMS.EL::JSR     R1,$SAVE5                       ;                               6725
000004  005746                          TST     -(SP)
000006  016646  000020                  MOV     20(SP), (SP)                    ; INDEX,*                       6741
000012  012746  000102                  MOV     #102,-(SP)
000016  004737  0000000G                JSR     PC,BL$MUL
000022  062700  0000000G                ADD     #ELOG.PKT,R0
000026  010001                          MOV     R0,R1                           ; *,ELOG.ADDR
000030  116166  000016  000004          MOVB    16(R1),4(SP)                    ; *(ELOG.ADDR),REASON           6742
000036  105066  000005                  CLRB    5(SP)                           ; REASON
000042  116105  000012                  MOVB    12(R1),R5                       ; *(ELOG.ADDR),DISK.NUM         6743
000046  116100  000020                  MOVB    20(R1),R0                       ; *(ELOG.ADDR),*                6744
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         6745
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  0000000G                MOV     #ELG.00,(SP)                    ;                               6746
000110  012746  000001                  MOV     #1,-(SP)
000114  010600                          MOV     SP,R0                           ; SP,*
000116  104414                          TRAP    14
000120  016602  000006                  MOV     6(SP),R2                        ; REASON,*                      6751
000124  006302                          ASL     R2
000126  005766  000006                  TST     6(SP)                           ; REASON                        6748
000132  001404                          BEQ     1$
000134  026627  000006  000001          CMP     6(SP),#1                        ; REASON,*                      6749
000142  001007                          BNE     2$
000144  016216  0000000G          1$:     MOV     ELG.FMT(R2),(SP)                ;                               6751
000150  012746  000001                  MOV     #1,-(SP)
000154  010600                          MOV     SP,R0                           ; SP,*
000156  104414                          TRAP    14
000160  000411                          BR      3$                              ;                               6748
000162  005016                  2$:     CLR     (SP)                            ;                               6753
000164  110516                          MOVB    R5,(SP)                         ; DISK.NUM,*
000166  016246  0000000G                MOV     ELG.FMT(R2),-(SP)
000172  012746  000002                  MOV     #2,-(SP)
000176  010600                          MOV     SP,R0                           ; SP,*
000200  104414                          TRAP    14
000202  005726                          TST     (SP)+
000204  105704                  3$:     TSTB    R4                              ; ELOG.CODE                     6755
000206  001423                          BEQ     4$
000210  120427  000013                  CMPB    R4,#13                          ; ELOC.CODE,*                   6756
000214  101020                          BHI     4$
000216  012716  0000000G                MOV     #ASTERISK,(SP)                  ;                               6759
000222  012746  000001                  MOV     #1,-(SP)
000226  010600                          MOV     SP,R0                           ; SP,*
000230  104415                          TRAP    15
```

```
000232  005000                          CLR     R0                              ;                               6760
000234  150400                          BISB    R4,R0                           ; 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$                              ;                               6758
000256  120427  000037          4$:     CMPB    R4,#37                          ; ELOG.CODE,•                   6764
000262  001015                          BNE     6$
000264  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6767
000270  012746  000001                  MOV     #1,-(SP)
000274  010600                          MOV     SP,R0                           ; SP,•
000276  104415                          TRAP    15
000300  013716  000030G                 MOV     ERR.COD+30,(SP)                 ;                               6768
000304  012746  000001                  MOV     #1,-(SP)
000310  010600                          MOV     SP,R0                           ; SP,•
000312  104415                          TRAP    15
000314  022626                  5$:     CMP     (SP)+,(SP)+                     ;                               6766
000316  120427  000005          6$:     CMPB    R4,#5                           ; ELOG.CODE,•                   6771
000322  001031                          BNE     7$
000324  020327  000012                  CMP     R3,#12                          ; ELOG.SUB,•                    6772
000330  101026                          BHI     7$
000332  012716  000000G                 MOV     #CRLF,(SP)                      ;                               6775
000336  012746  000001                  MOV     #1,-(SP)
000342  010600                          MOV     SP,R0                           ; SP,•
000344  104415                          TRAP    15
000346  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6776
000352  012746  000001                  MOV     #1,-(SP)
000356  010600                          MOV     SP,R0                           ; SP,•
000360  104415                          TRAP    15
000362  010300                          MOV     R3,R0                           ; ELOG.SUB,•                    6777
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                           ;                               6774
000406  120427  000010          7$:     CMPB    R4,#10                          ; ELOG.CODE,•                   6780
000412  001031                          BNE     8$
000414  020327  000017                  CMP     R3,#17                          ; ELOG.SUB,•                    6781
000420  101026                          BHI     8$
000422  012716  000000G                 MOV     #CRLF,(SP)                      ;                               6784
000426  012746  000001                  MOV     #1,-(SP)
000432  010600                          MOV     SP,R0                           ; SP,•
000434  104415                          TRAP    15
000436  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6785
000442  012746  000001                  MOV     #1,-(SP)
000446  010600                          MOV     SP,R0                           ; SP,•
000450  104415                          TRAP    15
000452  010300                          MOV     R3,R0                           ; ELOG.SUB,•                    6786
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           ;                                       6783
000476  120427  000011          8$:       CMPB    R4,#11          ; ELOG.CODE,*                           6789
000502  001031                            BNE     9$
000504  020327  000004                    CMP     R3,#4           ; ELOG.SUB,*                            6790
000510  101026                            BHI     9$
000512  012716  000000G                   MOV     #CRLF,(SP)      ;                                       6793
000516  012746  000001                    MOV     #1,-(SP)
000522  010600                            MOV     SP,R0           ; SP,*
000524  104415                            TRAP    15
000526  012716  000000G                   MOV     #ASTERISK,(SP)  ;                                       6794
000532  012746  000001                    MOV     #1,-(SP)
000536  010600                            MOV     SP,R0           ; SP,*
000540  104415                            TRAP    15
000542  010300                            MOV     R3,R0           ; ELOG.SUB,*                            6795
000544  006300                            ASL     R0
000546  016016  000160'                   MOV     TBL.MST(R0),(SP)
000552  012746  000001                    MOV     #1,-(SP)
000556  010600                            MOV     SP,R0           ; SP,*
000560  104415                            TRAP    15
000562  062706  000006                    ADD     #6,SP           ;                                       6792
000566  120427  000012          9$:       CMPB    R4,#12          ; ELOG.CODE,*                           6798
000572  001031                            BNE     10$
000574  020327  000003                    CMP     R3,#3           ; ELOG.SUB,*                            6799
000600  101026                            BHI     10$
000602  012716  000000G                   MOV     #CRLF,(SP)      ;                                       6802
000606  012746  000001                    MOV     #1,-(SP)
000612  010600                            MOV     SP,R0           ; SP,*
000614  104415                            TRAP    15
000616  012716  000000G                   MOV     #ASTERISK,(SP)  ;                                       6803
000622  012746  000001                    MOV     #1,-(SP)
000626  010600                            MOV     SP,R0           ; SP,*
000630  104415                            TRAP    15
000632  010300                            MOV     R3,R0           ; ELOG.SUB,*                            6804
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           ;                                       6801
000656  120427  000013          10$:      CMPB    R4,#13          ; ELOG.CODE,*                           6807
000662  001031                            BNE     11$
000664  020327  000010                    CMP     R3,#10          ; ELOG.SUB,*                            6808
000670  101026                            BHI     11$
000672  012716  000000G                   MOV     #CRLF,(SP)      ;                                       6811
000676  012746  000001                    MOV     #1,-(SP)
000702  010600                            MOV     SP,R0           ; SP,*
000704  104415                            TRAP    15
000706  012716  000000G                   MOV     #ASTERISK,(SP)  ;                                       6812
000712  012746  000001                    MOV     #1,-(SP)
000716  010600                            MOV     SP,R0           ; SP,*
```

E3


```
000720  104415                          TRAP    15
000722  010300                          MOV     R3,R0                   ; ELOG.SUB,*            6813
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                   ;                       6810
000746  026627  000010  000002  11$:    CMP     10(SP),#2               ; REASON,*              6816
000754  001031                          BNE     14$
000756  032751  170000  000060          BIT     #170000,60(R1)          ; *,*(ELOG.ADDR)        6819
000764  001012                          BNE     12$
000766  016116  000056                  MOV     56(R1),(SP)             ; *(ELOG.ADDR),*        6821
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$                     ;                       6819
001012  016116  000056          12$:    MOV     56(R1),(SP)             ; *(ELOG.ADDR),*        6823
001016  011646                          MOV     (SP),-(SP)
001020  012746  000000G                 MOV     #EX.RBN,-(SP)
001024  012746  000003                  MOV     #3,-(SP)
001030  010600                          MOV     SP,R0                   ; SP,*
001032  104415                          TRAP    15
001034  062706  000006          13$:    ADD     #6,SP                   ;                       6819
001040  004737  000000V         14$:    JSR     PC,EMS.TIM              ;                       6825
001044  012716  000002                  MOV     #2,(SP)                 ;                       6826
001050  060116                          ADD     R1,(SP)                 ; ELOG.ADDR,*
001052  016146  000002                  MOV     2(R1),-(SP)             ; *(ELOG.ADDR),*
001056  005216                          INC     (SP)
001060  012746  000002                  MOV     #2,-(SP)
001064  004737  000000G                 JSR     PC,BL$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)          6827
001114  062706  000014                  ADD     #14,SP                  ;                       6725
001120  000207                          RTS     PC

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

```
;  6830  1     global routine EMS_CMP (ADDR) : novalue =
;  6831  1
;  6832  1     !+
;  6833  1     ! THIS ROUTINE IS CALLED FROM 'HOST_WRT_CHK' AND PRINTS RELEVENT DATA ON A HOST
;  6834  1     ! COMPARE ERROR
;  6835  1     !
;  6836  1
;  6837  2         begin
;  6838  2
;  6839  2         local
;  6840  2             ORIG_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;  6841  2
;  6842  2         ORIG_ADDR = .ADDR;                                          ! ADDRESS OF THE WRITE RETPKT
;  6843  2         PRINTB (ERR_00, .CDISK);                                    ! "DISK XXX'
;  6844  2         PRINTB (DASH);                                              !
;  6845  2         PRINTB (.ERR_COD [12]);                                     ! " - HOST COMPARE ERROR"
;  6846  2         PRINTX (EX_LBW, .ORIG_ADDR [LBN_LO], .ORIG_ADDR [LBN_LO]);  ! LBN (WRITE)
;  6847  2         PRINTX (EX_LBR, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);      ! LBN (READ)
;  6848  2         PRINTX (EX_CBW, .ORIG_ADDR [CBCNT_LO]);                     ! BYTE COUNT (WRITE)
;  6849  2         PRINTX (EX_BC, .ORIG_ADDR [BCNT_LO]);                       ! BYTE COUNT XMITTED (WRITE)
;  6850  2         PRINTX (EX_CBR, .RP_ADDR [CBCNT_LO]);                       ! BYTE COUNT (READ);
;  6851  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);                         ! BYTE COUNT XMITTED (READ)
;  6852  2         PRINTX (EX_BDW, .ORIG_ADDR [BUFF_1], .ORIG_ADDR [BUFF_0]);  ! BUFFER ADDRESS (WRITE)
;  6853  2         PRINTX (EX_BDR, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);      ! BUFFER ADDRESS (READ)
;  6854  2         EMS_TIM ();                                                 ! TIME
;  6855  1         end;


                                .SBTTL  EMS.CMP ERROR MESSAGE SUBROUTINES
000000  010146          EMS.CMP::
                        MOV     R1,-(SP)                        ;                               6830
000002  016601  000004  MOV     4(SP),R1                        ; ADDR,ORIG.ADDR                6842
000006  013746  000000G MOV     CDISK,-(SP)                     ;                               6843
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)                      ;                               6844
000032  012746  000001  MOV     #1,-(SP)
000036  010600          MOV     SP,R0                           ; SP,*
000040  104414          TRAP    14
000042  013716  000030G MOV     ERR.COD+30,(SP)                 ;                               6845
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),*                6846
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                      ;                               6847
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),*                6848
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),*                6849
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              ;                               6850
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              ;                               6851
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),*                6852
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              ;                               6853
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              ;                               6854
000316  062706  000062          ADD     #62,SP                  ;                               6837
000322  012601                  MOV     (SP)+,R1                ;                               6830
000324  000207                  RTS     PC

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

```
;    6856  1     global routine EMS_ERR : novalue =
;    6857  1
;    6858  2         begin
;    6859  2
;    6860  2     ! TABLE OF BASIC, HARD ERROR MESSAGE ADDRESSES, INDEXED BY STATUS CODE
;    6861  2     !
;    6862  2         PRINTB (ERR_00, .CDISK);                        ! "DISK XXX
;    6863  2         PRINTB (DASH);                                  !
;    6864  2
;    6865  2         if (.ST_CODE gtru 0) and                        ! IF STATUS CODE IS WITHIN RANGE
;    6866  3            (.ST_CODE lequ 11)
;    6867  2         then
;    6868  3             PRINTB (.ERR_COD [.ST_CODE - 1])            ! PRINTB APPROPRIATE MESSAGE
;    6869  2         else
;    6870  2
;    6871  2             if .ST_CODE eql ST_DIA
;    6872  2             then
;    6873  3                 PRINTB (.ERR_COD [11])                  ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    6874  2             else
;    6875  2                 PRINTB (EX_SC, .ST_CODE);               ! JUST PRINT STATUS CODE WHEN NO MATCH
;    6876  2
;    6877  2         EMS RP ();                                      ! PRINTX OTHER RETPKT FIELDS
;    6878  2
;    6879  1         end;

                                .SBTTL  EMS.ERR ERROR MESSAGE SUBROUTINES
000000  013746  000000G  EMS.ERR::
                        MOV     CDISK,-(SP)                     ;                       6862
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)              ;                       6863
000024  012746  000001          MOV     #1,-(SP)
000030  010600                  MOV     SP,R0                   ; SP,*
000032  104414                  TRAP    14
000034  013700  000000G         MOV     ST.CODE,R0              ;                       6865
000040  001413                  BEQ     1$
000042  020027  000013          CMP     R0,#13                  ;                       6866
000046  101010                  BHI     1$
000050  006300                  ASL     R0                      ;                       6868
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$                      ;                       6865
000070  020027  000037  1$:     CMP     R0,#37                  ;                       6871
000074  001007                  BNE     2$
000076  013716  000026G         MOV     ERR.COD+26,(SP)         ;                       6873
000102  012746  000001          MOV     #1,-(SP)
000106  010600                  MOV     SP,R0                   ; SP,*
000110  104414                  TRAP    14
```

```
000112  000410                          BR      3$                              ;                       6871
000114  010016                  2$:     MOV     R0,(SP)                         ;                       6875
000116  012746  000000G                 MOV     #EX.SC, (SP)
000122  012746  000002                  MOV     #2, (SP)
000126  010600                          MOV     SP,R0                           ; SP,*
000130  104414                          TRAP    14
000132  005726                          TST     (SP)+
000134  004737  012326'          3$:     JSR     PC,EMS.RP                       ;                       6877
000140  062706  000012                  ADD     #12,SP                          ;                       6858
000144  000207                          RTS     PC                              ;                       6856

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

J3


```
;   6880  1     routine EMS_TIM : novalue =
;   6881  1
;   6882  1     !+
;   6883  1     ! THIS ROUTINE PRINTS THE TIME OF DAY MESSAGE
;   6884  1     !
;   6885  1
;   6886  1     PRINTX (EX_TIM, .HOURS, .MINUTES);


                                        .SBTTL  EMS.TIM ERROR MESSAGE SUBROUTINES
000000  005046                  EMS.TIM:CLR     -(SP)                           ;                       6886
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                              ;                       6880

; Routine Size:  15 words,      Routine Base:  $CODE$ + 1#22
; Maximum stack depth per invocation:  6 words
```

```
;   6887  1     BGNMSG (EMS_01);


                                        .SBTTL  EMS.01 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.01::JSR     PC,M$EMS.01                     ;                       6887
000004  104423                            TRAP    23
000006  000207                            RTS     PC

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


;   6888  2     PRINTB (EBS_01, MAX_UNITS);             ! "MORE THAN XX UNITS SPECIFIED'
;   6889  1     ENDMSG;


                                        .SBTTL  M$EMS.01 ERROR MESSAGE SUBROUTINES
000000  012746  000004           M$EMS.01:
                                        MOV     #4,-(SP)                        ;                       6888
000004  012746  000000G                 MOV     #EBS.01,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  062706  000006                  ADD     #6,SP                           ;                       6887
000024  000207                          RTS     PC

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

```
;   6890  1     BGNMSG (EMS_10);


                                        .SBTTL  EMS.10 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.10::JSR     PC,M$EMS.10                     ;                       6890
000004  104423                            TRAP    23
000006  000207                            RTS     PC

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


;   6891  2     PRINTB (EBD_10, .RDRX_ADDR + .OF_RC);           ! "NO RESPONSE AT ADDRESS XXXXXX"
;   6892  1     ENDMSG;


                                        .SBTTL  M$EMS.10 ERROR MESSAGE SUBROUTINES
000000  013746  000000G            M$EMS.10:
                                        MOV     RDRX.ADDR,-(SP)                 ;                       6891
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                           ;                       6890
000030  000207                          RTS     PC

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

M3


```
;   6893  1     BGNMSG (EMS_12);


                                        .SBTTL  EMS.12 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.12::JSR     PC,M$EMS.12                     ;                       6893
000004  104423                           TRAP    23
000006  000207                           RTS     PC

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


;   6894  2     PRINTB (EBD_12, .RDRX_ADDR);            ! "INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX"
;   6895  1     ENDMSG;


                                        .SBTT   M$EMS.12 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.12:
                                        MOV     RDRX.ADDR,-(SP)                 ;                       6894
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                           ;                       6893
000024  000207                          RTS     PC

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

N3


```
;   6896  1     BGNMSG (EMS_13);


                                        .SBTTL  EMS.13 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.13::JSR     PC,M$EMS.13                     ;                       6896
000004  104423                            TRAP    23
000006  000207                            RTS     PC

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


.   6897  2     PRINTB (EBD_13, .STEP);                 ! 'STEP x READ ERROR
;   6898  2     EMS_SA ();                              ! PRINTx SA CONTENTS
;   6899  1     ENDMSG;


                                        .SBTTL  M$EMS.13 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.13:
                                        MOV     STEP,-(SP)                      ;                       6897
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                       ;                       6898
000024  062706  000006                  ADD     #6,SP                           ;                       6896
000030  000207                          RTS     PC

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

```
;   6900  1     BGNMSG (EMS_14);


                                        .SBTTL  EMS.14 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.14::JSR     PC,M$EMS.14                     ;                       6900
000004  104423                           TRAP    23
000006  000207                           RTS     PC

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


;   6901  2     PRINTB (EBD_14, .IRDRX_ADDR);           ! "BAD SA CODE FROM DEVICE XXXXXX"
;   6902  2     EMS_SA ();                              ! PRINTX SA REGISTER CONTENTS
;   6903  1     ENDMSG;


                                        .SBTTL  M$EMS.14 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.14:
                                        MOV     IRDRX.ADDR,-(SP)                ;                       6901
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                       ;                       6902
000024  062706  000006                  ADD     #6,SP                           ;                       6900
000030  000207                          RTS     PC

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

```
;   6904  1     BGNMSG (EMS_18);


                                        .SBTTL  EMS.18 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.18::JSR     PC,M$EMS.18                     ;                       6904
000004  104423                            TRAP    23
000006  000207                            RTS     PC

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


;   6905  2     PRINTB (EBD_18, .CDISK);                ! "DISK XXX WENT OFFLINE"
;   6906  2     EMS_RP ();                              ! PRINTX RELEVANT RETPKT FIELDS
;   6907  1     ENDMSG;


                                        .SBTTL  M$EMS.18 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.18:
                                        MOV     CDISK,-(SP)                     ;                       6905
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                       ;                       6906
000024  062706  000006                  ADD     #6,SP                           ;                       6904
000030  000207                          RTS     PC

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

```
;   6908  1     BGNMSG (EMS_21);


                                          .SBTTL  EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.21::JSR     PC,M$EMS.21                     ;                       6908
000004  104423                               TRAP    23
000006  000207                               RTS     PC

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


;   6909  2     EMS_RP1 ();                                 ! CONTENTS OF RETURN PACKET
;   6910  1     ENDMSG;


                                          .SBTTL  M$EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  012446                   M$EMS.21:
                                               JSR     PC,EMS.RP1                      ;                       6909
000004  000207                               RTS     PC                              ;                       6908

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

```
;   6911  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                     ;                6911
000004  104423                                  TRAP    23
000006  000207                                  RTS     PC

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


;   6912  2     EMS_DBN ();                                                           !ZZZ
;   6913  1     ENDMSG;                                                               !ZZZ


                                          .SBTTL  M$EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  011400                   M$EMS.22:
                                                  JSR     PC,EMS.DBN                      ;                6912
000004  000207                                    RTS     PC                              ;                6911

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

```
;   6914  1     BGNMSG (EMS_24);


                                        .SBTTL  EMS.24 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.24::JSR     PC,M$EMS.24                     ;                       6914
000004  104423                            TRAP    23
000006  000207                            RTS     PC

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


;   6915  2     PRINTB (EBD_24, .CDISK);                ! "DISK XXX WENT TO THE AVAILABLE STATE"
;   6916  2     EMS_RP ();                              ! PRINTX RELEVANT RETPKT FIELDS
;   6917  1     ENDMSG;


                                        .SBTTL  M$EMS.24 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.24:
                                        MOV     CDISK,-(SP)                     ;                       6915
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                       ;                       6916
000024  062706  000006                  ADD     #6,SP                           ;                       6914
000030  000207                          RTS     PC

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

```
;   6918  1     BGNMSG (EMS_30);


                                        .SBTTL  EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.30::JSR     PC,M$EMS.30                     ;                       6918
000004  104423                           TRAP    23
000006  000207                           RTS     PC

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


;   6919  2     EMS_ERR ();                             ! PRINT ALL RELEVANT DATA ON DETECTING AN ERROR
;   6920  1     ENDMSG;


                                        .SBTTL  M$EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  014154'           M$EMS.30:
                                        JSR     PC,EMS.ERR                      ;                       6919
000004  000207                           RTS     PC                              ;                       6918

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


;   6921  1
;   6922  1     end
;   6923  1
;   6924  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]ZRQAF0.L16;11        406       297        73        21        00:00.1
```

H4


```
;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAF0.BL1/LIST=ZRQAF0.LS1/OBJECT=ZRQAF0.OB1/SOURCE=PAGE:53

; Size:          3155 code + 6635 data words
; Run Time:         03:04.6
; Elapsed Time:     09:19.5
; Lines/CPU Min:     2250
; Lexemes/CPU-Min: 22455
; Memory Used:  702 pages
; Compilation Complete
```

```
;    0001  0
;    0002  0     module ZRQAM3 (
;    0003  0
;    0004  0     %title 'RD/RX EXERCISER'
;    0005  0                     ident = 'V02.1',
;    0006  0                     addressing_mode (absolute),
;    0007  0                     environment (noeis)
;    0008  0                     ) =
;    0009  0
;    0010  1     begin
;    0011  1
;    0012  1     %sbttl 'DECLARATIONS
;    0013  1
;    0014  1     library 'ZRQAF0.L16';                       ! RDRX EXERCISER GLOBAL LIBRARY
;    0015  1
;    0016  1     require 'BLSMAC.REQ';                       ! DIAGNOSTIC SUPERVISOR LIBRARY
;    1507  1
;    1508  1     EQUALS;
;    1509  1
;    1510  1     forward routine                             ! ROUTINES APPEAR IN THIS ORDER
;    1511  1         INIT_TEST : novalue,                    !    INDENTATION IMPLIES CALLED SUBROUTINE
;    1512  1             DRIVER_INIT : novalue,
;    1513  1                 CTLR_INIT : novalue,
;    1514  1                     INI_CTLR_DAT : novalue,
;    1515  1                     REG_EXIST,
;    1516  1                     VEC_BR_TEST,
;    1517  1                     INT_GEN,
;    1518  1                     HARD_INIT,
;    1519  1                     INI_RRING : novalue,
;    1520  1                     SET_CTLR_CHAR,
;    1521  1                 UNIT_INIT : novalue,
;    1522  1                     DR_ERR : novalue,
;    1523  1             ACCESS : novalue,
;    1524  1         MULTI_DRIVE : novalue,
;    1525  1             MD_INIT : novalue,
;    1526  1                 INIT_IO_BUFF : novalue,
;    1527  1             FATAL_ERROR : novalue,
;    1528  1             QIO_OK,
;    1529  1                 QIO_OUT,
;    1530  1                 QIO_GEN : novalue,
;    1531  1                     GET_RANDOM : novalue,
;    1532  1                     QIO_UNIT : novalue,
;    1533  1                     QIO_FUNC : novalue,
;    1534  1                        DUP : NOVALUE,                          !ZZZ
;    1535  1                              DUPWRTDBN : NOVALUE,              !ZZZ
;    1536  1                              DUPREDDBN : NOVALUE,              !ZZZ
;    1537  1                                   DUPCOMMAND : NOVALUE,        !ZZZ
;    1538  1                                   DUPIDLE : NOVALUE,           !ZZZ
;    1539  1                     QIO_LBN : novalue,
;    1540  1                     QIO_SIZE : novalue,
;    1541  1                     FILL_BUFF : novalue,
;    1542  1                 PROC_RETPKT : novalue,
;    1543  1                     DIO_RETPKT : NOVALUE,                   !ZZZ
```

```
;    1544  1                        DUP_COMPARE : NOVALUE,                                  !ZZZ
;    1545  1                    IO_RETPKT : novalue,
;    1546  1                        FSET_UPAR : novalue,
;    1547  1                        HARD_ERROR : novalue,
;    1548  1                            ERR_HRD_RTNE : novalue,
;    1549  1                            ERR_HRD_RTNE_APT : novalue,
;    1550  1                        UPD_IO_TALLY : novalue,
;    1551  1                            OVF_CHK : novalue,
;    1552  1                            ROUND_OUTPUT : novalue,
;    1553  1                        HOST_WRT_CHK,
;    1554  1     !                      ERR_HRD_RTNE : novalue,
;    1555  1     !                      ERR_HRD_RTNE_APT : novalue,
;    1556  1                        SWEEP : novalue,
;    1557  1                            RPS_REM,
;    1558  1                    DR_RETPKT : novalue,
;    1559  1            AZINTO : L$ISR novalue,
;    1560  1                AZINT : novalue,
;    1561  1     !              FATAL_ERROR : novalue,
;    1562  1                    POLL_CRING : novalue,
;    1563  1                    POLL_RRING : novalue,
;    1564  1                        DUP_RSP  : NOVALUE,
 !ZZZ
;    1565  1                        DISK_RSP : novalue,
;    1566  1                        SEQUEN : novalue,
;    1567  1                            SCAN_ERRLOG : novalue,
;    1568  1                                ERR_SOFT_RTNE : novalue,
;    1569  1                                ERR_SOFT_RTNE_APT : novalue,
;    1570  1                                SOFT_ERROR : novalue,
;    1571  1                        DATAGM : novalue;
;    1572  1     !                      ERR_SOFT_RTNE : novalue,
;    1573  1     !                      ERR_SOFT_RTNE_APT : novalue,
;    1574  1     !                      SOFT_ERROR : novalue;
;    1575  1
;    1576  1     external
;    1577  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    1578  1                            ! RUN-TIME CONTROLLER STATUS TABLES
;    1579  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    1580  1                            ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    1581  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    1582  1                            ! DRIVER CONTROLLER TABLES
;    1583  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    1584  1                            ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    1585  1         RDRX_ADDR : ref rdrx field (RC_REG),
;    1586  1                            ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    1587  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;    1588  1                            ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    1589  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                                 !ZZZ
;    1590  1                            !BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS        !ZZZ
;    1591  1                            !RANDOM SEEK) MODE                                  !ZZZ
;    1592  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    1593  1                            ! STATISTICS TABLES
;    1594  1         T_ADDP : ref block [TALLY_LEN, word] field (T_FIELDS),
;    1595  1                            ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    1596  1         DUPPKT : BLOCK [257, WORD] FIELD (DP FIELDS),       !BUFFER FOR DUP  ZZZ
```

```
;    1597  1                          !INFO FROM RECEIVE AND SEND COMMANDS                ZZZ
;    1598  1          TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TK DIRECTION    ZZZ
;    1599  1          RDM_CNT : WORD,     !NO. OF RANDOM NOS.                KEEP\\        ZZZ
;    1600  1          RANDOM : VECTOR [RDM_LEN, WORD],    !RAND NO TABLE TOGET//HER       ZZZ
;    1601  1          C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    1602  1                          ! STATISTICS TABLE FOR CONTROLLER ERRORS
;    1603  1          MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS)
;    1604  1                          ! MSCP PACKET POOL
;    1605  1          IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    1606  1                          ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;    1607  1          PKT_USE : vector [PKT_CNT, byte, signed],
;    1608  1                          ! MSCP PACKET POOL ALLOCATION TABLE
;    1609  1          RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;    1610  1                          ! RETURN PACKET POOL
;    1611  1          RP_USE : vector [RP_CNT, byte, signed],
;    1612  1                          ! RETURN PACKET POOL ALLOCATION TABLE
;    1613  1          RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
;    1614  1          RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
;    1615  1                          ! CURRENT RETURN PACKET ADDRESS
;    1616  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;    1617  1                          ! ERROR-LOG PACKET SAVE AREA
;    1618  1          BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
;    1619  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;    1620  1          IODQ : vector [IODQ_LEN, byte],                      ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;    1621  1          IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
;    1622  1          IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
;    1623  1          ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
;    1624  1          EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
;    1625  1          DUP_FLAGS : WORD,                                    !DUP FLAGS                                  ZZZ
;    1626  1          CCTLR . word,                                        ! NUMBER OF "CURRENT" CONTROLLER
;    1627  1          CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    1628  1          CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
;    1629  1          CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;    1630  1          DUR : vector [MAX_UNITS, byte],                      ! DROP UNIT REASON
;    1631  1          QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;    1632  1          FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
;    1633  1          BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
;    1634  1          ST_CODE : word,                                      ! CURRENT STATUS CODE
;    1635  1          SB_CODE : word,                                      ! CURRENT SUB-CODE
;    1636  1          STEP : word,                                         ! CURRENT STEP IN HARD_INIT
;    1637  1          OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
;    1638  1          SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
;    1639  1          CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
;    1640  1          NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
;    1641  1          CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
;    1642  1          CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
;    1643  1          TEMP1 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN            !ZZZ
;    1644  1          TEMP2 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN            !ZZZ
;    1645  1          CREDIT_BAL : word,                                   ! CREDIT BALANCE
;    1646  1          NEXT_PKT_USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;    1647  1          HOURS : byte,                                        ! TIME OF DAY (HOURS)
;    1648  1          MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
;    1649  1          CLK_TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
```

```
;    1650  1          CLK_PRESENT : byte,                ! FLAG INDICATES IF LINE-CLOCK PRESENT
;    1651  1          HOE_FLAG : byte,                   ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;    1652  1          FORCED_ERROR : byte,               ! "FORCED ERROR" DETECTED IN LAST READ
;    1653  1          FER_LBN : word,                    ! LBN OF THE "FORCED ERROR" BLOCK
;    1654  1          FER_BC : word,                     ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    1655  1          INIT_OCCURED : byte,               ! EXERCISER INITIALIZATION COMPLETE
;    1656  1          ADDR_VECT_OK : byte,               ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    1657  1          S_PATTERN : WORD,                  !PATTERN WRITTEN TO DBNS                ZZZ
;    1658  1          S_DUPPKT : WORD,                   !DBN BYTE COUNTER                       ZZZ
;    1659  1          P_INDEX : SIGNED WORD,             !CURRENT MESSAGE PACKET INDEX           ZZZ
;    1660  1          RD_COUNT : WORD,                   ! NUMBER OF WINCHESTER UNITS            ZZZ
;    1661  1          BRLEVEL : word,                    ! CURRENT DEVICE'S BR LEVEL             ZZZ
;    1662  1          D_FAIL : BYTE,                     !SIGNIFIES DUP TYPE ERROR               ZZZ
;    1663  1          DBM12,
;    1664  1          DBM18,
;    1665  1          DBM19,
;    1666  1          DBM20,
;    1667  1          DBM21,
;    1668  1          DBM22,
;    1669  1          DBM23,
;    1670  1          DBM25,
;    1671  1          DBM26,
;    1672  1          DBM27,
;    1673  1          DBM29,
;    1674  1          DBM108,
;    1675  1          DBM109,
;    1676  1          DBM111,
;    1677  1          DBM112,
;    1678  1          DBM120,
;    1679  1          DBM121,
;    1680  1          EH_0,                                                          !ZZZ
;    1681  1          EH_1,                                                          !ZZZ
;    1682  1          EH_2,                                                          !ZZZ
;    1683  1          EH_3,                                                          !ZZZ
;    1684  1          EH_4,                                                          !ZZZ
;    1685  1          EH_5,                                                          !ZZZ
;    1686  1          EH_6,                                                          !ZZZ
;    1687  1          EH_7,                                                          !ZZZ
;    1688  1          EH_8,                                                          !ZZZ
;    1689  1          EH_9,                                                          !ZZZ
;    1690  1          EH_10,                                                         !ZZZ
;    1691  1          EH_12,                                                         !ZZZ
;    1692  1          EH_13,                                                         !ZZZ
;    1693  1          MSG_02,
;    1694  1          MSG_03,
;    1695  1          EGS_02,
;    1696  1          EGD_10,
;    1697  1          EGD_11,
;    1698  1          EGD_12,
;    1699  1          EGD_13,
;    1700  1          EGD_14,
;    1701  1          EGD_15,
;    1702  1          EGD_16,
```

```
;    1703  1          EGD_17,
;    1704  1          EGD_18,
;    1705  1          EGD_19,
;    1706  1          EGD_20,
;    1707  1          EGD_21,
;    1708  1          EGD_22,
;    1709  1          EGD_23,
;    1710  1          EGD_24,
;    1711  1          EGH_30,
;    1712  1          DF_MSG,
;    1713  1          HRD_MSG,
;    1714  1          SFT_MSG,
;    1715  1          HRD_SUB,
;    1716  1          CRLF,
;    1717  1          SWP_ERROR : word,                    ! HARD ERROR LIMIT FOR DROPPING UNIT
;    1718  1          SWP_XFER : word,                     ! TRANSFER LIMIT FOR DROPPING UNIT
;    1719  1          SWP_FLAGS : word,                    ! FLAGS (SEE DOCUMENTATION)
;    1720  1          DUPROUND  : WORD,                    !DUP TESTING RATIO                    ZZZ
;    1721  1          SWP_RAT : word,                      ! RD51/52 OPERATION RATIO
;    1722  1          SWP_DPAT : word,                     ! DATA PATTERN NUMBER
;    1723  1          SWP_UCNT : word,                     ! USER DATA PATTERN COUNT
;    1724  1          SWP_TIME : word,                     ! TIME OF DAY
;    1725  1          SWP_UDPAT : vector [MAX_UDP_CNT, word],    ! USER DATA PATTERN
;    1726  1          L$LUN,
;    1727  1          L$UNIT;
;    1728  1
;    1729  1      psect
;    1730  1          own = $GGG$(read, nowrite, execute, local, concatenate);
;    1731  1
;    1732  1      own
;    1733  1          COMM_AREA : blockvector [MAX_CTLR, COMM_LEN, word] field (COM_FIELDS),
;    1734  1                              ! COMMUNICATIONS AREA BETWEEN HOST AND AZTEC CONTROLLERS
;    1735  1      !!ZZZ    BST : vector [MAX_UNITS, word, signed],
;    1736  1                              ! BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
;    1737  1          DPST : vector [MAX_UNITS, byte],          ! DATA PATTERN SEQUENCE TABLE
;    1738  1          MAX_LBN : vector [MAX_UNITS, word],       ! LARGEST LBN ALLOWED
;    1739  1          STORAGE : vector [MAX_UNITS, word],       ! DUMMY STORAGE
;    1740  1          ICOM_ADDR : ref block [COMM_LEN, word] field (COM_FIELDS),
;    1741  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S COMMUNICATION AREA
;    1742  1          ICST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    1743  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S CST
;    1744  1          IDCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    1745  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    1746  1          INT_ADDR : vector [MAX_CTLR] initial (AZINT0 %(, AZINT1, AZINT2, AZINT3)%),
;    1747  1                              ! INTERRUPT SERVICE ROUTINE ADDRESS TABLE
;    1748  1      !!ZZZ    RDM_CNT : word initial (RDM_LEN),          ! NUMBER OF RANDOM NUMBERS  \ KEEP
;    1749  1      !!ZZZ    RANDOM : vector [RDM_LEN, word],           ! RANDOM NUMBER TABLE       / TOGETHER
;    1750  1          ICTLR : word,                         ! INTERRUPTING CONTROLLING NUMBER
;    1751  1          RW_BALANCE : WORD INITIAL (3),        ! FLAGS TOO MANY READS IN RD/WR RATIO      ZZZ
;    1752  1          MX1 : signed word,                    ! MSCP PKT INDEX FOR FIRST QIO
;    1753  1          MX2 : signed word,                    ! MSCP PKT INDEX FOR SECOND QIO
;    1754  1          MAD1 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    1755  1                              ! ADDRESS OF MSCP PACKET FOR FIRST QIO
```

```
;  1756  1          MAD2 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  1757  1                              ! ADDRESS OF MSCP PACKET FOR SECOND QIO
;  1758  1          LAST_PKT : blockvector [MAX_CTLR, LAST_PKT_LEN, word] field (LAST_PKT_FIELDS),
;  1759  1                              ! SAVE AREA FOR INFO ABOUT LAST RESPONSE PACKET
;  1760  1          PAT02 : vector [2] initial (1,                                  ! PATTERN 2
;  1761  1                  %o'000000'),
;  1762  1          PAT03 : vector [2] initial (1,                                  ! PATTERN 3
;  1763  1                  %o'177777'),
;  1764  1          PAT04 : vector [2] initial (1,                                  ! PATTERN 4
;  1765  1                  %o'105613'),
;  1766  1          PAT05 : vector [2] initial (1,                                  ! PATTERN 5
;  1767  1                  %o'031463'),
;  1768  1          PAT06 : vector [2] initial (1,                                  ! PATTERN 6
;  1769  1                  %o'030221'),
;  1770  1          PAT07 : vector [17] initial (16,                                ! PATTERN 7
;  1771  1                  %o'000001', %o'000003', %o'000007', %o'000017',
;  1772  1                  %o'000037', %o'000077', %o'000177', %o'000377',
;  1773  1                  %o'000777', %o'001777', %o'003777', %o'007777',
;  1774  1                  %o'017777', %o'037777', %o'077777', %o'177777'),
;  1775  1          PAT08 : vector [17] initial (16,                                ! PATTERN 8
;  1776  1                  %o'177776', %o'177774', %o'177770', %o'177760',
;  1777  1                  %o'177740', %o'177700', %o'177600', %o'177400',
;  1778  1                  %o'177000', %o'176000', %o'174000', %o'170000',
;  1779  1                  %o'160000', %o'140000', %o'100000', %o'000000'),
;  1780  1          PAT09 : vector [17] initial (16,                                ! PATTERN 9
;  1781  1                  rep 3 of (%o'000000'), rep 3 of (%o'177777'),
;  1782  1                  rep 2 of (%o'000000'), rep 2 of (%o'177777'),
;  1783  1                  %o'000000', %o'177777', %o'000000', %o'177777',
;  1784  1                  %o'000000', %o'177777'),
;  1785  1          PAT10 : vector [2] initial (1,                                  ! PATTERN 10
;  1786  1                  %o'133331'),
;  1787  1          PAT11 : vector [17] initial (16,                                ! PATTERN 11
;  1788  1                  rep 3 of (%o'052525'), rep 3 of (%o'125252'),
;  1789  1                  rep 2 of (%o'052525'), rep 2 of (%o'125252'),
;  1790  1                  %o'052525', %o'125252', %o'052525', %o'125252',
;  1791  1                  %o'052525', %o'125252'),
;  1792  1          PAT12 : vector [21] initial (20,                                ! PATTERN 12
;  1793  1                  rep 3 of (%o'026455'), rep 3 of (%o'151322'),
;  1794  1                  rep 2 of (%o'026455'), rep 2 of (%o'151322'),
;  1795  1                  rep 2 of (%o'026455'),
;  1796  1                  %o'151322', %o'026455', %o'151322', %o'026455',
;  1797  1                  %o'151322', %o'026455', %o'151322', %o'026455'),
;  1798  1          PAT13 : vector [2] initial (1,                                  ! PATTERN 13
;  1799  1                  %o'066666'),
;  1800  1          PAT14 : vector [17] initial (16,                                ! PATTERN 14
;  1801  1                  %o'000001', %o'000002', %o'000004', %o'000010',
;  1802  1                  %o'000020', %o'000040', %o'000100', %o'000200',
;  1803  1                  %o'000400', %o'001000', %o'002000', %o'004000',
;  1804  1                  %o'010000', %o'020000', %o'040000', %o'100000'),
;  1805  1          PAT15 : vector [17] initial (16,                                ! PATTERN 15
;  1806  1                  %o'177776', %o'177775', %o'177773', %o'177767',
;  1807  1                  %o'177757', %o'177737', %o'177677', %o'177577',
;  1808  1                  %o'177377', %o'176777', %o'175777', %o'173777',
```

```
;  1809  1                %o'167777', %o'157777', %o'137777', %o'077777' ),
;  1810  1            PAT16 : vector [17] initial (16,                        ! PATTERN 16
;  1811  1                    rep 3 of (%o'133331'), rep 3 of (%o'155554'),
;  1812  1                    rep 2 of (%o'133331'), rep 2 of (%o'155554'),
;  1813  1                    %o'133331', %o'155554', %o'133331', %o'155554',
;  1814  1                    %o'133331', %o'155554'),
;  1815  1            PAT17 : vector [22] initial (21,                        ! PATTERN 17
;  1816  1                    %o'000000', rep 2 of (%o'106466'),
;  1817  1                    rep 3 of (%o'071311'), rep 4 of (%o'106466'),
;  1818  1                    rep 5 of (%o'071311'), rep 6 of (%o'106466')),
;  1819  1            PAT18 : vector [22] initial (21,                        ! PATTERN 18
;  1820  1                    %o'106466', %o'000000', %o'071311',
;  1821  1                    rep 3 of (%o'106466'), rep 4 of (%o'071311'),
;  1822  1                    rep 5 of (%o'106466'), rep 6 of (%o'071311')),
;  1823  1            PAT19 : vector [22] initial (21,                        ! PATTERN 19
;  1824  1                    %o'000000', rep 2 of (%o'134631'),
;  1825  1                    rep 3 of (%o'043146'), rep 4 of (%o'134631'),
;  1826  1                    rep 5 of (%o'043146'), rep 6 of (%o'134631')),
;  1827  1            PAT20 : vector [22] initial (21,                        ! PATTERN 20
;  1828  1                    %o'134631', %o'000000', %o'043146',
;  1829  1                    rep 3 of (%o'134631'), rep 4 of (%o'043146'),
;  1830  1                    rep 5 of (%o'134631'), rep 6 of (%o'043146')),
;  1831  1            PAT21 : vector [2] initial (1,                          ! PATTERN 21
;  1832  1                    %o'000000'),                                    !    (LBN)
;  1833  1            DPA_TBL : vector [DP_CNT] initial                       ! DATA PATTERN ADDRESS TABLE
;  1834  1                      (RDM_CNT, PAT02, PAT03, PAT04, PAT05,
;  1835  1                      PAT06, PAT07, PAT08, PAT09, PAT10, PAT11,
;  1836  1                      PAT12, PAT13, PAT14, PAT15, PAT16, PAT17,
;  1837  1                      PAT18, PAT19, PAT20, PAT21),
;  1838  1            BST_CNT : word initial (0),                             ! CURRENT SEQUENTIAL BLOCK COUNT
;  1839  1            BST_DEV : word initial (0),                             ! CURRENT SEQUENTIAL BLOCK DEVICE
;  1840  1            CURRENT_VECTOR : word,                                  ! CURRENT DEVICE'S VECTOR ADDRESS
;  1841  1        !ZZZ    BRLEVEL : word,                                     ! CURRENT DEVICE'S BR LEVEL                ZZZ
;  1842  1            DUOFF   : WORD,                                         !DUP OFFSET INTO CST                       ZZZ
;  1843  1            DRS_START,                                              ! START OF THE SUPERVISOR
;  1844  1            APT_MODE : byte initial (byte (FALSE)),                 ! FLAG SET IF EXERCISER RUNNING UNDER APT
;  1845  1            MAIL_BOX_TESTNUM,                                       ! ADDRESS OF TEST NUMBER LOCATION IN APT MAIL-BOX
;  1846  1            MAIL_BOX_SUBTST,                                        ! ADDRESS OF SUB-TEST NUMBER LOCATION IN APT MAIL BOX
;  1847  1            COMPARE_DATA : byte,                                    ! FLAG CLEARED TO BYPASS HOST COMPARES
;  1848  1            DRS_FLAGS: word,                                        ! FLAGS USED IN START/RESTART OF THE EXERCISER
;  1849  1            RD_MAX_SEQ_CNT : word,                                  ! COUNT USED IN SEQUENTIAL ACCESS OPERATIONS
;  1850  1            RX_MAX_SEQ_CNT : word;                                  !
;  1851  1
;  1852  1        external routine
;  1853  1            NEX_TRAP : L$1SR novalue,
;  1854  1            TIME : L$ISR novalue,
;  1855  1            SET_CPAR : novalue,
;  1856  1            SET_UPAR : novalue,
;  1857  1            OUT_IODQ,
;  1858  1            IN_IODQ : novalue,
;  1859  1            GET_PKT,
;  1860  1            PUT_PKT : novalue,
;  1861  1            GET_RETPKT,
```

```
;    1862  1          PUT_RETPKT : novalue,
;    1863  1          GET_IO_BUFF : novalue,
;    1864  1          PUT_IO_BUFF : novalue,
;    1865  1          PUTA_BUFF : novalue,
;    1866  1          SEND,
;    1867  1          WAIT : novalue,
;    1868  1          MODULAS,                                        !ZZZ
;    1869  1          DROP_CTLR : novalue,
;    1870  1          DRV_CTLERR : novalue,
;    1871  1          EMS_RP1 : novalue,
;    1872  1          EMS_EL : novalue,
;    1873  1          EMS_CMP : novalue,
;    1874  1          EMS_ERR : novalue,
;    1875  1          EMS_10 : novalue,
;    1876  1          EMS_12 : novalue,
;    1877  1          EMS_13 : novalue,
;    1878  1          EMS_14 : novalue,
;    1879  1          EMS_18 : novalue,
;    1880  1          EMS_21 : novalue,
;    1881  1          EMS_22 : NOVALUE,                               !ZZZ
;    1882  1          EMS_24 : novalue,
;    1883  1          EMS_30 : novalue;
```

```
;    1884  1     %sbttl 'TEST SECTION
;    1885  1
;    1886  1
;    1887  1     !+
;    1888  1     !       THIS SECTION CONTAINS THE TOP-LEVEL TEST CODE FOR THE RDRX EXERCISER.
;    1889  1     !       THE EXERCISER CONSISTS OF ONE TEST WHICH IS SUBDIVIDED INTO A NUMBER OF
;    1890  1     !       SUBTESTS. ALL SUBTESTS ARE DECLARED WITHIN THIS BLOCK.
;    1891  1     !
;    1892  1
;    1893  1
;    1894  3     BGNTST;
;    1895  3
;    1896  3     local
;    1897  3         DUMMY_0 : word,
;    1898  3         DUMMY_1 : word;
;    1899  3
;    1900  3
;    1901  3
;    1902  3     EOP_FLAG = TRUE;                                        ! ASSUME NO UNIT AVAILABLE
;    1903  3     COMPARE_DATA = TRUE;                                    ! ALLOW HOST COMAPRES IF ASKED FOR
;    1904  3     DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT);              !CLEAR DUP INIT FLAG     ZZZ
;    1905  3     HOE_FLAG = FALSE;                                       ! ASSUME 'HOE' FLAG NOT SET
;    1906  3     FORCED_ERROR = FALSE;                                   ! INITIALIZE "FORCED ERROR" FLAG
;    1907  3
;    1908  3
;    1909  3     incr I from 0 to PKT_CNT - 1 do                         ! INITIALIZE PACKET AREA
;    1910  4         begin
;    1911  4
;    1912  4         incr J from 0 to PKT_LEN - 1 do
;    1913  4             MSCP_PKT [.I, .J, 0, 16, 0] = 0;
;    1914  4
;    1915  4         MSCP_PKT [.I, RSP_RECEIVED] = FALSE;
;    1916  3         end;
;    1917  3
;    1918  3     incr I from 0 to RP_CNT - 1 do                          ! INITIALIZE RESPONSE SAVE AREA
;    1919  3         incr J from 0 to RP_LEN - 1 do
;    1920  3             RETPKT [.I, .J, 0, 16, 0] = 0;
;    1921  3
;    1922  3     incr I from 0 to EP_CNT do                              ! INITIALIZE ERROR-LOG SAVE AREA
;    1923  4         begin
;    1924  4
;    1925  4         incr J from 0 to EP_LEN - 1 do
;    1926  4             ELOG_PKT [.I, .J, 0, 16, 0] = 0;
;    1927  4
;    1928  4         ELOG_PKT [.I, EL_CONTENTS] = EMPTY;
;    1929  3         end;
;    1930  3
;    1931  4     if BIT_TST (SWP_FLAGS, SWF_CWC)                         ! NO SIMULTANEOUS CNTR/HOST WRIE CHECKS
;    1932  3     then
;    1933  3         SWP_FLAGS = .SWP_FLAGS and (not SWF_HWC);
;    1934  3
;    1935  4     if BIT_TST (SWP_FLAGS, SWF_RDM)                         ! NO SIMULTANEOUS RANDOM/SEQUENTIAL SELECTS
;    1936  3     then
```

```
;  1937  3          SWP_FLAGS = .SWP_FLAGS and (not SWF_SEQ);
;  1938  3
;  1939  3      if not .INIT_OCCURED
;  1940  3      then
;  1941  4          begin
;  1942  4          DRS_START = .FREE_MEM_ADDR + 2 + (..FREE_MEM ADDR + 2);          ! START OF SUPERVISOR
;  1943  4
;  1944  4
;  1945  4
;  1946  4      !-  THE FOLLOWING DETERMINES WHETHER THE TEST IS TO BE RUN IN APT MODE:                    !Z
ZZ
;  1947  4                                                                                               !Z
ZZ
;  1948  5      IF BIT_TST (SWP_FLAGS, SWF_APT)                                  !IF APT                 !Z
ZZ
;  1949  4          then
;  1950  5              begin
;  1951  5              APT_MODE = TRUE;
;  1952  5              MAIL_BOX_TESTNUM = .DRS_START + %o'62' + %o'6';          ! APT MAIL-BOX IS OFFSET AT OCTAL 62 FROM
;  1953  5              MAIL_BOX_SUBTST = .DRS_START + %o'62' + %o'4';           ! BEGINNING OF SUPERVISOR
;  1954  4              end;
;  1955  4
;  1956  4
;  1957  4          NEX = FALSE;                                                 ! CHECK IF LINE-CLOCK PRESENT
;  1958  4          CLK_PRESENT = FALSE;                                         !
;  1959  4          SETVEC (4, NEX_TRAP, PRIO7);                                 ! SET TRAP CATCHER ADDRESS
;  1960  4          DUMMY_0 = .LINE_CLOCK;                                       ! TRY TO ADDRESS THE CLOCK
;  1961  4          DUMMY_1 = 0;                                                 ! DUMMY INSTRUCTION
;  1962  4          CLRVEC (4);                                                  ! RETURN LOC 4 TO THE SUPERVISOR
;  1963  4
;  1964  4
;  1965  4          if not .NEX
;  1966  4          then
;  1967  5              begin
;  1968  5              CLK_PRESENT = TRUE;                                      ! SET FLAG IF CLOCK PRESENT
;  1969  5              CLK_TICKS = 0;                                           ! INITIALIZE THE LINE-CLOCK TICK COUNT
;  1970  5              HOURS = .SWP_TIME / 100;                                 ! TIME OF DAY (HOURS)
;  1971  5              MINUTES = (.SWP_TIME mod 100) + 1;                       ! TIME OF DAY (MINUTES)
;  1972  5
;  1973  5              while .MINUTES gequ 60 do                                ! NORMALIZE MINUTES
;  1974  6                  begin
;  1975  6                  MINUTES = .MINUTES - 60;
;  1976  6                  HOURS = .HOURS + 1;
;  1977  5                  end;
;  1978  5
;  1979  5              HOURS = .HOURS mod 24;                                   ! NORMALIZE HOURS
;  1980  4              end;
;  1981  4
;  1982  3          end;
;  1983  3
;  1984  3
;  1985  3      if .CLK_PRESENT
;  1986  3      then
;  1987  4          begin
;  1988  4          SETVEC (%o'100 , TIME, PRIO6);                               ! LINE-CLOCK VECTOR
;  1989  4          LINE CLOCK = BIT6;                                           ! START THE CLOCK
```

```
;    1990  3          end;
;    1991  3
;    1992  3      RFLAGS (DRS_FLAGS);                                    ! READ DRS FLAGS INTO LOC DRS FLAGS
;    1993  3
;    1994  3      if BIT_TST (DRS FLAGS, HOE) eql HOE
;    1995  3      then
;    1996  3          HOE FLAG = TRUE;                                   ! SET FLAG IF 'HOE  SET
;    1997  3
;    1998  3
;    1999  3      INIT TEST ();                                          ! INITIALIZE TEST ENVIRONMENT
;    2000  3
;    2001  3      incr CTLR from 0 to (MAX_CTLR - 1) do                  ! FOR EVERY CONTROLLER
;    2002  3
;    2003  3          if (.CST [.CTLR, STATE] eql ONLINE) and            ! IF CONTROLLER ONLINE
;    2004  3             (.DCT [.CTLR, STAT] eql ONLINE) and
;    2005  4             (.CST [.CTLR, U_CNT] gequ 0)
;    2006  3          then
;    2007  3              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + 4) by UNIT_SIZE do
;    2008  3
;    2009  3                  if .CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE   ! IF AT LEAST ONE UNIT ALIVE
;    2010  3                  then
;    2011  4                      begin
;    2012  4                      EOP_FLAG = FALSE;                      ! NOT END OF PASS
;    2013  4                      exitloop;
;    2014  3                      end;
;    2015  3
;    2016  3      if not .EOP_FLAG
;    2017  3      then
;    2018  3          MULTI_DRIVE ();                                    ! RUN MULTI-DRIVE TEST
;    2019  1      ENDTST;


                                        .TITLE  ZRQAM3 RD/RX EXERCISER
                                        .IDENT  /V02.1/
                                        .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  000001                                  PAT02:  .WORD   1
000130  000000                                          .WORD   0
000132  000001                                  PAT03:  .WORD   1
000134  177777                                          .WORD   -1
000136  000001                                  PAT04:  .WORD   1
000140  105613                                          .WORD   -72165
000142  000001                                  PAT05:  .WORD   1
000144  031463                                          .WORD   31463
000146  000001                                  PAT06:  .WORD   1
000150  030221                                          .WORD   30221
000152  000020                                  PAT07:  .WORD   20
000154  000001                                          .WORD   1
000156  000003                                          .WORD   3
000160  000007                                          .WORD   7
000162  000017                                          .WORD   17
000164  000037                                          .WORD   37
000166  000077                                          .WORD   77
000170  000177                                          .WORD   177
000172  000377                                          .WORD   377
000174  000777                                          .WORD   777
000176  001777                                          .WORD   1777
000200  003777                                          .WORD   3777
000202  007777                                          .WORD   7777
000204  017777                                          .WORD   17777
000206  037777                                          .WORD   37777
000210  077777                                          .WORD   77777
000212  177777                                          .WORD   -1
000214  000020                                  PAT08:  .WORD   20
000216  177776                                          .WORD   -2
000220  177774                                          .WORD   -4
000222  177770                                          .WORD   -10
000224  177760                                          .WORD   -20
000226  177740                                          .WORD   -40
000230  177700                                          .WORD   -100
000232  177600                                          .WORD   -200
000234  177400                                          .WORD   -400
000236  177000                                          .WORD   -1000
000240  176000                                          .WORD   -2000
000242  174000                                          .WORD   -4000
000244  170000                                          .WORD   -10000
000246  160000                                          .WORD   -20000
000250  140000                                          .WORD   -40000
000252  100000                                          .WORD   -100000
000254  000000                                          .WORD   0
000256  000020                                  PAT09:  .WORD   20
000260  000000                                          .WORD   0
```

```
000262  000000                          .WORD   0
000264  000000                          .WORD   0
000266  177777                          .WORD    1
000270  177777                          .WORD    1
000272  177777                          .WORD   -1
000274  000000                          .WORD   0
000276  000000                          .WORD   0
000300  177777                          .WORD   -1
000302  177777                          .WORD   -1
000304  000000                          .WORD   0
000306  177777                          .WORD   -1
000310  000000                          .WORD   0
000312  177777                          .WORD   -1
000314  000000                          .WORD   0
000316  177777                          .WORD   -1
000320  000001                  PAT10:  .WORD   1
000322  133331                          .WORD   -44447
000324  000020                  PAT11:  .WORD   20
000326  052525                          .WORD   52525
000330  052525                          .WORD   52525
000332  052525                          .WORD   52525
000334  125252                          .WORD   -52526
000336  125252                          .WORD   -52526
000340  125252                          .WORD   -52526
000342  052525                          .WORD   52525
000344  052525                          .WORD   52525
000346  125252                          .WORD   -52526
000350  125252                          .WORD   -52526
000352  052525                          .WORD   52525
000354  125252                          .WORD   -52526
000356  052525                          .WORD   52525
000360  125252                          .WORD   -52526
000362  052525                          .WORD   52525
000364  125252                          .WORD   -52526
000366  000024                  PAT12:  .WORD   24
000370  026455                          .WORD   26455
000372  026455                          .WORD   26455
000374  026455                          .WORD   26455
000376  151322                          .WORD   -26456
000400  151322                          .WORD   -26456
000402  151322                          .WORD   -26456
000404  026455                          .WORD   26455
000406  026455                          .WORD   26455
000410  151322                          .WORD   -26456
000412  151322                          .WORD   -26456
000414  026455                          .WORD   26455
000416  026455                          .WORD   26455
000420  151322                          .WORD   -26456
000422  026455                          .WORD   26455
000424  151322                          .WORD   -26456
000426  026455                          .WORD   26455
000430  151322                          .WORD   -26456
000432  026455                          .WORD   26455
```

```
000434  151322                          .WORD   -26456
000436  026455                          .WORD   26455
000440  000001                  PAT13:  .WORD   1
000442  066666                          .WORD   66666
000444  000020                  PAT14:  .WORD   20
000446  000001                          .WORD   1
000450  000002                          .WORD   2
000452  000004                          .WORD   4
000454  000010                          .WORD   10
000456  000020                          .WORD   20
000460  000040                          .WORD   40
000462  000100                          .WORD   100
000464  000200                          .WORD   200
000466  000400                          .WORD   400
000470  001000                          .WORD   1000
000472  002000                          .WORD   2000
000474  004000                          .WORD   4000
000476  010000                          .WORD   10000
000500  020000                          .WORD   20000
000502  040000                          .WORD   40000
000504  100000                          .WORD   -100000
000506  000020                  PAT15:  .WORD   20
000510  177776                          .WORD   -2
000512  177775                          .WORD   -3
000514  177773                          .WORD   -5
000516  177767                          .WORD   -11
000520  177757                          .WORD   -21
000522  177737                          .WORD   -41
000524  177677                          .WORD   -101
000526  177577                          .WORD   -201
000530  177377                          .WORD   -401
000532  176777                          .WORD   -1001
000534  175777                          .WORD   -2001
000536  173777                          .WORD   -4001
000540  167777                          .WORD   -10001
000542  157777                          .WORD   -20001
000544  137777                          .WORD   -40001
000546  077777                          .WORD   77777
000550  000020                  PAT16:  .WORD   20
000552  133331                          .WORD   -44447
000554  133331                          .WORD   -44447
000556  133331                          .WORD   -44447
000560  155554                          .WORD   -22224
000562  155554                          .WORD   -22224
000564  155554                          .WORD   -22224
000566  133331                          .WORD   -44447
000570  133331                          .WORD   -44447
000572  155554                          .WORD   -22224
000574  155554                          .WORD   -22224
000576  133331                          .WORD   -44447
000600  155554                          .WORD   -22224
000602  133331                          .WORD   -44447
000604  155554                          .WORD   -22224
```

```
000606  133331                          .WORD   -44447
000610  155554                          .WORD    22224
000612  000025                  PAT17:  .WORD   25
000614  000000                          .WORD   0
000616  106466                          .WORD   -71312
000620  106466                          .WORD   -71312
000622  071311                          .WORD   71311
000624  071311                          .WORD   71311
000626  071311                          .WORD   71311
000630  106466                          .WORD   -71312
000632  106466                          .WORD   -71312
000634  106466                          .WORD   -71312
000636  106466                          .WORD   -71312
000640  071311                          .WORD   71311
000642  071311                          .WORD   71311
000644  071311                          .WORD   71311
000646  071311                          .WORD   71311
000650  071311                          .WORD   71311
000652  106466                          .WORD   -71312
000654  106466                          .WORD   -71312
000656  106466                          .WORD   -71312
000660  106466                          .WORD   -71312
000662  106466                          .WORD   -71312
000664  106466                          .WORD   -71312
000666  000025                  PAT18:  .WORD   25
000670  106466                          .WORD   -71312
000672  000000                          .WORD   0
000674  071311                          .WORD   71311
000676  106466                          .WORD   -71312
000700  106466                          .WORD   -71312
000702  106466                          .WORD   -71312
000704  071311                          .WORD   71311
000706  071311                          .WORD   71311
000710  071311                          .WORD   71311
000712  071311                          .WORD   71311
000714  106466                          .WORD   -71312
000716  106466                          .WORD   -71312
000720  106466                          .WORD   -71312
000722  106466                          .WORD   -71312
000724  106466                          .WORD   -71312
000726  071311                          .WORD   71311
000730  071311                          .WORD   71311
000732  071311                          .WORD   71311
000734  071311                          .WORD   71311
000736  071311                          .WORD   71311
000740  071311                          .WORD   71311
000742  000025                  PAT19:  .WORD   25
000744  000000                          .WORD   0
000746  134631                          .WORD   -43147
000750  134631                          .WORD   -43147
000752  043146                          .WORD   43146
000754  043146                          .WORD   43146
000756  043146                          .WORD   43146
```

```
000760  134631                          .WORD   -43147
000762  134631                          .WORD   -43147
000764  134631                          .WORD   -43147
000766  134631                          .WORD   -43147
000770  043146                          .WORD   43146
000772  043146                          .WORD   43146
000774  043146                          .WORD   43146
000776  043146                          .WORD   43146
001000  043146                          .WORD   43146
001002  134631                          .WORD   -43147
001004  134631                          .WORD   -43147
001006  134631                          .WORD   -43147
001010  134631                          .WORD   -43147
001012  134631                          .WCRD   -43147
001014  134631                          .WORD   -43147
001016  000025                  PAT20:  .WORD   25
001020  134631                          .WORD   -43147
001022  000000                          .WORD   0
001024  043146                          .WORD   43146
001026  134631                          .WORD   -43147
001030  134631                          .WORD   -43147
001032  134631                          .WORD   -43147
001034  043146                          .WORD   43146
001036  043146                          .WORD   43146
001040  043146                          .WORD   43146
001042  043146                          .WORD   43146
001044  134631                          .WORD   -43147
001046  134631                          .WORD   -43147
001050  134631                          .WORD   -43147
001052  134631                          .WORD   -43147
001054  134631                          .WORD   -43147
001056  043146                          .WORD   43146
001060  043146                          .WORD   43146
001062  043146                          .WORD   43146
001064  043146                          .WORD   43146
001066  043146                          .WORD   43146
001070  043146                          .WORD   43146
001072  000001                  PAT21:  .WORD   1
001074  000000                          .WORD   0
001076  000000G                 DPA.TBL:.WORD   RDM.CNT
001100  000126'                          .WORD   PAT02
001102  000132'                          .WORD   PAT03
001104  000136'                          .WORD   PAT04
001106  000142'                          .WORD   PAT05
001110  000146'                          .WORD   PAT06
001112  000152'                          .WORD   PAT07
001114  000214'                          .WORD   PAT08
001116  000256'                          .WORD   PAT09
001120  000320'                          .WORD   PAT10
001122  000324'                          .WORD   PAT11
001124  000366'                          .WORD   PAT12
001126  000440'                          .WORD   PAT13
001130  000444'                          .WORD   PAT14
```

```
001132  000506                          .WORD   PAT15
001134  000550'                         .WORD   PAT16
001136  000612'                         .WORD   PAT17
001140  000666'                         .WORD   PAT18
001142  000742                          .WORD   PAT19
001144  001016                          .WORD   PAT20
001146  001072'                         .WORD   PAT21
001150  000000                  BST.CNT:.WORD   0
001152  000000                  BST.DEV:.WORD   0
001154                          CURRENT.VECTOR:
                                        .BLKW   1
001156                          DUOFF:  .BLKW   1
001160                          DRS.START:
                                        .BLKW   1
001162                          APT.MODE:
001162  000                             .BYTE   0
                                        .EVEN
001164                          MAIL.BOX.TESTNUM:
                                        .BLKW   1
001166                          MAIL.BOX.SUBTST:
                                        .BLKW   1
001170                          COMPARE.DATA:
                                        .BLKB   1
                                        .EVEN
001172                          DRS.FLAGS:
                                        .BLKW   1
001174                          RD.MAX.SEQ.CNT:
                                        .BLKW   1
001176                          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, FER.LBN
                                .GLOBL  FER.BC, INIT.OCCURED, ADDR.VECT.OK
                                .GLOBL  S.PATTERN, S.DUPPKT, P.INDEX, RD.COUNT
                                .GLOBL  BRLEVEL, D.FAIL, DBM12, DBM18
                                .GLOBL  DBM19, DBM20, DBM21, DBM22, DBM23
                                .GLOBL  DBM25, DBM26, DBM27, DBM29, DBM108
                                .GLOBL  DBM109, DBM111, DBM112, DBM120
```

```
                .GLOBL  DBM121, EH.0, EH.1, EH.2, EH.3
                .GLOBL  EH.4, EH.5, EH.6, EH.7, EH.8, EH.9
                .GLOBL  EH.10, EH.12, EH.13, MSG.02, MSG.03
                .GLOBL  EGS.02, EGD.10, EGD.11, EGD.12
                .GLOBL  EGD.13, EGD.14, EGD.15, EGD.16
                .GLOBL  EGD.17, EGD.18, EGD.19, EGD.20
                .GLOBL  EGD.21, EGD.22, EGD.23, EGD.24
                .GLOBL  EGH.30, DF.MSG, HRD.MSG, SFT.MSG
                .GLOBL  HRD.SUB, CRLF, SWP.ERROR, SWP.XFER
                .GLOBL  SWP.FLAGS, DUPROUND, SWP.RAT, SWP.DPAT
                .GLOBL  SWP.UCNT, SWP.TIME, SWP.UDPAT
                .GLOBL  L$LUN, L$UNIT, NEX.TRAP, TIME
                .GLOBL  SET.CPAR, SET.UPAR, OUT.IODQ, IN.IODQ
                .GLOBL  GET.PKT, PUT.PKT, GET.RETPKT, PUT.RETPKT
                .GLOBL  GET.IO.BUFF, PUT.IO.BUFF, PUTA.BUFF
                .GLOBL  SEND, WAIT, MODULAS, DROP.CTLR
                .GLOBL  DRV.CTLERR, EMS.RP1, EMS.EL, EMS.CMP
                .GLOBL  EMS.ERR, EMS.10, EMS.12, EMS.13
                .GLOBL  EMS.14, EMS.18, EMS.21, EMS.22
                .GLOBL  EMS.24, EMS.30

100000          BIT15==         -100000
040000          BIT14==         40000
020000          BIT13==         20000
010000          BIT12==         10000
004000          BIT11==         4000
002000          BIT10==         2000
001000          BIT09==         1000
000400          BIT08==         400
000200          BIT07==         200
000100          BIT06==         100
000040          BIT05==         40
000020          BIT04==         20
000010          BIT03==         10
000004          BIT02==         4
000002          BIT01==         2
000001          BIT00==         1
001000          BIT9==          1000
000400          BIT8==          400
000200          BIT7==          200
000100          BIT6==          100
000040          BIT5==          40
000020          BIT4==          20
000010          BIT3==          10
000004          BIT2==          4
000002          BIT1==          2
000001          BIT0==          1
000040          EF.START==      40
000037          EF.RESTART==    37
000036          EF.CONTINUE==   36
000035          EF.NEW==        35
000034          EF.PWR==        34
```

N5


```
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               ;                       1883
000004 112737 000001 000000G            MOVB    #1,EOP.FLAG             ;                       1902
000012 112737 000001 001170'            MOVB    #1,COMPARE.DATA         ;                       1903
000020 042737 000002 000000G            BIC     #2,DUP.FLAGS            ;                       1904
000026 105037 000000G                   CLRB    HOE.FLAG                ;                       1905
000032 105037 000000G                   CLRB    FORCED.ERROR            ;                       1906
000036 005002                           CLR     R2                      ; I                     1909
000040 010246                   1$:     MOV     R2,-(SP)                ; I,*                   1913
000042 012746 000043                    MOV     #43,-(SP)
000046 004737 000000G                   JSR     PC,BL$MUL
000052 005001                           CLR     R1                      ; J                     1912
000054 010003                   2$:     MOV     R0,R3                   ;                       1913
000056 060103                           ADD     R1,R3                   ; J,*
000060 006303                           ASL     R3
000062 005063 000000G                   CLR     MSCP.PKT(R3)
000066 005201                           INC     R1                      ; J                     1912
000070 020127 000042                    CMP     R1,#42                  ; J,*
000074 003767                           BLE     2$
000076 010216                           MOV     R2,(SP)                 ; I,*                   1915
000100 012746 000106                    MOV     #106,-(SP)
000104 004737 000000G                   JSR     PC,BL$MUL
000110 105060 000005G                   CLRB    MSCP.PKT+5(R0)
000114 062706 000006                    ADD     #6,SP                   ;                       1910
000120 005202                           INC     R2                      ; I                     1909
000122 020227 000013                    CMP     R2,#13                  ; I,*
000126 003744                           BLE     1$
```

```
000130  005002                          CLR     R2                              ; I                             1918
000132  005001                  3$:     CLR     R1                              ; J                             1919
000134  010200                  4$:     MOV     R2,R0                           ; I,*                           1920
000136  060100                          ADD     R1,R0                           ; J,*
000140  006300                          ASL     R0
000142  005060  000000G                 CLR     RETPKT(R0)
000146  005201                          INC     R1                              ; J                             1919
000150  020127  000025                  CMP     R1,#25                          ; J,*
000154  003767                          BLE     4$
000156  062702  000026                  ADD     #26,R2                          ; *,I                           1918
000162  020227  000232                  CMP     R2,#232                         ; I,*
000166  003761                          BLE     3$
000170  005002                          CLR     R2                              ; I                             1922
000172  010246                  5$:     MOV     R2,-(SP)                        ; I,*                           1926
000174  012746  000041                  MOV     #41,-(SP)
000200  004737  000000G                 JSR     PC,BL$MUL
000204  005001                          CLR     R1                              ; J                             1925
000206  010003                  6$:     MOV     R0,R3                           ;                               1926
000210  060103                          ADD     R1,R3                           ; J,*
000212  006303                          ASL     R3
000214  005063  000000G                 CLR     ELOG.PKT(R3)
000220  005201                          INC     R1                              ; J                             1925
000222  020127  000040                  CMP     R1,#40                          ; J,*
000226  003767                          BLE     6$
000230  010216                          MOV     R2,(SP)                         ; I,*                           1928
000232  012746  000102                  MOV     #102,-(SP)
000236  004737  000000G                 JSR     PC,BL$MUL
000242  105060  000001G                 CLRB    ELOG.PKT+1(R0)
000246  062706  000006                  ADD     #6,SP                           ;                               1923
000252  005202                          INC     R2                              ; I                             1922
000254  020227  000014                  CMP     R2,#14                          ; I,*
000260  003744                          BLE     5$
000262  032737  000020  000000G         BIT     #20,SWP.FLAGS                   ;                               1931
000270  001403                          BEQ     7$
000272  042737  000040  000000G         BIC     #40,SWP.FLAGS                   ;                               1933
000300  032737  000002  000000G 7$:     BIT     #2,SWP.FLAGS                    ;                               1935
000306  001403                          BEQ     8$
000310  042737  001000  000000G         BIC     #1000,SWP.FLAGS                 ;                               1937
000316  132737  000001  000000G 8$:     BITB    #1,INIT.OCCURED                 ;                               1939
000324  001143                          BNE     13$
000326  017700  000000G                 MOV     @FREE.MEM.ADDR,R0               ;                               1942
000332  006300                          ASL     R0
000334  063700  000000G                 ADD     FREE.MEM.ADDR,R0
000340  010037  001160'                 MOV     R0,DRS.START
000344  062737  000002  001160'         ADD     #2,DRS.START
000352  032737  000001  000000G         BIT     #1,SWP.FLAGS                    ;                               1948
000360  001417                          BEQ     9$
000362  112737  000001  001162'         MOVB    #1,APT.MODE                     ;                               1951
000370  013737  001160' 001164'         MOV     DRS.START,MAIL.BOX.TESTNUM      ;                               1952
000376  062737  000070  001164'         ADD     #70,MAIL.BOX.TESTNUM
000404  013737  001160' 001166'         MOV     DRS.START,MAIL.BOX.SUBTST       ;                               1953
000412  062737  000066  001166'         ADD     #66,MAIL.BOX.SUBTST
000420  005037  000000G         9$:     CLR     NEX                             ;                               1957
```

Cf,


```
000424  105037  000000G                        CLRB    CLK.PRESENT                     ;                       1958
000430  012746  000340                        MOV     #340, (SP)                      ;                       1959
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                           ;                       1962
000456  104436                                TRAP    36
000460  032737  000001  000000G               BIT     #1,NEX                          ;                       1965
000466  001060                                BNE     12$
000470  112737  000001  000000G               MOVB    #1,CLK.PRESENT                  ;                       1968
000476  005037  000000G                        CLR     CLK.TICKS                       ;                       1969
000502  013716  000000G                        MOV     SWP.TIME,(SP)                   ;                       1970
000506  012746  000144                        MOV     #144,-(SP)
000512  004737  000000G                        JSR     PC,BL$DIV
000516  110037  000000G                        MOVB    R0,HOURS
000522  013716  000000G                        MOV     SWP.TIME,(SP)                   ;                       1971
000526  012746  000144                        MOV     #144,-(SP)
000532  004737  000000G                        JSR     PC,BL$MOD
000536  010001                                MOV     R0,R1
000540  005201                                INC     R1
000542  110137  000000G                        MOVB    R1,MINUTES
000546  123727  000000G 000074        10$:    CMPB    MINUTES,#74                     ;                       1973
000554  103412                                BLO     11$
000556  005000                                CLR     R0                              ;                       1975
000560  153700  000000G                        BISB    MINUTES,R0
000564  162700  000074                        SUB     #74,R0
000570  110037  000000G                        MOVB    R0,MINUTES
000574  105237  000000G                        INCB    HOURS                           ;                       1976
000600  000762                                BR      10$                             ;                       1973
000602  005016                        11$:    CLR     (SP)                            ;                       1979
000604  113716  000000G                        MOVB    HOURS,(SP)
000610  012746  000030                        MOV     #30,-(SP)
000614  004737  000000G                        JSR     PC,BL$MOD
000620  110037  000000G                        MOVB    R0,HOURS
000624  062706  000006                        ADD     #6,SP                           ;                       1967
000630  062706  000010                12$:    ADD     #10,SP                          ;                       1941
000634  132737  000001  000000G       13$:    BITB    #1,CLK.PRESENT                  ;                       1985
000642  001416                                BEQ     14$
000644  012746  000300                        MOV     #300,-(SP)                      ;                       1988
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                   ;                       1989
000674  062706  000010                        ADD     #10,SP                          ;                       1987
000700  104421                        14$:    TRAP    21                              ;                       1992
000702  010037  001172'                        MOV     R0,DRS.FLAGS
000706  042700  077777                        BIC     #77777,R0                       ;                       1994
000712  020027  100000                        CMP     R0,#-100000
000716  001003                                BNE     15$
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,MOE.FLAG             ;                               1996
000744  004737  000000V                  17$:    JSR     PC,INIT.TEST            ;                               1999
000750  005002                                  CLR     R2                      ; CTLR                          2001
000752  010246                          18$:    MOV     R2,-(SP)                ; CTLR,*                        2003
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,*                        2004
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,*                        2009
001020  012746  000053                          MOV     #53,-(SP)
001024  004737  000000G                        JSR     PC,BL$MUL
001030  012701  000003                          MOV     #3,R1                   ; *,OFFSET                      2007
001034  010003                          19$:    MOV     R0,R3                   ;                               2009
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                ;                               2012
001056  000405                                  BR      21$                     ;                               2011
001060  062701  000012                  20$:    ADD     #12,R1                  ; *,OFFSET                      2007
001064  020127  000042                          CMP     R1,#42                  ; OFFSET,*
001070  003761                                  BLE     19$
001072  022626                          21$:    CMP     (SP)+,(SP)+
001074  005202                          22$:    INC     R2                      ; CTLR                          2001
001076  000243                                  .WORD CLV!CLC

001100  003724                                  BLE     18$
001102  132737  000001  000000G                BITB    #1,EOP.FLAG             ;                               2016
001110  001002                                  BNE     23$
001112  004737  000000V                        JSR     PC,MULTI.DRIVE          ;                               2018
001116  000207                          23$:    RTS     PC                      ;                               1883

; 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                  ;                               2018
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
```

```
;    2020  1     %sbttl 'INITIALIZATION TEST ROUTINES
;    2021  1
;    2022  1     GLOBAL routine INIT_TEST : novalue =
;    2023  1
;    2024  1     !+
;    2025  1     !       THE INITIALIZATION TEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
;    2026  1     !       DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
;    2027  1     !       BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI-DRIVE TEST
;    2028  1     !       OR THE DM EXERCISER.
;    2029  1     !
;    2030  1     !       BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
;    2031  1     !       INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
;    2032  1     !       MARKED OFFLINE IN THEIR DCT AND CST. FOR THOSE DEVICES WHICH SURVIVE
;    2033  1     !       THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
;    2034  1     !       EACH DISK VIA ROUTINE "ACCESS". THE INITIALIZATION TEST IS DEEMED A
;    2035  1     !       SUCCESS IF A BLOCK ON THE INNER TRACK OF EACH DISK CAN BE ACCESSED.
;    2036  1     !
;    2037  1
;    2038  2         begin
;    2039  2         DRIVER_INIT ();                             ! INIT DRIVER DATA AND DEVICES
;    2040  2
;    2041  2         incr CTLR from 0 to (MAX_CTLR - 1) do       ! FOR EACH CONTROLLER
;    2042  3             begin
;    2043  3             SET_CPAR (.CTLR);                       ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    2044  3
;    2045  3             if .CST_ADDR [STATE] eql ONLINE         ! IF CONTROLLER IS STILL ALIVE
;    2046  3             then                                    ! FOR EACH DISK
;    2047  3
;    2048  3                 incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    2049  3
;    2050  3                 if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    2051  3                    (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    2052  4                    (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2053  3                 then
;    2054  4                     begin
;    2055  4                     SET_UPAR (.OFFSET);             ! SET UP UNIT-RELATED DATA ITEMS
;    2056  5                     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)          !ZZZ
;    2057  4                        THEN ACCESS ();                                 !ZZZ
;    2058  4                                                !SKIP IF DUP CAUSED INIT ZZZ
;    2059  4
;    2060  3                     end;                            ! IF UNIT IS PRESENT AND ONLINE
;    2061  3
;    2062  2             end;                                    ! CONTROLLER LOOP
;    2063  2
;    2064  1         end;                                        ! ROUTINE INIT_TEST


                                .SBTTL  INIT.TEST INITIALIZATION TEST ROUTINES
000000  004137  000000G     INIT.TEST::
                            JSR     R1,$SAVE2                   ;                       2022
000004  004737  000000V             JSR     PC,DRIVER.INIT              ;                       2039
000010  005002                      CLR     R2                          ; CTLR                  2041
000012  010246              1$:     MOV     R2,-(SP)                    ; CTLR,*                2043
```

```
000014  004737  000000G                        JSR     PC,SET.CPAR
000020  013700  000000G                        MOV     CST.ADDR,R0             ;                       2045
000024  005760  000002                         TST     2(R0)
000030  100035                                 BPL     4$
000032  012701  000003                         MOV     #3,R1                   ; *,OFFSET              2048
000036  010100                        2$:      MOV     R1,R0                   ; OFFSET,*              2050
000040  006300                                 ASL     R0
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  032710  040000                         BIT     #40000,(P0)
000052  001417                                 BEQ     3$
000054  032710  020000                         BIT     #20000,(R0)             ;                       2051
000060  001414                                 BEQ     3$
000062  032710  010000                         BIT     #10000,(R0)             ;                       2052
000066  001011                                 BNE     3$
000070  010116                                 MOV     R1,(SP)                 ; OFFSET,*              2055
000072  004737  000000G                        JSR     PC,SET.UPAR
000076  032737  000002  000000G                BIT     #2,DUP.FLAGS            ;                       2056
000104  001002                                 BNE     3$
000106  004737  000000V                        JSR     PC,ACCESS               ;                       2057
000112  062701  000012                3$:      ADD     #12,R1                  ; *,OFFSET              2048
000116  020127  000041                         CMP     R1,#41                  ; OFFSET,*
000122  003745                                 BLE     2$
000124  005726                        4$:      TST     (SP)+                   ;                       2042
000126  005202                                 INC     R2                      ; CTLR                  2041
000130  000243                                 .WORD CLV!CLC

000132  003727                                 BLE     1$
000134  000207                                 RTS     PC                      ;                       2022

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

```
;    2065  1     GLOBAL routine DRIVER_INIT : novalue =
;    2066  1
;    2067  1     !+
;    2068  1     !       THIS ROUTINE IS EQUIVALENT IN FUNCTION TO THE INITIALIZATION ENTRY
;    2069  1     !       POINT OF A STANDARD DEVICE DRIVER. ITS RESPONSIBILITY IS TO INITIALIZE
;    2070  1     !       DRIVER DATA, AND TO BRING EACH RDRX CONTROLLER AND UNIT (DISK)
;    2071  1     !       ONLINE.
;    2072  1     !
;    2073  1
;    2074  2         begin
;    2075  2
;    2076  2         local
;    2077  2             PKT_ADDR;
;    2078  2
;    2079  2         PKT_ADDR = MSCP_PKT + 10;                                   ! ADDR (TEXT + 0) OF 1ST MSCP PKT
;    2080  2         NEXT_PKT_USE = 0;                                           ! NEXT PACKET TO ALLOCATE
;    2081  2
;    2082  2         incr COUNT from 0 to (PKT_CNT - 1) do                       ! FOR EACH MSCP PACKET
;    2083  3             begin
;    2084  3             PKT_USE [.COUNT] = -1;                                  ! MARK PACKET FREE
;    2085  3             MSCP_PKT [.COUNT, PKT_LO] = .PKT_ADDR;                  ! LOAD ADDR INTO BUFFER DESCRIPTOR
;    2086  3             MSCP_PKT [.COUNT, PKT_HI] = 0;
;    2087  3             MSCP_PKT [.COUNT, CONNID] = CID_DISK;                   ! SET CONNECTION ID TO MSCP ID
;    2088  3             PKT_ADDR = .PKT_ADDR + (PKT_LEN * 2);                   ! ADVANCE ADDR TO NEXT PACKET
;    2089  2             end;
;    2090  2
;    2091  2         incr CTLR from 0 to (MAX_CTLR - 1) do                       ! FOR EACH CONTROLLER
;    2092  2
;    2093  2             if .CST [.CTLR, IP_ADDR] neqa 0                         ! IF CONTROLLER IS PRESENT
;    2094  2             then
;    2095  3                 begin
;    2096  3                 SET_CPAR (.CTLR);                                   ! CURRENT CONTROLLER PARAMETERS
;    2097  3                 CURRENT_VECTOR = .CST_ADDR [VEC_ADDR];              ! CURRENT CONTROLLER'S VECTOR
;    2098  3                 BRLEVEL = .CST_ADDR [BR_LEV] + 5;                   ! SET CURRENT CONTROLLER'S BR LEVEL
;    2099  3                 CTLR_INIT ();                                       ! INIT DEVICE AND CTLR DATA
;    2100  3
;    2101  3                 if .DCT_ADDR [STAT] eql ONLINE                      ! IF CONTROLLER IS STILL ALIVE
;    2102  3                 then                                                ! FOR EACH DIAK UNIT
;    2103  3
;    2104  3                     incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    2105  3
;    2106  3                         if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and  ! IF UNIT EXISTS
;    2107  4                            (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2108  3                         then
;    2109  4                             begin
;    2110  4                             CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_0] = %o'40';  ! BLANK DEVICE NAME
;    2111  4                             CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_1] = %o'40';  !
;    2112  4                             CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_2] = %o'40';  !
;    2113  4                             CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_3] = %o'40';  !
;    2114  4                             SET_UPAR (.OFFSET);                     ! SET UP UNIT-RELATED DATA ITEMS
;    2115  4                             UNIT_INIT ();                           ! BRING UNIT ONLINE
;    2116  3                             end;                                    ! IF UNIT EXISTS
;    2117  3
```

```
;  2118  2                end;                                          ! IF CONTROLLER IS PRESENT
;  2119  2
;  2120  1          end;                                                ! ROUTINE DRIVER_INIT

                                        .SBTTL  DRIVER.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G          DRIVER.INIT::
                                        JSR     R1,$SAVE3               ;                                       2065
000004  012702  000012G                 MOV     #MSCP.PKT+12,R2         ; *,PKT.ADDR                            2079
000010  105037  000000G                 CLRB    NEXT.PKT.USE            ;                                       2080
000014  005001                          CLR     R1                      ; COUNT                                 2082
000016  112761  000377  000000G  1$:    MOVB    #377,PKT.USE(R1)        ; *,*(COUNT)                            2084
000024  010146                          MOV     R1,-(SP)                ; COUNT,*                               2085
000026  012746  000106                 MOV     #106,-(SP)
000032  004737  000000G                 JSR     PC,BL$MUL
000036  010260  000000G                 MOV     R2,MSCP.PKT(R0)         ; PKT.ADDR,*
000042  005060  000002G                 CLR     MSCP.PKT+2(R0)          ;                                       2086
000046  105060  000011G                 CLRB    MSCP.PKT+11(R0)         ;                                       2087
000052  062702  000106                 ADD     #106,R2                 ; *,PKT.ADDR                            2088
000056  022626                          CMP     (SP)+,(SP)+             ;                                       2083
000060  005201                          INC     R1                      ; COUNT                                 2082
000062  020127  000013                 CMP     R1,#13                  ; COUNT,*
000066  003753                          BLE     1$
000070  005003                          CLR     R3                      ; CTLR                                  2091
000072  010346                  2$:     MOV     R3,-(SP)                ; CTLR,*                                2093
000074  012746  000126                 MOV     #126,-(SP)
000100  004737  000000G                 JSR     PC,BL$MUL
000104  022626                          CMP     (SP)+,(SP)+
000106  005760  000000G                 TST     CST(R0)
000112  001503                          BEQ     6$
000114  010346                          MOV     R3,-(SP)                ; CTLR,*                                2096
000116  004737  000000G                 JSR     PC,SET.CPAR
000122  013700  000000G                 MOV     CST.ADDR,R0             ;                                       2097
000126  016037  000002  001154'         MOV     2(R0),CURRENT.VECTOR
000134  042737  177000  001154'         BIC     #177000,CURRENT.VECTOR
000142  005016                          CLR     (SP)                    ;                                       2098
000144  116016  000004                 MOVB    4(R0),(SP)
000150  012746  000005                 MOV     #5,-(SP)
000154  004737  000000G                 JSR     PC,BL$SHF
000160  010037  000000G                 MOV     R0,BRLEVEL
000164  004737  000000V                 JSR     PC,CTLR.INIT            ;                                       2099
000170  005777  000000G                 TST     @DCT.ADDR               ;                                       2101
000174  100051                          BPL     5$
000176  012701  000003                 MOV     #3,R1                   ; *,OFFSET                              2104
000202  013702  000000G          3$:    MOV     CST.ADDR,R2             ;                                       2106
000206  010100                          MOV     R1,R0                   ; OFFSET,*
000210  006300                          ASL     R0
000212  060200                          ADD     R2,R0
000214  032710  040000                 BIT     #40000,(R0)
000220  001432                          BEQ     4$
000222  032710  010000                 BIT     #10000,(R0)             ;                                       2107
000226  001027                          BNE     4$
000230  010100                          MOV     R1,R0                   ; OFFSET,*                              2110
```

J6


```
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)              ;                       2111
000252  010100                                  MOV     R1,R0                   ; OFFSET,*              2112
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)              ;                       2113
000274  010116                                  MOV     R1,(SP)                 ; OFFSET,*              2114
000276  004737  000000G                         JSR     PC,SET.UPAR
000302  004737  000000V                         JSR     PC,UNIT.INIT            ;                       2115
000306  062701  000012                  4$:     ADD     #12,R1                  ; *,OFFSET              2104
000312  020127  000041                          CMP     R1,#41                  ; OFFSET,*
000316  003731                                  BLE     3$
000320  022626                          5$:     CMP     (SP)+,(SP)+             ;                       2095
000322  005203                          6$:     INC     R3                      ; CTLR                  2091
000324  000243                                  .WORD CLV!CLC

000326  003661                                  BLE     2$
000330  000207                                  RTS     PC                      ;                       2065

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

```
;    2121  1     GLOBAL routine CTLR INIT : novalue =
;    2122  1
;    2123  1     !+
;    2124  1     !       THIS "DRIVER" ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONTROLLER
;    2125  1     !       CONFIGURED FOR TESTING. ITS GENERAL PURPOSE IS TO BRING THE RDRX ONLINE
;    2126  1     !       TO THE HOST. SPECIFICALLY, IT IS WRITTEN TO:
;    2127  1     !
;    2128  1     !       1. INITIALIZE DRIVER CONTROLLER DATA, INCLUDING THE DCT,
;    2129  1     !       2. SET UP THE DEVICE'S INTERRUPT VECTOR ADDRESS,
;    2130  1     !       3. PERFORM A REGISTER EXISTENCE TEST TO VERIFY THE DEVICE'S PRESENCE,
;    2131  1     !       4. PERFORM A VECTOR AND BR LEVEL TEST TO VERIFY THE DEVICE'S VECTOR
;    2132  1     !            ADDRESS AND INTERRUPT REQUEST LEVEL,
;    2133  1     !       5. DO A HARD INITIALIZATION (FOUR STEPS) ON THE DEVICE.
;    2134  1     !
;    2135  1     !       IF ANY OF THESE INITIAL TESTS FAIL, THEN ALL UNITS ASSOCIATED WITH THE
;    2136  1     !       DEVICE ARE DROPPED.
;    2137  1     !
;    2138  1
;    2139  2         begin
;    2140  2
;    2141  2         local
;    2142  2             RESULT : byte;
;    2143  2
;    2144  2         INI_CTLR_DAT ();                                        ! INITIALIZE CONTROLLER DATA
;    2145  2     !ZZZ    SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], PRIO4);    ! SET DEVICE'S ASSUMED VECTOR ADDRESS
;    2146  2         SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], .BRLEVEL);     ! SET DEVICE'S ASSUMED VECTOR ADDRESS    ZZZ
;    2147  2         DCT_ADDR [IG_INT] = TRUE;                               ! SET "IGNORE INTERRUPT" BIT
;    2148  2         L$LUN = .CST_ADDR [OF_UN + OF_DATA, D_UNIT];            ! GET FIRST UNIT NUMBER OF CONTROLLER
;    2149  2                                                                 ! (USED BY DRS FOR DEVICE-FATAL CTLR ERRORS)
;    2150  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
;    2151  2     THEN                                    !CAUSED INIT, SKIP THIS CODE   ZZZ
;    2152  2
;    2153  2         if REG_EXIST () eql FAILURE                             ! REGISTER EXISTENCE TEST
;    2154  2         then
;    2155  3             begin
;    2156  3             DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
;    2157  3             return;
;    2158  2             end;
;    2159  2
;    2160  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
;    2161  2     THEN                                    !CAUSED INIT, SKIP THIS CODE   ZZZ
;    2162  2
;    2163  2         if VEC_BR_TEST () eql FAILURE                           ! VECTOR ADDR AND BR LEVEL TEST
;    2164  2         then
;    2165  3             begin
;    2166  3             DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
;    2167  3             return;
;    2168  2             end;
;    2169  2
;    2170  2         RESULT = HARD_INIT ();                                  ! ATTEMPT HARD DEVICE INIT
;    2171  2         DCT_ADDR [IG_INT] = FALSE;                              ! CLAER "IGNORE INTERRUPT" BIT
;    2172  2
;    2173  2         if .RESULT eql SUCCESS                                  ! IF HARD INIT WAS SUCCESSFUL
```

```
;  2174  2          then
;  2175  3              begin
;  2176  3              ADDR_VECT_OK = TRUE;                        ! ADDRESS/VECTOR TEST PASSED
;  2177  3              INI_RRING ();                               ! INITIALIZE RESPONSE RING
;  2178  3              WRT_RDRX (RCSA, RC_ALL, SA_GO);             ! SET "GO" BIT (START CTLR POLLING)
;  2179  3
;  2180  3              if SET_CTLR_CHAR () eql SUCCESS             ! SET CONTROLLER CHARACTERISTICS
;  2181  3              then
;  2182  4                  begin
;  2183  4                  DCT_ADDR [STAT] = ONLINE;               ! MARK CONTROLLER ONLINE IN "DRIVER"
;  2184  4                  CST_ADDR [STATE] = ONLINE;              ! MARK CONTROLLER ONLINE IN "PROGRAM"
;  2185  3                  end;
;  2186  3              end
;  2187  3
;  2188  2          else                                            ! HARD INIT FAILED
;  2189  3              begin
;  2190  3              DROP_CTLR (.CCTLR, DU INIT);                ! DROP ALL CONTROLLER'S UNITS
;  2191  2              end;
;  2192  2
;  2193  1          end;                                            ! ROUTINE CTLR_INIT


                                .SBTTL  CTLR.INIT INITIALIZATION TEST ROUTINES
000000  010146                CTLR.INIT::
                                MOV     R1,-(SP)                    ;                       2121
000002  004737  000000V         JSR     PC,INI.CTLR.DAT             ;                       2144
000006  013746  000000G         MOV     BRLEVEL,-(SP)               ;                       2146
000012  013700  000000G         MOV     CCTLR,R0
000016  006300                  ASL     R0
000020  016046  000102'         MOV     INT.ADDR(R0),-(SP)
000024  013746  001154'         MOV     CURRENT.VECTOR,-(SP)
000030  012746  000003          MOV     #3,-(SP)
000034  104437                  TRAP    37
000036  052777  040000  000000G BIS     #40000,@DCT.ADDR            ;                       2147
000044  013700  000000G         MOV     CST.ADDR,R0                 ;                       2148
000050  016001  000006          MOV     6(R0),R1
000054  000301                  SWAB    R1
000056  042701  177760          BIC     #177760,R1
000062  010137  000000G         MOV     R1,L$LUN
000066  032737  000002  000000G BIT     #2,DUP.FLAGS                ;                       2150
000074  001025                  BNE     2$
000076  004737  000000V         JSR     PC,REG.EXIST                ;                       2153
000102  005700                  TST     R0
000104  001410                  BEQ     1$                          ;                       2156
000106  032737  000002  000000G BIT     #2,DUP.FLAGS                ;                       2160
000114  001015                  BNE     2$
000116  004737  000000V         JSR     PC,VEC.BR.TEST              ;                       2163
000122  005700                  TST     R0
000124  001011                  BNE     2$
000126  013716  000000G     1$: MOV     CCTLR,(SP)                  ;                       2166
000132  012746  000002          MOV     #2,-(SP)
000136  004737  000000G         JSR     PC,DROP.CTLR
000142  062706  000012          ADD     #12,SP                      ;                       2167
```

M6,


```
000146  000453                                  BR      5$                              ;                               2165
000150  004737  000000V                 2$:     JSR     PC,HARD.INIT                    ;                               2170
000154  110001                                  MOVB    R0,R1                           ; *,RESULT
000156  042777  040000  000000G                BIC     #40000,@DCT.ADDR                ;                               2171
000164  120127  000001                          CMPB    R1,#1                           ; RESULT,*                      2173
000170  001031                                  BNE     3$
000172  112737  000001  000000G                MOVB    #1,ADDR.VECT.OK                 ;                               2176
000200  004737  000000V                         JSR     PC,INI.RRING                    ;                               2177
000204  012701  000001                          MOV     #1,R1                           ; *,RC.REG                      2178
000210  013700  000000G                         MOV     RDRX.ADDR,R0
000214  010160  000002                          MOV     R1,2(R0)                        ; RC.REG,*
000220  004737  000000V                         JSR     PC,SET.CTLR.CHAR                ;                               2180
000224  020027  000001                          CMP     R0,#1
000230  001020                                  BNE     4$
000232  052777  100000  000000G                BIS     #100000,@DCT.ADDR               ;                               2183
000240  013700  000000G                         MOV     CST.ADDR,R0                     ;                               2184
000244  052760  100000  000002                  BIS     #100000,2(R0)
000252  000407                                  BR      4$                              ;                               2173
000254  013716  000000G                 3$:     MOV     CCTLR,(SP)                      ;                               2190
000260  012746  000002                          MOV     #2,-(SP)
000264  004737  000000G                         JSR     PC,DROP.CTLR
000270  005726                                  TST     (SP)+                           ;                               2189
000272  062706  000010                 4$:     ADD     #10,SP                          ;                               2139
000276  012601                         5$:     MOV     (SP)+,R1                        ;                               2121
000300  000207                                  RTS     PC

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

N6


```
;  2194  1     GLOBAL routine INI_CTLR_DAT : novalue =
;  2195  1
;  2196  1     !+
;  2197  1     !       THIS ROUTINE IS RESPONSIBLE FOR INITIALIZING ALL CONTROLLER-RELATED
;  2198  1     !       DATA IN THE "DRIVER" PORTION OF THE EXERCISER. THIS INCLUDES THE
;  2199  1     !       CONTROLLER'S DCT AND OUTSTANDING COMMAND LIST.
;  2200  1     !
;  2201  1     !       IMPLICIT INPUTS:
;  2202  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  2203  1     !               DCT_ADDR - ADDRESS OF CURENT CONTROLLER'S DCT
;  2204  1     !
;  2205  1
;  2206  2         begin
;  2207  2         DCT_ADDR [WORD0] = 0;                                          ! CLEAR FIRST DCT WORD
;  2208  2         DCT_ADDR [RR_BEG] = COMM_AREA + 8 + (.CCTLR * COMM_LEN * 2);   ! START OF RESPONSE RING
;  2209  2         DCT_ADDR [RR_END] = .DCT_ADDR [RR_BEG] + ((RRING_LEN - 1) * 4); ! LAST SLOT IN RESPONSE RING
;  2210  2         DCT_ADDR [CR_BEG] = .DCT_ADDR [RR_END] + 4;                    ! START OF COMMAND RING
;  2211  2         DCT_ADDR [CR_END] = .DCT_ADDR [CR_BEG] + ((CRING_LEN - 1) * 4); ! LAST SLOT IN COMMAND RING
;  2212  2         DCT_ADDR [RR_POLL] = .DCT_ADDR [RR_BEG];                       ! FIRST PRING SLOT TO POLL
;  2213  2         DCT_ADDR [CR_POLL] = DCT_ADDR [CR_NEXT] = .DCT_ADDR [CR_BEG];  ! CRING POLL AND NEXT COMMAND POINTERS
;  2214  1         end;


                                .SBTTL  INI.CTLR.DAT INITIALIZATION TEST ROUTINES
000000  004137  000000G    INI.CTLR.DAT::
                                JSR     R1,$SAVE2                       ;                       2194
000004  013701  000000G            MOV     DCT.ADDR,R1                     .                       2207
000010  005011                    CLR     (R1)
000012  012702  000004            MOV     #4,R2                           ;                       2208
000016  060102                    ADD     R1,R2
000020  013746  000000G            MOV     CCTLR,-(SP)
000024  012746  000050            MOV     #50,-(SP)
000030  004737  000000G            JSR     PC,BL$MUL
000034  062700  000010'            ADD     #COMM.AREA+10,R0
000040  010012                    MOV     R0,(R2)
000042  010061  000006            MOV     R0,6(R1)                        ;                       2209
000046  062761  000014  000006    ADD     #14,6(R1)
000054  012700  000010            MOV     #10,R0                          ;                       2210
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)                     ;                       2211
000076  062761  000014  000012    ADD     #14,12(R1)
000104  011261  000014            MOV     (R2),14(R1)                     ;                       2212
000110  011061  000020            MOV     (R0),20(R1)                     ;                       2213
000114  011061  000016            MOV     (R0),16(R1)
000120  022626                    CMP     (SP)+,(SP)+                     ;                       2206
000122  000207                    RTS     PC                              ;                       2194

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

```
;    2215  1     GLOBAL routine REG_EXIST =
;    2216  1     !+
;    2217  1     !       THIS .S THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VERIFY
;    2218  1     !       THE PRESENCE OF AN RQRX DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
;    2219  1     !       SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
;    2220  1     !       ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE S SA AND IP
;    2221  1     !       REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
;    2222  1     !       THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
;    2223  1     !       DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
;    2224  1     !
;    2225  2         begin
;    2226  2
;    2227  2         local
;    2228  2             DUMMY_0 : word,                         ! TEMP FOR READING SA AND IP
;    2229  2             DUMMY_1 : word;                         !
;    2230  2
;    2231  2         if .ENTRY_REASON eql NEW_PASS
;    2232  2         then
;    2233  2             return SUCCESS;                         ! SKIP TEST FOR NEXT PASS
;    2234  2
;    2235  2         OF_RC = 2;                                  ! SET UP TO READ SA FIRST
;    2236  2
;    2237  2         do
;    2238  3             begin
;    2239  3             NEX = FALSE;                            ! SET TO "TRAP NOT RECEIVED"
;    2240  3             SETVEC (4, NEX_TRAP, PRI07);            ! SET LOCATION 4 TRAP VECTOR ADDRESS
;    2241  3             DUMMY_0 = .(.RQRX_ADDR + .OF_RC);       ! READ REGISTER (THEN TRAP OR CONTINUE)
;    2242  3             DUMMY_1 = 0;                            ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
;    2243  3                                                     !       (TRAP RETURNS TO NEXT INSTRUCTION)
;    2244  3             CLRVEC (4);                             ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS
;    2245  3
;    2246  3             if .NEX                                 ! IF NEX TRAP OCCURRED
;    2247  3             then
;    2248  4                 begin
;    2249  4                 C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C ERR HRD] + 1;
;    2250  4
;    2251  4                 if .APT_MODE
;    2252  4                 then
;    2253  5                     begin
;    2254  5                     .MAIL_BOX_TESTNUM = 1;
;    2255  5                     .MAIL_BOX_SUBTST = 0;
;    2256  4                     end;
;    2257  4
;    2258  4                 ERRDF (10, EGD_10, EMS_10);         ! REGISTER EXISTENCE TEST FAILED
;    2259  4                 SETPRI (PRI00);                     ! LOWER PRIORITY
;    2260  4                 return FAILURE;
;    2261  4                 end
;    2262  3             else
;    2263  3                 OF_RC = .OF RC   2;                 ! SET UP FOR IP REG OR QUIT
;    2264  3
;    2265  3             end
;    2266  2         until .OF_RC lss 0;
;    2267  2
```

```
;   2268  2          return SUCCESS;
;   2269  1          end;


                                                .SBTTL  REG.EXIST INITIALIZATION TEST ROUTINES
000000  004137  000000G                  REG.EXIST::
                                                JSR     R1,$SAVE2                       ;                               2215
000004  123727  000000G 000005                  CMPB    ENTRY.REASON,#5                 ;                               2231
000012  001472                                  BEQ     4$                              ;                               2233
000014  012737  000002  000000G                MOV     #2,OF.RC                        ;                               2235
000022  005037  000000G                1$:      CLR     NEX                             ;                               2239
000026  012746  000340                          MOV     #340,-(SP)                      ;                               2240
000032  012746  000000G                          MOV     #NEX.TRAP, (SP)
000036  012746  000004                          MOV     #4,-(SP)
000042  012746  000003                          MOV     #3,-(SP)
000046  104437                                  TRAP    37
000050  013700  000000G                          MOV     RDRX.ADDR,R0                    ;                               2241
000054  063700  000000G                          ADD     OF.RC,R0
000060  011001                                  MOV     (R0),R1                         ; #,DUMMY.0
000062  005002                                  CLR     R2                              ; DUMMY.1                       2242
000064  012700  000004                          MOV     #4,R0                           ;                               2244
000070  104436                                  TRAP    36
000072  032737  000001  000000G                BIT     #1,NEX                          ;                               2246
000100  001427                                  BEQ     3$
000102  013700  000000G                          MOV     CCTLR,R0                        ;                               2249
000106  006300                                  ASL     R0
000110  105260  000000G                          INCB    C.ERR.TBL(R0)
000114  032737  000001  001162'                 BIT     #1,APT.MODE                     ;                               2251
000122  001405                                  BEQ     2$
000124  012777  000001  001164                 MOV     #1,@MAIL.BOX.TESTNUM            ;                               2254
000132  005077  001166'                         CLR     @MAIL.BOX.SUBTST                ;                               2255
000136  104455                        2$:       TRAP    55                              ;                               2258
000140  000012                                  .WORD   12
000142  000000G                                  .WORD   EGD.10
000144  000000G                                  .WORD   EMS.10
000146  005000                                  CLR     R0                              ;                               2259
000150  104441                                  TRAP    41
000152  062706  000010                          ADD     #10,SP                          ;                               2260
000156  000413                                  BR      5$                              ;                               2248
000160  162737  000002  000000G        3$:       SUB     #2,OF.RC                        ;                               2263
000166  062706  000010                          ADD     #10,SP                          ;                               2238
000172  005737  000000G                          TST     OF.RC                           ;                               2266
000176  002311                                  BGE     1$
000200  012700  000001                4$:       MOV     #1,R0                           ;                               2225
000204  000207                                  RTS     PC
000206  005000                        5$:       CLR     R0                              ;                               2215
000210  000207                                  RTS     PC

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

```
;    2270  1     GLOBAL routine VEC_BR_TEST =
;    2271  1
;    2272  1     !+
;    2273  1     !       THIS ROUTINE ATTEMPTS TO VERIFY (A) THAT THE R^RX VECTOR ADDRESS GIVEN
;    2274  1     !       BY THE USER DURING THE HW DIALOG IS VALID, AND (B) THAT THE
;    2275  1     !       USER-SPECIFIED BUS REQUEST LEVEL FOR THE DEVICE IS CORRECT. THE FIRST
;    2276  1     !       OBJECTIVE IS ACCOMPLISHED BY SETTING THE CPU PRIORITY TO 0 AND FORCING
;    2277  1     !       AN RDRX INTERRUPT. IF THE USER SPECIFIED AN INCORRECT VECTOR ADDRESS,
;    2278  1     !       THEN THE RESULT MAY BE UNPREDICTABLE. FOR THIS REASON, THE MESSAGE
;    2279  1     !       "FUNCTIONAL TEST STARTED" IS PRINTED BEFORE THE TEST, AND
;    2280  1     !       "EXERCISER STARTED" IS PRINTED AT ITS SUCCESSFUL CONCLUSION. IF
;    2281  1     !       EITHER "FUNCTIONAL TEST ..." OR "EXERCISER ..." DOES NOT APPEAR, THEN
;    2282  1     !       PROGRAM CONTROL IS ASSUMED LOST AND A FATAL TRAP IS LIKELY TO OCCUR. AT
;    2283  1     !       THIS POINT, THE EXERCISER MUST BE STARTED AGAIN.
;    2284  1     !
;    2285  1     !       IF THIS TEST SUCCEEDS, THEN THE BR LEVEL TEST IS RUN BY SETTING THE
;    2286  1     !       PROCESSOR PRIORITY TO THE ASSUMED INTERRUPT PRIORITY GIVEN BY THE
;    2287  1     !       USER. A FORCED INTERRUPT SHOULD NOT OCCURR. THEN, BY LOWERING THE
;    2288  1     !       PRIORITY BY ONE, THE DELAYED INTERRUPT SHOULD OCCUR.
;    2289  1     !
;    2290  1
;    2291  2         begin
;    2292  2
;    2293  2         if .ENTRY_REASON eql NEW_PASS
;    2294  2         then
;    2295  3             begin
;    2296  3             SETPRI (PRI00);                         ! LOWER PRIORITY
;    2297  3             return SUCCESS;                         ! SKIP TEST IF NEXT PASS
;    2298  2             end;
;    2299  2
;    2300  2         PRINTF (MSG_02);                            ! "FUNCTIONAL TEST STARTED"
;    2301  2
;    2302  2         if INT_GEN () eql FALSE                     ! FORCE AN INTERRUPT
;    2303  2         then
;    2304  3             begin                                   ! IF INTERRUPT DID NOT OCCUR
;    2305  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2306  3
;    2307  3             if .APT_MODE
;    2308  3             then
;    2309  4                 begin
;    2310  4                 .MAIL_BOX_TESTNUM = 1;
;    2311  4                 .MAIL_BOX_SUBTST = 0;
;    2312  3                 end;
;    2313  3
;    2314  3             ERRDF (11, EGD_11, 0);                  ! VECTOR TEST FAILED
;    2315  3             return FAILURE;
;    2316  3             end
;    2317  2         else
;    2318  3             begin                                   ! INTERRUPT DID OCCUR
;    2319  3             PRINTF (MSG_03);                        ! "EXERCISER STARTED"
;    2320  3             SETPRI (.BRLEVEL);                      ! SET PRIORITY TO ASSUMED BR LEVEL
;    2321  3
;    2322  3             if INT_GEN () eql FALSE                 ! FORCE AN INTERRUPT (SHOULD NOT OCCUR)
```

```
;    2323  3              then
;    2324  4                  begin                                   ! IF INTERRUPT DID NOT OCCUR
;    2325  4                  SETPRI (.BRLEVEL    %o'40');            ! LOWER PRIORITY BY 1
;    2326  4                  DELAY (1);                              ! WAIT
;    2327  4
;    2328  4                  if .DCT_ADDR [SA_SAVE] neq 0            ! IF INTERRUPT DID OCCUR (SA_SAVE WOULD BE NON ZERO)
;    2329  4                  then
;    2330  5                      begin
;    2331  5                      SETPRI (PRIOO);                     ! RESTORE PROCESSOR PRIORITY TO 0
;    2332  5                      return SUCCESS;                     ! ONLY SUCCESSFUL EXIT POINT
;    2333  4                      end;
;    2334  4
;    2335  3                  end;
;    2336  3
;    2337  2              end;
;    2338  2
;    2339  2          SETPRI (PRIOO);                                 ! COME HERE ONLY FOR BR TEST FAILURE
;    2340  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2341  2
;    2342  2          if .APT_MODE
;    2343  2          then
;    2344  3              begin
;    2345  3              .MAIL_BOX_TESTNUM = 1;
;    2346  3              .MAIL_BOX_SUBTST = 0;
;    2347  2              end;
;    2348  2
;    2349  2          ERROF (12, EGD_12, EMS 12);
;    2350  2          return FAILURE;
;    2351  1          end;


                                        .GLOBL  L$DLY


                                        .SBTTL  VEC.BR.TEST INITIALIZATION TEST ROUTINES
000000  010146                  VEC.BR.TEST::
                                        MOV     R1,-(SP)                ;                                       2270
000002  005746                          TST     -(SP)
000004  123727  000000G 000005         CMPB    ENTRY.REASON,#5         ;                                       2293
000012  001003                          BNE     1$
000014  005000                          CLR     R0                      ;                                       2296
000016  104441                          TRAP    41
000020  000504                          BR      8$                      ;                                       2295
000022  012746  000000G          1$:    MOV     #MSG.02,-(SP)           ;                                       2300
000026  012746  000001                  MOV     #1,-(SP)
000032  010600                          MOV     SP,R0                   ; SP,*
000034  104417                          TRAP    17
000036  004737  000000V                 JSR     PC,INT.GEN              ;                                       2302
000042  005700                          TST     R0
000044  001023                          BNE     3$
000046  013700  000000G                 MOV     CCTLR,R0                ;                                       2305
000052  006300                          ASL     R0
000054  105260  000000G                 INCB    C.ERR.TBL(R0)
```

```
000060  032737  000001  001162                  BIT     #1,APT.MODE                     ;                               2307
000066  001405                                  BEQ     2$
000070  012777  000001  001164'                MOV     #1,@MAIL.BOX.TESTNUM            ;                               2310
000076  005077  001166'                        CLR     @MAIL.BOX.SUBTST                ;                               2311
000102  104455                          2$:     TRAP    55                              ;                               2314
000104  000013                                  .WORD   13
000106  000000G                                 .WORD   EGD.11
000110  000000                                  .WORD   0
000112  000477                                  BR      11$                             ;                               2315
000114  012716  000000G                 3$:     MOV     #MSG.03,(SP)                    ;                               2319
000120  012746  000001                          MOV     #1,-(SP)
000124  010600                                  MOV     SP,R0                           ; SP,*
000126  104417                                  TRAP    17
000130  013700  000000G                         MOV     BRLEVEL,R0                      ;                               2320
000134  104441                                  TRAP    41
000136  004737  000000V                         JSR     PC,INT.GEN                      ;                               2322
000142  005700                                  TST     R0
000144  001035                                  BNE     9$
000146  013700  000000G                         MOV     BRLEVEL,R0                      ;                               2325
000152  162700  000040                          SUB     #40,R0
000156  104441                                  TRAP    41
000160  012701  000001                          MOV     #1,R1                           ; *,$$TMP2                      2326
000164  001411                          4$:     BEQ     7$
000166  013700  000000G                         MOV     L$DLY,R0                        ; *,$$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$:     MOV     DCT.ADDR,R0                     ;                               2328
000214  005760  000002                          TST     2(R0)
000220  001407                                  BEQ     9$
000222  005000                                  CLR     R0                              ;                               2331
000224  104441                                  TRAP    41
000226  062706  000006                          ADD     #6,SP                           ;                               2332
000232  012700  000001                  8$:     MOV     #1,R0                           ;                               2330
000236  000427                                  BR      12$
000240  005726                          9$:     TST     (SP)+                           ;                               2318
000242  005000                                  CLR     R0                              ;                               2339
000244  104441                                  TRAP    41
000246  013700  000000G                         MOV     CCTLR,R0                        ;                               2340
000252  006300                                  ASL     R0
000254  105260  000000G                         INCB    C.ERR.TBL(R0)
000260  032737  000001  001162'                BIT     #1,APT.MODE                     ;                               2342
000266  001405                                  BEQ     10$
000270  012777  000001  001164'                MOV     #1,@MAIL.BOX.TESTNUM            ;                               2345
000276  005077  001166'                        CLR     @MAIL.BOX.SUBTST                ;                               2346
000302  104455                          10$:    TRAP    55                              ;                               2349
000304  000014                                  .WORD   14
000306  000000G                                 .WORD   EGD.12
000310  000000G                                 .WORD   EMS.12
000312  022626                          11$:    CMP     (SP)+,(SP)+                     ;                               2350
```

G7


```
000314  005000                          CLR     R0                              ;                               2270
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
```

```
;  2352  1     GLOBAL routine INT_GEN =
;  2353  1
;  2354  1     !+
;  2355  1     !       THIS ROUTINE BEGINS AN RDRX INITIALIZATION SEQUENCE, BUT ONLY
;  2356  1     !       COMPLETES THROUGH THE STEP 1 WRITE. ITS PURPOSE IS TO CREATE AN RDRX
;  2357  1     !       INTERRUPT (AT THE COMPLETEION OF STEP 1) IN ORDER TO HELP VERIFY THE
;  2358  1     !       THE USER-SPECIFIED VECTOR ADDRESS AND BUS REQUEST INTERRUPT LEVEL.
;  2359  1     !       A VALUE OF "TRUE" IS RETURNED TO THE CALLER IF AN INTERRUPT OCCURS,
;  2360  1     !       AND "FALSE" OTHERWISE. THE INTERRUPT IS VERIFIED BY A NON-ZERO VALUE
;  2361  1     !       IN THE "SA SAVE" WORD IN THE DEVICE'S DCT.
;  2362  1     !
;  2363  1
;  2364  2          begin
;  2365  2
;  2366  2          local
;  2367  2              SA : word;                                  ! STORAGE FOR STEP 1 READ AND WRITE
;  2368  2
;  2369  2          DCT_ADDR [SA_SAVE] = 0;                         ! ZERO OUT SA SAVE WORD IN DCT
;  2370  2          WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
;  2371  2          DELAY (2);                                      ! WAIT
;  2372  2          INCR COUNT FROM 1 TO 500 DO                     !MAKE SURE WE GET INTO STEP 1  ZZZ
;  2373  3              BEGIN                                       !BEFORE STEP 1 WRITE           ZZZ
;  2374  3              SA = .RDRX_ADDR [RCSA, RC_ALL];             ! STEP 1 READ
;  2375  3                IF (.SA AND S1_MASK) EQL SA_S1            !DID WE GET THE S1 BIT?        ZZZ
;  2376  3                THEN                                      !                              ZZZ
;  2377  3                EXITLOOP;                                 !EXIT IF SO                    ZZZ
;  2378  3              DELAY (1);                                  !                              ZZZ
;  2379  2              END;                                        !                              ZZZ
;  2380  2
;  2381  2         SA = (WR_RING ^ 8) or (.CURRENT_VECTOR ^ -2) or SA_INT;     ! STEP 1 WRITE VALUE
;  2382  2          WRT_RDRX (RCSA, RC_ALL, .SA);                  ! STEP 1 WRITE
;  2383  2
;  2384  2          incr COUNT from 1 to 8000 do
;  2385  3              begin
;  2386  3              DELAY (1);                                  ! TOTAL DELAY COUNT OF 8,000
;  2387  3
;  2388  3              if .DCT_ADDR [SA_SAVE] neq 0                ! IF SA WAS CHANGED
;  2389  3              then
;  2390  3                  return TRUE;                            ! INTERRUPT OCCURED
;  2391  3
;  2392  3              BREAK;
;  2393  2              end;
;  2394  2
;  2395  2          return FALSE;                                   ! IF INTERRUPT DID NOT OCCUR
;  2396  1          end;


                                        .SBTTL  INT.GEN INITIALIZATION TEST ROUTINES
000000  004137  000000G          INT.GEN::
                                        JSR     R1,$SAVE4                       ;                       2352
000004  024646                          CMP     -(SP),-(SP)
000006  013700  000000G                 MOV     DCT.ADDR,R0                     ;                       2369
000012  005060  000002                  CLR     2(R0)
```

```
000016  012700  177777                  MOV     #-1,R0          ; #-,RC.REG             2370
000022  010077  0000006G                MOV     R0,@RDRX.ADDR   ; RC.REG,*
000026  012701  000002                  MOV     #2,R1           ; #,$$TMP2              2371
000032  001411                  1$:     BEQ     4$
000034  013700  0000006G                MOV     L$DLY,R0        ; #,$$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              . $$TMP2
000054  000766                          BR      1$
000056  013702  0000006G        4$:     MOV     RDRX.ADDR,R2    ;                       2374
000062  012703  000764                  MOV     #764,R3         ; #,COUNT               2372
000066  016216  000002          5$:     MOV     2(R2),(SP)      ; #,RC.REG              2374
000072  011604                          MOV     (SP),R4         ; RC.REG,SA
000074  010400                          MOV     R4,R0           ; SA,#                  2375
000076  042700  001777                  BIC     #1777,R0
000102  020027  004000                  CMP     R0,#4000
000106  001416                          BEQ     10$             ;                       2377
000110  012701  000001                  MOV     #1,R1           ; #,$$TMP2              2378
000114  001411                  6$:     BEQ     9$
000116  013700  0000006G                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$:     DEC     R3              ; COUNT                 2372
000142  001351                          BNE     5$
000144  013700  001154          10$:    MOV     CURRENT.VECTOR,R0 ;                     2381
000150  006200                          ASR     R0
000152  006200                          ASR     R0
000154  010004                          MOV     R0,R4           ; #,SA
000156  052704  111200                  BIS     #111200,R4      ; #,SA
000162  010401                          MOV     R4,R1           ; SA,RC.REG             2382
000164  010162  000002                  MOV     R1,2(R2)        ; RC.REG,*
000170  012702  017500                  MOV     #17500,R2       ; #,COUNT               2384
000174  012701  000001          11$:    MOV     #1,R1           ; #,$$TMP2              2386
000200  001411                  12$:    BEQ     15$
000202  013700  0000006G                MOV     L$DLY,R0        ; #,$$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  0000006G        15$:    MOV     DCT.ADDR,R0     ;                       2388
000230  005760  000002                  TST     2(R0)
000234  001403                          BEQ     16$
000236  012700  000001                  MOV     #1,R0           ;                       2390
000242  000404                          BR      17$
000244  104422                  16$:    TRAP    22
```

J7


```
000246  005302                          DEC     R2                      ; COUNT                 2384
000250  001351                          BNE     11$
000252  005000                          CLR     R0                      ;                       2364
000254  022626                  17$:    CMP     (SP)+,(SP)+             ;                       2352
000256  000207                          RTS     PC

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

```
;    2397  1     GLOBAL routine HARD INIT =
;    2398  1
;    2399  1     !+
;    2400  1     !       THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
;    2401  1     !       INITIALIZE AN RDRX DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
;    2402  1     !       THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
;    2403  1     !       OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
;    2404  1     !       IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
;    2405  1     !
;    2406  1
;    2407  2         begin
;    2408  2
;    2409  2         local
;    2410  2             IE_VEC : word;                                  ! IE-BIT-AND-VECTOR ADDRESS/4 BYTE
;    2411  2                                                             !   (USED IN STEP 1 WRITE AND STEP 3 READ)
;    2412  2
;    2413  2         IE_VEC = .CURRENT_VECTOR + -2;                      ! GET VECTOR ADDR/4 (IE = 0)
;    2414  2
;    2415  2         incr ATTEMPTS from 1 to INI_ATT do
;    2416  3             begin
;    2417  3
;    2418  3             label
;    2419  3                 STEP_1_READ,
;    2420  3                 STEP_2_READ,
;    2421  3                 STEP_3_READ,
;    2422  3                 STEP_4_READ;
;    2423  3
;    2424  3             WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
;    2425  3     !
;    2426  3     !       STEP 1 READ
;    2427  3     !
;    2428  3             STEP = 1;
;    2429  3             STEP_1_READ:
;    2430  4                 begin
;    2431  4
;    2432  4                 incr COUNT from 1 to 500 do
;    2433  5                     begin
;    2434  5                     DELAY (1);                              ! TOTAL DELAY COUNT OF 500 FOR STEP 1
;    2435  5                     SA_REG = .RDRX_ADDR [RCSA, RC_ALL];     ! READ SA
;    2436  5
;    2437  5                     if (.SA_REG and S1_MASK) eql SA_S1      ! IF STEP 1 READ IS O.K.
;    2438  5                     then
;    2439  5                         leave STEP_1_READ;
;    2440  5
;    2441  5                     BREAK;
;    2442  4                     end;
;    2443  4
;    2444  4                 exitloop;
;    2445  3                 end;
;    2446  3
;    2447  3     !
;    2448  3     !       STEP 1 WRITE
;    2449  3     !
```

```
;    2450  3              SA_REG = (WR_RING ^ 8) or .IE_VEC;                    ! STEP 1 WRITE VALUE
;    2451  3              WRT_RDRX (RCSA, RC ALL, .SA_REG);                     ! STEP 1 WRITE
;    2452  3      !
;    2453  3      !       STEP 2 READ
;    2454  3      !
;    2455  3              STEP = .STEP + 1;
;    2456  3              STEP_2_READ:
;    2457  4                  begin
;    2458  4
;    2459  4                  incr COUNT from 1 to 10000 do
;    2460  5                      begin
;    2461  5                      DELAY (1);                                    ! TOTAL DELAY COUNT OF 10,000 FOR STEP 2
;    2462  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];           ! READ SA
;    2463  5
;    2464  6                      if (.SA_REG and S2_MASK) eql (SA_S2 or WR_RING) ! IF STEP 2 READ IS O.K.
;    2465  5                      then
;    2466  5                          leave STEP_2_READ;
;    2467  5
;    2468  5                      BREAK;
;    2469  4                      end;
;    2470  4
;    2471  4                  exitloop;
;    2472  3                  end;
;    2473  3
;    2474  3      !
;    2475  3      !       STEP 2 WRITE
;    2476  3      !
;    2477  3              WRT_RDRX (RCSA, RC_ALL, .DCT_ADDR [RR_BEG]);          ! RINGBASE LO, PI - 0
;    2478  3      !
;    2479  3      !       STEP 3 READ
;    2480  3      !
;    2481  3              STEP = .STEP + 1;
;    2482  3              STEP_3_READ:
;    2483  4                  begin
;    2484  4
;    2485  4                  incr COUNT from 1 to 10000 do
;    2486  5                      begin
;    2487  5                      DELAY (1);                                    ! TOTAL DELAY COUNT OF 10,000 FOR STEP 3 READ
;    2488  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];           ! READ SA
;    2489  5
;    2490  6                      if (.SA_REG and S3_MASK) eql (SA_S3 or .IE_VEC) ! IF STEP 3 READ IS O.K.
;    2491  5                      then
;    2492  5                          leave STEP_3_READ;
;    2493  5
;    2494  5                      BREAK;
;    2495  4                      end;
;    2496  4
;    2497  4                  exitloop;
;    2498  3                  end;
;    2499  3
;    2500  3      !
;    2501  3      !       STEP 3 WRITE
;    2502  3      !
```

```
;    2503  3              WRT_RDRX (RCSA, RC_ALL, 0);                          ! PP, RINGBASE-HI = 0
;    2504  3      !
;    2505  3      !       STEP 4 READ
;    2506  3      !
;    2507  3              STEP = .STEP + 1;
;    2508  3              STEP_4_READ:
;    2509  4                  begin
;    2510  4
;    2511  4                  incr COUNT from 1 to 10000 do
;    2512  5                      begin
;    2513  5                      DELAY (1);                                   ! TOTAL DELAY COUNT OF 10,000 FOR STEP 4 READ
;    2514  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];          ! READ SA
;    2515  5
;    2516  5                      if (.SA_REG and S4_MASK) eql SA_S4           ! IF STEP 4 READ IS O.K.
;    2517  5                      then
;    2518  5                          leave STEP_4_READ;
;    2519  5
;    2520  5                      BREAK;
;    2521  4                      end;
;    2522  4
;    2523  4                  exitloop;
;    2524  3                  end;
;    2525  3      !
;    2526  3      !       STEP 4 WRITE
;    2527  3      !
;    2528  3              CREDIT_BAL = 1;                                      ! START WITH A CREDIT BALANCE = 1
;    2529  3              WRT_RDRX (RCSA, RC_ALL, 0);                          ! BURST, LF, GO = 0
;    2530  3              return SUCCESS;                                      ! SUCCESS EXIT POINT
;    2531  3
;    2532  2                  end;                                             ! TRY AGAIN OR GIVE UP
;    2533  2
;    2534  2          CREDIT_BAL = 0;                                          ! NO CREDIT BALANCE
;    2535  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2536  2
;    2537  2          if .APT_MODE
;    2538  2          then
;    2539  3              begin
;    2540  3              .MAIL_BOX_TESTNUM = 1;
;    2541  3              .MAIL_BOX_SUBTST = 0;
;    2542  2              end;
;    2543  2
;    2544  2          ERRDF (13, EGD_13, EMS_13);                              ! INIT SEQUENCE FAILED
;    2545  2          return FAILURE;
;    2546  1          end;                                                     ! ROUTINE HARD INIT


                                        .SBTTL  HARD.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G          HARD.INIT::
                                        JSR     R1,$SAVE5                    ;                                  2397
000004  162706  000012                  SUB     #12,SP
000010  013704  001154'                 MOV     CURRENT.VECTOR,R4            ; #,IE.VEC                         2413
000014  006204                          ASR     R4                           ; IE.VEC
000016  006204                          ASR     R4                           ; IE.VEC
```

```
000020  012705  000002                        MOV     #2,R5                   ; *,ATTEMPTS            2415
000024  012700  177777                        MOV     # 1,R0                 ; *,RC.REG              2424
000030  010077  000000G                       MOV     R0,@RDRX.ADDR           ; RC.REG,*
000034  012737  000001  000000G               MOV     #1,STEP                 ;                       2428
000042  012702  000764                        MOV     #764,R2                 ; *,COUNT               2432
000046  012701  000001              1$:       MOV     #1,R1                   ; *,$$TMP2              2434
000052  001411                      2$:       BEQ     5$
000054  013700  000000G                       MOV     L$DLY,R0                ; *,$$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                      ; $$TMP2
000074  000766                              BR      2$
000076  013700  000000G             5$:       MOV     RDRX.ADDR,R0            ;                       2435
000102  016016  000002                        MOV     2(R0),(SP)              ; *,RC.REG
000106  011637  000000G                       MOV     (SP),SA.REG             ; RC.REG,*
000112  011600                              MOV     (SP),R0                 ; SA.REG,*              2437
000114  042700  001777                        BIC     #1777,R0
000120  020027  004000                        CMP     R0,#4000
000124  001404                              BEQ     6$                      ;                       2439
000126  104422                              TRAP    22
000130  005302                              DEC     R2                      ; COUNT                 2432
000132  001345                              BNE     1$
000134  000532                              BR      18$                     ;                       2416
000136  010437  000000G             6$:       MOV     R4,SA.REG               ; IE.VEC,*              2450
000142  052737  111000  000000G               BIS     #111000,SA.REG
000150  013701  000000G                       MOV     SA.REG,R1               ; *,RC.REG              2451
000154  013700  000000G                       MOV     RDRX.ADDR,R0
000160  010160  000002                        MOV     R1,2(R0)                ; RC.REG,*
000164  005237  000000G                       INC     STEP                    ;                       2455
000170  012702  023420                        MOV     #23420,R2               ; *,COUNT               2459
000174  012701  000001              7$:       MOV     #1,R1                   ; *,$$TMP2              2461
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$:      MOV     RDRX.ADDR,R0            ;                       2462
000230  016066  000002  000002               MOV     2(R0),2(SP)             ; *,RC.REG
000236  016637  000002  000000G               MOV     2(SP),SA.REG            ; RC.REG,*
000244  016600  000002                        MOV     2(SP),R0                ; SA.REG,*              2464
000250  042700  003400                        BIC     #3400,R0
000254  020027  010222                        CMP     R0,#10222
000260  001404                              BEQ     12$                     ;                       2466
000262  104422                              TRAP    22
000264  005302                              DEC     R2                      ; COUNT                 2459
000266  001342                              BNE     7$
000270  000537                              BR      26$                     ;                       2416
000272  013700  000000G             12$:      MOV     DCT.ADDR,R0             ;                       2477
```

```
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                    ;                       2481
000316  010403                                MOV     R4,R3                   ; IE.VEC,*              2490
000320  052703  020000                        BIS     #20000,R3
000324  012702  023420                        MOV     #23420,R2               ; *,COUNT               2485
000330  012701  000001              13$:      MOV     #1,R1                   ; *,$$TMP2              2487
000334  001411                      14$:      BEQ     17$
000336  013700  000000G                       MOV     L$DLY,R0                ; *,$$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                      ; $$TMP2
000356  000766                                BR      14$
000360  013700  000000G             17$:      MOV     RDRX.ADDR,R0            ;                       2488
000364  016066  000002  000004               MOV     2(R0),4(SP)             ; *,RC.REG
000372  016637  000004  000000G               MOV     4(SP),SA.REG            ; RC.REG,*
000400  016600  000004                        MOV     4(SP),R0                ; SA.REG,*              2490
000404  042700  003400                        BIC     #3400,R0
000410  020003                                CMP     R0,R3
000412  001404                                BEQ     19$                     ;                       2492
000414  104422                                TRAP    22
000416  005302                                DEC     R2                      ; COUNT                 2485
000420  001343                                BNE     13$
000422  000462                      18$:      BR      26$                     ;                       2416
000424  013700  000000G             19$:      MOV     RDRX.ADDR,R0            ;                       2503
000430  005060  000002                        CLR     2(R0)
000434  005237  000000G                       INC     STEP                    ;                       2507
000440  012703  023420                        MOV     #23420,R3               ; *,COUNT               2511
000444  012701  000001              20$:      MOV     #1,R1                   ; *,$$TMP2              2513
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$:      MOV     RDRX.ADDR,R0            ;                       2514
000500  016066  000002  000006               MOV     2(R0),6(SP)             ; *,RC.REG
000506  016637  000006  000000G               MOV     6(SP),SA.REG            ; RC.REG,*
000514  016600  000006                        MOV     6(SP),R0                ; SA.REG,*              2516
000520  042700  003777                        BIC     #3777,R0
000524  020027  040000                        CMP     R0,#40000
000530  001404                                BEQ     25$                     ;                       2518
000532  104422                                TRAP    22
000534  005303                                DEC     R3                      ; COUNT                 2511
000536  001342                                BNE     20$
000540  000413                                BR      26$                     ;                       2416
000542  012737  000001  000000G     25$:      MOV     #1,CREDIT.BAL           ;                       2528
000550  005001                                CLR     R1                      ; RC.REG                2529
```

C8


```
000552  013700  000000G                          MOV     RDRx.ADDR,R0
000556  005060  000002                           CLR     2(R0)
000562  012700  000001                           MOV     #1,R0                           ;                       2416
000566  000425                                   BR      28$
000570  005037  000000G                 26$:     CLR     CREDIT.BAL                      ;                       2534
000574  013700  000000G                          MOV     CCTLR,R0                        ;                       2535
000600  006300                                   ASL     R0
000602  105260  000000G                          INCB    C.ERR.TBL(R0)
000606  032737  000001  001162                   BIT     #1,APT.MODE                     ;                       2537
000614  001405                                   BEQ     27$
000616  012777  000001  001164                   MOV     #1,@MAIL.BOX.TESTNUM            ;                       2540
000624  005077  001166                           CLR     @MAIL.BOX.SUBTST                ;                       2541
000630  104455                          27$:     TRAP    55                              ;                       2544
000632  000015                                   .WORD   15
000634  000000G                                  .WORD   EGD.13
000636  000000G                                  .WORD   EMS.13
000640  005000                                   CLR     R0                              ;                       2407
000642  062706  000012                  28$:     ADD     #12,SP                          ;                       2397
000646  000207                                   RTS     PC

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

```
;    2547  1     GLOBAL routine INI_RRING : novalue =

;    2548  1
;    2549  1     !+
;    2550  1     !       THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP PACKETS TO
;    2551  1     !       FILL AN RQRX RESPONSE RING. THE BUFFER DESCRIPTOR OF EACH PACKET
;    2552  1     !       (LOCATED IN FRONT OF THE PACKET ITSELF) IS LOADED INTO SUCCESSIVE
;    2553  1     !       RRING SLOTS. NOTE THAT THE BUFFER DESCRIPTORS HAVE BEEN INITIALIZED
;    2554  1     !       WITH THE FLAG AND OWNERSHIP BITS SET TO "1", MAKING EACH SLOT
;    2555  1     !       CONTROLLER-OWNED.
;    2556  1     !
;    2557  1     !       IMPLICIT INPUTS:
;    2558  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2559  1     !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;    2560  1     !
;    2561  1
;    2562  2         begin
;    2563  2
;    2564  2         local
;    2565  2             index : word,
;    2566  2             RRING_ADDR;
;    2567  2
;    2568  2         RRING_ADDR = .DCT_ADDR [RR_BEG];                    ! FIRST RESPONSE RING SLOT
;    2569  2
;    2570  2         incr COUNT from 1 to RRING_LEN do
;    2571  3             begin
;    2572  3             index = GET_PKT (.CCTLR);                       ! GET AN MSCP PACKET
;    2573  3             .RRING_ADDR = .MSCP_PKT [.index, PKT_LO];       ! LOAD LO-ORDER BUFF DESC INTO SLOT
;    2574  3             RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO SECOND WORD
;    2575  3             .RRING_ADDR = .MSCP_PKT [.index, PKT_HI];       ! LOAD HI-ORDER BUFF DESC INTO SLOT
;    2576  3             PKT_USE [.index] = .CCTLR;                      ! PACKET IN USE
;    2577  3             .RRING_ADDR = ..RRING_ADDR or ED_OWN or ED_FLAG;! GIVE OWNERSHIP TO CONTRLLER
;    2578  3             RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO NEXT SLOT
;    2579  2             end;
;    2580  2
;    2581  1         end;


                                        .SBTTL  INI.RRING INITIALIZATION TEST ROUTINES
000000  004137  000000G        INI.RRING::
                                JSR     R1,$SAVE4                       ;                               2547
000004  013700  000000G                MOV     DCT.ADDR,R0             ;                               2568
000010  016001  000004                 MOV     4(R0),R1                ; *,RRING.ADDR
000014  013703  000000G                MOV     CCTLR,R3                ;                               2572
000020  012704  000004                 MOV     #4,R4                   ; *,COUNT                       2570
000024  010346                 1$:     MOV     R3,-(SP)                ;                               2572
000026  004737  000000G                JSR     PC,GET.PKT
000032  010002                         MOV     R0,R2                   ; *,INDEX
000034  010216                         MOV     R2,(SP)                 ; INDEX,*                       2573
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                  2575
000056  013703  000000G                MOV     CCTLR,R3                ;                               2576
```

```
000062  110362  000000G                        MOVB    R3,PKT.USE(R2)                  ; *,*(INDEX)
000066  052721  140000                         BIS     #140000,(R1)+                   ; *,RRING.ADDR                  2577
000072  022626                                 CMP     (SP)+,(SP)+                     ;                               2571
000074  005304                                 DEC     R4                              ; COUNT                         2570
000076  001352                                 BNE     1$
000100  000207                                 RTS     PC                              ;                               2547

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

```
;    2582  1     GLOBAL routine SET_CTLR_CHAP -
;    2583  1
;    2584  1     !+
;    2585  1     !       THIS ROUTINE IS CALLED BY CTLR_INIT AFTER THE RDRX HAS BEEN HARD
;    2586  1     !       INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
;    2587  1     !       CHARACTERISTICS" COMMAND, AND TO VALIDATE THE RESPONSE (END MESSAGE).
;    2588  1     !
;    2589  1     !       IMPLICIT INPUTS:
;    2590  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2591  1     !
;    2592  1
;    2593  2         begin
;    2594  2
;    2595  2         local
;    2596  2             P_INDEX : word;
;    2597  2
;    2598  2
;    2599  2     ! MISCELLANEOUS INITIALIZATION                                   !ZZZ
;    2600  2                                                                     !ZZZ
;    2601  2     QIO [.CCTLR] = 0;               !INIT NO OF OUTSTANDING QIOS    !ZZZ
;    2602  2     CST [.CCTLR, U_CNT] = 0;        !CLEAR UNITS IN CST TABLE       !ZZZ
;    2603  2     INCR COUNT FROM 0 TO (RP_CNT - 1) DO    !INIT RETURN PACKET POOL !ZZZ
;    2604  2          RP_USE [.COUNT] = -1;                                      !ZZZ
;    2605  2                                                                     !ZZZ
;    2606  2     IODQ_IN = IODQ_OUT = 0;         !INIT I/O DONE QUEUE POINTERS   !ZZZ
;    2607  2
;    2608  2
;    2609  2         P_INDEX = GET_PKT (.CCTLR);                                 ! GET AN MSCP PACKET
;    2610  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_SCC;                       ! PACKET SIZE
;    2611  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_SCC;                       ! OPCODE = SET CTLR CHAR
;    2612  2         MSCP_PKT [.P_INDEX, C_FLAGS] = CF_MASK;                     ! CONTROLLER FLAGS
;    2613  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = IMM_CMD;                    ! IMMEDIATE COMMAND
;    2614  2
;    2615  2         if SEND (.P_INDEX) eql FAILURE                              ! ATTEMPT SEND
;    2616  2         then
;    2617  3             begin                                                   ! IF SEND WAS UNSUCCESSFUL
;    2618  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2619  3
;    2620  3             if .APT_MODE
;    2621  3             then
;    2622  4                 begin
;    2623  4                 .MAIL_BOX_TESTNUM = 1;
;    2624  4                 .MAIL_BOX_SUBTST = 0;
;    2625  3                 end;
;    2626  3
;    2627  3             ERRDF (20, EGD_20, 0);                                  ! FATAL ERROR
;    2628  3             PUT_PKT (.P_INDEX);                                     ! RETURN PACKET TO POOL
;    2629  3             DROP_CTLR (.CCTLR, DU_CFATAL);                          ! DROP CONTROLLER
;    2630  3             return FAILURE;
;    2631  3             end
;    2632  2         else
;    2633  3             begin                                                   ! IF SEND WAS SUCCESSFUL
;    2634  3
```

```
;  2635  3              do
;  2636  4                  begin
;  2637  4                  WAIT ();                                          ! WAIT FOR RETPKT RESPONSE
;  2638  4                  RP_INDX = OUT_IODQ ();                            ! GET INDEX OF RETPKT
;  2639  4                  RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);       ! CALCULATE RETPKT ADDRESS
;  2640  4
;  2641  4                  if .RP_ADDR [MESTYP] neq MT_SEQ                   ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;  2642  4                  then
;  2643  4                      PUT_RETPKT (.RP_INDX);
;  2644  4
;  2645  4                  end
;  2646  3              until (.RP_ADDR [CONID] eql CID_DRIVER) or
;  2647  4                    ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;  2648  3                     ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;  2649  3
;  2650  3              if .RP_ADDR [CONID] eql CID_DRIVER                    ! IF RETPKT IS FROM "DRIVER'
;  2651  3              then
;  2652  4                  begin
;  2653  4                  PRINTF (DBM23);                                   ! "ERROR IN SET_CTLR_CHAR"
;  2654  4                  PUT_RETPKT (.RP_INDX);                            ! RELEASE RETURN PACKET
;  2655  4                  DR_ERR ();                                        ! DROP CONTROLLER
;  2656  4                  return FAILURE;
;  2657  4                  end
;  2658  3              else
;  2659  4                  begin                                             ! ELSE - RETPKT IS FROM DISK MSCP
;  2660  4
;  2661  4                  if (.RP_ADDR [ENDCOD] neq (OP_SCC or OP_END)) or  ! IF WRONG ENDCODE
;  2662  5                     ((.RP_ADDR [C_FLGS] and CF_MASK) neq CF_MASK)  ! OR FLAGS IN ERROR
;  2663  4                  then
;  2664  5                      begin
;  2665  5                      C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  2666  5
;  2667  5                      if .APT_MODE
;  2668  5                      then
;  2669  6                          begin
;  2670  6                          .MAIL_BOX_TESTNUM = 1;
;  2671  6                          .MAIL_BOX_SUBTST = 0;
;  2672  5                          end;
;  2673  5
;  2674  5                      ERRDF (21, EGD_21, EMS_21);                   ! FATAL ERROR
;  2675  5                      DROP_CTLR (.CCTLR, DU_CFATAL);                ! DROP CONTROLLER
;  2676  5                      PUT_RETPKT (.RP_INDX);                        ! RELEASE RETURN PACKET
;  2677  5                      return FAILURE;
;  2678  5                      end
;  2679  4                  else
;  2680  5                      begin                                         ! RETPKT HAS CORRECT ENDCODE
;  2681  5                      CMD_TIME = .RP_ADDR [C_TIME] * 2;
;  2682  5
;  2683  5      !ZZZ            if BIT_TST (SWP_FLAGS, SWF_TRC)
;  2684  5      !ZZZ            then
;  2685  5      !ZZZ                PRINTF (DBM25, .RP_ADDR [C_TIME]);
;  2686  5
;  2687  4                      end;                                          ! RETPKT HAS CORRECT ENDCODE
```

```
;    2688  4
;    2689  3                end;                                          ! IF RETPKT WAS SENT BY DISK MSCP
;    2690  3
;    2691  3            PUT_RETPKT (.RP_INDX);
;    2692  3            return SUCCESS;                                   ! IF SEND WAS SUCCESSFUL
;    2693  2            end;
;    2694  2
;    2695  1        end;                                                  ! ROUTINE SET_CTLR_CHAR


                                        .SBTTL  SET.CTLR.CHAR INITIALIZATION TEST ROUTINES
000000  010146                    SET.CTLR.CHAR::
                                        MOV     R1,-(SP)                ;                                       2582
000002  013701  000000G                 MOV     CCTLR,R1                ;                                       2601
000006  105061  000000G                 CLRB    QIO(R1)
000012  010146                          MOV     R1,-(SP)                ;                                       2602
000014  012746  000126                  MOV     #126,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  105060  000005G                 CLRB    CST+5(R0)
000030  005000                          CLR     R0                      ; COUNT                                 2603
000032  112760  000377  000000G  1$:    MOVB    #377,RP.USE(R0)         ; *,*(COUNT)                            2604
000040  005200                          INC     R0                      ; COUNT                                 2603
000042  020027  000007                  CMP     R0,#7                   ; COUNT,*
000046  003771                          BLE     1$
000050  005037  000000G                 CLR     IODQ.OUT                ;                                       2606
000054  005037  000000G                 CLR     IODQ.IN
000060  010116                          MOV     R1,(SP)                 ;                                       2609
000062  004737  000000G                 JSR     PC,GET.PKT
000066  010001                          MOV     R0,R1                   ; *,P.INDEX
000070  010116                          MOV     R1,(SP)                 ; P.INDEX,*                             2610
000072  012746  000106                  MOV     #106,-(SP)
000076  004737  000000G                 JSR     PC,BL$MUL
000102  012760  000040  000006G         MOV     #40,MSCP.PKT+6(R0)
000110  112760  000004  000022G         MOVB    #4,MSCP.PKT+22(R0)      ;                                       2611
000116  012760  000120  000030G         MOV     #120,MSCP.PKT+30(R0)    ;                                       2612
000124  105060  000004G                 CLRB    MSCP.PKT+4(R0)          ;                                       2613
000130  010116                          MOV     R1,(SP)                 ; P.INDEX,*                             2615
000132  004737  000000G                 JSR     PC,SEND
000136  005700                          TST     R0
000140  001036                          BNE     3$
000142  013700  000000G                 MOV     CCTLR,R0                ;                                       2618
000146  006300                          ASL     R0
000150  105260  000000G                 INCB    C.ERR.TBL(R0)
000154  032737  000001  001162'         BIT     #1,APT.MODE             ;                                       2620
000162  001405                          BEQ     2$
000164  012777  000001  001164'         MOV     #1,@MAIL.BOX.TESTNUM    ;                                       2623
000172  005077  001166'                 CLR     @MAIL.BOX.SUBTST        ;                                       2624
000176  104455                  2$:     TRAP    55                      ;                                       2627
000200  000024                          .WORD   24
000202  000000G                         .WORD   EGD.20
000204  000000                          .WORD   0
000206  010116                          MOV     R1,(SP)                 ; P.INDEX,*                             2628
000210  004737  000000G                 JSR     PC,PUT.PKT
```

```
000214  013716  000000G                        MOV     CCTLR,(SP)                      ;                       2629
000220  012746  000006                         MOV     #6,-(SP)
000224  004737  000000G                        JSR     PC,DROP.CTLR
000230  005726                                 TST     (SP)+                           ;                       2617
000232  005000                                 CLR     R0                              ;                       2633
000234  000554                                 BR      12$
000236  004737  000000G               3$:     JSR     PC,WAIT                         ;                       2637
000242  004737  000000G                       JSR     PC,OUT.IODQ                     ;                       2638
000246  010037  000000G                       MOV     R0,RP.INDX
000252  010016                               MOV     R0,(SP)                         ; RP.INDX,*             2639
000254  012746  000054                       MOV     #54,-(SP)
000260  004737  000000G                       JSR     PC,BL$MUL
000264  062700  000000G                       ADD     #RETPKT,R0
000270  010037  000000G                       MOV     R0,RP.ADDR
000274  132760  000360  000002               BITB    #360,2(R0)                      ;                       2641
000302  001404                               BEQ     4$
000304  013716  000000G                       MOV     RP.INDX,(SP)                    ;                       2643
000310  004737  000000G                       JSR     PC,PUT.RETPKT
000314  005726                       4$:     TST     (SP)+                           ;                       2636
000316  013701  000000G                       MOV     RP.ADDR,R1                      ;                       2646
000322  005000                               CLR     R0
000324  126127  000003  000003               CMPB    3(R1),#3
000332  001002                               BNE     5$
000334  005200                               INC     R0
000336  000407                               BR      6$
000340  132761  000360  000002       5$:     BITB    #360,2(R1)                      ;                       2647
000346  001333                               BNE     3$
000350  105761  000014                       TSTB    14(R1)                          ;                       2648
000354  100330                               BPL     3$
000356  006000                       6$:     ROR     R0                              ;                       2650
000360  103015                               BCC     7$
000362  012716  000000G                       MOV     #OBM23,(SP)                     ;                       2653
000366  012746  000001                       MOV     #1,-(SP)
000372  010600                               MOV     SP,R0                           ; SP,*
000374  104417                               TRAP    17
000376  013716  000000G                       MOV     RP.INDX,(SP)                    ;                       2654
000402  004737  000000G                       JSR     PC,PUT.RETPKT
000406  004737  000000V                       JSR     PC,DR.ERR                       ;                       2655
000412  000447                               BR      10$                             ;                       2656
000414  126127  000014  000204       7$:     CMPB    14(R1),#204                     ;                       2661
000422  001007                               BNE     8$
000424  016100  000022                       MOV     22(R1),R0                       ;                       2662
000430  042700  177657                       BIC     #177657,R0
000434  020027  000120                       CMP     R0,#120
000440  001437                               BEQ     11$
000442  013700  000000G               8$:     MOV     CCTLR,R0                        ;                       2665
000446  006300                               ASL     R0
000450  105260  000000G                       INCB    C.ERR.TBL(R0)
000454  032737  000001  001162'              BIT     #1,APT.MODE                     ;                       2667
000462  001405                               BEQ     9$
000464  012777  000001  001164'              MOV     #1,@MAIL.BOX.TESTNUM            ;                       2670
000472  005077  001166'                       CLR     @MAIL.BOX.SUBTST                ;                       2671
000476  104455                       9$:     TRAP    55                              ;                       2674
```

J8


```
000500  000025                                  .WORD   25
000502  000000G                                 .WORD   EGD.21
000504  000000G                                 .WORD   EMS.21
000506  013716  000000G                         MOV     CCTLR,(SP)              ;                       2675
000512  012746  000006                          MOV     #6,-(SP)
000516  004737  000000G                         JSR     PC,DROP CTLR
000522  013716  000000G                         MOV     RP.INDX,(SP)            ;                       2676
000526  004737  000000G                         JSR     PC,PUT.RETPKT
000532  062706  000010                  10$:    ADD     #10,SP                  ;                       2677
000536  000416                                  BR      13$                     ;                       2664
000540  016137  000024  000000G         11$:    MOV     24(R1),CMD.TIME         ;                       2681
000546  006337  000000G                         ASL     CMD.TIME
000552  013716  000000G                         MOV     RP.INDX,(SP)            ;                       2691
000556  004737  000000G                         JSR     PC,PUT.RETPKT
000562  012700  000001                          MOV     #1,R0                   ;                       2633
000566  062706  000006                  12$:    ADD     #6,SP                   ;                       2615
000572  000401                                  BR      14$                     ;                       2593
000574  005000                          13$:    CLR     R0                      ;                       2582
000576  012601                          14$:    MOV     (SP)+,R1
000600  000207                                  RTS     PC

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

```
;    2696  1      routine UNIT_INIT : novalue =
;    2697  1
;    2698  1      !+
;    2699  1      !       THIS ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONFIGURED UNIT
;    2700  1      !       (DISK) WHICH IS ATTACHED TO A CONTROLLER THAT SURVIVED
;    2701  1      !       INITIALIZATION. ITS PURPOSE IS TO FORMAT AND SEND AN "ONLINE"
;    2702  1      !       MESSAGE, AND TO VERIFY THE RESPONSE.
;    2703  1      !
;    2704  1      !       IMPLICIT INPUTS:
;    2705  1      !               CCTLR   CURRENT CONTROLLER NUMBER
;    2706  1      !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    2707  1      !               L$LUN   CURRENT (DRS) UNIT NUMBER
;    2708  1      !               CST ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    2709  1      !
;    2710  1
;    2711  2          begin
;    2712  2          local
;    2713  2              MAX0_LBNS : WORD UNSIGNED,              ! UNIT'S MAXIMUM LO WORD LBN
;    2714  2              MAX1_LBNS : WORD UNSIGNED;              ! UNIT'S MAXIMUM HI WORD LBN
;    2715  2
;    2716  2          P_INDEX = GET_PKT (.CCTLR);                 ! GET AN MSCP PACKET
;    2717  2          MSCP_PKT [.P_INDEX, MSGLEN] = SZ_ONL;       ! PACKET SIZE
;    2718  2          MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;       ! SET DISK ADDRESS (RD/RX DISK NUMBER)
;    2719  2          MSCP_PKT [.P_INDEX, OPCODE] = OP_ONL;       ! OPCODE FOR "ONLINE"
;    2720  2      !ZZZ MSCP_PKT [.P_INDEX, DDPAR] = BIT00;        ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
;    2721  2          MSCP_PKT [.P_INDEX, CMD_TYPE] = SEQ_CMD;    ! SEQUENTIAL COMMAND
;    2722  2
;    2723  2          if SEND (.P_INDEX) eql FAILURE              ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
;    2724  2          then
;    2725  3              begin
;    2726  3              T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    2727  3
;    2728  3              if .APT_MODE                            !ZZZ
;    2729  3              then
;    2730  4                  begin
;    2731  4                  .MAIL_BOX_TESTNUM = 1;
;    2732  4                  .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK NUM];
;    2733  3                  end;
;    2734  3
;    2735  3              CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
;    2736  3              ERRDF (22, EGD_22, 0);                  !
;    2737  3              DUR [.L$LUN] = DU_ONLINE;               ! SETUP REASON TO DROP UNIT
;    2738  3              DODU (.L$LUN);                          ! DROP UNIT
;    2739  3              PUT_PKT (.P_INDEX);                     ! RETURN PACKET TO POOL
;    2740  3              end
;    2741  2          else
;    2742  3              begin                                   ! OTHERWISE (SEND WAS SUCCESSFUL)
;    2743  3
;    2744  3              do
;    2745  4                  begin
;    2746  4                  WAIT ();                            ! WAIT FOR RETPKT RESPONSE
;    2747  4                  RP_INDX = OUT_IODQ ();              ! GET INDEX OF RETPKT
;    2748  4                  RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
```

```
;    2749  4
;    2750  4                 if .RP_ADDR [MESTYP] neq MT_SEQ     ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    2751  4                 then
;    2752  4                     PUT_RETPKT (.RP_INDX);
;    2753  4
;    2754  4                 end
;    2755  3             until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    2756  4                   ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    2757  3                    ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    2758  3
;    2759  3             if .RP_ADDR [CONID] eql CID_DRIVER      ! IF RETPKT IS FROM "DRIVER"
;    2760  3             then
;    2761  4                 begin
;    2762  4                 PRINTF (DBM26);                     ! "ERROR IN UNIT_INIT"
;    2763  4                 DR_ERR ();                          ! DROP CONTROLLER
;    2764  4                 end
;    2765  3             else
;    2766  3
;    2767  4                 if .RP_ADDR [ENDCOD] neq (OP_ONL or OP_END) ! IF RETPKT IS FROM DISK MSCP
;    2768  3                 then
;    2769  4                     begin
;    2770  4                     T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2771  4
;    2772  4                     if .APT_MODE                    !ZZZ
;    2773  4                     then
;    2774  5                         begin
;    2775  5                         .MAIL_BOX_TESTNUM = 1;
;    2776  5                         .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    2777  4                         end;
;    2778  4
;    2779  4                     CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    2780  4                     ERRDF (23, EGD_23, EMS_21);     ! FATAL ERROR
;    2781  4                     DUR [.L$LUN] = DU_ONLINE;       ! SETUP REASON TO DROP UNIT
;    2782  4                     DODU (.L$LUN);                  ! DROP UNIT
;    2783  4                     end
;    2784  3                 else
;    2785  4                     begin                           ! RETPKT HAS GOOD ENDCODE
;    2786  4                     ST_CODE = .RP_ADDR [STSCOD];    ! GET STATUS CODE
;    2787  4                     SB_CODE = .RP_ADDR [SUBCOD];    ! GET SUB-CODE
;    2788  4
;    2789  4                     CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_0] = .RP_ADDR [NAME_0] + %o'100';                  ! UNIT NAME
ZZZ
;    2790  4                     CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .RP_ADDR [NAME_1_HI] * 16;                    !ZZZ
;    2791  4                     CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] +    !ZZZ
;    2792  4                                                               .RP_ADDR [NAME_1_LO] + %o'100';               !ZZZ
;    2793  4                     CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_2] = .RP_ADDR [NAME_NUM] / 10 + %o'60';            !ZZZ
;    2794  4                     CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_3] = (.RP_ADDR [NAME_NUM] mod 10) + %o'60';        !ZZZ
;    2795  4
;    2796  4
;    2797  4
;    2798  4                     IF .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] EQL %O'104'         !IF NAME IS _D          !ZZZ
;    2799  4                     THEN                                                                                    !ZZZ
;    2800  4                     CST_ADDR [.CUOFF, D_TYPE] = FIXED                               !ITS FIXED.             !ZZZ
;    2801  4                     ELSE                                                                                    !ZZZ
```

```
;    2802  4                    CST_ADDR [.CUOFF, D_TYPE] = REMOVABLE;                          !OTHERWISE REMOVABLE    !ZZZ
;    2803  4
;    2804  4
;    2805  4
;    2806  4                    if .ST_CODE neq ST_SUC          ! IF STATUS CODE IS NOT SUCCESSFUL
;    2807  4                    then
;    2808  5                        begin
;    2809  5                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2810  5
;    2811  5                        if .APT_MODE                !ZZZ
;    2812  5                        then
;    2813  6                            begin
;    2814  6                            .MAIL_BOX_TESTNUM = 1;
;    2815  6                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D DISK NUM];
;    2816  5                            end;
;    2817  5
;    2818  5                        CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    2819  5                        ERRDF (15, EGD_15, EMS_30); ! ONLINE FAILED
;    2820  5                        DUR [.L$LUN] = DU_ONLINE;   ! SET UP REASON FOR DROPPING UNIT
;    2821  5                        DODU (.L$LUN);              ! DROP UNIT
;    2822  5                        end
;    2823  4                    else
;    2824  5                        begin                       ! SUCCESFUL OPERATION
;    2825  5
;    2826  5                        MAX0_LBNS = .RP_ADDR [SIZE0];                               ! LOAD LOWER WORD OF UNIT SIZE
;    2827  5                        MAX1_LBNS = .RP_ADDR [SIZE1];                               ! LOAD UPPER WORD OF UNIT SIZE
;    2828  5
;    2829  6                        if (.MAX0_LBNS eql 0)                                       ! THIS SUBTRACTS ONE FROM THE TOTAL
;    2830  5                        then                                                        ! BECAUSE EVERYTHING STARTS AT 0
;    2831  6                            begin                                                   ! THROUGH (MAXIMUM   1)
;    2832  6                            MAX0_LBNS = %O'177777';
;    2833  6                            MAX1_LBNS = .MAX1_LBNS - 1;
;    2834  6                            end
;    2835  5                        else
;    2836  5                            MAX0_LBNS = .MAX0_LBNS - 1;
;    2837  5
;    2838  5                        if  (.CST_ADDR [.CUOFF + 2, D BEG1] gtru .MAX1_LBNS) or     ! THIS SECTION CHECKS TO SEE
 IF LBNs LISTED
;    2839  5
;    2840  6                            ((.CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .MAX1_LBNS) and   ! IN SOFTWARE QUESTIONS WERE
 TO LARGE FOR
;    2841  6                            (.CST_ADDR [.CUOFF + 1, D_BEG0] gtru (.MAX0_LBNS - 1)))  ! DEVICE SPECIFIED
;    2842  6                                                                                    ! note 1 less then max. or diagnost
c  will error
;    2843  5                        then                                                        ! operator error
;    2844  6                            begin
;    2845  6                            CST_ADDR [.CUOFF + 2, D_BEG1] = 0;
;    2846  6                            CST_ADDR [.CUOFF + 1, D_BEG0] = 0;                      ! change beginning lbn to 0
;    2847  5                            end;
;    2848  5
;    2849  5                         if
;    2850  5                            (.CST_ADDR [.CUOFF + 4, D_END1] gtru .MAX1_LBNS) or
;    2851  5
;    2852  6                            ((.CST_ADDR [.CUOFF + 4, D_END1] eqlu .MAX1_LBNS) and
;    2853  6                            (.CST_ADDR [.CUOFF + 3, D_END0] gtru .MAX0_LBNS))
;    2854  5                         then
```

```
;  2855  6                          begin
;  2856  6                          CST_ADDR [.CUOFF + 4, D_END1] = .MAX1_LBNS;
;  2857  6                          CST_ADDR [.CUOFF + 3, D_END0] = .MAX0_LBNS;          ! and ending lbn to max lbn
;  2858  5                          end;
;  2859  5
;  2860  5
;  2861  5
;  2862  7                          if (((.ENTRY_REASON eql RESTART) or                  ! if restart or
;  2863  6                              (.ENTRY_REASON eql START)) and                    ! if continue
;  2864  6
;  2865  6                              (.CRN_LOW leq 8) and                             ! and
;  2866  6                              (.CRN_HIGH eql 0))                               ! first initialization
;  2867  6
;  2868  5                          THEN                                                 ! intialize block numbers
;  2869  6                              begin
;  2870  6                              BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0];     ! LOAD sequential LBN table
;  2871  6                              BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1];     !
;  2872  6                              TRK_SGN [.L$LUN] = 1;                                      ! POSITIVE TRACKING DIRECTIO
N
;  2873  5                              end;
;  2874  5
;  2875  5
;  2876  5                                                                               ! THIS SECTION LOADS TYPE INTO  CST TABLE
;  2877  5     !ZZZ                         selectoneu .RP_ADDR [R_MODEL] of             ! MODEL BYTE TELLS WHAT TYPE OF UNIT
 IN
;  2878  5     !ZZZ                                                                      ! INDENTIFICATION BLOCK
;  2879  5     !ZZZ                             set
;  2880  5     !ZZZ
;  2881  5     !ZZZ                             [%O'6'] :   CST_ADDR [.CUOFF, D_TYPE] = RD 51;     ! RD 51
;  2882  5     !ZZZ                             [%O'7'] :   CST_ADDR [.CUOFF, D_TYPE] = RX 50;     ! RX 50
;  2883  5     !ZZZ                             [%O'10'] :  CST_ADDR [.CUOFF, D_TYPE] = RD 52;     ! RD 52
;  2884  5     !ZZZ
;  2885  5     !ZZZ                             [otherwise] : BEGIN
;  2886  5     !ZZZ                                           ERRDF (25 ,EGD_24 ,EMS 30);         ! ERROR UNKNOWN DEVICE
;  2887  5     !ZZZ                                           END;
;  2888  5     !ZZZ                             tes;
;  2889  5
;  2890  5
;  2891  5
;  2892  5                          if ((.RP_ADDR [U_FLGS] and UF_WPH) eql UF_WPH) and   ! STATUS CODE IS O.K.
;  2893  6                             (.CST_ADDR [.CUOFF, D_PROT] eql UNPROTECTED)
;  2894  5                          then
;  2895  6                              begin
;  2896  6                              T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  2897  6
;  2898  6                              if .APT_MODE                    !ZZZ
;  2899  6                              then
;  2900  7                                  begin
;  2901  7                                  .MAIL_BOX_TESTNUM = 1;
;  2902  7                                  .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF DATA, D_DISK NUM];
;  2903  6                                  end;
;  2904  6
;  2905  6                              CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;  2906  6                              ERRDF (16, EGD_16, EMS_30);                      ! WRITE PROTECT CONFLICT
;  2907  6                              DUR [.L$LUN] = DU_PROTECT;                       ! SET REASON TO DROP UNIT
```

```
;  2908  6                        DODU (.L$LUN);                                  ! DROP UNIT
;  2909  6                        end
;  2910  5                    else
;  2911  6                        begin                                           ! WRITE PROTECT SWITCH IS O.K.
;  2912  6                        CST_ADDR [.CUOFF, D_STAT] = ONLINE;             ! SET ONLINE FLAG
;  2913  6                        CST [.CCTLR, U_CNT] = .CST [.CCTLR, U_CNT] + 1; ! ADD UNIT TO CTRL TABLE
;  2914  5                        end;
;  2915  4                    end;
;  2916  3                end;                                                    ! IF RETPKT HAS CORRECT ENDCODE
;  2917  3
;  2918  3            PUT RETPKT (.RP_INDX);
;  2919  2            end;                                                        ! IF SEND WAS SUCCESSFUL
;  2920  2
;  2921  1        end;                                                            ! ROUTINE UNIT INIT


                                .SBTTL  UNIT.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G        UNIT.INIT:
                                JSR     R1,$SAVE5                       ;                       2696
000004  024646                  CMP     -(SP),-(SP)
000006  013746  000000G         MOV     CCTLR,-(SP)                     ;                       2716
000012  004737  000000G         JSR     PC,GET.PKT
000016  010037  000000G         MOV     R0,P.INDEX
000022  010016                  MOV     R0,(SP)                         ; P.INDEX,*             2717
000024  012746  000106          MOV     #106,-(SP)
000030  004737  000000G         JSR     PC,BL$MUL
000034  012760  000044  000006G MOV     #44,MSCP.PKT+6(R0)
000042  013760  000000G 000016G MOV     CDISK,MSCP.PKT+16(R0)           ;                       2718
000050  112760  000011  000022G MOVB    #11,MSCP.PKT+22(R0)             ;                       2719
000056  112760  000001  000004G MOVB    #1,MSCP.PKT+4(R0)               ;                       2721
000064  013716  000000G         MOV     P.INDEX,(SP)                    ;                       2723
000070  004737  000000G         JSR     PC,SEND
000074  005700                  TST     R0
000076  001054                  BNE     2$
000100  013700  000000G         MOV     T.ADDR,R0                       ;                       2726
000104  105260  000051          INCB    51(R0)
000110  032737  000001  001162  BIT     #1,APT.MODE                     ;                       2728
000116  001415                  BEQ     1$
000120  012777  000001  001164' MOV     #1,@MAIL.BOX.TESTNUM            ;                       2731
000126  013700  000000G         MOV     CUOFF,R0                        ;                       2732
000132  006300                  ASL     R0
000134  063700  000000G         ADD     CST.ADDR,R0
000140  111077  001166'         MOVB    (R0),@MAIL.BOX.SUBTST
000144  042777  177760  001166' BIC     #177760,@MAIL.BOX.SUBTST
000152  013700  000000G  1$:    MOV     CUOFF,R0                        ;                       2735
000156  006300                  ASL     R0
000160  063700  000000G         ADD     CST.ADDR,R0
000164  052710  010000          BIS     #10000,(R0)
000170  104455                  TRAP    55                              ;                       2736
000172  000026                  .WORD   26
000174  000000G                 .WORD   EGD.22
000176  000000                  .WORD   0
000200  013700  000000G         MOV     L$LUN,R0                        ;                       2737
```

```
000204  112760  000007  000000G                 MOVB    #7,DUR(R0)
000212  104451                                  TRAP    51                      ;                       2738
000214  013716  000000G                         MOV     P.INDEX,(SP)            ;                       2739
000220  004737  000000G                         JSR     PC,PUT.PKT
000224  000137  006706                          JMP     26$                     ;                       2723
000230  004737  000000G               2$:       JSR     PC,WAIT                 ;                       2746
000234  004737  000000G                         JSR     PC,OUT.IODQ             ;                       2747
000240  010037  000000G                         MOV     R0,RP.INDX
000244  010016                                  MOV     R0,(SP)                 ; RP.INDX,*             2748
000246  012746  000054                          MOV     #54,-(SP)
000252  004737  000000G                         JSR     PC,BL$MUL
000256  062700  000000G                         ADD     #RETPKT,R0
000262  010037  000000G                         MOV     R0,RP.ADDR
000266  132760  000360  000002                 BITB    #360,2(R0)              ;                       2750
000274  001404                                  BEQ     3$
000276  013716  000000G                         MOV     RP.INDX,(SP)            ;                       2752
000302  004737  000000G                         JSR     PC,PUT.RETPKT
000306  005726                        3$:       TST     (SP)+                   ;                       2745
000310  013702  000000G                         MOV     RP.ADDR,R2              ;                       2755
000314  005000                                  CLR     R0
000316  126227  000003  000003                 CMPB    3(R2),#3
000324  001002                                  BNE     4$
000326  005200                                  INC     R0
000330  000407                                  BR      5$
000332  132762  000360  000002        4$:       BITB    #360,2(R2)              ;                       2756
000340  001333                                  BNE     2$
000342  105762  000014                          TSTB    14(R2)                  ;                       2757
000346  100330                                  BPL     2$
000350  006000                        5$:       ROR     R0                      ;                       2759
000352  103012                                  BCC     6$
000354  012716  000000G                         MOV     #DBM26,(SP)             ;                       2762
000360  012746  000001                          MOV     #1,-(SP)
000364  010600                                  MOV     SP,R0                   ; SP,*
000366  104417                                  TRAP    17
000370  004737  000000V                         JSR     PC,DR.ERR               ;                       2763
000374  005726                                  TST     (SP)+                   ;                       2761
000376  000457                                  BR      8$                      ;                       2759
000400  013705  000000G               6$:       MOV     CST.ADDR,R5             ;                       2779
000404  013766  000000G  000004                 MOV     CUOFF,4(SP)
000412  006366  000004                          ASL     4(SP)
000416  060566  000004                          ADD     R5,4(SP)
000422  126227  000014  000211                 CMPB    14(R2),#211             ;                       2767
000430  001444                                  BEQ     9$
000432  013700  000000G                         MOV     T.ADDR,R0               ;                       2770
000436  105260  000050                          INCB    50(R0)
000442  032737  000001  001162'                BIT     #1,APT.MODE             ;                       2772
000450  001415                                  BEQ     7$
000452  012777  000001  001164'                MOV     #1,@MAIL.BOX.TESTNUM    ;                       2775
000460  013700  000000G                         MOV     CUOFF,R0                ;                       2776
000464  006300                                  ASL     R0
000466  063700  000000G                         ADD     CST.ADDR,R0
000472  111077  001166'                         MOVB    (R0),@MAIL.BOX.SUBTST
000476  042777  177760  001166'                BIC     #177760,@MAIL.BOX.SUBTST
```

```
000504  052776  010000  000004          7$:     BIS     #10000,@4(SP)                   ;                               2779
000512  104455                                  TRAP    55                              ;                               2780
000514  000027                                  .WORD   27
000516  000000G                                 .WORD   EGD.23
000520  000000G                                 .WORD   EMS.21
000522  013700  000000G                         MOV     L$LUN,R0                        ;                               2781
000526  112760  000007  000000G                 MOVB    #7,DUR(R0)
000534  10445'                                  TRAP    51                              ;                               2782
000536  000137  006676                  8$:     JMP     25$                             ;                               2767
000542  116237  000016  000000G         9$:     MOVB    16(R2),ST.CODE                  ;                               2786
000550  042737  177740  000000G                 BIC     #177740,ST.CODE
000556  016200  000016                          MOV     16(R2),R0                       ;                               2787
000562  006200                                  ASR     R0
000564  006200                                  ASR     R0
000566  006200                                  ASR     R0
000570  006200                                  ASR     R0
000572  006200                                  ASR     R0
000574  042700  174000                          BIC     #174000,R0
000600  010037  000000G                         MOV     R0,SB.CODE
000604  013701  000000G                         MOV     CUOFF,R1                        ;                               2789
000610  006301                                  ASL     R1
000612  060501                                  ADD     R5,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                       ;                               2790
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                              ;                               2791
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                           ;                               2792
000722  062701  000100                          ADD     #100,R1
000726  110163  000001                          MOVB    R1,1(R3)
000732  013701  000000G                         MOV     CUOFF,R1                        ;                               2793
000736  006301                                  ASL     R1
000740  060501                                  ADD     R5,R1
000742  116216  000034                          MOVB    34(R2),(SP)
```

```
000746  042716  177700                          BIC     #177700,(SP)
000752  012746  000012                          MOV     #12,-(SP)
000756  004737  000000G                         JSR     PC,BL$DIV
000762  010004                                  MOV     R0,R4
000764  062704  000060                          ADD     #60,R4
000770  110461  000014                          MOVB    R4,14(R1)
000774  116216  000034                          MOVB    34(R2),(SP)             ;                       2794
001000  042716  177700                          BIC     #177700,(SP)
001004  012746  000012                          MOV     #12,-(SP)
001010  004737  000000G                         JSR     PC,BL$MOD
001014  010004                                  MOV     R0,R4
001016  062704  000060                          ADD     #60,R4
001022  110461  000015                          MOVB    R4,15(R1)
001026  126327  000001  000104                  CMPB    1(R3),#104              ;                       2798
001034  001004                                  BNE     10$
001036  152776  000020  000010                  BISB    #20,@10(SP)             ;                       2800
001044  000403                                  BR      11$                     ;                       2798
001046  142776  000020  000010          10$:    BICB    #20,@10(SP)             ;                       2802
001054  005737  000000G                 11$:    TST     ST.CODE                 ;                       2806
001060  001440                                  BEQ     13$
001062  013700  000000G                         MOV     T.ADDR,R0               ;                       2809
001066  105260  000050                          INCB    50(R0)
001072  032737  000001  001162'                BIT     #1,APT.MODE             ;                       2811
001100  001411                                  BEQ     12$
001102  012777  000001  001164                  MOV     #1,@MAIL.BOX.TESTNUM    ;                       2814
001110  117677  000010  001166'                MOVB    @10(SP),@MAIL.BOX.SUBTST ;                      2815
001116  042777  177760  001166'                BIC     #177760,@MAIL.BOX.SUBTST
001124  052776  010000  000010          12$:    BIS     #10000,@10(SP)          ;                       2818
001132  104455                                  TRAP    55                      ;                       2819
001134  000017                                  .WORD   17
001136  000000G                                 .WORD   EGD.15
001140  000000G                                 .WORD   EMS.30
001142  013700  000000G                         MOV     L$LUN,R0                ;                       2820
001146  112760  000007  000000G                 MOVB    #7,DUR(R0)
001154  104451                                  TRAP    51                      ;                       2821
001156  000137  006674'                         JMP     24$                     ;                       2806
001162  016203  000044                  13$:    MOV     44(R2),R3               ; *,MAX0.LBNS           2826
001166  016204  000046                          MOV     46(R2),R4               ; *,MAX1.LBNS           2827
001172  005703                                  TST     R3                      ; MAX0.LBNS             2829
001174  001004                                  BNE     14$
001176  012703  177777                          MOV     #-1,R3                  ; *,MAX0.LBNS           2832
001202  005304                                  DEC     R4                      ; MAX1.LBNS             2833
001204  000401                                  BR      15$                     ;                       2829
001206  005303                          14$:    DEC     R3                      ; MAX0.LBNS             2836
001210  013701  000000G                 15$:    MOV     CUOFF,R1                ;                       2838
001214  006301                                  ASL     R1
001216  060501                                  ADD     R5,R1
001220  012766  000004  000012                  MOV     #4,12(SP)
001226  060166  000012                          ADD     R1,12(SP)
001232  027604  000012                          CMP     @12(SP),R4              ; *,MAX1.LBNS
001236  101012                                  BHI     16$
001240  001021                                  BNE     17$                     ;                       2840
001242  013701  000000G                         MOV     CUOFF,R1                ;                       2841
```

```
001246  006301                                  ASL     R1
001250  060501                                  ADD     R5,R1
001252  010300                                  MOV     R3,R0           ; MAX0.LBNS,*
001254  005300                                  DEC     R0
001256  026100  000002                          CMP     2(R1),R0
001262  101410                                  BLOS    17$
001264  005076  000012              16$:        CLR     @12(SP)         ;                               2845
001270  013701  000000G                         MOV     CUOFF,R1        ;                               2846
001274  006301                                  ASL     R1
001276  060501                                  ADD     R5,R1
001300  005061  000002                          CLR     2(R1)
001304  013701  000000G             17$:        MOV     CUOFF,R1        ;                               2850
001310  006301                                  ASL     R1
001312  060501                                  ADD     R5,R1
001314  026104  000010                          CMP     10(R1),R4       ; *,MAX1.LBNS
001320  101010                                  BHI     18$
001322  001017                                  BNE     19$             ;                               2852
001324  013700  000000G                         MOV     CUOFF,R0        ;                               2853
001330  006300                                  ASL     R0
001332  060500                                  ADD     R5,R0
001334  026003  000006                          CMP     6(R0),R3        ; *,MAX0.LBNS
001340  101410                                  BLOS    19$
001342  010461  000010              18$:        MOV     R4,10(R1)       ; MAX1.LBNS,*                   2856
001346  013701  000000G                         MOV     CUOFF,R1        ;                               2857
001352  006301                                  ASL     R1
001354  060501                                  ADD     R5,R1
001356  010361  000006                          MOV     R3,6(R1)        ; MAX0.LBNS,*
001362  123727  000000G 000002      19$:        CMPB    ENTRY.REASON,#2 ;                               2862
001370  001404                                  BEQ     20$
001372  123727  000000G 000001                  CMPB    ENTRY.REASON,#1 ;                               2863
001400  001031                                  BNE     21$
001402  023727  000000G 000010      20$:        CMP     CRN.LOW,#10     ;                               2865
001410  003025                                  BGT     21$
001412  005737  000000G                         TST     CRN.HIGH        ;                               2866
001416  001022                                  BNE     21$
001420  013703  000000G                         MOV     L$LUN,R3        ;                               2870
001424  010304                                  MOV     R3,R4
001426  006304                                  ASL     R4
001430  006304                                  ASL     R4
001432  013701  000000G                         MOV     CUOFF,R1
001436  006301                                  ASL     R1
001440  060501                                  ADD     R5,R1
001442  016164  000002  000000G                 MOV     2(R1),BST(R4)
001450  017664  000012  000002G                 MOV     @12(SP),BST+2(R4)       ;                       2871
001456  112763  000001  000000G                 MOVB    #1,TRK.SGN(R3)          ;                       2872
001464  032762  020000  000022      21$:        BIT     #20000,22(R2)           ;                       2892
001472  001442                                  BEQ     23$
001474  005776  000010                          TST     @10(SP)                 ;                       2893
001500  100037                                  BPL     23$
001502  013700  000000G                         MOV     T.ADDR,R0               ;                       2896
001506  105260  000050                          INCB    50(R0)
001512  032737  000001  001162'                 BIT     #1,APT.MODE             ;                       2898
001520  001411                                  BEQ     22$
```

```
001522  012777  000001  001164                MOV     #1,@MAIL.BOX.TESTNUM            ;                       2901
001530  117677  000010  001166                MOVB    @10(SP),@MAIL.BOX.SUBTST        ;                       2902
001536  042777  177760  001166                BIC     #177760,@MAIL.BOX.SUBTST
001544  052776  010000  000010        22$:    BIS     #10000,@10(SP)                  ;                       2905
001552  104455                                TRAP    55                              ;                       2906
001554  000020                                .WORD   20
001556  000000G                               .WORD   EGD.16
001560  000000G                               .WORD   EMS.30
001562  013700  000000G                       MOV     L$LUN,R0                        ;                       2907
001566  112760  000011  000000G               MOVB    #11,DUR(R0)
001574  104451                                TRAP    51                              ;                       2908
001576  000414                                BR      24$                             ;                       2892
001600  052776  020000  000010        23$:    BIS     #20000,@10(SP)                  ;                       2912
001606  013716  000000G                       MOV     CCTLR,(SP)                      ;                       2913
001612  012746  000126                        MOV     #126.,-(SP)
001616  004737  000000G                       JSR     PC,BL$MUL
001622  105260  000005G                       INCB    CST+5(R0)
001626  005726                                TST     (SP)+                           ;                       2911
001630  022626                        24$:    CMP     (SP)+,(SP)+                     ;                       2785
001632  013716  000000G               25$:    MOV     RP.INDX,(SP)                    ;                       2918
001636  004737  000000G                       JSR     PC,PUT.RETPKT
001642  062706  000010                26$:    ADD     #10,SP                          ;                       2696
001646  000207                                RTS     PC

; Routine Size:  468 words,     Routine Base:  $CODE$ + 5044
; Maximum stack depth per invocation:  14 words
```

```
;  2922  1     GLOBAL routine DR_ERR : novalue =
;  2923  1
;  2924  1     !+
;  2925  1     !       THIS ROUTINE IS DESIGNED TO PROCESS RETURN PACKETS THAT ORIGINATE AT
;  2926  1     !       THE "DRIVER" RATHER THAN THE DEVICE. DRIVER-ORIGINATED PACKETS INDICATE
;  2927  1     !       EITHER A FATAL DEVICE ERROR OR A COMMAND TIMEOUT. SINCE THIS ROUTINE IS
;  2928  1     !       ONLY CALLED DURING THE INITIALIZATION TEST, IT TREATS A COMMAND TIMEOUT
;  2929  1     !       AS AN INITIALIZATION ERROR.
;  2930  1     !
;  2931  1     !       IMPLICIT INPUTS:
;  2932  1     !               RP_ADDR - ADDRESS OF A RETPKT THAT ORIGINATED AT THE "DRIVER"
;  2933  1     !                       (I.E., CONNECTION ID = CID_DRIVER)
;  2934  1     !
;  2935  1
;  2936  2         begin
;  2937  2
;  2938  2         local
;  2939  2             REASON : word initial (DU_TIME);        ! ASSUME COMMAND TIMEOUT
;  2940  2
;  2941  2         if .RP_ADDR [MESTYP] eql MT_FATAL           ! IF FATAL DEVICE ERROR
;  2942  2         then
;  2943  2
;  2944  2         DROP_CTLR (.CCTLR, .REASON);                ! DROP ALL UNITS ON CONTROLLER
;  2945  1         end;


                                        .SBTTL  DR.ERR INITIALIZATION TEST ROUTINES
000000  010146                 DR.ERR::MOV     R1,-(SP)                        ;                       2922
000002  012701  000012                  MOV     #12,R1                          ; =,REASON              2936
000006  013700  000000G                 MOV     RP.ADDR,R0                      ;                       2941
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)                     ;                       2944
000034  010146                         MOV     R1,-(SP)                        ; REASON,=
000036  004737  000000G                 JSR     PC,DROP.CTLR
000042  022626                         CMP     (SP)+,(SP)+
000044  012601                 1$:     MOV     (SP)+,R1                        ;                       2922
000046  000207                         RTS     PC

; Routine Size:  20 words,      Routine Base:  $CODE$ + 6714
; Maximum stack depth per invocation:  4 words
```

```
;    2946  1     routine ACCESS : novalue =
;    2947  1
;    2948  1     !+
;    2949  1     !       THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT DISK
;    2950  1     !       CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
;    2951  1     !       SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE DISK, AND CHECKING
;    2952  1     !       THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
;    2953  1     !
;    2954  1     !       IMPLICIT INPUTS:
;    2955  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2956  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    2957  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    2958  1     !-
;    2959  1
;    2960  2         begin
;    2961  2
;    2962  2     local
;    2963  2             RESULT : word initial (FAILURE),        ! GUILTY UNTIL PROVEN INNOCENT
;    2964  2             LBN : word,
;    2965  2             PASS : word initial (1);                ! LOOP PASS COUNT
;    2966  2
;    2967  2         ST_CODE = SB_CODE = 0;                      ! STATUS CODE AND SUB-CODE
;    2968  2         LBN = (((.MAX_LBN [.L$LUN] + 1) ^ -1) and %o'77777') - 1;
;    2969  2                                                     ! START WITH LAST LBN ON TOP SURFACE: [(X+1)/2]  1
;    2970  2
;    2971  2         do
;    2972  3             begin                                   ! LOOP STARTS HERE
;    2973  3             P_INDEX = GET_PKT (.CCTLR);             ! GET AN MSCP PACKET
;    2974  3             MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;   ! SET DISK ADDR (RD/RX DISK NUMBER)
;    2975  3             MSCP_PKT [.P_INDEX, OPCODE] = OP_ACC;   ! ACCESS OPCODE
;    2976  3             MSCP_PKT [.P_INDEX, BC_LO] = 512;       ! BYTE COUNT (1 BLOCK)
;    2977  3             MSCP_PKT [.P_INDEX, LBN_L] = .LBN;      ! LOGICAL BLOCK NUMBER
;    2978  3             MSCP_PKT [.P_INDEX, CMD_TYPE] = NON_SEQ_CMD;    ! NON-SEQUENTIAL COMMAND
;    2979  3
;    2980  3             if SEND (.P_INDEX) eql FAILURE          ! ATTEMPT TO SEND; IF CTLR NOT ONLINE
;    2981  3             then
;    2982  4                 begin
;    2983  4                 PUT_PKT (.P_INDEX);                 ! RETURN PACKET TO POOL
;    2984  4                 PASS = 2;                           ! NO MORE TRIES
;    2985  4                 end
;    2986  3             else
;    2987  4                 begin                               ! IF SEND WAS SUCCESSFUL
;    2988  4
;    2989  4                 do
;    2990  5                     begin
;    2991  5                     WAIT ();                        ! WAIT FOR RESPONSE
;    2992  5                     RP_INDX = OUT_IODQ ();          ! GET RETPKT (RESPONSE) INDEX
;    2993  5                     RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
;    2994  5
;    2995  5                     if .RP_ADDR [MESTYP] neq MT_SEQ ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    2996  5                     then
;    2997  5                         PUT_RETPKT (.RP_INDX);
;    2998  5
```

```
;  2999  5                    end
;  3000  4                until (.RP_ADDR [CONID] eql CID_DRIVER) or
;  3001  5                      ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;  3002  4                       ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;  3003  4
;  3004  4                if .RP_ADDR [CONID] eql CID_DRIVER  ! IF RETPKT CAME FROM "DRIVER"
;  3005  4                then
;  3006  4                    PASS = 2                        ! NO MORE TRIES
;  3007  4                else
;  3008  4
;  3009  5                    if .RP_ADDR [ENDCOD] neq (OP_ACC or OP_END)
;  3010  4                    then
;  3011  5                        begin
;  3012  5                        PRINTF (D8M29);             ! "RETPKT HAS BAD ENDCODE'
;  3013  5    !ZZZ                EMSCMD ();
;  3014  5                        end
;  3015  4                    else
;  3016  5                        begin                       ! RETPKT HAS CORRECT ENDCODE
;  3017  5                        ST_CODE = .RP_ADDR [STSCOD];! GET STATUS CODE FROM PACKET
;  3018  5                        SB_CODE = .RP_ADDR [SUBCOD];! GET SUB-CODE FROM PACKET
;  3019  5
;  3020  5                        if .ST_CODE eql ST_SUC      ! IF STATUS CODE INDICATES SUCCESS
;  3021  5                        then
;  3022  6                            begin
;  3023  6                            RESULT = SUCCESS;
;  3024  6                            PASS = 2;               ! NO NEED TO TRY AGAIN
;  3025  5                            end;
;  3026  5
;  3027  4                        end;                        ! IF RETPKT HAS CORRECT ENDCODE
;  3028  4
;  3029  4                PUT_RETPKT (.RP_INDX);
;  3030  3                end;                                ! IF SEND WAS SUCCESSFUL
;  3031  3
;  3032  3            LBN = .LBN + 1;                         ! ADVANCE TO FIRST LBN OF BOTTOM SURFACE
;  3033  3            PASS = .PASS + 1;                       ! SECOND PASS
;  3034  3            end                                     ! END OF PASS LOOP
;  3035  2        until .PASS gequ 3;
;  3036  2
;  3037  2        if .RESULT eql FAILURE
;  3038  2        then
;  3039  3            begin
;  3040  3            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  3041  3            CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
;  3042  3            ERRDF (17, EGD_17, EMS_30);                 ! ACCESS FAILED
;  3043  3            DUR [.L$LUN] = DU_ACCESS;               ! SET REASON TO DROP UNIT
;  3044  3            DODU (.L$LUN);                          ! DROP UNIT
;  3045  2            end;                                    ! IF ACCESS FAILED
;  3046  2
;  3047  1        end;                                        ! ROUTINE ACCESS


                                .SBTTL  ACCESS INITIALIZATION TEST ROUTINES
000000  004137  000000G    ACCESS: JSR     R1,$SAVE4                       ;                                   2946
```

```
000004  005003                          CLR     R3                      ; RESULT                2960
000006  012702  000001                  MOV     #1,R2                   ; *,PASS
000012  005037  000000G                 CLR     SB.CODE                 ;                       2967
000016  005037  000000G                 CLR     ST.CODE
000022  013700  000000G                 MOV     L$LUN,R0                ;                       2968
000026  006300                          ASL     R0
000030  016000  000054'                  MOV     MAX.LBN(R0),R0
000034  060200                          ADD     R2,R0
000036  006200                          ASR     R0
000040  010004                          MOV     R0,R4                   ; *,LBN
000042  042704  100000                  BIC     #100000,R4              ; *,LBN
000046  005304                          DEC     R4                      ; LBN
000050  013746  000000G         1$:     MOV     CCTLR,-(SP)             ;                       2973
000054  004737  000000G                 JSR     PC,GET.PKT
000060  010037  000000G                 MOV     R0,P.INDEX
000064  010016                          MOV     R0,(SP)                 ; P.INDEX,*             2974
000066  012746  000106                  MOV     #106,-(SP)
000072  004737  000000G                 JSR     PC,BL$MUL
000076  013760  000000G  000016G        MOV     CDISK,MSCP.PKT+16(R0)
000104  112760  000020  000022G        MOVB    #20,MSCP.PKT+22(R0)     ;                       2975
000112  012760  001000  000026G        MOV     #1000,MSCP.PKT+26(R0)   ;                       2976
000120  010460  000046G                 MOV     R4,MSCP.PKT+46(R0)      ; LBN,*                 2977
000124  112760  000002  000004G        MOVB    #2,MSCP.PKT+4(R0)       ;                       2978
000132  013716  000000G                 MOV     P.INDEX,(SP)            ;                       2980
000136  004737  000000G                 JSR     PC,SEND
000142  005700                          TST     R0
000144  001007                          BNE     2$
000146  013716  000000G                 MOV     P.INDEX,(SP)            ;                       2983
000152  004737  000000G                 JSR     PC,PUT.PKT
000156  012702  000002                  MOV     #2,R2                   ; *,PASS                2984
000162  000522                          BR      9$                      ;                       2980
000164  004737  000000G         2$:     JSR     PC,WAIT                 ;                       2991
000170  004737  000000G                 JSR     PC,OUT.IODQ             ;                       2992
000174  010037  000000G                 MOV     R0,RP.INDX
000200  010016                          MOV     R0,(SP)                 ; RP.INDX,*             2993
000202  012746  000054                  MOV     #54,-(SP)
000206  004737  000000G                 JSR     PC,BL$MUL
000212  062700  000000G                 ADD     #RETPKT,R0
000216  010037  000000G                 MOV     R0,RP.ADDR
000222  132760  000360  000002        BITB    #360,2(R0)              ;                       2995
000230  001404                          BEQ     3$
000232  013716  000000G                 MOV     RP.INDX,(SP)            ;                       2997
000236  004737  000000G                 JSR     PC,PUT.RETPKT
000242  005726                  3$:     TST     (SP)+                   ;                       2990
000244  013701  000000G                 MOV     RP.ADDR,R1              ;                       3000
000250  005000                          CLR     R0
000252  126127  000003  000003        CMPB    3(R1),#3
000260  001002                          BNE     4$
000262  005200                          INC     R0
000264  000407                          BR      5$
000266  132761  000360  000002  4$:     BITB    #360,2(R1)              ;                       3001
000274  001333                          BNE     2$
000276  105761  000014                  TSTB    14(R1)                  ;                       3002
```

L9


```
000302  100330                          BPL     2$
000304  006000                  5$:     ROR     R0                      ;                       3004
000306  103442                          BLO     7$                      ;                       3006
000310  126127  000014  000220          CMPB    14(R1),#220             ;                       3009
000316  001410                          BEQ     6$
000320  012716  000000G                 MOV     #DBM29,(SP)             ;                       3012
000324  012746  000001                  MOV     #1,-(SP)
000330  010600                          MOV     SP,R0                   ; SP,*
000332  104417                          TRAP    17
000334  005726                          TST     (SP)+                   ;                       3011
000336  000430                          BR      8$                      ;                       3009
000340  116137  000016  000000G  6$:     MOVB    16(R1),ST.CODE          ;                       3017
000346  042737  177740  000000G          BIC     #177740,ST.CODE
000354  016100  000016                  MOV     16(R1),R0               ;                       3018
000360  006200                          ASR     R0
000362  006200                          ASR     R0
000364  006200                          ASR     R0
000366  006200                          ASR     R0
000370  006200                          ASR     R0
000372  042700  174000                  BIC     #174000,R0
000376  010037  000000G                 MOV     R0,SB.CODE
000402  005737  000000G                 TST     ST.CODE                 ;                       3020
000406  001004                          BNE     8$
000410  012703  000001                  MOV     #1,R3                   ; *,RESULT              3023
000414  012702  000002          7$:     MOV     #2,R2                   ; *,PASS                3024
000420  013716  000000G          8$:     MOV     RP.INDX,(SP)            ;                       3029
000424  004737  000000G                 JSR     PC,PUT.RETPKT
000430  005204                  9$:     INC     R4                      ; LBN                   3032
000432  005202                          INC     R2                      ; PASS                  3033
000434  022626                          CMP     (SP)+,(SP)+             ;                       2972
000436  020227  000003                  CMP     R2,#3                   ; PASS,*                3035
000442  103602                          BLO     1$
000444  005703                          TST     R3                      ; RESULT                3037
000446  001025                          BNE     10$
000450  013700  000000G                 MOV     T.ADDR,R0               ;                       3040
000454  105260  000050                  INCB    50(R0)
000460  013700  000000G                 MOV     CUOFF,R0                ;                       3041
000464  006300                          ASL     R0
000466  063700  000000G                 ADD     CST.ADDR,R0
000472  052710  010000                  BIS     #10000,(R0)
000476  104455                          TRAP    55                      ;                       3042
000500  000021                          .WORD   21
000502  000000G                         .WORD   EGD.17
000504  000000G                         .WORD   EMS.30
000506  013700  000000G                 MOV     L$LUN,R0                ;                       3043
000512  112760  000010  000000G         MOVB    #10,DUR(R0)
000520  104451                          TRAP    51                      ;                       3044
000522  000207                  10$:    RTS     PC                      ;                       2946

; Routine Size:  170 words,    Routine Base:  $CODE$ + 6764
; Maximum stack depth per invocation:  10 words
```

```
;  3048  1     %sbttl 'MULTI-DRIVE TEST ROUTINES'
;  3049  1
;  3050  1
;  3051  1     GLOBAL routine MULTI_DRIVE : novalue =
;  3052  1
;  3053  1
;  3054  1     !+
;  3055  1     !       THIS SUBTEST IS THE MOST SIGNIFICANT PART OF THE ENTIRE PROGRAM. THE
;  3056  1     !       MULTI-DRIVE TEST IS A HOST-CONTROLLED EXERCISER DESIGNED TO GIVE THE
;  3057  1     !       USER AN INDICATION OF HOW ONE OR SEVERAL RDRX DRIVES WOULD PERFORM IN
;  3058  1     !       AN OPERATING SYSTEM ENVIRONMENT.
;  3059  1     !
;  3060  1     !       THIS ROUTINE ACTS AS AN "EXECUTIVE" TO THE WHOLE PROCESS. AFTER
;  3061  1     !       INVOKING MD_INIT TO INITIALIZE MULTI-DRIVE TEST DATA, THIS ROUTINE
;  3062  1     !       ENTERS A LOOP WHICH ISSUES QIOs TO ALL ACTIVE CONTROLLERS AND PROCESSES
;  3063  1     !       ANY RESPONSES. IN ADDITION, ALL OUTSTANDING COMMANDS ARE TIMED IN
;  3064  1     !       DRV_TIMCHK WHICH IS INVOKED EVERY SECOND. NORMAL TERMINATION OF THIS
;  3065  1     !       LOOP OCCURS WHEN QIOs ARE NO LONGER BEING ISSUED, AND ALL OUTSTANDING
;  3066  1     !       QIOS HAVE COMPLETED.
;  3067  1     !
;  3068  1
;  3069  1
;  3070  2         begin
;  3071  2
;  3072  2         local
;  3073  2             CUR_PRIORITY : word;
;  3074  2
;  3075  2         label
;  3076  2             SEND_COMMANDS;
;  3077  2
;  3078  2         MD_INIT ();                                          ! INIT MULTI-DRIVE TEST DATA
;  3079  2         INIT_OCCURED = TRUE;                                 !
;  3080  2
;  3081  2
;  3082  3         do  begin                                            ! START OF EXECUTIVE LOOP
;  3083  3
;  3084  3             incr CTLR from 0 to (MAX_CTLR - 1) do            ! FOR EACH CONTROLLER
;  3085  4                 begin
;  3086  4                 SET_CPAR (.CTLR);                            ! SET UP CURRENT CONTROLLER PARAMETERS
;  3087  4                 GETPRI (CUR_PRIORITY);                       !
;  3088  4     !ZZZ        SETPRI (PRI04);                              ! NO INTERRUPTS WHEN EXAMINING SA
;  3089  4                 SETPRI (.BRLEVEL);                           ! NO INTERRUPTS WHEN EXAMINING SA          ZZ
Z
;  3090  4                 ICTLR = .CCTLR;                              ! FAKE INTERRUPTING CONTROLLER'S NUMBER
;  3091  4                 ICST_ADDR = .CST_ADDR;                       ! FAKE INTERRUPTING CONTROLLER'S CST ADDR
;  3092  4                 IDCT_ADDR = .DCT_ADDR;                       ! FAKE INTERRUPTING CONTROLLER'S DCT ADDR
;  3093  4                 IRDRX_ADDR = .ICST_ADDR [IP_ADDR];           ! FAKE INTERRUPTING CONTROLLER'S ADDRESS
;  3094  4                 IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC ALL];  ! CONTENTS OF THE SA REGISTER
;  3095  4
;  3096  5                 if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)     ! IF SA SHOWS AN ERROR
;  3097  4                 then
;  3098  5                     begin
;  3099  5                     FATAL_ERROR ();                          ! DECLARE FATAL ERROR
;  3100  5                     SETPRI (.CUR PRIORITY),                  ! LOWER PRIORITY
```

```
;    3101  5                    exitloop;                                   ! QUIT
;    3102  5                    end
;    3103  5
;    3104  4                else
;    3105  4                    SETPRI (.CUR_PRIORITY);                     ! IF NO ERROR, CONTINUE
;    3106  4
;    3107  4                if QIO_OK ()                                    ! IF O.K. TO ISSUE QIO(S) TO CONTROLLER
;    3108  4                then
;    3109  4                    SEND_COMMANDS:
;    3110  5                        begin
;    3111  5                        QIO_GEN ();                             ! GENERATE 1 OR 2 QIOs
;    3112  5
;    3113  5                        if (.MX1 geq 0) and                     ! IF SUCCESS ON FIRST QIO
;    3114  6                           (not .EOP_FLAG)
;    3115  5                        then
;    3116  5
;    3117  5                            if SEND (.MX1) eql SUCCESS          ! ATTEMPT TO SEND IT. IF SUCCESS
;    3118  5                            then
;    3119  6                            BEGIN                               !ZZZ
;    3120  6                                QIO [.CTLR] = .QIO [.CTLR] + 1; ! INCR OUTSTANDING QIO COUNT ZZZ
;    3121  6                                RW_BALANCE = .RW_BALANCE + 1;   !INCR RD/WR RATIO COUNT
;    3122  6                            END                                 !ZZZ
;    3123  6
;    3124  5                            else
;    3125  6                                begin
;    3126  6                                PUT_PKT (.MX1);                 ! RETURN PACKET TO POOL
;    3127  6                                leave SEND_COMMANDS;
;    3128  5                                end;
;    3129  5
;    3130  5
;    3131  5                if (.MX2 geq 0) and                             ! IF SUCCESS ON SECOND QIO
;    3132  6                   (not .EOP_FLAG)
;    3133  5                then
;    3134  6                    begin
;    3135  6
;    3136  6                    do
;    3137  6                        BREAK                                   ! WAIT TILL 1 MORE SLOT AVAILABLE IN CRING
;    3138  6                    until (.DCT_ADDR [CRING_CNT] lssu CRING_LEN);   !
;    3139  6
;    3140  6                    if SEND (.MX2) eql SUCCESS                  ! ATTEMPT TO SEND IT.
;    3141  6                    then
;    3142  7                    BEGIN                                       !ZZZ
;    3143  7                        QIO [.CTLR] = .QIO [.CTLR] + 1;         ! IF SUCCESS, INCR OUTSTANDING QIO COUNT
;    3144  7                        RW_BALANCE = 0;                         !INDICATE RD/WT PAIR WAS LAST ISSUED  ZZZ
;    3145  7                    END                                         !ZZZ
;    3146  7
;    3147  6                    else
;    3148  7                        begin
;    3149  7                        PRINTF (DBM121, .CRN_HIGH, .CRN_LOW);
;    3150  7                        COMPARE_DATA = FALSE;                   ! NO SENSE IN COMPARING WRITE DATA
;    3151  7                        PUT_PKT (.MX2);                         ! RETURN PACKET TO POOL
;    3152  6                        end;
;    3153  6
```

```
;    3154  5                          end;
;    3155  5
;    3156  4                      end;                                  ! O.K. TO ISSUE QIO(S)
;    3157  3                  end;                                      ! CONTROLLER LOOP
;    3158  3
;    3159  3
;    3160  3              BREAK;                                        ! LET SUPERVISOR CATCH USER RFQUESTS
;    3161  3              PROC_RETPKT ();                               ! PROCESS ANY RETURN PACKETS
;    3162  3
;    3163  3              end                                           ! EXECUTIVE PROCESSING LOOP
;    3164  3          until ((not QIO_OUT ()) or
;    3165  4                 ((.DCT_ADDR [CRING_CNT] eql 0) and
;    3166  2                  (.EOP_FLAG)));
;    3167  2
;    3168  2
;    3169  2          DCT_ADDR [IG_INT] = TRUE;                         ! NO FURTHER INTERRUPTS ON THIS CONTROLLER
;    3170  2
;    3171  2
;    3172  1          end;                                              ! EXERCISER


                                        .SBTTL  MULTI.DRIVE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G           MULTI.DRIVE::
                                        JSR     R1,$SAVE3               ;                                       3051
000004  005746                         TST     -(SP)
000006  004737  000000V                 JSR     PC,MD.INIT              ;                                       3078
000012  112737  000001  000000G         MOVB    #1,INIT.OCCURED         ;                                       3079
000020  005001                 1$:     CLR     R1                      ; CTLR                                  3084
000022  010146                 2$:     MOV     R1,-(SP)                ; CTLR,*                                3086
000024  004737  000000G                 JSR     PC,SET.CPAR
000030  104440                         TRAP    40                      ;                                       3087
000032  010003                         MOV     R0,R3                   ; *,CUR.PRIORITY
000034  013700  000000G                 MOV     BRLEVEL,R0              ;                                       3089
000040  104441                         TRAP    41
000042  013737  000000G 000104'         MOV     CCTLR,ICTLR             ;                                       3090
000050  013737  000000G 000076'         MOV     CST.ADDR,ICST.ADDR      ;                                       3091
000056  013737  000000G 000100'         MOV     DCT.ADDR,IDCT.ADDR      ;                                       3092
000064  017737  000076' 000000G         MOV     @ICST.ADDR,IRDRX.ADDR   ;                                       3093
000072  013700  000100'                 MOV     IDCT.ADDR,R0            ;                                       3094
000076  013702  000000G                 MOV     IRDRX.ADDR,R2
000102  016266  000002  000002         MOV     2(R2),2(SP)             ; *,RC_REG
000110  016660  000002  000002         MOV     2(SP),2(R0)             ; RC.REG,*
000116  016600  000002                 MOV     2(SP),R0                ;                                       3096
000122  042700  077777                 BIC     #77777,R0
000126  020027  100000                 CMP     R0,#-100000
000132  001006                         BNE     3$
000134  004737  000000V                 JSR     PC,FATAL.ERROR          ;                                       3099
000140  010300                         MOV     R3,R0                   ; CUR.PRIORITY,*                        3100
000142  104441                         TRAP    41
000144  005726                         TST     (SP)+                   ;                                       3098
000146  000515                         BR      9$
000150  010300                 3$:     MOV     R3,R0                   ; CUR.PRIORITY,*                        3105
000152  104441                         TRAP    41
```

```
000154  004737  000000V                        JSR     PC,QIO.OK               ;                       3107
000160  006000                                  ROR     R0
000162  103103                                  BCC     8$
000164  004737  000000V                        JSR     PC,QIO.GEN              ;                       3111
000170  013700  000110                          MOV     MX1,R0                  ;                       3113
000174  002424                                  BLT     5$
000176  132737  000001  000000G                BITB    #1,EOP.FLAG             ;                       3114
000204  001020                                  BNE     5$
000206  010016                                  MOV     R0,(SP)                 ;                       3117
000210  004737  000000G                        JSR     PC,SEND
000214  020027  000001                          CMP     R0,#1
000220  001005                                  BNE     4$
000222  105261  000000G                        INCB    QIO(R1)                 ; *(CTLR)               3120
000226  005237  000106'                        INC     RW.BALANCE              ;                       3121
000232  000405                                  BR      5$                      ;                       3117
000234  013716  000110'              4$:       MOV     MX1,(SP)                ;                       3126
000240  004737  000000G                        JSR     PC,PUT.PKT
000244  000452                                  BR      8$                      ;                       3125
000246  005737  000112'              5$:       TST     MX2                     ;                       3131
000252  002447                                  BLT     8$
000254  132737  000001  000000G                BITB    #1,EOP.FLAG             ;                       3132
000262  001043                                  BNE     8$
000264  104422                      6$:         TRAP    22                      ;                       3136
000266  127727  000000G 000004                 CMPB    @DCT.ADDR,#4            ;                       3138
000274  103373                                  BHIS    6$
000276  013716  000112'                        MOV     MX2,(SP)                ;                       3140
000302  004737  000000G                        JSR     PC,SEND
000306  020027  000001                          CMP     R0,#1
000312  001005                                  BNE     7$
000314  105261  000000G                        INCB    QIO(R1)                 ; *(CTLR)               3143
000320  005037  000106'                        CLR     RW.BALANCE              ;                       3144
000324  000422                                  BR      8$                      ;                       3140
000326  013716  000000G              7$:       MOV     CRN.LOW,(SP)            ;                       3149
000332  013746  000000G                        MOV     CRN.HIGH,-(SP)
000336  012746  000000G                        MOV     #DBM121,-(SP)
000342  012746  000003                          MOV     #3,-(SP)
000346  010600                                  MOV     SP,R0                   ; SP,*
000350  104417                                  TRAP    17
000352  105037  001170'                        CLRB    COMPARE.DATA            ;                       3150
000356  013716  000112'                        MOV     MX2,(SP)                ;                       3151
000362  004737  000000G                        JSR     PC,PUT.PKT
000366  062706  000006                          ADD     #6,SP                   ;                       3148
000372  005726                      8$:         TST     (SP)+                   ;                       3085
000374  005201                                  INC     R1                      ; CTLR                  3084
000376  000243                                  .WORD CLV!CLC

000400  003610                                  BLE     2$
000402  104422                      9$:         TRAP    22                      ;                       3157
000404  004737  000000V                        JSR     PC,PROC.RETPKT          ;                       3161
000410  004737  000000V                        JSR     PC,QIO.OUT              ;                       3164
000414  006000                                  ROR     R0
000416  103011                                  BCC     12$
000420  105777  000000G                        TSTB    @DCT.ADDR               ;                       3165
```

```
000424  001402                                  BEQ     11$
000426  000137  007530'                 10$:    JMP     1$
000432  132737  000001  000000G         11$:    BITB    #1,EOP.FLAG                     ;                       3166
000440  001772                                  BEQ     10$
000442  052777  040000  000000G         12$:    BIS     #40000,@DCT.ADDR                ;                       3169
000450  005726                                  TST     (SP)+                           ;                       3051
000452  000207                                  RTS     PC

; Routine Size:  150 words,     Routine Base:  $CODE$ + 7510
; Maximum stack depth per invocation:  11 words


;   3173  1
```

```
;    3174  1     GLOBAL routine MD_INIT : novalue =
;    3175  1
;    3176  1     !+
;    3177  1     !       THIS ROUTINE IS CALLED BY ROUTINE MULTI DRIVE TO INITIALIZE DATA ITEMS
;    3178  1     !       USED BY THE MULTI-DRIVE TEST.
;    3179  1     !
;    3180  1
;    3181  2         begin
;    3182  2
;    3183  2     !!ZZZ    local
;    3184  2     !!ZZZ    AVG_XFER_SIZE : word,                          ! SIZE (BYTES) OF AN AVERAGE I/O XFER
;    3185  2     !!ZZZ    QUICK_PASS_CNT : word;                         ! AVG NO. OF I/O OPERATIONS IN A QUICK PASS
;    3186  2
;    3187  2         if not .INIT_OCCURED                               ! IF THIS IS A START
;    3188  2         then
;    3189  2             INIT_IO_BUFF ();                               ! PARTITION FREE MEMORY INTO I/O BUFFERS
;    3190  2
;    3191  2         if (.ENTRY_REASON neq CONT) and                    ! IF START, RESTART, OR PWR FAIL
;    3192  3            (.ENTRY_REASON neq NEW_PASS)
;    3193  2         then
;    3194  2
;    3195  2             incr CTLR from 0 to (MAX_CTLR - 1) do
;    3196  3                 begin
;    3197  3                 SET_CPAR (.CTLR);
;    3198  3
;    3199  4                 INCR DISK FROM (0 + OF_UN) TO (3 + UNIT_SIZE                    !ZZZ
;    3200  3                      + OF_UN) BY UNIT_SIZE DO                                   !ZZZ
;    3201  4                      BEGIN                                                      !ZZZ
;    3202  4                      SET_UPAR (.DISK);                                          !ZZZ
;    3203  4                      DPST [.L$LUN] = DP_CNT;        !INIT DATA PTRN SEQ TABLEZZZ
;    3204  3                      END;                                                       !ZZZ
;    3205  3
;    3206  2                 END;                                                           !ZZZ
;    3207  2             INCR COUNT FROM 0 TO (QIO_PER_CTLR * MAX_CTLR - 1) DO   !INIT      !ZZZ
;    3208  2                 BUFF_OWN [.COUNT] = -1;             !I/O BUFF ALLOC TABLE      !ZZZ
;    3209  1             END;                                    !END MD_INIT               !ZZZ


                                        .SBTTL  MD.INIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G            MD.INIT::
                                    JSR     R1,$SAVE2                       ;                       3174
000004  132737  000001  000000G            BITB    #1,INIT.OCCURED         ;                       3187
000012  001002                      BNE     1$
000014  004737  000000V                    JSR     PC,INIT.IO.BUFF         ;                       3189
000020  123727  000000G 000003     1$:    CMPB    ENTRY.REASON,#3         ;                       3191
000026  001433                      BEQ     4$
000030  123727  000000G 000005             CMPB    ENTRY.REASON,#5         ;                       3192
000036  001427                      BEQ     4$
000040  005002                      CLR     R2                              ; CTLR                  3195
000042  010246                      MOV     R2,-(SP)                        ; CTLR,*                3197
000044  004737  000000G            JSR     PC,SET.CPAR
000050  012701  000003             MOV     #3,R1                           ; *,DISK                3199
000054  010116                      MOV     R1,(SP)                         ; DISK,*                3202
```

```
000056  004737  000000G                        JSR     PC,SET.UPAR
000062  013700  000000G                        MOV     L$LUN,R0                ;                               3203
000066  112760  000025  000050'                MOVB    #25,DPST(R0)
000074  062701  000012                         ADD     #12,R1                  ; *,DISK                        3199
000100  020127  000041                         CMP     R1,#41                  ; DISK,*
000104  003763                                 BLE     3$
000106  005726                                 TST     (SP)+                   ;                               3196
000110  005202                                 INC     R2                      ; CTLR                          3195
000112  000243                                 .WORD CLV!CLC

000114  003752                                 BLE     2$
000116  005000                         4$:     CLR     R0                      ; COUNT                         3207
000120  112760  000377  000000G       5$:     MOVB    #377,BUFF.OWN(R0)       ; *,*(COUNT)                    3208
000126  005200                                 INC     R0                      ; COUNT                         3207
000130  020027  000007                         CMP     R0,#7                   ; COUNT,*
000134  003771                                 BLE     5$
000136  000207                                 RTS     PC                      ;                               3174

; Routine Size:  48 words,      Routine Base:  $CODE$ + 10164
; Maximum stack depth per invocation:  5 words


;   3210  1
```

```
;    3211  1     GLOBAL routine INIT_IO_BUFF : novalue =
;    3212  1
;    3213  1     !+
;    3214  1     !       THIS ROUTINE IS CALLED BY MD_INIT WHEN THE MULTI-DRIVE TEST IS FIRST
;    3215  1     !       STARTED. IT IS RESPONSIBLE FOR PARTITIONING FREE MEMORY INTO A
;    3216  1     !       COLLECTION OF I/O BUFFERS. THE SIZE OF EACH I/O BUFFER IS DETERMINED
;    3217  1     !       BY A NUMBER OF FACTORS, INCLUDING THE NUMBER OF UNITS, THE NUMBER OF
;    3218  1     !       CONTROLLERS, AND THE SIZE OF FREE MEMORY.
;    3219  1     !
;    3220  1     !       ONCE THE BUFFER SIZE IS DETERMINED, THE NUMBER OF I/O BUFFERS IS
;    3221  1     !       CALCULATED. FINALLY, THE BUFFER ADDRESS (BUFF_ADDR) TABLE IS LOADED
;    3222  1     !       WITH FIXED BUFFER DESCRIPTORS THAT ARE USED IN THE ALLOCATION AND
;    3223  1     !       DEALLOCATION PROCESS.
;    3224  1     !
;    3225  1     !       IMPLICIT INPUTS:
;    3226  1     !               CTLR_CNT - THE NUMBER OF CONTROLLERS CONFIGURED
;    3227  1     !               L$UNIT - THE NUMBER OF UNITS AVAILABLE FOR TESTING
;    3228  1     !               FREE_MEM_ADDR - START OF FREE MEMORY
;    3229  1     !-
;    3230  1
;    3231  2         begin
;    3232  2         BUFF_ADDR [0] = (.FREE_MEM_ADDR + 2 + 1) and %o'177776' ;          ! START OF READ/WRITE BUFFERS
;    3233  2
;    3234  2         while (.BUFF_ADDR [0] and %o'37') neq 0 do                         ! FORCE FIRST I/O BUFFER TO START
;    3235  2             BUFF_ADDR [0] = .BUFF_ADDR [0] + 2;                            ! ON EVEN BOUNDARY
;    3236  2
;    3237  2         BYTS_PER_QIO = ((.DRS_START - .BUFF_ADDR [0]) / (QIO_PER_CTLR * MAX_CTLR)) and %o'177740';
;    3238  2                                                                            ! MAX TRANSFER SIZE
;    3239  2
;    3240  2         if .BYTS_PER_QIO gtru MAX_XFER_SIZE
;    3241  2         then
;    3242  2             BYTS_PER_QIO = MAX_XFER_SIZE;                                  ! ADJUST TRANSFER SIZE LOWER
;    3243  2
;    3244  2         if .BYTS_PER_QIO lssu 32
;    3245  2         then
;    3246  3             begin
;    3247  3             ERRSF (2, EGS_02, 0);                                          ! ERROR IF NOT ENOUGH MEMORY
;    3248  3             DOCLN;
;    3249  2             end;
;    3250  2
;    3251  2         if (QIO_PER_CTLR * MAX_CTLR) gtru 1
;    3252  2         then
;    3253  2
;    3254  2             incr index from 1 to (QIO_PER_CTLR * MAX_CTLR - 1) do          ! INIT REMAINING TABLE ENTRIES
;    3255  2                 BUFF_ADDR [.index] = .BUFF_ADDR [.index - 1] + .BYTS_PER_QIO;  ! FIXED BUFFER ADDRESS
;    3256  2
;    3257  1         end;                                                               ! ROUTINE INIT_IO_BUFF


                                        .SBTTL  INIT.IO.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          INIT.IO.BUFF::
                                        JSR     R1,$SAVE3                       ;  3211
000004  013700  000000G                 MOV     FREE.MEM.ADDR,R0                ;  3232
```

```
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$:     BIT     #37,BUFF.ADDR                   ;                       3234
000034  001404                                BEQ     2$
000036  062737  000002  000000G               ADD     #2,BUFF ADDR                    ;                       3235
000044  000770                                BR      1$                              ;                       3234
000046  013746  001160'              2$:     MOV     DRS.START, (SP)                 ;                       3237
000052  163716  000000G                       SUB     BUFF.ADDR,(SP)
000056  012746  000010                        MOV     #10,-(SP)
000062  004737  000000G                       JSR     PC,BL$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              ;                       3240
000106  101403                                BLOS    3$
000110  012737  001400  000000G               MOV     #1400,BYTS.PER.QIO              ;                       3242
000116  023727  000000G 000040        3$:     CMP     BYTS.PER.QIO,#40                ;                       3244
000124  103005                                BHIS    4$
000126  104454                                TRAP    54                              ;                       3247
000130  000002                                .WORD   2
000132  000000G                               .WORD   EGS.02
000134  000000                                .WORD   0
000136  104444                                TRAP    44
000140  012702  000001              4$:     MOV     #1,R2                           ; *,INDEX               3251
000144  010200                      5$:     MOV     R2,R0                           ; INDEX,*               3255
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                 3251
000172  020227  000007                      CMP     R2,#7                           ; INDEX,*
000176  003762                              BLE     5$
000200  022626                              CMP     (SP)+,(SP)+                     ;                       3231
000202  000207                              RTS     PC                              ;                       3211

; Routine Size:  66 words,      Routine Base:  $CODE$ + 10324
; Maximum stack depth per invocation:  8 words
```

```
;    3258  1     GLOBAL routine QIO_OK =
;    3259  1
;    3260  1
;    3261  1     !+
;    3262  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE "EXECUTIVE" IN ORDER TO
;    3263  1     !       DETERMINE WHETHER OR NOT A QIO REQUEST (OR QIO PAIR) SHOULD BE
;    3264  1     !       GENERATED TO THE CURRENT CONTROLLER. A VALUE OF "TRUE" IS RETURNED IF
;    3265  1     !       THE CONTROLLER MEETS 3 REQUIREMENTS:
;    3266  1     !
;    3267  1     !               A. THE CONTROLLER IS ONLINE;
;    3268  1     !               B. THE NUMBER OF OUTSTANDING QIOs IS AT LEAST 2 LESS THAN THE
;    3269  1     !                       MAXIMUM ALLOWED FOR ANY ONE CONTROLLER;
;    3270  1     !               C. THERE IS AT LEAST ONE DISK ONLINE TO THE CONTROLLER.
;    3271  1     !
;    3272  1     !       IF ANY OF THESE TEST FAIL, THEN A VALUE OF "FALSE" IS RETURNED.
;    3273  1     !
;    3274  1     !       IMPLICIT INPUTS:
;    3275  1     !               CCTLR   CURRENT CONTROLLER NUMBER
;    3276  1     !               CST_ADDR   ADDRESS OF CURRENT CONTROLLER'S CST
;    3277  1     !
;    3278  1
;    3279  1
;    3280  1         if (.CST_ADDR [STATE] eql ONLINE) and               ! IF CONTROLLER IS ONLINE
;    3281  1            (not .EOP_FLAG) and
;    3282  1            ((.QIO [.CCTLR] + 2) lequ QIO_PER_CTLR) and      ! IF OUTSTANDING QIO COUNT IS O.K.
;    3283  2            (.CST_ADDR [U_CNT] neq 0)                        ! IF THERE IS VALID UNIT
;    3284  2
;    3285  1         then
;    3286  1             return TRUE                                     ! "TRUE' EXIT POINT
;    3287  1
;    3288  1         else
;    3289  1             return FALSE;                                   ! "FALSE" EXIT POINT


                                        .SBTTL  QIO.OK MULTI-DRIVE TEST ROUTINES
000000  013700  000000G           QIO.OK::MOV     CST.ADDR,R0                     ;                       3280
000004  005760  000002                    TST     2(R0)
000010  100027                           BPL     1$
000012  132737  000001  000000G          BITB    #1,EOP.FLAG                     ;                       3281
000020  001023                           BNE     1$
000022  013700  000000G                  MOV     CCTLR,R0                        ;                       3282
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                     ;                       3283
000054  105760  000005                   TSTB    5(R0)
000060  001403                           BEQ     1$
000062  012700  000001                   MOV     #1,R0                           ;                       3289
000066  000207                           RTS     PC
000070  005000                    1$:    CLR     R0
000072  000207                           RTS     PC                              ;                       3258
```

```
; Routine Size:  30 words,      Routine Base:  $CODE$ + 10530
; Maximum stack depth per invocation:  0 words

;   3290  1
```

```
;    3291  1     GLOBAL routine QIO_OUT -
;    3292  1
;    3293  1     !+
;    3294  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR DETERMINING THE
;    3295  1     !       END OF THE MULTI-DRIVE TEST. ITS PURPOSE IS TO EXAMINE THE QIO VECTOR
;    3296  1     !       FOR ANY OUTSTANDING QIOs ON ANY CONTROLLER. A VALUE OF "TRUE" IS
;    3297  1     !       RETURNED IF THERE IS AT LEAST ONE QIO OUTSTANDING ON ANY CONTROLLER.
;    3298  1     !       OTHERWISE, "FALSE" IS RETURNED INDICATING NO OUTSTANDING QIOs.
;    3299  1     !
;    3300  1
;    3301  2         begin
;    3302  2
;    3303  2         incr CTLR from 0 to (MAX_CTLR   1) do
;    3304  3             begin
;    3305  3             SET_CPAR (.CTLR);                       ! SET UP CURRENT CONTROLLER PARAMETERS
;    3306  3
;    3307  3             if .CST_ADDR [STATE] eql ONLINE         ! IF CONTROLLER IS ONLINE
;    3308  3             then
;    3309  3                 return TRUE;
;    3310  3
;    3311  2             end;
;    3312  2
;    3313  2         return FALSE;                               ! EXIT - NO CONTROLLERS ONLINE
;    3314  1         end;


                                        .SBTTL  QIO.OUT MULTI-DRIVE TEST ROUTINES
000000  010146                  QIO.OUT::
                                        MOV     R1,-(SP)                        ;                       3291
000002  005001                          CLR     R1                              ; CTLR                  3303
000004  010146                  1$:     MOV     R1,-(SP)                        ; CTLR,*                3305
000006  004737  000000G                 JSR     PC,SET.CPAR
000012  013700  000000G                 MOV     CST.ADDR,R0                     ;                       3307
000016  005760  000002                  TST     2(R0)
000022  100004                          BPL     2$
000024  005726                          TST     (SP)+                           ;                       3309
000026  012700  000001                  MOV     #1,R0
000032  000405                          BR      3$
000034  005726                  2$:     TST     (SP)+                           ;                       3304
000036  005201                          INC     R1                              ; CTLR                  3303
000040  000243                          .WORD CLV!CLC

000042  003760                          BLE     1$
000044  005000                          CLR     R0                              ;                       3301
000046  012601                  3$:     MOV     (SP)+,R1                        ;                       3291
000050  000207                          RTS     PC

; Routine Size:  21 words,      Routine Base:  $CODE$ + 10624
; Maximum stack depth per invocation:  3 words
```

```
;    3315  1     GLOBAL routine QIO_GEN : novalue =
;    3316  1
;    3317  1     !+
;    3318  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR AN ONLINE
;    3319  1     !       CONTROLLER ELIGIBLE TO RECEIVE I/O TRANSFER REQUESTS. IT IS
;    3320  1     !       RESPONSIBLE FOR SECURING ONE OR TWO MSCP PACKETS AND LOADING THEM
;    3321  1     !       WITH VARIOUS PARAMETERS COMPRISING THE I/O REQUEST. THE I/O REQUEST
;    3322  1     !       GENERATED HERE IS DESTINED TO A PARTICULAR UNIT SELECTED AT RANDOM FROM
;    3323  1     !       THOSE CONFIGURED UNDER THE CURRENT CONTROLLER.
;    3324  1     !
;    3325  1     !       EACH FIELD OF THE PACKET(S) IS LOADED WITHIN INDIVIDUAL ROUTINES
;    3326  1     !       (QIO_FUNC, QIO_LBN, QIO_SIZE, ETC.). MOST OF THE VALUES SELECTED FOR
;    3327  1     !       EACH FIELD ARE BASED ON A SET OF RANDOM NUMBER GENERATED AT THE START.
;    3328  1     !
;    3329  1     !       UNDER NORMAL CIRCUMSTANCES, ONLY ONE I/O REQUEST IS GENERATED. HOWEVER,
;    3330  1     !       IF THIS I/O REQUEST IS A "WRITE", AND IF THE OPERATOR SELECTED THE
;    3331  1     !       OPTION FOR HOST WRITE-COMPARES, THEN A SECOND "READ" REQUEST WILL BE
;    3332  1     !       GENERATED WITH THE SAME LBN AND BYTE COUNT.
;    3333  1     !
;    3334  1     !       AFTER THE PACKET(S) HAVE BEEN LOADED, THIS ROUTINE REGAINS CONTROL
;    3335  1     !       AND ATTEMPTS TO GET ONE OR TWO I/O BUFFERS FOR THE ACTUAL DATA
;    3336  1     !       TRANSFERS. THE SUCCESS / FAIL STATUS OF THIS ENTIRE OPERATION IS
;    3337  1     !       PASSED BACK TO THE CALLER THROUGH THE GLOBALS "MX1" AND "MX2"; THEY
;    3338  1     !       CONTAIN VALID MSCP PACKET INDECES, OR -1.
;    3339  1     !
;    3340  1     !       IMPLICIT INPUTS:
;    3341  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3342  1     !
;    3343  1
;    3344  2         begin
;    3345  2         MX2 = -1;                                   ! ASSUME FAILURE IN SECURING 2ND PACKET
;    3346  2
;    3347  2         if (MX1 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 1ST PACKET. IF FAILURE
;    3348  2         then
;    3349  2             return;                                 ! NO POINT IN CONTINUING
;    3350  2
;    3351  2         if (MX2 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 2ND PACKET. IF FAILURE
;    3352  2         then
;    3353  3             begin
;    3354  3             PUT_PKT (.MX1);                         ! RETURN 1ST PACKET TO POOL
;    3355  3             MX1 = -1;                               ! INDICATE FAILURE
;    3356  3             return;                                 ! DONE
;    3357  2             end;
;    3358  2
;    3359  2         MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2);     ! CALCULATE STARTING ADDRESSES
;    3360  2         MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2);     !     OF BOTH PACKETS
;    3361  2         GET_RANDOM ();                              ! GENERATE A SET OF RANDOM NUMBERS
;    3362  2         QIO_UNIT ();                                ! LOAD RANDOM UNIT NUMBER INTO PACKETS
;    3363  2
;    3364  2         if .EOP_FLAG                                ! RETURN IF NO UNIT ONLINE
;    3365  2         then
;    3366  2             return;
;    3367  2
```

M10


```
;  3368  2          QIO_FUNC ();                            ! LOAD RANDOM FUNCTION CODE (OPCODE)
;  3369  2          QIO_LBN ();                             ! LOAD LBN (RANDOM OR SEQUENTIAL)
;  3370  2          QIO_SIZE ();                            ! LOAD RANDOM BYTE COUNT
;  3371  2          GET IO_BUFF (MAD1 [BUF 0]);             ! TRY TO GET AN I/O BUFFER
;  3372  2
;  3373  2          if .MX2 geq 0                           ! IF TWO QIOs ARE TO BE ISSUED
;  3374  2          then
;  3375  3              begin
;  3376  3              GET IO_BUFF (MAD2 [BUF 0]);         ! TRY TO GET 2ND I/O BUFFER
;  3377  3
;  3378  3              if .MAD2 [BUF 0] eqla 0             ! IF 2ND BUFFER ALLOCATION FAILED
;  3379  3              then
;  3380  4                  begin
;  3381  4
;  3382  4                  if .MAD1 [BUF 0] neqa 0         ! IF 1ST I/O BUFFER WAS ALLOCATED
;  3383  4                  then
;  3384  5                      begin
;  3385  5                      PUT_IO_BUFF (MAD1 [BUF 0]); ! RETURN 1ST I/O BUFFER TO POOL
;  3386  5                      MAD1 [BUF_0] = 0;           ! MARK IT AS FAILED
;  3387  4                      end;
;  3388  4
;  3389  4                  PUT_PKT (.MX2);                 ! RETURN 2ND PACKET TO POOL
;  3390  4                  MX2 =  1;                       ! INDICATE FAILURE
;  3391  3                  end;                            ! IF 2ND I/O BUFFER ALLOCATION FAILED
;  3392  3
;  3393  2              end;                                ! IF TWO QIOs ARE TO BE ISSUED
;  3394  2
;  3395  2          if .MAD1 [BUF_0] eqla 0                 ! IF 1ST I/O BUFFER ALLOCATION FAILED
;  3396  2          then
;  3397  3              begin
;  3398  3              PUT_PKT (.MX1);                     ! RETURN 1ST PACKET TO POOL
;  3399  3              MX1 = -1;                           ! INDICATE FAILURE
;  3400  3              end
;  3401  2          else
;  3402  2
;  3403  2              if .MAD1 [OPCODE] eql OP WRT        ! OTHERWISE, IF 1ST OPCODE IS A WRITE  (ALL IS O.K.)
;  3404  2              then
;  3405  2                  FILL_BUFF ();                   ! FILL 1ST I/O BUFFER WITH APPROPRIATE DATA PATTERN
;  3406  2
;  3407  1          end;                                    ! ROUTINE QIO GEN


                                        .SBTTL  QIO.GEN MULTI DRIVE TEST ROUTINES
000000  012737  177777  000112'  QIO.GEN::
                                        MOV     #-1,MX2                         ;                       3345
000006  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3347
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$                              ;                       3349
000030  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3351
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)               ;                       3354
000056  004737  000000G                       JSR     PC,PUT.PKT
000062  012737  177777  000110'               MOV     #-1,MX1                 ;                       3355
000070  005726                                TST     (SP)+                   ;                       3356
000072  000207                                RTS     PC                      ;                       3353
000074  013746  000110'             1$:       MOV     MX1,-(SP)               ;                       3359
000100  012746  000106                        MOV     #106,-(SP)
000104  004737  000000G                       JSR     PC,BL$MUL
000110  062700  000000G                       ADD     #MSCP.PKT,R0
000114  010037  000114'                       MOV     R0,MAD1
000120  013716  000112'                       MOV     MX2,(SP)                ;                       3360
000124  012746  000106                        MOV     #106,-(SP)
000130  004737  000000G                       JSR     PC,BL$MUL
000134  062700  000000G                       ADD     #MSCP.PKT,R0
000140  010037  000116'                       MOV     R0,MAD2
000144  004737  000000V                       JSR     PC,GET.RANDOM           ;                       3361
000150  004737  000000V                       JSR     PC,QIO.UNIT             ;                       3362
000154  132737  000001  000000G             BITB    #1,EOP.FLAG             ;                       3364
000162  001103                                BNE     5$                      ;                       3315
000164  004737  000000V                       JSR     PC,QIO.FUNC             ;                       3368
000170  004737  000000V                       JSR     PC,QIO.LBN              ;                       3369
000174  004737  000000V                       JSR     PC,QIO.SIZE             ;                       3370
000200  013716  000114'                       MOV     MAD1,(SP)               ;                       3371
000204  062716  000032                        ADD     #32,(SP)
000210  004737  000000G                       JSR     PC,GET.IO.BUFF
000214  005737  000112'                       TST     MX2                     ;                       3373
000220  002437                                BLT     3$
000222  013716  000116'                       MOV     MAD2,(SP)               ;                       3376
000226  062716  000032                        ADD     #32,(SP)
000232  004737  000000G                       JSR     PC,GET.IO.BUFF
000236  013700  000116'                       MOV     MAD2,R0                 ;                       3378
000242  005760  000032                        TST     32(R0)
000246  001024                                BNE     3$
000250  013700  000114'                       MOV     MAD1,R0                 ;                       3382
000254  062700  000032                        ADD     #32,R0
000260  005710                                TST     (R0)
000262  001407                                BEQ     2$
000264  010016                                MOV     R0,(SP)                 ;                       3385
000266  004737  000000G                       JSR     PC,PUT.IO.BUFF
000272  013700  000114'                       MOV     MAD1,R0                 ;                       3386
000276  005060  000032                        CLR     32(R0)
000302  013716  000112'             2$:       MOV     MX2,(SP)                ;                       3389
000306  004737  000000G                       JSR     PC,PUT.PKT
000312  012737  177777  000112'               MOV     #-1,MX2                 ;                       3390
000320  013700  000114'             3$:       MOV     MAD1,R0                 ;                       3395
000324  005760  000032                        TST     32(R0)
000330  001010                                BNE     4$
000332  013716  000110'                       MOV     MX1,(SP)                ;                       3398
000336  004737  000000G                       JSR     PC,PUT.PKT
```

```
000342  012737  177777  000110              MOV     #1,MX1                          ;                       3399
000350  000410                              BR      5$                              ;                       3395
000352  013700  000114              4$:     MOV     MAD1,R0                         ;                       3403
000356  126027  000022  000042            CMPB    22(R0),#42
000364  001002                              BNE     5$
000366  004737  000000V                     JSR     PC,FILL BUFF                    ;                       3405
000372  062706  000006              5$:     ADD     #6,SP                           ;                       3344
000376  000207                      6$:     RTS     PC                              ;                       3315

; Routine Size:  128 words,     Routine Base:  $CODE$ + 10676
; Maximum stack depth per invocation:  4 words
```

```
;    3408  1     GLOBAL routine GET_RANDOM : novalue =
;    3409  1
;    3410  1     !+
;    3411  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO GENERATE A SET OF RANDOM NUMBERS,
;    3412  1     !       AND TO STORE THEM INTO THE RANDOM NUMBER TABLE (RANDOM). THE RANDOM
;    3413  1     !       NUMBERS ARE USED TO SELECT I/O REQUEST PARAMETERS FOR THE CURRENT QIO
;    3414  1     !       OR QIO PAIR. IN ADDITION, IF DATA PATTERN #1 IS BEING USED, THESE
;    3415  1     !       RANDOM NUMBERS WILL BE USED IN THE WRITE OPERATION.
;    3416  1     !
;    3417  1
;    3418  2         begin
;    3419  2
;    3420  2         own
;    3421  2             SEED : word initial (173),
;    3422  2             NEXT_RANDNUM : word initial (245);
;    3423  2
;    3424  2         incr COUNT from 0 to (RDM_LEN - 1) do
;    3425  3             begin
;    3426  3             SEED = (.SEED * .NEXT_RANDNUM + 1) * 4;
;    3427  3             NEXT_RANDNUM = (.NEXT_RANDNUM / 4) * .SEED;
;    3428  3             RANDOM [.COUNT] = .NEXT_RANDNUM;
;    3429  2             end;
;    3430  2
;    3431  1         end;


001200                                  .PSECT  $GGG$,  RO
001200  000255                 SEED:  .WORD   255
001202  000365                 NEXT.RANDNUM:
                                        .WORD   365


                                        .SBTTL  GET.RANDOM MULTI-DRIVE TEST ROUTINES
011276                                  .PSECT  $CODE$,  RO

000000  004137  000000G        GET.RANDOM::
                                        JSR     R1,$SAVE3                       ;                       3408
000004  013703  001200'                 MOV     SEED,R3                         ;                       3426
000010  013702  001202'                 MOV     NEXT.RANDNUM,R2
000014  005001                          CLR     R1                              ; COUNT                 3424
000016  010200                 1$:     MOV     R2,R0                           ;                       3426
000020  060300                          ADD     R3,R0
000022  006300                          ASL     R0
000024  006300                          ASL     R0
000026  010037  001200'                 MOV     R0,SEED
000032  062737  000004  001200'        ADD     #4,SEED
000040  010246                          MOV     R2,-(SP)                        ;                       3427
000042  012746  000004                  MOV     #4,-(SP)
000046  004737  000000G                 JSR     PC,BL$DIV
000052  013703  001200'                 MOV     SEED,R3
000056  060300                          ADD     R3,R0
000060  010037  001202'                 MOV     R0,NEXT.RANDNUM
```

```
000064  010002                          MOV     R0,R2                           ; NEXT.RANDNUM,*                        3428
000066  010261  0000006                 MOV     R2,RANDOM(R1)                   ; *,*(COUNT)
000072  022626                          CMP     (SP)+,(SP)+                     ;                                       3425
000074  062701  000002                  ADD     #2,R1                           ; *,COUNT                               3424
000100  020127  000036                  CMP     R1,#36                          ; COUNT,*
000104  003744                          BLE     1$
000106  000207                          RTS     PC                              ;                                       3408

; Routine Size:  36 words.      Routine Base:  $CODE$ + 11276
; Maximum stack depth per invocation:  7 words
```

```
;    3432  1      GLOBAL routine QIO_UNIT : novalue =
;    3433  1
;    3434  1      !+
;    3435  1      !       THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
;    3436  1      !       CONFIGURED UNDER THE CURRENT CONTROLLER (CCTLR) TO BE USED FOR THE
;    3437  1      !       CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
;    3438  1      !       UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
;    3439  1      !       RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
;    3440  1      !
;    3441  1      !       IMPLICIT INPUTS:
;    3442  1      !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    3443  1      !
;    3444  1      !       IMPLICIT OUTPUTS:
;    3445  1      !               THE RD/RX DISK NUMBER (DISK ADDRESS) IS LOADED INTO THE
;    3446  1      !                       APPROPRIATE FIELD OF BOTH MSCP PACKETS.
;    3447  1      !
;    3448  1
;    3449  2          begin
;    3450  2
;    3451  2          local
;    3452  2              MOD_COUNT : byte,
;    3453  2              TBL_COUNT : byte,
;    3454  2              SELECT_RD : byte initial (byte (TRUE)),
;    3455  2      !ZZZ    RD_COUNT : word initial (0),
;    3456  2              RX_COUNT : word initial (0);
;    3457  2
;    3458  2      !
;    3459  2      ! THE UNITS WILL BE SELECTED ON AN ADJUSTABLE RATIO, RD51/52 TO RX50,
;    3460  2      ! SELECTED VIA THE SOFTWARE PARAMETERS
;    3461  2      !
;    3462  2      ! THIS MODE IS FOR SELECTING DEVICES ON THE FOLLOWING SCHEME:
;    3463  2      ! CHOOSE A DEVICE AND KEEP IT SELECTED FOR A CONSTANT TIME, THEN
;    3464  2      ! MOVE TO THE NEXT. THIS IS NON-RANDOM, FIXED SEQUENTIAL OPERATIONAL
;    3465  2      ! MODE
;    3466  2      !
;    3467  2
;    3468  2                  RD_COUNT = 0;                                          !ZZZ
;    3469  2                  RX_COUNT = 0;                                          !ZZZ
;    3470  2
;    3471  2                  incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3472  2
;    3473  2                      if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3474  2                         (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3475  3                         (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3476  2                      then
;    3477  2
;    3478  3                          if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;    3479  2                          then
;    3480  2                              RD_COUNT = .RD_COUNT + 1        ! NUMBER OF RD51/52s UNDER TEST
;    3481  2                          else
;    3482  2                              RX_COUNT = .RX_COUNT + 1;       ! NUMBER OF RX50s UNDER TEST
;    3483  2
;    3484  2
```

```
;    3485  2
;    3486  2          if (not BIT_TST (SWP_FLAGS, SWF_RDM)) and          ! NOT RANDOM MODE
;    3487  3             (not BIT_TST (SWP_FLAGS, SWF_SEQ))              ! NOT RANDOM SEQUENTIAL MODE
;    3488  2          then
;    3489  2
;    3490  2              if (.BST_CNT neq 0) and
;    3491  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;    3492  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;    3493  3                 (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;    3494  2              then
;    3495  3                  begin                                      ! ALREADY WITHIN DEVICE
;    3496  3                  BST_CNT = .BST_CNT   1;
;    3497  3                  SET_UPAR (.BST_DEV);
;    3498  3                  MAD1 [DK_NUM] = .CDISK;
;    3499  3                  MAD2 [DK_NUM] = .CDISK;
;    3500  3                  return;
;    3501  3                  end
;    3502  2              else
;    3503  3                  begin                                      ! GET NEW DEVICE
;    3504  3
;    3505  3      !ZZZ        incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR   1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3506  3      !ZZZ
;    3507  3      !ZZZ            if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3508  3      !ZZZ               (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3509  3      !ZZZ               (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3510  3      !ZZZ            then
;    3511  3      !ZZZ
;    3512  3      !ZZZ                if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;    3513  3      !ZZZ                then
;    3514  3      !ZZZ                    RD_COUNT = .RD_COUNT + 1       ! NUMBER OF RD51/52s UNDER TEST
;    3515  3      !ZZZ                else
;    3516  3      !ZZZ                    RX_COUNT = .RX_COUNT + 1;      ! NUMBER OF RX50s UNDER TEST
;    3517  3
;    3518  3                      incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF UN) by UNIT_SIZE do
;    3519  4                          begin
;    3520  4
;    3521  4                          if (.BST_DEV eql 0) or
;    3522  5                             (.BST_DEV eql ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN))
;    3523  4                          then
;    3524  4                              BST_DEV = OF_UN
;    3525  4                          else
;    3526  4                              BST_DEV = .BST_DEV + UNIT_SIZE;
;    3527  4
;    3528  4                          if (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;    3529  4                             (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;    3530  5                             (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;    3531  4                          then
;    3532  5                              begin
;    3533  5
;    3534  5                              if .CST_ADDR [.BST_DEV + OF_DATA, D_TYPE] eql REMOVABLE
;    3535  5                              then
;    3536  5                                  BST_CNT = .RX_MAX_SEQ_CNT / .RX_COUNT
;    3537  5                              else
```

```
;    3538  5                          BST_CNT = .RD_MAX_SEQ_CNT / .RD_COUNT;
;    3539  5
;    3540  5                      if .BST_CNT eql 0
;    3541  5                      then
;    3542  5                          BST_CNT = 1;
;    3543  5
;    3544  5                      SET_UPAR (.BST_DEV);
;    3545  5                      MAD1 [DK_NUM] = .CDISK;
;    3546  5                      MAD2 [DK_NUM] = .CDISK;
;    3547  5                      return;
;    3548  4                      end;
;    3549  4
;    3550  3                  end;
;    3551  3
;    3552  2              end;
;    3553  2
;    3554  2      !
;    3555  2      ! RANDOM SELECTION OF DRIVES
;    3556  2      !
;    3557  2      !
;    3558  2      ! DETERMINE IF RD51/52s ARE TO BE SELECTED
;    3559  2      !
;    3560  2
;    3561  2          if ((.RANDOM [RDM_LEN - 1] and %o'077777') mod 100) gequ .SWP_RAT
;    3562  2          then
;    3563  2              SELECT_RD = FALSE;
;    3564  2
;    3565  2      !
;    3566  2      ! IF RD51/52s SELECTED
;    3567  2      !
;    3568  2      ! COUNT NUMBER OF RD51/52s AVAILABLE
;    3569  2      !
;    3570  2
;    3571  2          if .SELECT_RD
;    3572  2          then
;    3573  3              begin
;    3574  3              MOD_COUNT = 0;                                  ! COUNT THE NUMBER OF RDs UNDER TEST
;    3575  3
;    3576  3              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3577  3
;    3578  3                  if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3579  3                     (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3580  3                     (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED) and
;    3581  4                     (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3582  3                  then
;    3583  4                      begin
;    3584  4                      STORAGE [.MOD_COUNT] = .OFFSET;
;    3585  4                      MOD_COUNT = .MOD_COUNT + 1;
;    3586  3                      end;
;    3587  3
;    3588  3      !
;    3589  3      ! SELECT ON OF THE RD51/52s
;    3590  3      !
```

```
;    3591  3
;    3592  3              if .MOD_COUNT neq 0                                ! IF AT LEAST ONE RD51/52 PRESENT
;    3593  3              then
;    3594  4                  begin
;    3595  4                  TBL_COUNT = 0;
;    3596  4
;    3597  4                  do
;    3598  5                      begin
;    3599  5                      SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3600  5                      TBL_COUNT = .TBL_COUNT + 1;
;    3601  5                      end
;    3602  5                  until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3603  5                         (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3604  4                         (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3605  4                        (.TBL_COUNT eql RDM_LEN);
;    3606  4
;    3607  4                  MAD1 [DK_NUM] = .CDISK;
;    3608  4                  MAD2 [DK_NUM] = .CDISK;
;    3609  4                  return;
;    3610  3                  end;
;    3611  3
;    3612  2              end;
;    3613  2
;    3614  2          !
;    3615  2          ! IF NO RD51/52 SELECTED, SELECT AN RX50
;    3616  2          !
;    3617  2          ! COUNT THE NUMBER OF RX50s
;    3618  2          !
;    3619  2
;    3620  2              MOD_COUNT = 0;
;    3621  2
;    3622  2              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3623  2
;    3624  2                  if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3625  2                     (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3626  2                     (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql REMOVABLE) and
;    3627  3                     (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3628  2                  then
;    3629  3                      begin
;    3630  3                      STORAGE [.MOD_COUNT] = .OFFSET;
;    3631  3                      MOD_COUNT = .MOD_COUNT + 1;
;    3632  2                      end;
;    3633  2
;    3634  2          !
;    3635  2          ! AND CHOOSE ONE!
;    3636  2          !
;    3637  2
;    3638  2              if .MOD_COUNT neq 0
;    3639  2              then
;    3640  3                  begin
;    3641  3                  TBL_COUNT = 0;
;    3642  3
;    3643  3                  do
```

```
;    3644  4                begin
;    3645  4                SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3646  4                TBL_COUNT = .TBL_COUNT + 1;
;    3647  4                end
;    3648  4            until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3649  4                   (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3650  3                   (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3651  3                  (.TBL_COUNT eql RDM_LEN);
;    3652  3
;    3653  3            MAD1 [DK_NUM] = .CDISK;
;    3654  3            MAD2 [DK_NUM] = .CDISK;
;    3655  3            return;
;    3656  2            end;
;    3657  2
;    3658  2        !
;    3659  2        ! IF NO UNIT SELECTED SO FAR BY ABOVE METHOD, SELECT ANY ONE AT RANDOM
;    3660  2        !
;    3661  2        ! COUNT ALL UNITS AVAILABLE
;    3662  2        !
;    3663  2
;    3664  2            MOD_COUNT = 0;
;    3665  2
;    3666  2            incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3667  2
;    3668  2                if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3669  2                   (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3670  3                   (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3671  2                then
;    3672  3                    begin
;    3673  3                    STORAGE [.MOD_COUNT] = .OFFSET;
;    3674  3                    MOD_COUNT = .MOD_COUNT + 1;
;    3675  2                    end;
;    3676  2
;    3677  2        !
;    3678  2        ! SELECT ANY ONE ONE UNIT AT RANDOM
;    3679  2        !
;    3680  2            if .MOD_COUNT neq 0
;    3681  2            then
;    3682  3                begin
;    3683  3                TBL_COUNT = 0;
;    3684  3
;    3685  3                do
;    3686  4                    begin
;    3687  4                    SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD COUNT]);
;    3688  4                    TBL_COUNT = .TBL_COUNT + 1;
;    3689  4                    end
;    3690  4                until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3691  4                       (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3692  3                       (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3693  3                      (.TBL_COUNT eql RDM_LEN);
;    3694  3
;    3695  3                MAD1 [DK_NUM] = .CDISK;
;    3696  3                MAD2 [DK_NUM] = .CDISK;
```

```
;    3697  3              return
;    3698  3              end
;    3699  3
;    3700  3      !
;    3701  3      ! DECLARE END-OF-PASS IF NO UNIT ONLINE
;    3702  3      !
;    3703  3
;    3704  2          else
;    3705  2              EOP_FLAG = TRUE;
;    3706  2
;    3707  1          end;                                          ! ROUTINE QIO_UNIT


                                        .SBTTL  QIO.UNIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          QIO.UNIT::
                                        JSR     R1,$SAVE4                       ;                                3432
000004  112704  000001                  MOVB    #1,R4                           ; *,SELECT.RD                    3449
000010  005003                          CLR     R3                              ; RX.COUNT
000012  005037  000000G                 CLR     RD.COUNT                        ;                                3468
000016  013702  000000G                 MOV     CST.ADDR,R2                     ;                                3473
000022  012701  000006                  MOV     #6,R1                           ; *,OFFSET                       3471
000026  010100              1$:         MOV     R1,R0                           ; OFFSET,*                       3473
000030  060200                          ADD     R2,R0
000032  032710  040000                  BIT     #40000,(R0)
000036  001415                          BEQ     3$
000040  032710  020000                  BIT     #20000,(R0)                     ;                                3474
000044  001412                          BEQ     3$
000046  032710  010000                  BIT     #10000,(R0)                     ;                                3475
000052  001007                          BNE     3$
000054  132710  000020                  BITB    #20,(R0)                        ;                                3478
000060  001403                          BEQ     2$
000062  005237  000000G                 INC     RD.COUNT                        ;                                3480
000066  000401                          BR      3$                              ;                                3478
000070  005203              2$:         INC     R3                              ; RX.COUNT                       3482
000072  062701  000024      3$:         ADD     #24,R1                          ; *,OFFSET                       3471
000076  020127  000102                  CMP     R1,#102                         ; OFFSET,*
000102  003751                          BLE     1$
000104  032737  000002  000000G         BIT     #2,SWP.FLAGS                    ;                                3486
000112  001163                          BNE     13$
000114  032737  001000  000000G         BIT     #1000,SWP.FLAGS                 ;                                3487
000122  001157                          BNE     13$
000124  005737  001150'                 TST     BST.CNT                         ;                                3490
000130  001447                          BEQ     4$
000132  013700  001152'                 MOV     BST.DEV,R0                      ;                                3491
000136  006300                          ASL     R0
000140  060200                          ADD     R2,R0
000142  032710  040000                  BIT     #40000,(R0)
000146  001440                          BEQ     4$
000150  013700  001152'                 MOV     BST.DEV,R0                      ;                                3492
000154  006300                          ASL     R0
000156  060200                          ADD     R2,R0
000160  032710  020000                  BIT     #20000,(R0)
000164  001431                          BEQ     4$
```

```
000166  013700  001152                        MOV     BST.DEV,R0              ;                       3493
000172  006300                                  ASL     R0
000174  060200                                  ADD     R2,R0
000176  032710  010000                        BIT     #10000,(R0)
000202  001022                                  BNE     4$
000204  005337  001150'                        DEC     BST.CNT                 ;                       3496
000210  013746  001152'                        MOV     BST.DEV,-(SP)           ;                       3497
000214  004737  000000G                       JSR     PC,SET.UPAR
000220  013700  000114'                        MOV     MAD1,R0                 ;                       3498
000224  013760  000000G 000016                MOV     CDISK,16(R0)
000232  013700  000116'                        MOV     MAD2,R0                 ;                       3499
000236  013760  000000G 000016                MOV     CDISK,16(R0)
000244  005726                                  TST     (SP)+                   ;                       3500
000246  000207                                  RTS     PC                      ;                       3495
000250  012702  000003              4$:       MOV     #3,R2                   ; *,OFFSET              3518
000254  013700  001152'              5$:       MOV     BST.DEV,R0              ;                       3521
000260  001403                                  BEQ     6$
000262  020027  000041                        CMP     R0,#41                  ;                       3522
000266  001004                                  BNE     7$
000270  012737  000003  001152'      6$:       MOV     #3,BST.DEV              ;                       3524
000276  000403                                  BR      8$                      ;                       3521
000300  062737  000012  001152'      7$:       ADD     #12,BST.DEV             ;                       3526
000306  013700  001152'              8$:       MOV     BST.DEV,R0              ;                       3528
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)             ;                       3529
000332  001446                                  BEQ     12$
000334  032710  010000                        BIT     #10000,(R0)             ;                       3530
000340  001043                                  BNE     12$
000342  132710  000020                        BITB    #20,(R0)                ;                       3534
000346  001004                                  BNE     9$
000350  013746  001176                        MOV     RX.MAX.SEQ.CNT,-(SP)    ;                       3536
000354  010346                                  MOV     R3,-(SP)                ; RX.COUNT,*
000356  000404                                  BR      10$
000360  013746  001174'              9$:       MOV     RD.MAX.SEQ.CNT,-(SP)    ;                       3538
000364  013746  000000G                       MOV     RD.COUNT,-(SP)
000370  004737  000000G              10$:      JSR     PC,BL$DIV
000374  010037  001150'                        MOV     R0,BST.CNT
000400  001003                                  BNE     11$                     ;                       3540
000402  012737  000001  001150'              MOV     #1,BST.CNT              ;                       3542
000410  013716  001152'              11$:      MOV     BST.DEV,(SP)            ;                       3544
000414  004737  000000G                       JSR     PC,SET.UPAR
000420  013700  000114'                        MOV     MAD1,R0                 ;                       3545
000424  013760  000000G 000016                MOV     CDISK,16(R0)
000432  013700  000116'                        MOV     MAD2,R0                 ;                       3546
000436  013760  000000G 000016                MOV     CDISK,16(R0)
000444  022626                                  CMP     (SP)+,(SP)+             ;                       3547
000446  000207                                  RTS     PC                      ;                       3532
000450  062702  000012              12$:      ADD     #12,R2                  ; *,OFFSET              3518
000454  020227  000041                        CMP     R2,#41                  ; OFFSET,*
000460  003675                                  BLE     5$
```

```
000462  013746  000036G            13$:    MOV     RANDOM+36,-(SP)             ;                       3561
000466  042716  100000                    BIC     #100000,(SP)
000472  012746  000144                    MOV     #144,-(SP)
000476  004737  000000G                    JSR     PC,BL$MOD
000502  022626                            CMP     (SP)+,(SP)+
000504  020037  000000G                    CMP     R0,SWP.RAT
000510  103401                            BLO     14$
000512  105004                            CLRB    R4                          ; SELECT.RD             3563
000514  006004            14$:            ROR     R4                          ; SELECT.RD             3571
000516  103105                            BCC     19$
000520  105003                            CLRB    R3                          ; MOD.COUNT             3574
000522  012701  000003                    MOV     #3,R1                       ; *,OFFSET              3576
000526  010100            15$:            MOV     R1,R0                       ; OFFSET,*              3578
000530  006300                            ASL     R0
000532  063700  000000G                    ADD     CST.ADDR,R0
000536  032710  040000                    BIT     #40000,(R0)
000542  001417                            BEQ     16$
000544  032710  020000                    BIT     #20000,(R0)                 ;                       3579
000550  001414                            BEQ     16$
000552  132710  000020                    BITB    #20,(R0)                    ;                       3580
000556  001411                            BEQ     16$
000560  032710  010000                    BIT     #10000,(R0)                 ;                       3581
000564  001006                            BNE     16$
000566  005000                            CLR     R0                          ;                       3584
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             3585
000602  062701  000012            16$:    ADD     #12,R1                      ; *,OFFSET              3576
000606  020127  000041                    CMP     R1,#41                      ; OFFSET,*
000612  003745                            BLE     15$
000614  105703                            TSTB    R3                          ; MOD.COUNT             3592
000616  001445                            BEQ     19$
000620  105002                            CLRB    R2                          ; TBL.COUNT             3595
000622  005000            17$:            CLR     R0                          ;                       3599
000624  150200                            BISB    R2,R0                       ; TBL.COUNT,*
000626  006300                            ASL     R0
000630  016046  000000G                    MOV     RANDOM(R0),-(SP)
000634  042716  100000                    BIC     #100000,(SP)
000640  005046                            CLR     -(SP)
000642  110316                            MOVB    R3,(SP)                     ; MOD.COUNT,*
000644  004737  000000G                    JSR     PC,BL$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             3600
000664  022626                            CMP     (SP)+,(SP)+                 ;                       3598
000666  013700  000000G                    MOV     CUOFF,R0                    ;                       3602
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)                 ;                       3603
```

```
000712  001403                          BEQ     18$
000714  032710  010000                  BIT     #10000,(R0)             ;                       3604
000720  001510                          BEQ     24$
000722  120227  000020          18$:    CMPB    R2,#20                  ; TBL.COUNT,*           3605
000726  001335                          BNE     17$
000730  000504                          BR      24$                     ;                       3607
000732  105003                  19$:    CLRB    R3                      ; MOD.COUNT             3620
000734  012701  000003                  MOV     #3,R1                   ; *,OFFSET              3622
000740  010100                  20$:    MOV     R1,R0                   ; OFFSET,*              3624
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)             ;                       3625
000762  001414                          BEQ     21$
000764  132710  000020                  BITB    #20,(R0)                ;                       3626
000770  001011                          BNE     21$
000772  032710  010000                  BIT     #10000,(R0)             ;                       3627
000776  001006                          BNE     21$
001000  005000                          CLR     R0                      ;                       3630
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             3631
001014  062701  000012          21$:    ADD     #12,R1                  ; *,OFFSET              3622
001020  020127  000041                  CMP     R1,#41                  ; OFFSET,*
001024  003745                          BLE     20$
001026  105703                          TSTB    R3                      ; MOD.COUNT             3638
001030  001445                          BEQ     25$
001032  105002                          CLRB    R2                      ; TBL.COUNT             3641
001034  005000                  22$:    CLR     R0                      ;                       3645
001036  150200                          BISB    R2,R0                   ; TBL.COUNT,*
001040  006300                          ASL     R0
001042  016046  000000G                 MOV     RANDOM(R0),-(SP)
001046  042716  100000                  BIC     #100000,(SP)
001052  005046                          CLR     -(SP)
001054  110316                          MOVB    R3,(SP)                 ; MOD.COUNT,*
001056  004737  000000G                 JSR     PC,BL$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             3646
001076  022626                          CMP     (SP)+,(SP)+             ;                       3644
001100  013700  000000G                 MOV     CUOFF,R0                ;                       3648
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)             ;                       3649
001124  001403                          BEQ     23$
001126  032710  010000                  BIT     #10000,(R0)             ;                       3650
001132  001505                          BEQ     30$
001134  120227  000020          23$:    CMPB    R2,#20                  ; TBL.COUNT,*           3651
```

```
001140  001335                          BNE     22$
001142  000501                  24$:    BR      30$                             ;                               3653
001144  105003                  25$:    CLRB    R3                              ; MOD.COUNT                     3664
001146  012701  000003                  MOV     #3,R1                           ; *,OFFSET                      3666
001152  010100                  26$:    MOV     R1,R0                           ; OFFSET,*                      3668
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)                     ;                               3669
001174  001411                          BEQ     27$
001176  032710  010000                  BIT     #10000,(R0)                     ;                               3670
001202  001006                          BNE     27$
001204  005000                          CLR     R0                              ;                               3673
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                     3674
001220  062701  000012          27$:    ADD     #12,R1                          ; *,OFFSET                      3666
001224  020127  000041                  CMP     R1,#41                          ; OFFSET,*
001230  003750                          BLE     26$
001232  105703                          TSTB    R3                              ; MOD.COUNT                     3680
001234  001457                          BEQ     31$
001236  105002                          CLRB    R2                              ; TBL.COUNT                     3683
001240  005000                  28$:    CLR     R0                              ;                               3687
001242  150200                          BISB    R2,R0                           ; TBL.COUNT,*
001244  006300                          ASL     R0
001246  016046  000000G                 MOV     RANDOM(R0),-(SP)
001252  042716  100000                  BIC     #100000,(SP)
001256  005046                          CLR     -(SP)
001260  110316                          MOVB    R3,(SP)                         ; MOD.COUNT,*
001262  004737  000000G                 JSR     PC,BL$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                     3688
001302  022626                          CMP     (SP)+,(SP)+                     ;                               3686
001304  013700  000000G                 MOV     CUOFF,R0                        ;                               3690
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)                     ;                               3691
001330  001403                          BEQ     29$
001332  032710  010000                  BIT     #10000,(R0)                     ;                               3692
001336  001403                          BEQ     30$
001340  120227  000020          29$:    CMPB    R2,#20                          ; TBL.COUNT,*                   3693
001344  001335                          BNE     28$
001346  013700  000114'          30$:    MOV     MAD1,R0                         ;                               3695
001352  013760  000000G 000016          MOV     CDISK,16(R0)
001360  013700  000116'                 MOV     MAD2,R0                         ;                               3696
001364  013760  000000G 000016          MOV     CDISK,16(R0)
001372  000207                          RTS     PC                              ;                               3682
```

```
001374  112737  000001  000000G          31$:    MOVB    #1,EOP.FLAG                     ;                3705
001402  000207                                   RTS     PC                              ;                3432

; Routine Size:  386 words,     Routine Base:  $CODE$ + 11406
; Maximum stack depth per invocation:  8 words
```

```
;  3708  1     GLOBAL routine QIO_FUNC : novalue =
;  3709  1
;  3710  1     !+
;  3711  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
;  3712  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
;  3713  1     !       BY THE FOLLWING ALGORITHM:
;  3714  1     !
;  3715  1     !               IF THE CHOSEN UNIT IS PROTECTED
;  3716  1     !               THEN
;  3717  1     !                       FUNCTION = READ
;  3718  1     !               ELSE (UNPROTECTED)
;  3719  1     !                       FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
;  3720  1     !                       NUMBER
;  3721  1     !
;  3722  1     !       IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE
;  3723  1     !       COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
;  3724  1     !       THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
;  3725  1     !       PACKET. OTHERWISE, THE SECOND MSCP PACKET IS RETURNED TO THE POOL.
;  3726  1     !
;  3727  1     !
;  3728  1     !       PERIODIACLLY, THIS ROUTINE WILL CALL THE DUP ROUTINE BEFORE IT   ZZZ
;  3729  1     !       BEGINS ITS OWN TASK. IF THE OPERATOR HAS SELECTED, "ALSO RUN     ZZZ
;  3730  1     !       DUP EXERCISER," THEN DUP TESTING OF DBNS WILL BE INTERLEAVED     ZZZ
;  3731  1     !       WITH THE REGULAR MSCP TESTING OF THE LBNS.                       ZZZ
;  3732  1     !
;  3733  1     !       TO AVOID LONG, CUMULATIVE INIT TIMES, THE DUP CODE IS ONLY       ZZZ
;  3734  1     !       EXECUTED AFTER (25 TIMES 'DUPROUND') MSCP I/O'S HAVE BEEN DONE.  ZZZ
;  3735  1     !       THE NUMBER OF DUP I/O'S IS 'DUPROUND'. THIS GIVES US A 25 TO 1   ZZZ
;  3736  1     !       INTERLEAVE.                                                      ZZZ
;  3737  1     !
;  3738  1     !       THE DUP TESTING IS DONE BY EXECUTING CONTROLLER LOCAL PROGRAMS   ZZZ
;  3739  1     !       TO READ OR WRITE/READ DBNS. AFTER THE DUP TESTING, THE CON-      ZZZ
;  3740  1     !       TROLLER IS REINITIALIZED, AND QIO_FUNC ROUTINE CONTINUES FROM    ZZZ
;  3741  1     !       WHERE IT LEFT OFF.                                               ZZZ
;  3742  1     !
;  3743  1     !
;  3744  1     !       IMPLICIT INPUTS:
;  3745  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  3746  1     !               CUOFF - CURRENT UNIT CST OFFSET
;  3747  1     !
;  3748  1     !       IMPLICIT OUTPUTS:
;  3749  1     !               THE OPCODE FIELD OF ONE OR BOTH MSCP PACKETS IS LOADED.
;  3750  1     !-
;  3751  1
;  3752  2         begin
;  3753  2
;  3754  2         local
;  3755  2             FUNC : word;                                          ! OPCODE (READ OR WRITE)
;  3756  2
;  3757  2     DUOFF = .CUOFF;                                  !SAVE IN CASE OTHER CMDS ZZZ
;  3758  2                                                      !LEFT IN QUEUE           ZZZ
;  3759  3     IF ((.CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] LEQ 0) AND !MSCP CNT=0       ZZZ
;  3760  3         (.CST_ADDR [.DUOFF, D_TYPE] NEQ RX_50) AND         !FIXED DISK       ZZZ
```

```
;  3761  3          (.CST_ADDR [.DUOFF + OF_DUPFLAGS, NODUPMEDIA] NEQ 1)) !MEDIA IN      ZZZ
;  3762  3                                                                               !ZZZ
;  3763  2      THEN                                                                     !ZZZ
;  3764  3          BEGIN                                                                !ZZZ
;  3765  3          PUT_PKT (.MX2);                          !RETURN 2ND ENVLOPE         !ZZZ
;  3766  3          MX2 = -1;                                !INDICATE FAILURE           ZZZ
;  3767  3          DUP ();                                  !DO DUP TEST                ZZZ
;  3768  3          CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] =  !REINIT MSCP FUN            ZZZ
;  3769  3                  (25 * .DUPROUND);                !CTION COUNTER              ZZZ
;  3770  3                                                                               !ZZZ
;  3771  3      !       THE FOLLOWING REINITS 2 ENVELOPES, SO THAT THE MSCP EXERCISER    ZZZ
;  3772  3      !       CAN PROCEED AS BEFORE THE DUP EXERCISER WAS CALLED.              ZZZ
;  3773  3      !                                                                        ZZZ
;  3774  3          DUP_FLAGS = .DUP_FLAGS OR SWP_DINT;      !SET DUP INIT FLAG          ZZZ
;  3775  3          INIT_TEST ();                            !REINIT CONTROLLER          ZZZ
;  3776  3          DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT); !CLR DUP INIT DLAG        ZZZ
;  3777  3                                                                               !ZZZ
;  3778  3          MX2 = -1;                                !ASSUME NO 2ND ENVELOPE     ZZZ
;  3779  3          IF (MX1 = GET_PKT (.CCTLR)) LSS 0        !TRY FOR 1ST ENVELOPE       ZZZ
;  3780  4              OR (.EOP_FLAG)                       !IF FAILURE                 ZZZ
;  3781  3          THEN     RETURN;                         !NO POINT TO GO ON          ZZZ
;  3782  3          IF (MX2 = GET_PKT (.CCTLR)) LSS 0        !TRY FOR 2ND ENVELOPE       ZZZ
;  3783  4              OR (.EOP_FLAG)                       !IF FAILURE                 ZZZ
;  3784  4          THEN     BEGIN                                                       !ZZZ
;  3785  4                   PUT_PKT (.MX1);                 !PUT 1ST BACK IN POOL       ZZZ
;  3786  4                   MX1 = -1;                       !INDICATE FAILURE           ZZZ
;  3787  4                   RETURN;                         !DONE                       ZZZ
;  3788  3                   END;                                                        !ZZZ
;  3789  3                                                                               !ZZZ
;  3790  3                   MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2); !CALC START ADDR    ZZZ
;  3791  3                   MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2); !OF BOTH ENVELOPES  ZZZ
;  3792  3                   GET_RANDOM ();                  !GET SET OF RANDOM NOS      ZZZ
;  3793  3                   QIO_UNIT ();                    !PUT RAND UNIT NO IN        ZZZ
;  3794  2                   END;                            !ENVELOPES                  ZZZ
;  3795  2                                                                               !ZZZ
;  3796  2      !       MSCP CODE STARTS HERE                                            ZZZ
;  3797  2      !                                                                        ZZZ
;  3798  2
;  3799  2      CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] =      !                           ZZZ
;  3800  2          .CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] -1;  !DECR MSCP FUNCTION CNTR ZZZ
;  3801  2
;  3802  2          MAD2 [OPCODE] = 0;                                                   ! ASSUME 2ND PACKET NOT NEEDED
;  3803  2
;  3804  2          if (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED) and        ! IF "FORCED ERROR" SET IN LAST READ,
;  3805  2             (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and              !       REWRITE SAME BLOCK
;  3806  3             (.FORCED_ERROR)                                                   !
;  3807  2          then
;  3808  2              FUNC = OP_WRT
;  3809  2          else
;  3810  2
;  3811  2              if .CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql PROTECTED            ! IF UNIT IS PROTECTED
;  3812  2              then
;  3813  2                  FUNC = OP_RD                                                 ! SET FUNCTION TO READ
```

```
;    3814  2              else
;    3815  2
;    3816  3                  if (.RANDOM [1] and 1)                          ! USE 2ND RANDOM NUMBER TO SELECT
;    3817  2                  then
;    3818  2                      FUNC = OP_RD                                !     READ
;    3819  2                  else
;    3820  2                      FUNC = OP_WRT;                              !     WRITE
;    3821  2
;    3822  2
;    3823  2
;    3824  3          IF (.CST_ADDR [.CUOFF + OF_DATA, D PROT] eql UNPROTECTED)   !I/O'S ARE CANCELLED WHEN CMD ZZZ
;    3825  2          THEN                                                        !RING IS FULL. DON'T LET THIS ZZZ
;    3826  2              IF .RW_BALANCE GEQU TOO_MANY_READS                      !UPSET THE BALANCE BETWEEN    ZZZ
;    3827  2              THEN                                                    !THE NUMBER OF READS AND      ZZZ
;    3828  2                  FUNC = OP_WRT;                                      !WRITES.                      ZZZ
;    3829  2
;    3830  2
;    3831  2          if (MAD1 [OPCODE] = .FUNC) eql OP_WRT                       ! LOAD CHOSEN OPCODE. IF WRITE
;    3832  2          then
;    3833  3              begin
;    3834  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                          ! NON-SEQUENTIAL COMMAND
;    3835  3
;    3836  4              if BIT_TST (SWP_FLAGS, SWF_CWC)                         ! IF CONTROLLER DOES WRITE-COMPARES
;    3837  3              then
;    3838  3                  MAD1 [MODIFY] = MD_CMP                              ! ADD COMPARE MODIFIER
;    3839  3              else
;    3840  3
;    3841  4                  if BIT_TST (SWP_FLAGS, SWF_HWC)                     ! IF HOST DOES WRITE-COMPARES
;    3842  3                  then
;    3843  4                      begin
;    3844  4                      MAD1 [MODIFY] = MD_EXP;                         ! SET WRITE AS AN EXPRESS REQUEST
;    3845  4                      MAD2 [OPCODE] = OP_RD;                          ! SET READ OPCODE INTO 2ND MSCP PACKET
;    3846  4                      MAD2 [MODIFY] = MD_EXP;                         ! SET READ AS AN EXPRESS REQUEST TOO
;    3847  4                      MAD2 [CMD_TYPE] = NON_SEQ_CMD;                  ! NON-SEQUENTIAL COMMAND
;    3848  3                      end;
;    3849  3              end
;    3850  2          else
;    3851  3              begin
;    3852  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                          ! NON-SEQUENTIAL COMMAND
;    3853  3
;    3854  4              if BIT_TST (SWP_FLAGS, SWF_CRC)                         ! IF READ-COMPARES   FUNCTION IS READ
;    3855  3              then
;    3856  3                  MAD1 [MODIFY] = MD_CMP;                             ! ADD COMPARE MODIFIER
;    3857  3
;    3858  2              end;
;    3859  2
;    3860  2          if .MAD2 [OPCODE] eql 0                                     ! IF NO OPCODE IN 2ND PACKET
;    3861  2          then
;    3862  3              begin
;    3863  3              PUT_PKT (.MX2);                                         ! RETURN 2ND PACKET TO POOL
;    3864  3              MX2 = -1;                                               ! MARK IT UNUSED
;    3865  2              end;
;    3866  2
```

```
;  3867  1        end;                                          ! ROUTINE QIO FUNC

                                        .SBTTL  QIO.FUNC MULTI-DRIVE TEST ROUTINES
000000  004137  000000G           QIO.FUNC::
                                        JSR     R1,$SAVE4                       ;                       3708
000004  013737  000000G 001156          MOV     CUOFF,DUOFF                     ;                       3757
000012  013702  000000G                 MOV     CST.ADDR,R2                     ;                       3759
000016  013701  001156                  MOV     DUOFF,R1
000022  010100                          MOV     R1,R0
000024  006300                          ASL     R0
000026  060200                          ADD     R2,R0
000030  005760  000022                  TST     22(R0)
000034  003146                          BGT     4$
000036  010100                          MOV     R1,R0                           ;                       3760
000040  006300                          ASL     R0
000042  060200                          ADD     R2,R0
000044  132710  000020                  BITB    #20,(R0)
000050  001540                          BEQ     4$
000052  010100                          MOV     R1,R0                           ;                       3761
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)                       ;                       3765
000072  004737  000000G                 JSR     PC,PUT.PKT
000076  012737  177777  000112'        MOV     #-1,MX2                         ;                       3766
000104  004737  000000V                 JSR     PC,DUP                          ;                       3767
000110  013701  001156'                 MOV     DUOFF,R1                        ;                       3768
000114  006301                          ASL     R1
000116  063701  000000G                 ADD     CST.ADDR,R1
000122  013716  000000G                 MOV     DUPROUND,(SP)                   ;                       3769
000126  012746  000031                  MOV     #31,-(SP)
000132  004737  000000G                 JSR     PC,BL$MUL
000136  010061  000022                  MOV     R0,22(R1)
000142  052737  000002  000000G         BIS     #2,DUP.FLAGS                    ;                       3774
000150  004737  001134'                 JSR     PC,INIT.TEST                    ;                       3775
000154  042737  000002  000000G         BIC     #2,DUP.FLAGS                    ;                       3776
000162  012737  177777  000112'        MOV     #-1,MX2                         ;                       3778
000170  013716  000000G                 MOV     CCTLR,(SP)                      ;                       3779
000174  004737  000000G                 JSR     PC,GET.PKT
000200  010037  000110'                 MOV     R0,MX1
000204  002426                          BLT     2$
000206  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                       3780
000214  001022                          BNE     2$                              ;                       3708
000216  013716  000000G                 MOV     CCTLR,(SP)                      ;                       3782
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                     ;                       3783
000242  001411                          BEQ     3$
000244  013716  000110'           1$:   MOV     MX1,(SP)                        ;                       3785
000250  004737  000000G                 JSR     PC,PUT.PKT
```

```
000254  012737  177777  000110'                MOV     #-1,MX1                 ;                       3786
000262  022626                         2$:     CMP     (SP)+,(SP)+             ;                       3787
000264  000207                                 RTS     PC                      ;                       3784
000266  013716  000110'                3$:     MOV     MX1,(SP)                ;                       3790
000272  012746  000106                         MOV     #106.,-(SP)
000276  004737  000000G                        JSR     PC,BL$MUL
000302  062700  000000G                        ADD     #MSCP.PKT,R0
000306  010037  000114'                        MOV     R0,MAD1
000312  013716  000112'                        MOV     MX2,(SP)                ;                       3791
000316  012746  000106                         MOV     #106.,-(SP)
000322  004737  000000G                        JSR     PC,BL$MUL
000326  062700  000000G                        ADD     #MSCP.PKT,R0
000332  010037  000116'                        MOV     R0,MAD2
000336  004737  011276'                        JSR     PC,GET.RANDOM           ;                       3792
000342  004737  011406'                        JSR     PC,QIO.UNIT             ;                       3793
000346  062706  000010                         ADD     #10,SP                  ;                       3764
000352  013700  000000G                4$:     MOV     CUOFF,R0                ;                       3799
000356  006300                                 ASL     R0
000360  063700  000000G                        ADD     CST.ADDR,R0
000364  005360  000022                         DEC     22(R0)                  ;                       3800
000370  013701  000116'                        MOV     MAD2,R1                 ;                       3802
000374  012704  000022                         MOV     #22,R4
000400  060104                                 ADD     R1,R4
000402  105014                                 CLRB    (R4)
000404  013700  000000G                        MOV     CUOFF,R0                ;                       3804
000410  006300                                 ASL     R0
000412  063700  000000G                        ADD     CST.ADDR,R0
000416  005003                                 CLR     R3
000420  005710                                 TST     (R0)
000422  100010                                 BPL     5$
000424  005203                                 INC     R3
000426  132710  000020                         BITB    #20,(R0)                ;                       3805
000432  001404                                 BEQ     5$
000434  132737  000001  000000G                BITB    #1,FORCED.ERROR         ;                       3806
000442  001012                                 BNE     7$                      ;                       3808
000444  032710  100000                5$:     BIT     #100000,(R0)            ;                       3811
000450  001404                                 BEQ     6$                      ;                       3813
000452  032737  000001  000002G                BIT     #1,RANDOM+2             ;                       3816
000460  001403                                 BEQ     7$
000462  012702  000041                6$:     MOV     #41,R2                  ; *,FUNC                3818
000466  000402                                 BR      8$                      ;                       3816
000470  012702  000042                7$:     MOV     #42,R2                  ; *,FUNC                3820
000474  006003                        8$:     ROR     R3                      ;                       3824
000476  103006                                 BCC     9$
000500  023727  000106'  000002                CMP     RW.BALANCE,#2           ;                       3826
000506  103402                                 BLO     9$
000510  012702  000042                         MOV     #42,R2                  ; *,FUNC                3828
000514  013700  000114'                9$:     MOV     MAD1,R0                 ;                       3831
000520  013703  000000G                        MOV     SWP.FLAGS,R3            ;                       3836
000524  110260  000022                         MOVB    R2,22(R0)               ; FUNC,*                3831
000530  020227  000042                         CMP     R2,#42                  ; FUNC,*
000534  001025                                 BNE     10$
000536  112760  000002  000004                MOVB    #2,4(R0)                ;                       3834
```

H12


```
000544  032703  000020                          BIT     #20,R3                          ;                       3836
000550  001025                                  BNE     11$                             ;                       3838
000552  032703  000040                          BIT     #40,R3                          ;                       3841
000556  001425                                  BEQ     12$
000560  012760  100000  000024                  MOV     #-100000,24(R0)                 ;                       3844
000566  112714  000041                          MOVB    #41,(R4)                        ;                       3845
000572  012761  100000  000024                  MOV     #-100000,24(R1)                 ;                       3846
000600  112761  000002  000004                  MOVB    #2,4(R1)                        ;                       3847
000606  000411                                  BR      12$                             ;                       3831
000610  112760  000002  000004          10$:    MOVB    #2,4(R0)                        ;                       3852
000616  032703  000004                          BIT     #4,R3                           ;                       3854
000622  001403                                  BEQ     12$
000624  012760  040000  000024          11$:    MOV     #40000,24(R0)                   ;                       3856
000632  105714                          12$:    TSTB    (R4)                            ;                       3860
000634  001010                                  BNE     13$
000636  013746  000112                          MOV     MX2,-(SP)                       ;                       3863
000642  004737  000000G                         JSR     PC,PUT.PKT
000646  012737  177777  000112'                 MOV     #-1,MX2                         ;                       3864
000654  005726                                  TST     (SP)+                           ;                       3862
000656  000207                          13$:    RTS     PC                              ;                       3708

; Routine Size:  216 words,    Routine Base:  $CODE$ + 13012
; Maximum stack depth per invocation:  10 words
```

```
;  3868  1
;  3869  1     GLOBAL ROUTINE DUP : NOVALUE =                                          !ZZZ
;  3870  1     !*                                                                     !ZZZ
;  3871  1     !       THIS ROUTINE IS CALLED BY QIO_FUNC AFTER 25 * 'DUPROUND' RD/WTS.  ZZZ
;  3872  1     !       THIS EXERCISER WAS PLACED IN THE MIDDLE OF THE MSCP EXERCISER,    ZZZ
;  3873  1     !       SO COMMON INIT AND OTHER ROUTINES COULD BE USED.                  ZZZ
;  3874  1     !                                                                        ZZZ
;  3875  1     !       THE DUP EXERCISER WILL PERFORM EITHER READ-ONLY, OR WRITE-READ    ZZZ
;  3876  1     !       COMPARE OPERATIONS ON THE DIAGNOSTIC BLOCKS (DBNS). IT WILL       ZZZ
;  3877  1     !       RECORD THE STATISTICS IN THE TALLY TABLES.                        ZZZ
;  3878  1     !                                                                        ZZZ
;  3879  1     !       THE PROGRAM USES CONTROLLER LOCAL PROGRAMS TO WRITE AND READ     ZZZ
;  3880  1     !       DBNS. WHEN WRITING TO THE DBNS, A ONE WORD PATTERN WILL BE       ZZZ
;  3881  1     !       SELECTED, AND REPLICATED THROUGH A 256 WORD BLOCK FOR DATA.       ZZZ
;  3882  1     !       THE ROUTINE WILL WRITE 'DUPROUND' NUMBER OF SEQUENTIAL DBN       ZZZ
;  3883  1     !       BLOCKS. IF THE CONTROLLER LOCAL PROGRAMS EXIST, AND THE OPERATOR ZZZ
;  3884  1     !       SELECTS 'WRITE TO DIAGNOSTIC AREA', WRITE-READ-COMPARES WILL BE  ZZZ
;  3885  1     !       PERFORMED ON THE DBNS. OTHERWISE, READS WITH NO COMPARES WILL BE ZZZ
;  3886  1     !       DONE. BAD BLOCKS FOUND IN THE COMPARISON TESTS WILL NOT BE LIST-  ZZZ
;  3887  1     !       ED IN THE RCT TABLES.                                            ZZZ
;  3888  1     !                                                                        ZZZ
;  3889  1     !       AFTER 'DUPROUND' NUMBER OF DBNS HAVE BEEN TESTED, THE ENVELOPES  ZZZ
;  3890  1     !       WILL BE REINITIATED, SO THAT THE MSCP EXERCISER CAN CONTINUE      ZZZ
;  3891  1     !       AS BEFORE.                                                       ZZZ
;  3892  1     !                                                                        ZZZ
;  3893  1     !       IMPLICIT INPUTS:                                                 ZZZ
;  3894  1     !                       CCTLR - CURRENT CONTROLLER NUMBER                ZZZ
;  3895  1     !                       CST_ADDRS - CONTAINS THE CURRENT CONTROLLER       ZZZ
;  3896  1     !                                   STATUS TABLE                         ZZZ
;  3897  1     !                       CUOFF - CURRENT OFFSET IN CST TABLE FOR          ZZZ
;  3898  1     !                               PARTICULAR DRIVE                         ZZZ
;  3899  1     !                                                                        ZZZ
;  3900  1     !       IMPLICIT OUTPUTS:                                                ZZZ
;  3901  1     !                       S_PATTERN - PATTERN BEING WRITTEN TO DBNS        ZZZ
;  3902  1     !                                                                        ZZZ
;  3903  1     !                                                                       !ZZZ
;  3904  1     !                                                                       !ZZZ
;  3905  1
```

```
;    3906  1
;    3907  1
;    3908  2     BEGIN
;    3909  2     OWN
;    3910  2             TEMP : WORD;
;    3911  2
;    3912  2     !PRINTX (DBM110);
;    3913  2     !PRINTX (DER10);
;    3914  2
;    3915  2     until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;    3916  2           (.EOP_FLAG eql true) do                   ! Make sure all MSCP commands are completed
;    3917  3         begin
;    3918  3         BREAK;                                      ! BREAK FOR ACT
;    3919  3         PROC_RETPKT();                              ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;    3920  3         RP_INDX = .RP_INDX + 1;                     ! INCREMENT RP_INDX
;    3921  3         if .RP_INDX geq RP_CNT then (RP_INDX = 0;); ! MAKE SURE THE COUNTER DOES NOT GET TO BIG
;    3922  3         RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;    3923  2         end;
;    3924  2
;    3925  2
;    3926  2     S_PATTERN = .RANDOM [1];                                    !OTHER UNIT VARIABLES
;    3927  2
;    3928  2     IF (.CST_ADDR [.DUOFF + OF_DBN, D_DBN] + .dupround) GEQ 144     ! TEST TO SEE IF NEXT DBN'S TO LARGE
;    3929  2     THEN (CST_ADDR [.DUOFF + OF_DBN, D_DBN] = 0);               ! CIRCLE ARROUND IF DBN TO LARGE
;    3930  2
;    3931  2     DUPIDLE ();                                                 ! DO A GET DUST STATUS TO FIND IF LOCAL DUP MEDIA
;    3932  2     IF .CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] EQL 1 THEN RETURN; ! IF DUP LOCAL MEDIA NOT THERE THEN RETURN
;    3933  2
;    3934  2     TEMP = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];
;    3935  2     INCR DBNCNT FROM (.TEMP + 1) TO (.TEMP + .duproound) DO                 ! INCREMENT FROM RELATIVE DBN TO DBN + dupro
und
;    3936  3             BEGIN
;    3937  3             IF .CST_ADDR [.DUOFF + OF_DBN, DUPWRITE]                            ! IF WRITE FLAG SET IN CST TABLE THE
N WRITE DBN'S
;    3938  3             THEN
;    3939  4                     BEGIN
;    3940  4                     DUPIDLE ();                                     ! MAKE SURE THE CONTROLLER IS IN AN IDLE STA
TE
;    3941  4                     DUPWRTDBN ();                                   ! CALL ROUTINE TO HANDLE WRITTING ROUTINES
;    3942  3                     END;
;    3943  3
;    3944  3             DUPIDLE ();                                             ! MAKE SURE CONTROLLER IN IDLE STATE
;    3945  3             DUPREDDBN ();                                           ! CALL ROUTINE TO HANDLE READING DBN'S
;    3946  3
;    3947  3             CST_ADDR [.DUOFF + OF_DBN, D_DBN] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN] + 1; ! INCREMENT RELATIVE DBN COUNTER
;    3948  3
;    3949  3             IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1          ! ERROR IN DUP REINITIALIZE
;    3950  3             THEN RETURN;                                            ! AND RETURN
;    3951  2             END;
;    3952  1     END;


001204                                  .PSECT  $GGG$,  RO
001204                          TEMP:   .BLKW   1
```

```
                                        .SBTTL  DUP MULTI-DRIVE TEST ROUTINES
013672                                  .PSECT  $CODE$,  RO

000000  004137  000000G           DUP::  JSR     R1,$SAVE3                       ;                       3869
000004  013700  000000G           1$:    MOV     RP.ADDR,RO                      ;                       3915
000010  023760  000000G 000004           CMP     CRN.LOW,4(RO)
000016  001433                           BEQ     3$
000020  123727  000000G 000001           CMPB    EOP.FLAG,#1                     ;                       3916
000026  001427                           BEQ     3$
000030  104422                           TRAP    22                              ;                       3917
000032  004737  000000V                  JSR     PC,PROC.RETPKT                  ;                       3919
000036  005237  000000G                  INC     RP.INDX                         ;                       3920
000042  023727  000000G 000010           CMP     RP.INDX,#10                     ;                       3921
000050  002402                           BLT     2$
000052  005037  000000G                  CLR     RP.INDX
000056  013746  000000G           2$:    MOV     RP.INDX,-(SP)                   ;                       3922
000062  012746  000054                   MOV     #54,-(SP)
000066  004737  000000G                  JSR     PC,BL$MUL
000072  062700  000000G                  ADD     #RETPKT,RO
000076  010037  000000G                  MOV     RO,RP.ADDR
000102  022626                           CMP     (SP)+,(SP)+                     ;                       3917
000104  000737                           BR      1$                              ;                       3915
000106  013737  000002G 000000G  3$:     MOV     RANDOM+2,S.PATTERN              ;                       3926
000114  013700  001156'                  MOV     DUOFF,RO                        ;                       3928
000120  006300                           ASL     RO
000122  063700  000000G                  ADD     CST.ADDR,RO
000126  005001                           CLR     R1
000130  156001  000020                   BISB    20(RO),R1
000134  063701  000000G                  ADD     DUPROUND,R1
000140  020127  000220                   CMP     R1,#220
000144  002402                           BLT     4$
000146  105060  000020                   CLRB    20(RO)                          ;                       3929
000152  004737  000000V           4$:    JSR     PC,DUPIDLE                      ;                       3931
000156  013700  001156'                  MOV     DUOFF,RO                        ;                       3932
000162  006300                           ASL     RO
000164  063700  000000G                  ADD     CST.ADDR,RO
000170  005760  000020                   TST     20(RO)
000174  100456                           BMI     8$
000176  116037  000020  001204'         MOVB    20(RO),TEMP                     ;                       3934
000204  105037  001205'                  CLRB    TEMP+1
000210  013703  001204'                  MOV     TEMP,R3                         ;                       3935
000214  063703  000000G                  ADD     DUPROUND,R3
000220  013700  001156'                  MOV     DUOFF,RO                        ;                       3937
000224  006300                           ASL     RO
000226  063700  000000G                  ADD     CST.ADDR,RO
000232  010001                           MOV     RO,R1
000234  062701  000020                   ADD     #20,R1
000240  013702  001204'                  MOV     TEMP,R2                         ; *,DBNCNT              3935
000244  000427                           BR      7$
000246  032711  010000           5$:     BIT     #10000,(R1)                     ;                       3937
000252  001404                           BEQ     6$
000254  004737  000000V                  JSR     PC,DUPIDLE                      ;                       3940
```

```
000260  004737  000000V                        JSR     PC,DUPWRTDBN                    ;                       3941
000264  004737  000000V                6$:     JSR     PC,DUPIDLE                      ;                       3944
000270  004737  000000V                        JSR     PC,DUPREDDBN                    ;                       3945
000274  013700  001156'                        MOV     DUOFF,R0                        ;                       3947
000300  006300                                ASL     R0
000302  063700  000000G                        ADD     CST.ADDR,R0
000306  010001                                MOV     R0,R1
000310  062701  000020                        ADD     #20,R1
000314  105211                                INCB    (R1)
000316  032711  040000                        BIT     #40000,(R1)                     ;                       3949
000322  001003                                BNE     8$                              ;                       3950
000324  005202                7$:     INC     R2                              ; DBNCNT                        3935
000326  020203                                CMP     R2,R3                           ; DBNCNT,*
000330  003746                                BLE     5$
000332  000207                8$:     RTS     PC                              ;                       3869

; Routine Size:  110 words,    Routine Base:  $CODE$ + 13672
; Maximum stack depth per invocation:  7 words


;   3953  1
```

```
;    3954  1      GLOBAL ROUTINE DUPWRTDBN : NOVALUE =
;    3955  1
;    3956  1      !+
;    3957  1      ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;    3958  1      ! "WRTDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;    3959  1      ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM WRITES TO A DIAGONOSTIC BLOCK (DBN)
;    3960  1      ! THE WORD IN "S_PATTERN" IS WRITTEN TO THE 256 WORDS IN THE DBN.  IF AN ERROR OCCURS
;    3961  1      ! WHILE RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;    3962  1      ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;    3963  1      !
;    3964  1      !       IMPLICIT INPUTS:
;    3965  1      !               CST_ADDRS  - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;    3966  1      !               DUOFF      - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;    3967  1      !               S PATTERN    CONTAINS PATTERN WORD!
;    3968  2      BEGIN
;    3969  2      LOCAL                                                  !                                                ZZZ
;    3970  2            TRYNUM : WORD,                                   !                                                ZZZ
;    3971  2            MAX_TRY COUNT : word initial (9);                ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR ZZZ
;    3972  2                                                             !                                                ZZZ
;    3973  2      LABEL                                                  !                                                ZZZ
;    3974  2           DUP_WLOOP;                                        !START OF DUP WRITE RETRY LOOP                   ZZZ
;    3975  2
;    3976  2      !PRINTX (DER11);
;    3977  2      T_ADDR [T_DBN_WT] = .T_ADDR [T_DBN_WT] + 1;            ! INCREMENT # OF WRITES GIVEN
;    3978  2
;    3979  2      TRYNUM = 0;                                            !ZERO TRY COUNTER                                ZZZ
;    3980  2      DUP_WLOOP:                                             !LABEL FOR LOOP EXCAPE ON GOOD WRITE             ZZZ
;    3981  3      BEGIN                                                  !BEGIN DUP_WLOOP                                 ZZZ
;    3982  3      INCR TRIES FROM 1 TO 10 DO                             !START TRYING DUP WRITES                         ZZZ
;    3983  4      BEGIN                                                  !BEGIN LARGE DO LOOP                             ZZZ
;    3984  4                                                             !                                                ZZZ
;    3985  4
;    3986  4      MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                      ! PACKET SIZE                                EXECUTE LOCAL PROGRAM WRT DB
N
;    3987  4      MSCP_PKT [.MX1, OPCODE] = OP_ELP;                      ! OPCODE = EXECUTE LOCAL PROGRAM
;    3988  4      MSCP_PKT [.MX1, L1] = %ascii'W';                       ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;    3989  4      MSCP_PKT [.MX1, L2] = %ascii'R';
;    3990  4      MSCP_PKT [.MX1, L3] = %ascii'T';
;    3991  4      MSCP_PKT [.MX1, L4] = %ascii'D';
;    3992  4      MSCP_PKT [.MX1, L5] = %ascii'B';
;    3993  4      MSCP_PKT [.MX1, L6] = %ascii'N';
;    3994  4      MSCP_PKT [.MX1, MODIFY] = 1;                           ! STANDALONE MODIFIER
;    3995  4      MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                     ! CALL IT IMMEDIATE
;    3996  4      DUPCOMMAND ();                                         ! SENDS AND RECEIVES THE COMMAND
;    3997  4
;    3998  4      IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1   !status error
;    3999  4      THEN RETURN;                                           ! AND RETURN
;    4000  4
;    4001  5      DO (MX1 = GET_PKT (.CCTLR))
;    4002  4      UNTIL (.MX1 GE 0);                        ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4003  4
;    4004  4              MSCP_PKT [.MX1, MSGLEN] = SZ_REC;              ! PACKET SIZE                                RECIEVE DATA
;    4005  4              MSCP_PKT [.MX1, OPCODE] = OP_RCD;              ! OPCODE = RECEIVE DATA
;    4006  4              MSCP_PKT [.MX1, BC_LO] = 80;                   ! BYTE COUNT TO BE TRANSFERED EQUALS 2 ***see pg 26 of DUP s
pec
```

```
;  4007  4          MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;  4008  4          MSCP_PKT [.MX1, MODIFY] = 0;                    !
;  4009  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;  4010  4          DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;  4011  4
;  4012  4          IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;  4013  4             (.DUPPKT [DUPTYPE] NEQU 1) OR                !dup type error
;  4014  5             (.DUPPKT [DUPMESG] NEQU 6)
;  4015  4          THEN
;  4016  5                  (D_FAIL = 1;                            !TELL HARD_ERROR IT WAS A DUP PROBLEM          ZZZ
;  4017  5                   HARD_ERROP ();                         !                                              ZZZ
;  4018  5                   D_FAIL = 0;                            !                                              ZZZ
;  4019  5                   CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;  4020  4                   RETURN;);                              ! NO POINT IN CONTINUING
;  4021  4
;  4022  5          DO (MX1 = GET_PKT (.CCTLR))
;  4023  4          UNTIL (.MX1 GEQ 0);                             ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4024  4
;  4025  4          MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;               ! PACKET SIZE                                  SEND DATA
;  4026  4          MSCP_PKT [.MX1, OPCODE] = OP_SDD;               ! OPCODE = SEND DATA
;  4027  4          MSCP_PKT [.MX1, BC_LO] = 6;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 6
;  4028  4          MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;  4029  4          DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; !LOAD UNIT NUMBER (RDRX)
;  4030  4          DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;  4031  4          DUPPKT [DUPBF2] = .S_PATTERN;                   ! LOAD PATTERN
;  4032  4          MSCP_PKT [.MX1, MODIFY] = 0;                    !
;  4033  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;  4034  4          DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;  4035  4
;  4036  4          IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  ! status error
;  4037  4          THEN RETURN;
;  4038  4
;  4039  5          DO (MX1 = GET_PKT (.CCTLR))
;  4040  4          UNTIL (.MX1 GEQ 0);                        ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4041  4
;  4042  4          MSCP_PKT [.MX1, MSGLEN] = SZ_REC;               ! PACKET SIZE                                  RECEIVE DATA
;  4043  4          MSCP_PKT [.MX1, OPCODE] = OP_RCD;               ! OPCODE = RECEIVE DATA
;  4044  4          MSCP_PKT [.MX1, BC_LO] = 4;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;  4045  4          MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;  4046  4          MSCP_PKT [.MX1, MODIFY] = 0;                    !
;  4047  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;  4048  4          DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;  4049  4
;  4050  4
;  4051  4          IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND                      ZZZ
;  4052  4             (.DUPPKT [DUPTYPE] EQL 3) AND                !NO dup type error                             ZZZ
;  4053  4             (.DUPPKT [DUPMESG] EQL 3) AND                !                                              ZZZ
;  4054  5             (.DUPPKT [DUPBF1]  EQL 0)                    !AND A successful write code                   ZZZ
;  4055  5                                                          !                                              ZZZ
;  4056  4          THEN                                            !THEN                                          ZZZ
;  4057  4             LEAVE DUP_WLOOP                              !I/O OK. EXIT RETRY LOOP.                      ZZZ
;  4058  4          ELSE                                            !                                              ZZZ
;  4059  5             BEGIN                                        !                                              ZZZ
```

```
;    4060  5                TRYNUM = .TRYNUM + 1;                          !INCR ATTEMPT COUNT                          ZZZ
;    4061  5                IF .TRYNUM  EQL .MAX_TRY_COUNT                 !IF IT FAILED ALL RETRIES, THEN              ZZZ
;    4062  5                THEN                                           !REPORT THE ERROR.                           ZZZ
;    4063  6                        (D_FAIL = 1;                           !TELL HARD_ERROR IT WAS A DUP PROBLEM        ZZZ
;    4064  6                         HARD_ERROR ();                        !                                            ZZZ
;    4065  6                         D_FAIL = 0;                           !                                            ZZZ
;    4066  6                         CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;    ! SET FLAG                        ZZZ
;    4067  5                         RETURN;);                             ! NO POINT IN CONTINUING                     ZZZ
;    4068  4                END;                                           !                                            ZZZ
;    4069  3         END;                                                  !END LARGE DO LOOP                           ZZZ
;    4070  3                                                               !                                            ZZZ
;    4071  2         END;                                                  !END DUP WLOOP                               ZZZ
;    4072  2
;    4073  2
;    4074  3         DO (MX1 = GET_PKT (.CCTLR))
;    4075  2         UNTIL (.MX1 GEQ 0);                                   ! TRY TO GET AN ENVELOPE.
;    4076  2
;    4077  2         T_ADDR [T_BLK_WT] = .T_ADDR [T_BLK_WT] + 1;           !INCREMENT COUNTER IF A SUCCESS
;    4078  2
;    4079  1         END;


                                        .SBTTL  DUPWRTDBN MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          DUPWRTDBN::
                                        JSR     R1,$SAVE4                       ;                                    3954
000004  012704  000011                  MOV     #11,R4                          ; #,MAX.TRY.COUNT                    3968
000010  013700  000000G                 MOV     T.ADDR,R0                       ;                                    3977
000014  005260  000060                  INC     60(R0)
000020  005002                          CLR     R2                              ; TRYNUM                             3979
000022  012703  000012                  MOV     #12,R3                          ; #,TRIES                            3982
000026  013746  000110'         1$:     MOV     MX1,-(SP)                       ;                                    3986
000032  012746  000106                  MOV     #106,-(SP)
000036  004737  000000G                 JSR     PC,BL$MUL
000042  012760  000022  000006G        MOV     #22,MSCP.PKT+6(R0)
000050  112760  000003  000022G        MOVB    #3,MSCP.PKT+22(R0)              ;                                    3987
000056  112760  000127  000026G        MOVB    #127,MSCP.PKT+26(R0)            ;                                    3988
000064  112760  000122  000027G        MOVB    #122,MSCP.PKT+27(R0)            ;                                    3989
000072  112760  000124  000030G        MOVB    #124,MSCP.PKT+30(R0)            ;                                    3990
000100  112760  000104  000031G        MOVB    #104,MSCP.PKT+31(R0)            ;                                    3991
000106  112760  000102  000032G        MOVB    #102,MSCP.PKT+32(R0)            ;                                    3992
000114  112760  000116  000033G        MOVB    #116,MSCP.PKT+33(R0)            ;                                    3993
000122  012760  000001  000024G        MOV     #1,MSCP.PKT+24(R0)              ;                                    3994
000130  142760  000360  000010G        BICB    #360,MSCP.PKT+10(R0)            ;                                    3995
000136  004737  000000V                 JSR     PC,DUPCOMMAND                   ;                                    3996
000142  013700  001156'                 MOV     DUOFF,R0                        ;                                    3998
000146  006300                          ASL     R0
000150  063700  000000G                 ADD     CST.ADDR,R0
000154  032760  040000  000020         BIT     #40000,20(R0)
000162  001402                          BEQ     2$
000164  022626                          CMP     (SP)+,(SP)+                     ;                                    3954
000166  000207                          RTS     PC                              ;                                    3999
000170  013716  000000G         2$:     MOV     CCTLR,(SP)                      ;                                    4001
000174  004737  000000G                 JSR     PC,GET.PKT
```

```
000200  010037  000110                          MOV     R0,MX1
000204  002771                                  BLT     2$                              ;4002
000206  010016                                  MOV     R0,(SP)                         ;4004
000210  012746  000106                          MOV     #106,-(SP)
000214  004737  000000G                         JSR     PC,BL$MUL
000220  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000226  112760  000005  000022G                 MOVB    #5,MSCP.PKT+22(R0)              ;4005
000234  012760  000120  000026G                 MOV     #120,MSCP.PKT+26(R0)            ;4006
000242  012760  000000G 000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0)         ;4007
000250  005060  000024G                         CLR     MSCP.PKT+24(R0)                 ;4008
000254  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)            ;4009
000262  152760  000020  000010G                 BISB    #20,MSCP.PKT+10(R0)
000270  004737  000000V                         JSR     PC,DUPCOMMAND                   ;4010
000274  013700  001156'                         MOV     DUOFF,R0                        ;4012
000300  006300                                  ASL     R0
000302  063700  000000G                         ADD     CST.ADDR,R0
000306  032760  040000  000020                  BIT     #40000,20(R0)
000314  001016                                  BNE     3$
000316  013700  000000G                         MOV     DUPPKT,R0                       ;4013
000322  042700  007777                          BIC     #7777,R0
000326  020027  010000                          CMP     R0,#10000
000332  001007                                  BNE     3$
000334  013700  000000G                         MOV     DUPPKT,R0                       ;4014
000340  042700  170000                          BIC     #170000,R0
000344  020027  000006                          CMP     R0,#6
000350  001422                                  BEQ     4$
000352  112737  000001  000000G         3$:     MOVB    #1,D.FAIL                       ;4016
000360  004737  000000V                         JSR     PC,HARD.ERROR                   ;4017
000364  105037  000000G                         CLRB    D.FAIL                          ;4018
000370  013700  001156'                         MOV     DUOFF,R0                        ;4019
000374  006300                                  ASL     R0
000376  063700  000000G                         ADD     CST.ADDR,R0
000402  052760  040000  000020                  BIS     #40000,20(R0)
000410  062706  000006                          ADD     #6,SP                           ;4020
000414  000207                                  RTS     PC                              ;4016
000416  013716  000000G                 4$:     MOV     CCTLR,(SP)                      ;4022
000422  004737  000000G                         JSR     PC,GET.PKT
000426  010037  000110'                         MOV     R0,MX1
000432  002771                                  BLT     4$                              ;4023
000434  010016                                  MOV     R0,(SP)                         ;4025
000436  012746  000106                          MOV     #106,-(SP)
000442  004737  000000G                         JSR     PC,BL$MUL
000446  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000454  112760  000004  000022G                 MOVB    #4,MSCP.PKT+22(R0)              ;4026
000462  012760  000006  000026G                 MOV     #6,MSCP.PKT+26(R0)              ;4027
000470  012760  000000G 000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0)         ;4028
000476  013701  001156'                         MOV     DUOFF,R1                        ;4029
000502  006301                                  ASL     R1
000504  063701  000000G                         ADD     CST.ADDR,R1
000510  111137  000000G                         MOVB    (R1),DUPPKT
000514  042737  177760  000000G                 BIC     #177760,DUPPKT
000522  013701  001156'                         MOV     DUOFF,R1                        ;4030
000526  006301                                  ASL     R1
```

```
000530  063701  000000G                        ADD     CST.ADDR,R1
000534  116137  000020  000002G                MOVB    20(R1),DUPPKT+2
000542  105037  000003G                        CLRB    DUPPKT+3
000546  013737  000000G 000004G                MOV     S.PATTERN,DUPPKT+4      ;                       4031
000554  005060  000024G                        CLR     MSCP.PKT+24(R0)         ;                       4032
000560  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)    ;                       4033
000566  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000574  004737  000000V                        JSR     PC,DUPCOMMAND           ;                       4034
000600  013700  001156                         MOV     DUOFF,R0                ;                       4036
000604  006300                                 ASL     R0
000606  063700  000000G                        ADD     CST.ADDR,R0
000612  032760  040000  000020                 BIT     #40000,20(R0)
000620  001403                                 BEQ     5$
000622  062706  000010                         ADD     #10,SP                  ;                       3954
000626  000207                                 RTS     PC                      ;                       4037
000630  013716  000000G             5$:         MOV     CCTLR,(SP)              ;                       4039
000634  004737  000000G                        JSR     PC,GET.PKT
000640  010037  000110'                        MOV     R0,MX1
000644  002771                                 BLT     5$                      ;                       4040
000646  010016                                 MOV     R0,(SP)                 ;                       4042
000650  012746  000106                         MOV     #106,-(SP)
000654  004737  000000G                        JSR     PC,BL$MUL
000660  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000666  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)      ;                       4043
000674  012760  000004  000026G                MOV     #4,MSCP.PKT+26(R0)      ;                       4044
000702  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0) ;                       4045
000710  005060  000024G                        CLR     MSCP.PKT+24(R0)         ;                       4046
000714  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)    ;                       4047
000722  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000730  004737  000000V                        JSR     PC,DUPCOMMAND           ;                       4048
000734  013700  001156'                        MOV     DUOFF,R0                ;                       4051
000740  006300                                 ASL     R0
000742  063700  000000G                        ADD     CST.ADDR,R0
000746  032760  040000  000020                 BIT     #40000,20(R0)
000754  001024                                 BNE     6$
000756  013700  000000G                        MOV     DUPPKT,R0               ;                       4052
000762  042700  007777                         BIC     #7777,R0
000766  020027  030000                         CMP     R0,#30000
000772  001015                                 BNE     6$
000774  013700  000000G                        MOV     DUPPKT,R0               ;                       4053
001000  042700  170000                         BIC     #170000,R0
001004  020027  000003                         CMP     R0,#3
001010  001006                                 BNE     6$
001012  005737  000002G                        TST     DUPPKT+2                ;                       4054
001016  001003                                 BNE     6$
001020  062706  000012                         ADD     #12,SP                  ;                       4057
001024  000433                                 BR      8$
001026  005202                     6$:         INC     R2                      ; TRYNUM                4060
001030  020204                                 CMP     R2,R4                   ; TRYNUM,MAX.TRY.COUNT  4061
001032  001022                                 BNE     7$
001034  112737  000001  000000G                MOVB    #1,D.FAIL               ;                       4063
001042  004737  000000V                        JSR     PC,HARD.ERROR           ;                       4064
001046  105037  000000G                        CLRB    D.FAIL                  ;                       4065
```

```
001052  013700  001156                          MOV     DUOFF,R0                        ;                               4066
001056  006300                                  ASL     R0
001060  063700  000000G                         ADD     CST.ADDR,R0
001064  052760  040000  000020                  BIS     #40000,20(R0)
001072  062706  000012                          ADD     #12,SP                          ;                               4067
001076  000207                                  RTS     PC                              ;                               4063
001100  062706  000012                  7$:     ADD     #12,SP                          ;                               3983
001104  005303                                  DEC     R3                              ; TRIES                         3982
001106  001402                                  BEQ     8$
001110  000137  014254'                         JMP     1$
001114  013746  000000G                 8$:     MOV     CCTLR,-(SP)                     ;                               4074
001120  004737  000000G                         JSR     PC,GET.PKT
001124  010037  000110'                         MOV     R0,MX1
001130  005726                                  TST     (SP)+
001132  005700                                  TST     R0                              ; MX1                           4075
001134  002767                                  BLT     8$
001136  013700  000000G                         MOV     T.ADDR,R0                       ;                               4077
001142  005260  000056                          INC     56(R0)
001146  000207                                  RTS     PC                              ;                               3954

; Routine Size:  308 words,     Routine Base:  $CODE$ + 14226
; Maximum stack depth per invocation:  11 words
```

```
;  4080  1     GLOBAL ROUTINE DUPREDDBN : NOVALUE =
;  4081  1
;  4082  1     !+
;  4083  1     ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;  4084  1     ! "REDDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;  4085  1     ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM READS A DIAGONOSTIC BLOCK (DBN)
;  4086  1     ! AND PLACES IT IN THE DUP BUFFER CALLED "DUPPKT".  IF AN ERROR OCCURS WHILE
;  4087  1     ! RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;  4088  1     ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;  4089  1     !
;  4090  1     !
;  4091  1     !       IMPLICIT INPUTS:
;  4092  1     !               CST_ADDR    - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;  4093  1     !               DUOFF       - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;  4094  1     !
;  4095  2     BEGIN
;  4096  2     LOCAL                                                   !                                                       ZZZ
;  4097  2           TRYNUM : WORD,                                    !                                                       ZZZ
;  4098  2           MAX_TRY_COUNT : word initial (9);                 ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR               ZZZ
;  4099  2                                                             !                                                       ZZZ
;  4100  2     LABEL                                                   !                                                       ZZZ
;  4101  2          DUP_RLOOP;                                         !START OR DUP READ RETRY LOOP                          ZZZ
;  4102  2
;  4103  2
;  4104  2     !PRINTX (DER12);
;  4105  2     T_ADDR [T_DBN_RD] = .T_ADDR [T_DBN_RD] + 1;             ! INCREMENT # OF READS GIVEN
;  4106  2
;  4107  2     TRYNUM = 0;                                             !ZERO TRY COUNTER                                      ZZZ
;  4108  2     DUP_RLOOP:                                              !LABEL FOR LOOP EXCAPE ON GOOD READ                    ZZZ
;  4109  3     BEGIN                                                   !BEGIN DUP_RLOOP                                       ZZZ
;  4110  3     INCR TRIES FROM 1 TO 10 DO                              !START TRYING DUP READS                                ZZZ
;  4111  4     BEGIN                                                   !BEGIN LARGE DO LOOP                                   ZZZ
;  4112  4                                                             !                                                       ZZZ
;  4113  4
;  4114  4      MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                      ! PACKET SIZE                                  EXECUTF REDDBN PROGRAM
;  4115  4      MSCP_PKT [.MX1, OPCODE] = OP_ELP;                      ! OPCODE = EXECUTE LOCAL PROGRAM
;  4116  4      MSCP_PKT [.MX1, L1] = %ascii'R';                       ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;  4117  4      MSCP_PKT [.MX1, L2] = %ascii'E';
;  4118  4      MSCP_PKT [.MX1, L3] = %ascii'D';
;  4119  4      MSCP_PKT [.MX1, L4] = %ascii'D';
;  4120  4      MSCP_PKT [.MX1, L5] = %ascii'B';
;  4121  4      MSCP_PKT [.MX1, L6] = %ascii'N';
;  4122  4      MSCP_PKT [.MX1, MODIFY] = 1;                           ! STANDALONE MODIFIER
;  4123  4      MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                     ! CALL IT IMMEDIATE
;  4124  4      DUPCOMMAND ();                                         ! SENDS AND RECEIVES THE COMMAND
;  4125  4
;  4126  4      IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1          !status error
;  4127  4      THEN RETURN;
;  4128  4
;  4129  5      DO (MX1 = GET_PKT (.CCTLR))
;  4130  4      UNTIL (.MX1 GEQ 0);                    ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4131  4
;  4132  4            MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                ! PACKET SIZE                                  RECIEVE DATA
```

```
;  4133  4          MSCP_PKT [.MX1, OPCODE] = OP_RCD;                ! OPCODE = RECEIVE DATA
;  4134  4          MSCP_PKT [.MX1, BC_LO] = 80;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 2 *******see pg 26 DUP sp
ec
;  4135  4          MSCP_PKT [.MX1, BUF_O] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4136  4          MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4137  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4138  4          DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4139  4
;  4140  4          IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;  4141  4             (.DUPPKT [DUPTYPE] NEQU 1) OR                 !dup type error
;  4142  5             (.DUPPKT [DUPMESG] NEQU 5)
;  4143  4          THEN
;  4144  5                  (D_FAIL = 1;                             !TELL HARD_ERROR IT WAS A DUP PROBLEM            ZZZ
;  4145  5                   HARD_ERROR ();
;  4146  5                   D_FAIL = 0;                             !                                                ZZZ
;  4147  5                   CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;  4148  4                   RETURN;);                               ! NO POINT IN CONTINUING
;  4149  4
;  4150  5          DO (MX1 = GET_PKT (.CCTLR))
;  4151  4          UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4152  4
;  4153  4          MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;                ! PACKET SIZE                                SEND DATA
;  4154  4          MSCP_PKT [.MX1, OPCODE] = OP_SDD;                ! OPCODE = SEND DATA
;  4155  4          MSCP_PKT [.MX1, BC_LO] = 4;                      ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;  4156  4          MSCP_PKT [.MX1, BUF_O] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4157  4          DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; ! LOAD UNIT NUMBER (RDRX)
;  4158  4          DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;  4159  4          MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4160  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4161  4          DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4162  4
;  4163  4           IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1          !status error
;  4164  4           THEN RETURN;
;  4165  4
;  4166  5          DO (MX1 = GET_PKT (.CCTLR))
;  4167  4          UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4168  4
;  4169  4          MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                ! PACKET SIZE                                RECEIVE DATA
;  4170  4          MSCP_PKT [.MX1, OPCODE] = OP_RCD;                ! OPCODE = GET DUST STATUS
;  4171  4          MSCP_PKT [.MX1, BC_LO] = 514;                    ! BYTE COUNT TO BE TRANSFERED EQUALS 512
;  4172  4          MSCP_PKT [.MX1, BUF_O] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4173  4          MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4174  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4175  4          DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4176  4
;  4177  4          IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND                 ZZZ
;  4178  4             (.DUPPKT [DUPTYPE] EQL 6) AND                 !NO dup type error                       ZZZ
;  4179  5             (.DUPPKT [DUPMESG] EQL 2)                     !                                        ZZZ
;  4180  4          THEN                                             !THEN                                    ZZZ
;  4181  4             LEAVE DUP_RLOOP                               !I/O OK. EXIT RETRY LOOP.                ZZZ
;  4182  4          ELSE                                             !                                        ZZZ
;  4183  5             BEGIN                                         !                                        ZZZ
;  4184  5             TRYNUM = .TRYNUM + 1;                         !INCR ATTEMPT COUNT                      ZZZ
;  4185  5             IF .TRYNUM  EQL .MAX_TRY_COUNT                !IF IT FAILED ALL RETRIES, THEN          ZZZ
```

```
;  4186  5                THEN                                        !REPORT THE ERROR.                                  ZZZ
;  4187  6                        (D_FAIL = 1;                        !TELL HARD_ERROR IT WAS A DUP PROBLEM               ZZZ
;  4188  6                         HARD_ERROR ();                     !                                                   ZZZ
;  4189  6                         D_FAIL = 0;                        !                                                   ZZZ
;  4190  6                         CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;   ! SET FLAG                                 ZZZ
;  4191  5                         RETURN;);                          ! NO POINT IN CONTINUING                            ZZZ
;  4192  4                END;                                        !                                                   ZZZ
;  4193  3       END;                                                 !END LARGE DO LOOP                                  ZZZ
;  4194  3                                                            !                                                   ZZZ
;  4195  2       END;                                                 !END DUP_RLOOP                                      ZZZ
;  4196  2
;  4197  3       DO (MX1 = GET_PKT (.CCTLR))
;  4198  2       UNTIL (.MX1 GEQ 0);                    ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4199  2
;  4200  2       T_ADDR [T_BLK_RD] = .T_ADDR [T_BLK_RD] + 1;          !IF DUP NO ERROR THEN INCREMENT COUNTER
;  4201  2
;  4202  1       END;


                                        .SBTTL  DUPREDDBN MULTI-DRIVE TEST ROUTINES
000000  004137  000000G           DUPREDDBN::
                                        JSR     R1,$SAVE4                       ;                                   4080
000004  012704  000011                  MOV     #11,R4                          ; #,MAX.TRY.COUNT                   4095
000010  013700  000000G                 MOV     T.ADDR,R0                       ;                                   4105
000014  005260  000064                  INC     64(R0)
000020  005002                          CLR     R2                              ; TRYNUM                            4107
000022  012703  000012                  MOV     #12,R3                          ; #,TRIES                           4110
000026  013746  000110'        1$:       MOV     MX1,-(SP)                       ;                                   4114
000032  012746  000106                  MOV     #106,-(SP)
000036  004737  000000G                 JSR     PC,BL$MUL
000042  012760  000022  000006G         MOV     #22,MSCP.PKT+6(R0)
000050  112760  000003  000022G         MOVB    #3,MSCP.PKT+22(R0)              ;                                   4115
000056  112760  000122  000026G         MOVB    #122,MSCP.PKT+26(R0)            ;                                   4116
000064  112760  000105  000027G         MOVB    #105,MSCP.PKT+27(R0)            ;                                   4117
000072  112760  000104  000030G         MOVB    #104,MSCP.PKT+30(R0)            ;                                   4118
000100  112760  000104  000031G         MOVB    #104,MSCP.PKT+31(R0)            ;                                   4119
000106  112760  000102  000032G         MOVB    #102,MSCP.PKT+32(R0)            ;                                   4120
000114  112760  000116  000033G         MOVB    #116,MSCP.PKT+33(R0)            ;                                   4121
000122  012760  000001  000024G         MOV     #1,MSCP.PKT+24(R0)              ;                                   4122
000130  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)            ;                                   4123
000136  004737  000000V                 JSR     PC,DUPCOMMAND                   ;                                   4124
000142  013700  001156'                 MOV     DUOFF,R0                        ;                                   4126
000146  006300                          ASL     R0
000150  063700  000000G                 ADD     CST.ADDR,R0
000154  032760  040000  000020         BIT     #40000,20(R0)
000162  001402                          BEQ     2$
000164  022626                          CMP     (SP)+,(SP)+                     ;                                   4080
000166  000207                          RTS     PC                              ;                                   4127
000170  013716  000000G         2$:       MOV     CCTLR,(SP)                      ;                                   4129
000174  004737  000000G                 JSR     PC,GET.PKT
000200  010037  000110                  MOV     R0,MX1
000204  002771                          BLT     2$                              ;                                   4130
000206  010016                          MOV     R0,(SP)                         ;                                   4132
```

```
000210  012746  000106                          MOV     #106,-(SP)
000214  004737  000000G                         JSR     PC,BL$MUL
000220  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000226  112760  000005  000022G                 MOVB    #5,MSCP.PKT+22(R0)      ;                       4133
000234  012760  000120  000026G                 MOV     #120,MSCP.PKT+26(R0)    ;                       4134
000242  012760  000000G 000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0) ;                       4135
000250  005060  000024G                         CLR     MSCP.PKT+24(R0)         ;                       4136
000254  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)    ;                       4137
000262  152760  000020  000010G                 BISB    #20,MSCP.PKT+10(R0)
000270  004737  000000V                         JSR     PC,DUPCOMMAND           ;                       4138
000274  013700  001156'                         MOV     DUOFF,R0                ;                       4140
000300  006300                                  ASL     R0
000302  063700  000000G                         ADD     CST.ADDR,R0
000306  032760  040000  000020                  BIT     #40000,20(R0)
000314  001016                                  BNE     3$
000316  013700  000000G                         MOV     DUPPKT,R0               ;                       4141
000322  042700  007777                          BIC     #7777,R0
000326  020027  010000                          CMP     R0,#10000
000332  001007                                  BNE     3$
000334  013700  000000G                         MOV     DUPPKT,R0               ;                       4142
000340  042700  170000                          BIC     #170000,R0
000344  020027  000005                          CMP     R0,#5
000350  001422                                  BEQ     4$
000352  112737  000001  000000G         3$:     MOVB    #1,D.FAIL               ;                       4144
000360  004737  000000V                         JSR     PC,HARD.ERROR           ;                       4145
000364  105037  000000G                         CLRB    D.FAIL                  ;                       4146
000370  013700  001156'                         MOV     DUOFF,R0                ;                       4147
000374  006300                                  ASL     R0
000376  063700  000000G                         ADD     CST.ADDR,R0
000402  052760  040000  000020                  BIS     #40000,20(R0)
000410  062706  000006                          ADD     #6,SP                   ;                       4148
000414  000207                                  RTS     PC                      ;                       4144
000416  013716  000000G                 4$:     MOV     CCTLR,(SP)              ;                       4150
000422  004737  000000G                         JSR     PC,GET.PKT
000426  010037  000110'                         MOV     R0,MX1
000432  002771                                  BLT     4$                      ;                       4151
000434  010016                                  MOV     R0,(SP)                 ;                       4153
000436  012746  000106                          MOV     #106,-(SP)
000442  004737  000000G                         JSR     PC,BL$MUL
000446  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000454  112760  000004  000022G                 MOVB    #4,MSCP.PKT+22(R0)      ;                       4154
000462  012760  000004  000026G                 MOV     #4,MSCP.PKT+26(R0)      ;                       4155
000470  012760  000000G 000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0) ;                       4156
000476  013701  001156'                         MOV     DUOFF,R1                ;                       4157
000502  006301                                  ASL     R1
000504  063701  000000G                         ADD     CST.ADDR,R1
000510  111137  000000G                         MOVB    (R1),DUPPKT
000514  042737  177760  000000G                 BIC     #177760,DUPPKT
000522  013701  001156'                         MOV     DUOFF,R1                ;                       4158
000526  006301                                  ASL     R1
000530  063701  000000G                         ADD     CST.ADDR,R1
000534  116137  000020  000002G                 MOVB    20(R1),DUPPKT+2
000542  105037  000003G                         CLRB    DUPPKT+3
```

```
000546  005060  000024G                        CLR     MSCP.PKT+24(R0)                 ;                                       4159
000552  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)            ;                                       4160
000560  152760  000020  000010G                 BISB    #20,MSCP.PKT+10(R0)
000566  004737  000000V                         JSR     PC,DUPCOMMAND                   ;                                       4161
000572  013700  001156                          MOV     DUOFF,R0                        ;                                       4163
000576  006300                                  ASL     R0
000600  063700  000000G                         ADD     CST.ADDR,R0
000604  032760  040000  000020                  BIT     #40000,20(R0)
000612  001403                                  BEQ     5$
000614  062706  000010                          ADD     #10,SP                          ;                                       4080
000620  000207                                  RTS     PC                              ;                                       4164
000622  013716  000000G                 5$:     MOV     CCTLR,(SP)                      ;                                       4166
000626  004737  000000G                         JSR     PC,GET.PKT
000632  010037  000110'                         MOV     R0,MX1
000636  002771                                  BLT     5$                              ;                                       4167
000640  010016                                  MOV     R0,(SP)                         ;                                       4169
000642  012746  000106                          MOV     #106,-(SP)
000646  004737  000000G                         JSR     PC,BL$MUL
000652  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000660  112760  000005  000022G                 MOVB    #5,MSCP.PKT+22(R0)              ;                                       4170
000666  012760  001002  000026G                 MOV     #1002,MSCP.PKT+26(R0)           ;                                       4171
000674  012760  000000G 000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                                       4172
000702  005060  000024G                         CLR     MSCP.PKT+24(R0)                 ;                                       4173
000706  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)            ;                                       4174
000714  152760  000020  000010G                 BISB    #20,MSCP.PKT+10(R0)
000722  004737  000000V                         JSR     PC,DUPCOMMAND                   ;                                       4175
000726  013700  001156'                         MOV     DUOFF,R0                        ;                                       4177
000732  006300                                  ASL     R0
000734  063700  000000G                         ADD     CST.ADDR,R0
000740  032760  040000  000020                  BIT     #40000,20(R0)
000746  001021                                  BNE     6$
000750  013700  000C00G                         MOV     DUPPKT,R0                       ;                                       4178
000754  042700  007777                          BIC     #7777,R0
000760  020027  060000                          CMP     R0,#60000
000764  001012                                  BNE     6$
000766  013700  000000G                         MOV     DUPPKT,R0                       ;                                       4179
000772  042700  170000                          BIC     #170000,R0
000776  020027  000002                          CMP     R0,#2
001002  001003                                  BNE     6$
001004  062706  000012                          ADD     #12,SP                          ;                                       4181
001010  000433                                  BR      8$
001012  005202                          6$:     INC     R2                              ; TRYNUM                                4184
001014  020204                                  CMP     R2,R4                           ; TRYNUM,MAX.TRY.COUNT                  4185
001016  001022                                  BNE     7$
001020  112737  000001  000000G                 MOVB    #1,D.FAIL                       ;                                       4187
001026  004737  000000V                         JSR     PC,HARD.ERROR                   ;                                       4188
001032  105037  000000G                         CLRB    D.FAIL                          ;                                       4189
001036  013700  001156'                         MOV     DUOFF,R0                        ;                                       4190
001042  006300                                  ASL     R0
001044  063700  000000G                         ADD     CST.ADDR,R0
001050  052760  040000  000020                  BIS     #40000,20(R0)
001056  062706  000012                          ADD     #12,SP                          ;                                       4191
001062  000207                                  RTS     PC                              ;                                       4187
```

```
001064  062706  000012                  1$:     ADD     #12,SP                          ;                               4111
001070  005303                                  DEC     R3                              ; TRIES                         4110
001072  001402                                  BEQ     8$
001074  000137  015424'                         JMP     1$
001100  013746  000000G                 8$:     MOV     CCTLR,-(SP)                     ;                               4197
001104  004737  000000G                         JSR     PC,GET.PKT
001110  010037  000110'                         MOV     R0,MX1
001114  005726                                  TST     (SP)+
001116  005700                                  TST     R0                              ; MX1                           4198
001120  002767                                  BLT     8$
001122  013700  000000G                         MOV     T.ADDR,R0                       ;                               4200
001126  005260  000062                          INC     62(R0)
001132  000207                                  RTS     PC                              ;                               4080

; Routine Size:  302 words,    Routine Base:  $CODE$ + 15376
; Maximum stack depth per invocation:  11 words


;   4203  1
```

```
;    4204  1
;    4205  1     GLOBAL ROUTINE DUPCOMMAND : NOVALUE =
;    4206  1
;    4207  1     !+
;    4208  1     ! THIS ROUTINE IS CALLED BY DUP TO PROCESS COMMANDS.
;    4209  1     ! THE COMMAND ENVOLOPES ARE FILLED IN DUP ROUTINES IN THE "MX1" INDEX.
;    4210  1     ! WITH THE INDEX THIS ROUTINE SENDS THE COMMAND, WAITS FOR A
;    4211  1     ! RESPONSES AND THEN PROCESSES THE RETURN PACKET.
;    4212  1     !
;    4213  2     BEGIN
;    4214  2     !PRINTX (DER13);
;    4215  2
;    4216  2     MSCP_PKT [.MX1, CREDITS] = 0;                   ! DUP DOES NOT USE THE CREDIT SYSTEM
;    4217  2     MSCP_PKT [.MX1, CONNID] = CID_DUP;              ! MAKE PACKAGE EQUAL A DUP COMMAND
;    4218  2     MSCP_PKT [.MX1, DK_NUM] = 0;                    ! DISK NUMBER (NOT APPLICABLE)
;    4219  2
;    4220  2     IF SEND (.MX1) EQLU FAILURE                     ! ATTEMPT SEND; IF CTLR IS OFFLINE
;    4221  2     THEN
;    4222  3             BEGIN
;    4223  3             PUT_PKT (.MX1);
;    4224  3             MX1 = -1;                               ! RETURN ENVELOPE TO POOL
;    4225  3             CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;
;    4226  3     !       PRINTF (DBM112);                        ! "DUP: PKT NOT AVAILABLE"  ZZZ
;    4227  3             END
;    4228  3
;    4229  2     ELSE
;    4230  2             do
;    4231  3                     begin
;    4232  3                     BREAK;                                  ! BREAK FOR ACT
;    4233  3                     PROC_RETPKT ();                                 ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;    4234  3                     end
;    4235  2             until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;    4236  2             (.EOP_FLAG eql true);                           ! or end of pass caused by error
;    4237  1     END;


                                        .SBTTL  DUPCOMMAND MULTI-DRIVE TEST ROUTINES
000000  013746  000110'              DUPCOMMAND::
                                        MOV     MX1,-(SP)                       ;                       4216
000004  012746  000106                  MOV     #106,-(SP)
000010  004737  000000G                 JSR     PC,BL$MUL
000014  142760  000017  000010G         BICB    #17,MSCP.PKT+10(R0)
000022  112760  000002  000011G         MOVB    #2,MSCP.PKT+11(R0)              ;                       4217
000030  005060  000016G                 CLR     MSCP.PKT+16(R0)                 ;                       4218
000034  013716  000110'                 MOV     MX1,(SP)                        ;                       4220
000040  004737  000000G                 JSR     PC,SEND
000044  005700                          TST     R0
000046  001020                          BNE     1$
000050  013716  000110'                 MOV     MX1,(SP)                        ;                       4223
000054  004737  000000G                 JSR     PC,PUT.PKT
000060  012737  177777  000110'         MOV     #-1,MX1                         ;                       4224
000066  013700  001156'                 MOV     DUOFF,R0                        ;                       4225
000072  006300                          ASL     R0
```

M13


```
000074  063700  000000G                        ADD     CST.ADDR,R0
000100  052760  040000  000020                 BIS     #40000,20(R0)
000106  000415                                 BR      2$                      ;                4220
000110  104422                          1$:    TRAP    22                      ;                4231
000112  004737  000000V                        JSR     PC,PROC.RETPKT          ;                4233
000116  013700  000000G                        MOV     RP.ADDR,R0              ;                4235
000122  023760  000000G 000004                 CMP     CRN.LOW,4(R0)
000130  001404                                 BEQ     2$
000132  123727  000000G 000001                 CMPB    EOP.FLAG,#1             ;                4236
000140  001363                                 BNE     1$
000142  022626                          2$:    CMP     (SP)+,(SP)+             ;                4213
000144  000207                                 RTS     PC                      ;                4205

; Routine Size:  51 words,      Routine Base:  $CODE$ + 16532
; Maximum stack depth per invocation:  4 words
```

```
;  4238  1
;  4239  1     GLOBAL ROUTINE DUPIDLE : NOVALUE =
;  4240  1     !+
;  4241  1     ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO INSURE THAT THE CONTROLLER
;  4242  1     ! IS NOT IN A ACTIVE STATE. IF CALLED AND THE CONTROLLER IS IN AN ACTIVE
;  4243  1     ! STATE THE CONTROLLER WILL GIVE AN ABORT COMMAND WHICH SHOULD KILL THE
;  4244  1     ! CURRENT JOB OR LOCAL PROGRAM.
;  4245  1     !
;  4246  2     BEGIN
;  4247  2     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 0;               !CLEAR DUP ERROR FLAG;
;  4248  2
;  4249  2     MSCP_PKT [.MX1, MSGLEN] = SZ_GDS;                ! PACKET SIZE                                  GET DUST STATUS
;  4250  2     MSCP_PKT [.MX1, OPCODE] = OP_GDS;                ! OPCODE = GET DUST STATUS
;  4251  2     MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4252  2     MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;               ! CALL IT IMMEDIATE
;  4253  2     DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4254  2                                                      ! GDS ONLY RETURNS SUCCESS or it don t return
;  4255  2
;  4256  3     DO (MX1 = GET_PKT (.CCTLR))
;  4257  2     UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;  4258  2
;  4259  2     if .CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] neq IDLE       ! if not in idle state then abort the program
;  4260  2     then
;  4261  3         begin
;  4262  3         MSCP_PKT [.MX1, MSGLEN] = SZ_ABT;                   ! PACKET SIZE                                  ABORT CMD
;  4263  3         MSCP_PKT [.MX1, OPCODE] = OP_ABT;                   ! OPCODE = ABORT PROGRAM
;  4264  3         MSCP_PKT [.MX1, MODIFY] = 0;                        !
;  4265  3         MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                  ! CALL IT IMMEDIATE
;  4266  3         DUPCOMMAND ();                                      ! SENDS AND RECEIVES THE COMMAND
;  4267  3                                                             !ONLY ERROR IS already in idle state
;  4268  4         DO (MX1 = GET_PKT (.CCTLR))
;  4269  3         UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;  4270  2         end;
;  4271  1     end;


                                        .SBTTL  DUPIDLE MULTI-DRIVE TEST ROUTINES
000000  010146                 DUPIDLE::
                                        MOV     R1,-(SP)                        ;                               4239
000002  013700  001156'                 MOV     DUOFF,R0                        ;                               4247
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)                       ;                               4249
000026  012746  000106                  MOV     #106,-(SP)
000032  004737  000000G                 JSR     PC,BL$MUL
000036  012760  000014  000006G         MOV     #14,MSCP.PKT+6(R0)
000044  112760  000001  000022G         MOVB    #1,MSCP.PKT+22(R0)              ;                               4250
000052  005060  000024G                 CLR     MSCP.PKT+24(R0)                 ;                               4251
000056  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)            ;                               4252
000064  004737  016532'                 JSR     PC,DUPCOMMAND                   ;                               4253
000070  013716  000000G         1$:     MOV     CCTLR,(SP)                      ;                               4256
000074  004737  000000G                 JSR     PC,GET.PKT
```

```
000100  010037  000110                        MOV     R0,MX1
000104  010001                                MOV     R0,R1                   ; MX1,*                 4257
000106  002770                                BLT     1$
000110  013700  001156                        MOV     DUOFF,R0                ;                       4259
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)                 ;                       4262
000134  012746  000106                        MOV     #106,-(SP)
000140  004737  000000G                       JSR     PC,BL$MUL
000144  012760  000014  000006G               MOV     #14,MSCP.PKT+6(R0)
000152  112760  000006  000022G               MOVB    #6,MSCP.PKT+22(R0)      ;                       4263
000160  005060  000024G                       CLR     MSCP.PKT+24(R0)         ;                       4264
000164  142760  000360  000010G               BICB    #360,MSCP.PKT+10(R0)    ;                       4265
000172  004737  016532                        JSR     PC,DUPCOMMAND           ;                       4266
000176  013716  000000G             2$:       MOV     CCTLR,(SP)              ;                       4268
000202  004737  000000G                       JSR     PC,GET.PKT
000206  010037  000110                        MOV     R0,MX1
000212  002771                                BLT     2$                      ;                       4269
000214  005726                                TST     (SP)+                   ;                       4261
000216  022626                      3$:       CMP     (SP)+,(SP)+             ;                       4246
000220  012601                                MOV     (SP)+,R1                ;                       4239
000222  000207                                RTS     PC

; Routine Size:  74 words,      Routine Base:  $CODE$ + 16700
; Maximum stack depth per invocation:  5 words
```

```
;    4272  1     GLOBAL ROUTINE QIO_LBN : NOVALUE =
;    4273  1
;    4274  1     !+
;    4275  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE LOGICAL BLOCK NUMBER TO
;    4276  1     !       BE USED FOR THE CURRENT QIO OR QIO PAIR.
;    4277  1     !
;    4278  1     !       IF THE OPERATOR CHOSE THE RANDOM BLOCK MODE OPTION, THEN THE LBN IS
;    4279  1     !       RANDOMLY CHOSEN WITHIN THE SPECIFIED LIMITS FOR THE LBN.
;    4280  1     !       OTHERWISE, THE NEXT SEQUENTIAL LBN IS DERIVED FROM THE BLOCK SEQUENCE
;    4281  1     !       TABLE (BST).
;    4282  1     !
;    4283  1     !       IMPLICIT INPUTS:
;    4284  1     !               L$LUN - CURRENT (DIAGNOSTIC SUPERVIOR) UNIT NUMBER
;    4285  1     !
;    4286  1     !       IMPLICIT OUTPUTS:
;    4287  1     !               THE LBN IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;    4288  1     !
;    4289  1
;    4290  2     begin
;    4291  2
;    4292  2     local
;    4293  2         RD_DISK : byte;      ! FLAG TO INDICATE WINCHESTER DISK SELECTED
;    4294  2
;    4295  2
;    4296  2     if .CST_ADDR [.CUOFF, D_TYPE] eql RX_50
;    4297  2     then
;    4298  2         RD_DISK = FALSE
;    4299  2     else
;    4300  2         RD_DISK = TRUE;
;    4301  2
;    4302  2     MAD1 [LBN_L] = .BST [.L$LUN, LO_WRD];                   ! LOAD LBN INTO 1ST PACKET
;    4303  2     MAD1 [LBN_H] = .BST [.L$LUN, HI_WRD];                   ! LOAD LBN INTO 1ST PACKET
;    4304  2
;    4305  2     if .MX2 geq 0                               ! IF 2 QIOS
;    4306  2     then
;    4307  3         begin
;    4308  3         MAD2 [LBN_L] = .BST [.L$LUN, LO_WRD];                       ! LOAD LBN INTO 2ND PACKET
;    4309  3         MAD2 [LBN_H] = .BST [.L$LUN, HI_WRD];               ! LOAD LBN INTO 2ND PACKET
;    4310  2         end;
;    4311  2
;    4312  3     if BIT_TST (SWP_FLAGS, SWF_BLK)             ! IF RANDOM BLOCK MODE
;    4313  2     then
;    4314  3         begin
;    4315  4         if  NOT ((.RD_DISK) and
;    4316  4           (((.RANDOM [0] and %o'077777') mod (99)) lequ 33))                    ! REDUCES SEEKS TO 66% OF TRANFERS o
n winchesters
;    4317  3         then
;    4318  4             begin                                                             ! NOTE IF NOT CALLED THE LBN REMAINS
 THE SAME
;    4319  4             if .CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .CST_ADDR [.CUOFF + 4, D END1]
;    4320  4             then                                                              ! if upper word of beg trk and endin
g trk same
;    4321  4                     BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0] +             ! select low lbn from random
 number WINDOW
;    4322  4                     MODULAS (.CST_ADDR [.CUOFF + 1, D_BEG0], .CST_ADDR [.CUOFF + 3, D_END0])
;    4323  4
;    4324  4             else                                                              ! if upper word of beg trk and end t
rk different
```

```
;    4325  5                begin
;    4326  5                BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1] +          ! select upper lbn from window
;    4327  5                MODULAS (.CST_ADDR [.CUOFF + 2, D_BEG1], .CST_ADDR [.CUOFF + 4, D_END1]);
;    4328  5
;    4329  5                if .BST [.L$LUN, HI_WRD] eql .CST_ADDR [.CUOFF + 4, D_END1]      ! IF UPPER WORD EQUALS HI LIMIT BE S
URE LOWER
;    4330  5                then .BST [.L$LUN, LO_WRD] = MODULAS (0, .CST_ADDR [.CUOFF + 3, D_ENDO]); ! WORD DOES NOT PASS HI LIMIT
;    4331  5
;    4332  5                if .BST [.L$LUN, HI_WRD] eql .CST_ADDR [.CUOFF + 2, D_BEG1]      ! if upper word equal lo limit make
sure lower
;    4333  5                then BST [.L$LUN, LO_WRD] = %O'177777'   MODULAS (.CST_ADDR [.CUOFF + 1, D_BEGO], %O'177777');
;    4334  5                                                                                 ! word is above lo limit
;    4335  5
;    4336  5                if .BST [.L$LUN, HI_WRD] gtr .CST_ADDR [.CUOFF + 2, D_BEG1] and
;    4337  5                   .BST [.L$LUN, HI_WRD] lss .CST_ADDR [.CUOFF + 4, D_END1]      ! if neither of the above then any n
umber is good
;    4338  5                then BST [.L$LUN, LO_WRD] = .RANDOM [5];
;    4339  4                end;
;    4340  3            end;
;    4341  3        end
;    4342  2    else
;    4343  3        begin                                          ! ELSE - SEQUENTIAL LBN MODE
;    4344  4        if (.TRK_SGN [.L$LUN] geq 1)
;    4345  3        then                                           ! if positive track direction add one to multiword
;    4346  4            (if .BST [.L$LUN, LO_WRD] eqlu %o'177777'
;    4347  4            then
;    4348  5                    begin
;    4349  5                    BST [.L$LUN, LO_WRD] = 0;
;    4350  5                    BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] + 1;
;    4351  5                    end
;    4352  4            else
;    4353  4                    BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] + 1)
;    4354  3        else                                           ! if negative track direction subtract one from multiword
;    4355  3            if .BST [.L$LUN, LO_WRD] eqlu %o'0'
;    4356  3            then
;    4357  4                    begin
;    4358  4                    BST [.L$LUN, LO_WRD] = %o'177777';
;    4359  4                    BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD]   1;
;    4360  4                    end
;    4361  3            else
;    4362  3                    BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] - 1;
;    4363  3
;    4364  3
;    4365  3        if  .BST [.L$LUN, LO_WRD] gequ (.CST_ADDR [.CUOFF + 3, D_ENDO]) and ! if hi limit then change direction
;    4366  4            .BST [.L$LUN, HI_WRD] gequ (.CST_ADDR [.CUOFF + 4, D_END1])
;    4367  3        then TRK_SGN [.L$LUN] =  -1;
;    4368  3
;    4369  3        if  .BST [.L$LUN, LO_WRD] lequ (.CST_ADDR [.CUOFF + 1, D_BEGO] + 1)  and    ! if low limit then change direction
;    4370  4            .BST [.L$LUN, HI_WRD] lequ (.CST_ADDR [.CUOFF + 2, D_BEG1])
;    4371  3        then TRK_SGN [.L$LUN] = 1;
;    4372  2        end;
;    4373  1    end;                                               ! ROUTINE QIO_LBN


                                        .SBTTL  QIO.LBN MULTI-DRIVE TEST ROUTINES
000000  004137  000000G              QIO.LBN::
```

```
                                                JSR     R1,$SAVE5                       ;                               4272
000004  013705  000000G                         MOV     CST.ADDR,R5                     ;                               4296
000010  013700  000000G                         MOV     CUOFF,R0
000014  006300                                  ASL     R0
000016  060500                                  ADD     R5,R0
000020  132710  000020                          BITB    #20,(R0)
000024  001002                                  BNE     1$
000026  105004                                  CLRB    R4                              ; RD.DISK                       4298
000030  000402                                  BR      2$                              ;                               4296
000032  112704  000001              1$:         MOVB    #1,R4                           ; *,RD.DISK                     4300
000036  013700  000114              2$:         MOV     MAD1,R0                         ;                               4302
000042  013701  000000G                         MOV     L$LUN,R1
000046  010103                                  MOV     R1,R3
000050  006303                                  ASL     R3
000052  006303                                  ASL     R3
000054  012702  000000G                         MOV     #BST,R2
000060  060302                                  ADD     R3,R2
000062  011260  000046                          MOV     (R2),46(R0)
000066  062703  000002G                         ADD     #BST+2,R3                       ;                               4303
000072  011360  000050                          MOV     (R3),50(R0)
000076  005737  000112'                         TST     MX2                             ;                               4305
000102  002406                                  BLT     3$
000104  013700  000116'                         MOV     MAD2,R0                         ;                               4308
000110  011260  000046                          MOV     (R2),46(R0)
000114  011360  000050                          MOV     (R3),50(R0)                     ;                               4309
000120  032737  040000  000000G     3$:         BIT     #40000,SWP.FLAGS                ;                               4312
000126  001002                                  BNE     4$
000130  000137  017700'                         JMP     10$
000134  006004                      4$:         ROR     R4                              ; RD.DISK                       4315
000136  103015                                  BCC     5$
000140  013746  000000G                         MOV     RANDOM,-(SP)                    ;                               4316
000144  042716  100000                          BIC     #100000,(SP)
000150  012746  000143                          MOV     #143,-(SP)
000154  004737  000000G                         JSR     PC,BL$MOD
000160  022626                                  CMP     (SP)+,(SP)+
000162  020027  000041                          CMP     R0,#41
000166  101001                                  BHI     5$
000170  000207                                  RTS     PC
000172  013700  000000G             5$:         MOV     CUOFF,R0                        ;                               4319
000176  006300                                  ASL     R0
000200  060500                                  ADD     R5,R0
000202  016004  000004                          MOV     4(R0),R4
000206  013700  000000G                         MOV     CUOFF,R0
000212  006300                                  ASL     R0
000214  060500                                  ADD     R5,R0
000216  020460  000010                          CMP     R4,10(R0)
000222  001022                                  BNE     6$
000224  013701  000000G                         MOV     CUOFF,R1                        ;                               4321
000230  006301                                  ASL     R1
000232  060501                                  ADD     R5,R1
000234  016146  000002                          MOV     2(R1),-(SP)                     ;                               4322
000240  013700  000000G                         MOV     CUOFF,R0
000244  006300                                  ASL     R0
```

```
000246  060500                                  ADD     R5,R0
000250  016046  000006                          MOV     6(R0),-(SP)
000254  004737  000000G                         JSR     PC,MODULAS
000260  066100  000002                          ADD     2(R1),R0        ;                       4321
000264  010012                                  MOV     R0,(R2)
000266  000530                                  BR      9$              ;                       4319
000270  010446                          6$:     MOV     R4,-(SP)        ;                       4327
000272  016046  000010                          MOV     10(R0),-(SP)
000276  004737  000000G                         JSR     PC,MODULAS
000302  060400                                  ADD     R4,R0           ;                       4326
000304  010013                                  MOV     R0,(R3)
000306  013701  000000G                         MOV     L$LUN,R1        ;                       4329
000312  006301                                  ASL     R1
000314  006301                                  ASL     R1
000316  013700  000000G                         MOV     CUOFF,R0
000322  006300                                  ASL     R0
000324  063700  000000G                         ADD     CST.ADDR,R0
000330  026160  000002G 000010                  CMP     BST+2(R1),10(R0)
000336  001015                                  BNE     7$
000340  005016                                  CLR     (SP)            ;                       4330
000342  013700  000000G                         MOV     CUOFF,R0
000346  006300                                  ASL     R0
000350  063700  000000G                         ADD     CST.ADDR,R0
000354  016046  000006                          MOV     6(R0),-(SP)
000360  004737  000000G                         JSR     PC,MODULAS
000364  010071  000000G                         MOV     R0,@BST(R1)
000370  005726                                  TST     (SP)+
000372  013701  000000G                 7$:     MOV     L$LUN,R1        ;                       4332
000376  006301                                  ASL     R1
000400  006301                                  ASL     R1
000402  013700  000000G                         MOV     CUOFF,R0
000406  006300                                  ASL     R0
000410  063700  000000G                         ADD     CST.ADDR,R0
000414  026160  000002G 000004                  CMP     BST+2(R1),4(R0)
000422  001021                                  BNE     8$
000424  013700  000000G                         MOV     CUOFF,R0        ;                       4333
000430  006300                                  ASL     R0
000432  063700  000000G                         ADD     CST.ADDR,R0
000436  016016  000002                          MOV     2(R0),(SP)
000442  012746  177777                          MOV     #-1,-(SP)
000446  004737  000000G                         JSR     PC,MODULAS
000452  012761  177777  000000G                 MOV     #-1,BST(R1)
000460  160061  000000G                         SUB     R0,BST(R1)
000464  005726                                  TST     (SP)+
000466  013700  000000G                 8$:     MOV     L$LUN,R0        ;                       4336
000472  006300                                  ASL     R0
000474  006300                                  ASL     R0
000476  013701  000000G                         MOV     CUOFF,R1
000502  006301                                  ASL     R1
000504  063701  000000G                         ADD     CST.ADDR,R1
000510  026061  000002G 000004                  CMP     BST+2(R0),4(R1)
000516  003414                                  BLE     9$
000520  013701  000000G                         MOV     CUOFF,R1        ;                       4337
```

```
000524  006301                                  ASL     R1
000526  063701  000000G                         ADD     CST.ADDR,R1
000532  026061  000002G 000010                  CMP     BST+2(R0),10(R1)
000540  002003                                  BGE     9$
000542  013760  000012G 000000G                 MOV     RANDOM+12,BST(R0)       ;                4338
000550  022626                          9$:     CMP     (SP)+,(SP)+             ;                4318
000552  000207                                  RTS     PC                      ;                4312
000554  062701  000000G                 10$:    ADD     #TRK.SGN,R1             ;                4344
000560  105711                                  TSTB    (R1)
000562  003410                                  BLE     12$
000564  021227  177777                          CMP     (R2),#-1                ;                4346
000570  001003                                  BNE     11$
000572  005012                                  CLR     (R2)                    ;                4349
000574  005213                                  INC     (R3)                    ;                4350
000576  000411                                  BR      14$                     ;                4346
000600  005212                          11$:    INC     (R2)                    ;                4353
000602  000407                                  BR      14$                     ;                4344
000604  005712                          12$:    TST     (R2)                    ;                4355
000606  001004                                  BNE     13$
000610  012712  177777                          MOV     #-1,(R2)                ;                4358
000614  005313                                  DEC     (R3)                    ;                4359
000616  000401                                  BR      14$                     ;                4355
000620  005312                          13$:    DEC     (R2)                    ;                4362
000622  013700  000000G                 14$:    MOV     CUOFF,R0                ;                4365
000626  006300                                  ASL     R0
000630  060500                                  ADD     R5,R0
000632  021260  000006                          CMP     (R2),6(R0)
000636  103411                                  BLO     15$
000640  013700  000000G                         MOV     CUOFF,R0                ;                4366
000644  006300                                  ASL     R0
000646  060500                                  ADD     R5,R0
000650  021360  000010                          CMP     (R3),10(R0)
000654  103402                                  BLO     15$
000656  112711  000377                          MOVB    #377,(R1)               ;                4367
000662  013700  000000G                 15$:    MOV     CUOFF,R0                ;                4369
000666  006300                                  ASL     R0
000670  060500                                  ADD     R5,R0
000672  016000  000002                          MOV     2(R0),R0
000676  005200                                  INC     R0
000700  021200                                  CMP     (R2),R0
000702  101011                                  BHI     16$
000704  013700  000000G                         MOV     CUOFF,R0                ;                4370
000710  006300                                  ASL     R0
000712  060500                                  ADD     R5,R0
000714  021360  000004                          CMP     (R3),4(R0)
000720  101002                                  BHI     16$
000722  112711  000001                          MOVB    #1,(R1)                 ;                4371
000726  000207                          16$:    RTS     PC                      ;                4272

; Routine Size:  236 words,     Routine Base:  $CODE$ + 17124
; Maximum stack depth per invocation:  10 words
```

```
;  4374  1     !!ZZZ routine QIO_SIZE : novalue *
;  4375  1     GLOBAL ROUTINE QIO_SIZE : NOVALUE *
;  4376  1
;  4377  1     !*
;  4378  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O TRANSFER BYTE COUNT
;  4379  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE BYTE COUNT IS
;  4380  1     !       DETERMINED BY A RANDOM NUMBER, AND WILL ALWAYS FALL BETWEEN 1 AND THE
;  4381  1     !       I/O BUFFER SIZE (BYTS_PER_QIO). It is assumed that BYTS_PER_QIO will
;  4382  1     !       never be larger than one binary word or  65000 bytes.
;  4383  1     !
;  4384  1     !       IMPLICIT OUTPUTS:
;  4385  1     !               THE BYTE COUNT IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;  4386  1     !
;  4387  1
;  4388  2          begin
;  4389  2
;  4390  2          local
;  4391  2              SIZE : word,                                                    ! BYTE COUNT
;  4392  2              BLOCKS_LEFT : word;                                             ! REMAINING BLOCKS LEFT
;  4393  2
;  4394  2          SIZE = ((.RANDOM [4] and %o'077777') mod (.BYTS_PER_QIO + 1)) and %o'177760';!GET BYTE COUNT FROM RANDOM NUMBER
;  4395  2
;  4396  2          if .SIZE eql 0
;  4397  2          then
;  4398  2              SIZE = 16;
;  4399  2
;  4400  2          if .CST_ADDR [.CUOFF + 4, D_END1] gtru .MAD1 [LBN_H]
;  4401  2          then BLOCKS_LEFT = %o'177777'                                       ! find
;  4402  2          else BLOCKS_LEFT = .CST_ADDR [.CUOFF + 3, D_END0] - .MAD1 [LBN_L] + 1;          ! REMAINING BLOCK COUNT
;  4403  2
;  4404  2          if ((.SIZE + BYTES_PER_SECT - 1) / BYTES_PER_SECT) gtru .BLOCKS_LEFT    ! IF BLOCK COUNT NOT ENOUGH
;  4405  2          then
;  4406  2              SIZE = .BLOCKS_LEFT * BYTES_PER_SECT;                           ! ADJUST BYTE COUNT DOWN
;  4407  2
;  4408  2          MAD1 [BC_LO] = .SIZE;                                               ! LOAD SIZE INTO 1ST MSCP PACKET
;  4409  2
;  4410  2          if .MX2 geq 0                                                       ! IF 2 QIOS
;  4411  2          then
;  4412  2              MAD2 [BC_LO] = .SIZE;                                           ! LOAD SIZE INTO 2ND MSCP PACKET
;  4413  2
;  4414  1          end;                                                                ! ROUTINE QIO_SIZE


                                        .SBTTL  QIO.SIZE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          QIO.SIZE::
                                        JSR     R1,$SAVE3                       ;                                       4375
000004  013746  000010G                 MOV     RANDOM+10,-(SP)                 ;                                       4394
000010  042716  100000                  BIC     #100000,(SP)
000014  013746  000000G                 MOV     BYTS.PER.QIO,-(SP)
000020  005216                          INC     (SP)
000022  004737  000000G                 JSR     PC,BL$MOD
000026  010003                          MOV     R0,R3                           ; *,SIZE
000030  042703  000017                  BIC     #17,R3                          ; *,SIZE
```

```
000034  001002                          BNE     1$                      ;                               4396
000036  012703  000020                  MOV     #20,R3                  ; *,SIZE                        4398
000042  013700  000000G         1$:     MOV     CUOFF,R0                ;                               4400
000046  006300                          ASL     R0
000050  063700  000000G                 ADD     CST.ADDR,R0
000054  013701  000114'                 MOV     MAD1,R1
000060  026061  000010  000050          CMP     10(R0),50(R1)
000066  101403                          BLOS    2$
000070  012702  177777                  MOV     #-1,R2                  ; *,BLOCKS.LEFT                 4401
000074  000413                          BR      3$                      ;                               4400
000076  013700  000000G         2$:     MOV     CUOFF,R0                ;                               4402
000102  006300                          ASL     R0
000104  063700  000000G                 ADD     CST.ADDR,R0
000110  016000  000006                  MOV     6(R0),R0
000114  166100  000046                  SUB     46(R1),R0
000120  010002                          MOV     R0,R2                   ; *,BLOCKS.LEFT
000122  005202                          INC     R2                      ; BLOCKS.LEFT
000124  010316                  3$:     MOV     R3,(SP)                 ; SIZE,*                        4404
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,*                 4406
000152  000300                          SWAB    R0
000154  105000                          CLRB    R0
000156  006300                          ASL     R0
000160  010003                          MOV     R0,R3                   ; *,SIZE
000162  010361  000026          4$:     MOV     R3,26(R1)               ; SIZE,*                        4408
000166  005737  000112'                 TST     MX2                     ;                               4410
000172  002404                          BLT     5$
000174  013700  000116'                 MOV     MAD2,R0                 ;                               4412
000200  010360  000026                  MOV     R3,26(R0)               ; SIZE,*
000204  022626                  5$:     CMP     (SP)+,(SP)+             ;                               4388
000206  000207                          RTS     PC                      ;                               4375

; Routine Size:  68 words,      Routine Base:  $CODE$ + 20054
; Maximum stack depth per invocation:  8 words
```

```
;  4415  1     GLOBAL routine FILL_BUFF : novalue =
;  4416  1
;  4417  1     !+
;  4418  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO LOAD THE I/O BUFFER DESCRIBED IN
;  4419  1     !       THE FIRST MSCP PACKET WITH THE APPROPRIATE DATA PATTERN.
;  4420  1     !
;  4421  1     !       THE DATA PATTERN TO BE SELECTED IS BASED ON THE FOLLOWING ALGORITHM:
;  4422  1     !
;  4423  1     !               IF THE OPERATOR DEFINED A DATA PATTERN
;  4424  1     !               THEN
;  4425  1     !                       SELECT IT
;  4426  1     !               ELSE
;  4427  1     !                       GET DATA PATTERN NUMBER FROM SW P-TABLE
;  4428  1     !                       IF DATA PATTERN NUMBER = 0
;  4429  1     !                       THEN
;  4430  1     !                               GET DATA PATTERN NUMBER FROM THE UNIT'S ENTRY
;  4431  1     !                               IN THE DATA PATTERN SEQUENCE TABLE (DPST)
;  4432  1     !
;  4433  1     !       NOTE THAT PATTERN # 1 CONSISTS OF RANDOM NUMBERS, AND PATTERNS # 17 -
;  4434  1     !       21 USE THE ACTUAL LBN OF THE WRITE REQUEST.
;  4435  1     !
;  4436  1     !       IMPLICIT INPUTS:
;  4437  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;  4438  1     !-
;  4439  1
;  4440  2         begin
;  4441  2
;  4442  2         local
;  4443  2             DP_NUM : word,                              ! DATA PATTERN NUMBER SELECTED
;  4444  2             DP_ADDR,                                    ! ADDR OF DATA PATTERN (LENGTH)
;  4445  2             IOB_ADDR,                                   ! I/O BUFFER ADDRESS (DESTINATION)
;  4446  2             SRC_ADDR,                                   ! WORKING SOURCE ADDRESS
;  4447  2             COUNT : word;                               ! NO. OF WORDS IN DATA PATTERN
;  4448  2
;  4449  3         if BIT_TST (SWP_FLAGS, SWF UDP)                 ! IF USER DEFINED A DATA PATTERN
;  4450  2         then
;  4451  2             DP_ADDR = SWP UCNT                          ! SELECT IT
;  4452  2         else
;  4453  3             begin
;  4454  3
;  4455  3             if .SWP_DPAT neq 0                          ! IF USER SELECTED A PRE-DEFINED DATA PATTERN
;  4456  3             then
;  4457  3                 DP_NUM = .SWP_DPAT                      ! SELECT IT
;  4458  3             else
;  4459  4                 begin
;  4460  4                 DP_NUM = .DPST [.L$LUN];                ! GET PATTERN NUMBER FROM SEQUENCE TABLE
;  4461  4                 DPST [.L$LUN] = .DPST [.L$LUN] + 1;     ! ADVANCE TO NEXT PATTERN NUMBER
;  4462  4
;  4463  4                 if .DPST [.L$LUN] gtru DP_CNT           ! CHECK FOR HIGH LIMIT
;  4464  4                 then
;  4465  4                     DPST [.L$LUN] = 1;
;  4466  4
;  4467  3                 end;
```

```
;  4468  3
;  4469  3          DP_ADDR = .DPA_TBL [.DP_NUM   1];              ! ADDRESS OF DATA PATTERN (COUNT)
;  4470  3
;  4471  3          if .DP_NUM gequ 17
;  4472  3          then
;  4473  3
;  4474  3              if .DP_NUM                                 ! CHECK MACRO (IF PATTERN 17, 19, OR 21)
;  4475  3              then
;  4476  3                  (.DP_ADDR + 2) = .MAD1 [LBN_L]         ! LOAD LBN INTO FIRST WORD OF PATTERN
;  4477  3              else
;  4478  3                  (.DP_ADDR + 4) = .MAD1 [LBN_L];        ! LOAD LBN INTO SECOND WORD OF PATTERN
;  4479  3
;  4480  2              end;
;  4481  2
;  4482  2          IOB_ADDR = .MAD1 [BUF_0];                      ! I/O BUFFER ADDRESS
;  4483  2          COUNT = ..DP_ADDR;                             ! NO. OF WORDS IN DATA PATTERN
;  4484  2          SRC_ADDR = .DP_ADDR + 2;                       ! START OF THE ACTUAL DATA PATTERN
;  4485  2
;  4486  2          incr N from 1 to ((.MAD1 [BC_LO] + 1) / 2) do  ! FOR EACH WORD IN THIS WRITE REQUEST
;  4487  3              begin
;  4488  3              .IOB_ADDR = ..SRC_ADDR;                    ! MOVE 1 WORD
;  4489  3              IOB_ADDR = .IOB_ADDR + 2;                  ! ADVANCE DESTINATION ADDRESS
;  4490  3              SRC_ADDR = .SRC_ADDR + 2;                  ! ADVANCE SOURCE ADDRESS
;  4491  3              COUNT = .COUNT - 1;                        ! DECREMENT COUNT
;  4492  3
;  4493  3              if .COUNT eql 0                            ! IF END OF DATA PATTERN
;  4494  3              then
;  4495  4                  begin
;  4496  4                  COUNT = ..DP_ADDR;                     ! REPEAT DATA PATTERN
;  4497  4                  SRC_ADDR = .DP_ADDR + 2;
;  4498  3                  end;
;  4499  3
;  4500  2              end;                                       ! WORD TRANSFER LOOP
;  4501  2
;  4502  1          end;                                           ! ROUTINE FILL_BUFF


                                        .SBTTL  FILL.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          FILL.BUFF::
                                        JSR     R1,$SAVE5                       ;                       4415
000004  005746                          TST     -(SP)
000006  032737  000100  000000G         BIT     #100,SWP.FLAGS                  ;                       4449
000014  001403                          BEQ     1$
000016  012701  000000G                 MOV     #SWP.UCNT,R1                    ; *,DP.ADDR             4451
000022  000443                          BR      5$                              ;                       4449
000024  013700  000000G          1$:    MOV     SWP.DPAT,R0                     ;                       4455
000030  001402                          BEQ     2$
000032  010002                          MOV     R0,R2                           ; *,DP.NUM              4457
000034  000414                          BR      3$                              ;                       4455
000036  013700  000000G          2$:    MOV     L$LUN,R0                        ;                       4460
000042  062700  000050'                 ADD     #DPST,R0
000046  005002                          CLR     R2                              ; DP.NUM
000050  151002                          BISB    (R0),R2                         ; *,DP.NUM
```

```
000052  105210                          INCB    (R0)                    ;                               4461
000054  121027  000025                  CMPB    (R0),#25                ;                               4463
000060  101402                          BLOS    3$
000062  112710  000001                  MOVB    #1,(R0)                 ;                               4465
000066  010200                  3$:     MOV     R2,R0                   ; DP.NUM,*                      4469
000070  006300                          ASL     R0
000072  016001  001074'                  MOV     DPA.TBL-2(R0),R1        ; *,DP.ADDR
000076  020227  000021                  CMP     R2,#21                  ; DP.NUM,*                      4471
000102  103413                          BLO     5$
000104  013700  000114'                  MOV     MAD1,R0                 ;                               4476
000110  006002                          ROR     R2                      ; DP.NUM                        4474
000112  103004                          BCC     4$
000114  016061  000046  000002          MOV     46(R0),2(R1)            ; *,*(DP.ADDR)                  4476
000122  000403                          BR      5$                      ;                               4474
000124  016061  000046  000004  4$:     MOV     46(R0),4(R1)            ; *,*(DP.ADDR)                  4478
000132  013700  000114'          5$:     MOV     MAD1,R0                 ;                               4482
000136  016004  000032                  MOV     32(R0),R4               ; *,IOB.ADDR
000142  011103                          MOV     (R1),R3                 ; DP.ADDR,COUNT                 4483
000144  012705  000002                  MOV     #2,R5                   ;                               4484
000150  060105                          ADD     R1,R5                   ; DP.ADDR,*
000152  010502                          MOV     R5,R2                   ; *,SRC.ADDR
000154  016046  000026                  MOV     26(R0),-(SP)            ;                               4486
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$:     MOV     (R2)+,(R4)+             ; SRC.ADDR,IOB.ADDR             4488
000204  005303                          DEC     R3                      ; COUNT                         4491
000206  001002                          BNE     7$                      ;                               4493
000210  011103                          MOV     (R1),R3                 ; DP.ADDR,COUNT                 4496
000212  010502                          MOV     R5,R2                   ; *,SRC.ADDR                    4497
000214  005200                  7$:     INC     R0                      ; N                             4486
000216  020066  000004                  CMP     R0,4(SP)                ; N,*
000222  003767                          BLE     6$
000224  062706  000006                  ADD     #6,SP                   ;                               4415
000230  000207                          RTS     PC

; Routine Size:  77 words,      Routine Base:  $CODE$ + 20264
; Maximum stack depth per invocation:  10 words


;   4503  1
;   4504  1
```

```
;    4505  1     GLOBAL ROUTINE PROC_RETPKT : NOVALUE =
;    4506  1
;    4507  1     !+
;    4508  1     !       THIS ROUTINE IS CALLED FROM THE MULTI_DRIVE "EXECUTIVE" AND DUP_COMMAND TO CHECK FOR
;    4509  1     !       AND PROCESS ANY RETURN PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER"
;    4510  1     !       PORTION OF THE PROGRAM. THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK
;    4511  1     !       BETWEEN THE TWO PROGRAM PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH
;    4512  1     !       REQUIRE PROCESSING.
;    4513  1     !
;    4514  1     !       UNDER THE MULTI-DRIVE TEST, RETURN PACKETS ORIGINATE FROM TWO SOURCES:
;    4515  1     !               1. MSCP          - THE MORE COMMON, DESCRIBING A COMPLETED I/O
;    4516  1     !                                  OPERATION.
;    4517  1     !               2. DUP           - THE LESS COMMON, DESCRIBING A PORTION OF I/O
;    4518  1     !                                  COMMUNICATIONS WITH THE CONTROLLER PROGRAM.
;    4519  1     !               3. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
;    4520  1     !                                  COMMAND TIMEOUT.
;    4521  1     !
;    4522  1
;    4523  1         while .IODQ_IN neq .IODQ_OUT do                         ! DO UNTIL I/O DONE QUEUE IS EMPTY
;    4524  2             begin
;    4525  2             RP_INDX = OUT_IODQ ();                              ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
;    4526  2             RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);         ! CALCULATE RETPKT ADDRESS
;    4527  3             if NOT (.RP_ADDR [CONID] eql CID_DUP)               ! if not DUP then
;    4528  2             then (SET_CPAR (.RP_ADDR [CTLR]););                 ! SET UP CURRENT CONTROLLER PARAMETERS
;    4529  2
;    4530  2             selectoneu .RP_ADDR [CONID] of                      ! CONNECTION ID INDICATES PACKET SOURCE
;    4531  2                 set
;    4532  2
;    4533  2                 [CID_MSCP] :        IO_RETPKT ();               ! DISK MSCP (I/O TRANSFER DONE)
;    4534  2                 [CID_DUP] :         DIO_RETPKT ();              ! DUP (I/O TRANSFER DONE)
;    4535  2                 [CID_DRIVER] :      DR_RETPKT ();               ! MESSAGE FROM "DRIVER"
;    4536  2
;    4537  2                 [otherwise] :       PRINTF (DBM12, .RP_ADDR [CONID]);!"CONN ID = XXXXX RECEIVED"
;    4538  2                 tes;
;    4539  2
;    4540  1             end;                                                ! UNITL I/O DONE QUEUE IS EMPTY


                                        .SBTTL  PROC.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                  PROC.RETPKT::
                                        MOV     R1,-(SP)                    ;                                       4505
000002  023737  000000G 000000G 1$:     CMP     IODQ.IN,IODQ.OUT            ;                                       4523
000010  001467                          BEQ     7$
000012  004737  000000G                 JSR     PC,OUT.IODQ                 ;                                       4525
000016  010037  000000G                 MOV     R0,RP.INDX
000022  010046                          MOV     R0,-(SP)                    ; RP.INDX,*                             4526
000024  012746  000054                  MOV     #54,-(SP)
000030  004737  000000G                 JSR     PC,$L$MUL
000034  062700  000000G                 ADD     #RETPKT,R0
000040  010037  000000G                 MOV     R0,RP.ADDR
000044  126027  000003  000002          CMPB    3(R0),#2                    ;                                       4527
000052  001406                          BEQ     2$
000054  116016  000002                  MOVB    2(R0),(SP)                  ;                                       4528
```

```
000060  042716  177760                          BIC     #177760,(SP)
000064  004737  000000G                         JSR     PC,SET.CPAR
000070  013700  000000G               2$:       MOV     RP.ADDR,R0              ;                       4530
000074  005001                                   CLR     R1
000076  156001  000003                          BISB    3(R0),R1
000102  005701                                   TST     R1                      ;                       4533
000104  001003                                   BNE     3$
000106  004737  000000V                         JSR     PC,IO.RETPKT
000112  000424                                   BR      6$                      ;                       4530
000114  020127  000002               3$:       CMP     R1,#2                   ;                       4534
000120  001003                                   BNE     4$
000122  004737  000000V                         JSR     PC,DIO.RETPKT
000126  000416                                   BR      6$                      ;                       4530
000130  020127  000003               4$:       CMP     R1,#3                   ;                       4535
000134  001003                                   BNE     5$
000136  004737  000000V                         JSR     PC,DR.RETPKT
000142  000410                                   BR      6$                      ;                       4530
000144  010116               5$:       MOV     R1,(SP)                 ;                       4537
000146  012746  000000G                         MOV     #DBM12,-(SP)
000152  012746  000002                          MOV     #2,-(SP)
000156  010600                                   MOV     SP,R0                   ; SP,*
000160  104417                                   TRAP    17
000162  022626                                   CMP     (SP)+,(SP)+
000164  022626               6$:       CMP     (SP)+,(SP)+             ;                       4524
000166  000705                                   BR      1$                      ;                       4523
000170  012601               7$:       MOV     (SP)+,R1                ;                       4505
000172  000207                                   RTS     PC

; Routine Size:  62 words,      Routine Base:  $CODE$ + 20516
; Maximum stack depth per invocation:  7 words
```

```
;    4541  1     !*
;    4542  1     GLOBAL ROUTINE DIO_RETPKT : NOVALUE *
;    4543  1
;    4544  1     !*
;    4545  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL DUP I/O TRANSFER
;    4546  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;    4547  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS.
;    4548  1     !
;    4549  1     !       IMPLICIT INPUTS:
;    4550  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    4551  1     !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    4552  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    4553  1     !               DUOFF - CST OFFSET FOR THE CURRENT UNIT
;    4554  1     !               L$LUN - CURRENT UNIT NUMBER
;    4555  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    4556  1     !
;    4557  1     !       IMPLICIT OUTPUTS
;    4558  1     !               CST_ADDR [.DUOFF * OF DBN, NODUPMEDIA]   IF THIS BIT SET NO DUP EXERCISER
;    4559  1     !
;    4560  1     !
;    4561  2     BEGIN
;    4562  2
;    4563  2     LOCAL   FLAG : BYTE INITIAL(BYTE(TRUE)),
;    4564  2             SUM2 : WORD,
;    4565  2             SUM : WORD;                                    ! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
;    4566  2     !PRINTX (DER18);
;    4567  2
;    4568  2     IF .RP_ADDR [STATUS] NEQU ST_SUC                       ! IF STATUS CODE INDICATES ERROR
;    4569  2     THEN
;    4570  3             BEGIN
;    4571  3             CST_ADDR [.DUOFF * OF_DBN, DUPERROR] * 1.      ! SET DUP ERROR FLAG
;    4572  3             HARD_ERROR ();
;    4573  3             IF .RP_ADDR [ENDCOD] EQLU (OP_ELP * OP_END) OR  ! IF ENDCODE IS EXECUTE LOCAL PROGRAM
;    4574  4                .RP_ADDR [ENDCOD] EQLU (OP_GDS * OP_END)     !             OR GET DUST STATUS
;    4575  4             THEN    BEGIN
;    4576  4                     CST_ADDR [.DUOFF * OF DBN, NODUPMEDIA] * 1;    ! TURN OFF DUP EXERCISR
;    4577  3                     END;
;    4578  3             END
;    4579  2     ELSE                                                   ! ELSE   I/O WAS SUCCESSFUL
;    4580  3         BEGIN
;    4581  3
;    4582  4         IF .RP_ADDR [ENDCOD] EQLU (OP_GDS * OP_END) ! IF ENDCODE IS GET DUST STATUS
;    4583  3         THEN
;    4584  4             BEGIN
;    4585  4             IF .RP_ADDR [9,11,1,0] EQL 1
;    4586  4                     THEN CST_ADDR [.DUOFF * OF_DBN, D_ACTIVE] * ACTIVE     ! CONTROLLER IN AN ACTIVE STAE
;    4587  4                     ELSE CST_ADDR [.DUOFF * OF_DBN, D_ACTIVE] * IDLE;      ! CONTROLLER IN AN IDLE STATE
;    4588  4             IF .RP_ADDR [9,9,1,0] NEQ 1 THEN       ! TEST TO SEE IF CONTROLLER LOCAL PROGRAMS(PG 18 OF DUP DOC)
;    4589  5                     BEGIN
;    4590  5                     HARD_ERROR ();
;    4591  5                     CST_ADDR [ DUOFF * OF_DBN, NODUPMEDIA] * 1;    ! TURN OFF DUP EXERCISR
;    4592  4                     END;
;    4593  3             END;
```

```
;    4594  3
;    4595  3
;    4596  3     IF (.RP_ADDR [ENDCOD] EQL (OP_RCD + OP_END)) AND
;    4597  3        (.DUPPKT [DUPTYPE] EQL 6) AND
;    4598  3        (.DUPPKT [DUPMESG] EQL 2) AND                    !IF IT IS A RECEIVE DBN COMMAND WITH TYPE 6 AND MESSAGE 2 THEN
;    4599  4        (.CST_ADDR [.DUOFF + OF_DBN, DUPWRITE] EQLU 1)       ! IF WRITE FLAG SET IN CST TABLE THEN COMPARE BLOCKS
;    4600  3             THEN DUP_COMPARE ();
;    4601  3
;    4602  2     END;                                                ! COMPARE THE FOLLOWING 512 BYTES
;    4603  2
;    4604  2     PUT RETPKT (.RP_INDX);
;    4605  1     END;                                                ! ROUTINE DIO_RETPKT

                                        .SBTTL  DIO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                    DIO.RETPKT::
                                        MOV     R1,-(SP)                        ;                                       4542
000002  112700  000001                  MOVB    #1,R0                           ; #,FLAG                                4561
000006  013701  000000G                 MOV     RP.ADDR,R1                      ;                                       4568
000012  005761  000016                  TST     16(R1)
000016  001435                          BEQ     2$
000020  013700  001156                  MOV     DUOFF,R0                        ;                                       4571
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                   ;                                       4572
000044  013700  000000G                 MOV     RP.ADDR,R0                      ;                                       4573
000050  126027  000014  000203          CMPB    14(R0),#203
000056  001404                          BEQ     1$
000060  126027  000014  000201          CMPB    14(R0),#201                     ;                                       4574
000066  001112                          BNE     6$
000070  013700  001156'          1$:     MOV     DUOFF,R0                        ;                                       4576
000074  006300                          ASL     R0
000076  063700  000000G                 ADD     CST.ADDR,R0
000102  052760  100000  000020          BIS     #100000,20(R0)
000110  000501                          BR      6$                              ;                                       4568
000112  126127  000014  000201  2$:     CMPB    14(R1),#201                     ;                                       4582
000120  001036                          BNE     5$
000122  013700  001156'                 MOV     DUOFF,R0                        ;                                       4586
000126  006300                          ASL     R0
000130  063700  000000G                 ADD     CST.ADDR,R0
000134  032761  004000  000022          BIT     #4000,22(R1)                    ;                                       4585
000142  001404                          BEQ     3$
000144  052760  020000  000020          BIS     #20000,20(R0)                   ;                                       4586
000152  000403                          BR      4$                              ;                                       4585
000154  042760  020000  000020  3$:     BIC     #20000,20(R0)                   ;                                       4587
000162  032761  001000  000022  4$:     BIT     #1000,22(R1)                    ;                                       4588
000170  001012                          BNE     5$
000172  004737  000000V                 JSR     PC,HARD.ERROR                   ;                                       4590
000176  013700  001156'                 MOV     DUOFF,R0                        ;                                       4591
000202  006300                          ASL     R0
000204  063700  000000G                 ADD     CST.ADDR,R0
000210  052760  100000  000020          BIS     #100000,20(R0)
```

```
000216  013700  000000G                5$:     MOV     RP.ADDR,R0              ;                       4596
000222  126027  000014  000205                CMPB    14(R0),#205
000230  001031                                BNE     6$
000232  013700  000000G                        MOV     DUPPKT,R0               ;                       4597
000236  042700  007777                        BIC     #7777,R0
000242  020027  060000                        CMP     R0,#60000
000246  001022                                BNE     6$
000250  013700  000000G                        MOV     DUPPKT,R0               ;                       4598
000254  042700  170000                        BIC     #170000,R0
000260  020027  000002                        CMP     R0,#2
000264  001013                                BNE     6$
000266  013700  001156                        MOV     DUOFF,R0                ;                       4599
000272  006300                                ASL     R0
000274  063700  000000G                        ADD     CST.ADDR,R0
000300  032760  010000  000020                BIT     #10000,20(R0)
000306  001402                                BEQ     6$
000310  004737  000000V                        JSR     PC,DUP.COMPARE          ;                       4600
000314  013746  000000G                6$:     MOV     RP.INDX,-(SP)           ;                       4604
000320  004737  000000G                        JSR     PC,PUT.RETPKT
000324  005726                                TST     (SP)+                   ;                       4561
000326  012601                                MOV     (SP)+,R1                ;                       4542
000330  000207                                RTS     PC

; Routine Size:  109 words,    Routine Base:  $CODE$ + 20712
; Maximum stack depth per invocation:  3 words


;   4606  1
```

```
;  4607  1     GLOBAL ROUTINE DUP_COMPARE : NOVALUE =
;  4608  1
;  4609  1
;  4610  1     !+
;  4611  1     !       THIS ROUTINE IS CALLED BY DIO_RETPKT WHEN THE RECEIVE DATA COMMAND
;  4612  1     !       IS BEING PROCESSED. THIS COMMAND COMPARES THE WRITTEN BUFFER WITH
;  4613  1     !       THE PATERN WORD GIVEN IN SEND DATA COMMAND.  FOR EVERY WORD COMPARED
;  4614  1     !       THE ROUTINE INCREMENTS THE TALLY TABLE.  IF THE COMPARE SHOWS AN
;  4615  1     !       ERROR. THE DBN HARD ERROR COUNTER WILL BE INCREMENTED AND THE
;  4616  1     !       THE DBN NUMBER AND BYTE COUNT WILL BE PRINTED.
;  4617  1     !
;  4618  1     !       IMPLICIT INPUTS:
;  4619  1     !               S_PATTERN       ! THE SAVED PATTERN WRITTEN TO THE DBN'S
;  4620  1     !               S_DUPPKT        ! THE POINTER FOR DUP BUFFER
;  4621  1     !               T_ADDR          ! THE ADDRESS OF THE TALLY TABLE FOR THIS UNIT
;  4622  1     !               CST_ADDR        ! THE ADDRESS OF PRESENT CONTROLLER STATUS TABLE
;  4623  1     !-
;  4624  2     BEGIN
;  4625  2
;  4626  2     OWN
;  4627  2             COUNT : WORD;
;  4628  2
;  4629  2     !PRINTX (DER19);
;  4630  2     S_DUPPKT = 0;
;  4631  2     INCR COUNT FROM 1 TO 256 DO     !INDEX PIONTER FOR DATA STORED IN MSCP ENV PACKET
;  4632  3             BEGIN
;  4633  3             S_DUPPKT = .S_DUPPKT + 2;                       ! INITIALLY THIS SKIPS THE FIRST WORD OF DUPPKT
;  4634  3             IF .(DUPPKT + .S_DUPPKT) NEQ .S_PATTERN THEN    !IF THE CONTENTS OF DBN DOESN'T EQUAL PATTERN
;  4635  4                     BEGIN
;  4636  4                     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;       ! SET DUP ERROR FLAG
;  4637  4                     ERRHRD (46, EH_10, EMS_22);             !LIST ERROR
;  4638  4                     EXITLOOP;
;  4639  3                     END;
;  4640  2             END;                                    !GO THROUGH ALL DBN WORDS
;  4641  1     END;                                            !END ROUTINE DUP-COMPARE


001206                                  .PSECT  $GGG$,  RO
001206                          COUNT:  .BLKW   1


                                        .SBTTL  DUP.COMPARE MULTI-DRIVE TEST ROUTINES
021244                                  .PSECT  $CODE$,  RO

000000  010146                  DUP.COMPARE::
                                        MOV     R1,-(SP)                        ;                               4607
000002  005037  000000G                 CLR     S.DUPPKT                        ;                               4630
000006  012701  000400                  MOV     #400,R1                         ; #.COUNT                       4631
000012  062737  000002  000000G 1$:     ADD     #2,S.DUPPKT                     ;                               4633
000020  013700  000000G                 MOV     S.DUPPKT,R0                     ;                               4634
000024  026037  000000G 000000G         CMP     DUPPKT(R0),S.PATTERN
000032  001415                          BEQ     2$
```

```
000034  013700  001156                          MOV     DUOFF,R0                        ;                       4636
000040  006300                                  ASL     R0
000042  063700  000000G                         ADD     CST.ADDR,R0
000046  052760  040000  000020                  BIS     #40000,20(R0)
000054  104456                                  TRAP    56                              ;                       4637
000056  000056                                  .WORD   56
000060  000000G                                 .WORD   EH.10
000062  000000G                                 .WORD   EMS.22
000064  000402                                  BR      3$                              ;                       4635
000066  005301                          2$:     DEC     R1                              ; COUNT                 4631
000070  001350                                  BNE     1$
000072  012601                          3$:     MOV     (SP)+,R1                        ;                       4607
000074  000207                                  RTS     PC

; Routine Size:  31 words,      Routine Base:  $CODE$ + 21244
; Maximum stack depth per invocation:  3 words


;   4642  1
;   4643  1
;   4644  1
```

```
;  4645  1     GLOBAL routine IO_RETPKT : novalue =
;  4646  1
;  4647  1     !+
;  4648  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
;  4649  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;  4650  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
;  4651  1     !       PERFORMING HOST WRITE-COMPARES IF REQUIRED.
;  4652  1     !
;  4653  1     !       IMPLICIT INPUTS:
;  4654  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  4655  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  4656  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  4657  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  4658  1     !               L$LUN - CURRENT UNIT NUMBER
;  4659  1     !-
;  4660  1
;  4661  2         begin
;  4662  2
;  4663  2         local
;  4664  2             FLAG : byte initial (byte (TRUE));
;  4665  2
;  4666  2         FSET_UPAR ();                           ! FIND UNIT'S ENTRY IS CST AND SET UP UNIT-RELATED DATA
;  4667  2         ST_CODE = .RP_ADDR [STSCOD];            ! GET STATUS CODE FROM RETPKT
;  4668  2         SB_CODE = .RP_ADDR [SUBCOD];            ! GET SUB-CODE, IF ANY
;  4669  2
;  4670  3         if (.ST_CODE neq ST_SUC)                ! IF STATUS CODE INDICATES ERROR
;  4671  2         then
;  4672  3             begin
;  4673  3             HARD_ERROR ();                      ! UPDATE ERROR COUNT
;  4674  3             COMPARE_DATA = FALSE;               ! NO POINT IN DOING HOST COMPARES ON ERRORS
;  4675  3
;  4676  3             if (.ST_CODE neq ST_OFL) and        ! DROP UNIT IF ERROR COUNTS EXCEEDS LIMIT
;  4677  3                (.ST_CODE neq ST_AVL) and
;  4678  4                (.T_ADDR [ERR_HARD] gequ .SWP_ERROR)
;  4679  3             then
;  4680  4                 begin
;  4681  4                 DUR [.L$LUN] = DU_HERR;         ! LOAD REASON FOR DROPPING UNIT
;  4682  4                 DODU (.L$LUN);                  ! DROP UNIT
;  4683  3                 end;
;  4684  3
;  4685  3             end
;  4686  2         else                                    ! IF I/O WAS SUCCESSFUL
;  4687  3             begin
;  4688  3             UPD_IO_TALLY ();                    ! UPDATE I/O TALLY (STATISTICS)
;  4689  3
;  4690  4             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)
;  4691  3             then
;  4692  3                 COMPARE_DATA = TRUE;            ! HOST COMPARES MAY BE ALLOWED IF NO FURTHER ERRORS
;  4693  3
;  4694  3             if (BIT_TST (SWP_FLAGS, SWF_HWC)) and   ! IF HOST IS DOING WRITE-COMPARES
;  4695  4                (.COMPARE_DATA)
;  4696  3             then
;  4697  3                 FLAG = HOST_WRT_CHK ();         ! SAVE I/O PACKET OR DO WRITE-CHECK
```

```
;    4698  3
;    4699  2              end;
;    4700  2
;    4701  2          'f .FLAG                          ! IF FLAG IS STILL TRUE
;    4702  2          then
;    4703  2              SWEEP ();                     ! DEALLOCATE BUFFER(S) AND RETPKT(S)
;    4704  2
;    4705  2          QIO [.CCTLR] = .QIO [.CCTLR]   1; ! DECREMENT NO. OF OUTSTANDING QIOs
;    4706  1          end;                              ! ROUTINE IO_RETPKT


                                        .SBTTL  IO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          IO.RETPKT::
                                        JSR     R1,$SAVE2                       ;                       4645
000004  112701  000001                  MOVB    #1,R1                           ; *,FLAG                4661
000010  004737  000000V                  JSR     PC,FSET.UPAR                    ;                       4666
000014  013700  000000G                  MOV     RP.ADDR,R0                      ;                       4667
000020  116037  000016  000000G          MOVB    16(R0),ST.CODE
000026  042737  177740  000000G          BIC     #177740,ST.CODE
000034  016002  000016                  MOV     16(R0),R2                       ;                       4668
000040  006202                          ASR     R2
000042  006202                          ASR     R2
000044  006202                          ASR     R2
000046  006202                          ASR     R2
000050  006202                          ASR     R2
000052  042702  174000                  BIC     #174000,R2
000056  010237  000000G                  MOV     R2,SB.CODE
000062  005737  000000G                  TST     ST.CODE                         ;                       4670
000066  001431                          BEQ     1$
000070  004737  000000V                  JSR     PC,HARD.ERROR                   ;                       4673
000074  105037  001170'                  CLRB    COMPARE.DATA                    ;                       4674
000100  023727  000000G  000003         CMP     ST.CODE,#3                      ;                       4676
000106  001447                          BEQ     3$
000110  023727  000000G  000004         CMP     ST.CODE,#4                      ;                       4677
000116  001443                          BEQ     3$
000120  013700  000000G                  MOV     T.ADDR,R0                       ;                       4678
000124  026037  000014  000000G          CMP     14(R0),SWP.ERROR
000132  103435                          BLO     3$
000134  013700  000000G                  MOV     L$LUN,R0                        ;                       4681
000140  112760  000004  000000G          MOVB    #4,DUR(R0)
000146  104451                          TRAP    51                              ;                       4682
000150  000426                          BR      3$                              ;                       4670
000152  004737  000000V          1$:     JSR     PC,UPD.IO.TALLY                 ;                       4688
000156  013700  000000G                  MOV     RP.ADDR,R0                      ;                       4690
000162  126027  000014  000242         CMPB    14(R0),#242
000170  001003                          BNE     2$
000172  112737  000001  001170'         MOVB    #1,COMPARE.DATA                 ;                       4692
000200  032737  000040  000000G  2$:     BIT     #40,SWP.FLAGS                   ;                       4694
000206  001407                          BEQ     3$
000210  032737  000001  001170'         BIT     #1,COMPARE.DATA                 ;                       4695
000216  001403                          BEQ     3$
000220  004737  000000V                  JSR     PC,HOST.WRT.CHK                 ;                       4697
000224  110001                          MOVB    R0,R1                           ; *,FLAG
```

```
000226  006001                          3$:     ROR     R1                              ; FLAG                          4701
000230  103002                                  BCC     4$
000232  004737  000000V                        JSR     PC,SWEEP                        ;                               4703
000236  013700  000000G                4$:     MOV     CCTLR,R0                        ;                               4705
000242  105360  000000G                        DECB    QIO(R0)
000246  000207                                RTS     PC                              ;                               4645

; Routine Size:  84 words,      Routine Base:  $CODE$ + 21342
; Maximum stack depth per invocation:  5 words
```

```
;   4707  1     GLOBAL routine FSET_UPAR : novalue =
;   4708  1
;   4709  1     !+
;   4710  1     !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
;   4711  1     !       CONTROLLER STATUS TABLE (CST) FOR THE DISK ADDRESS WHICH IS
;   4712  1     !       CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
;   4713  1     !       CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT-RELATED
;   4714  1     !       DATA PARAMETERS.
;   4715  1     !
;   4716  1     !       IMPLICIT INPUTS:
;   4717  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;   4718  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;   4719  1     !
;   4720  1
;   4721  2         begin
;   4722  2
;   4723  2         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR   1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;   4724  2
;   4725  2             if .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .RP_ADDR [DISK]        ! IF RETPKT UNIT MATCHES CST ENTRY
;   4726  2             then
;   4727  3                 begin
;   4728  3                 SET_UPAR (.OFFSET);                                                ! SET UP UNIT-RELATED DATA
;   4729  3                 return;                                                            ! DONE
;   4730  2                 end;
;   4731  2
;   4732  2         PRINTF (DBM19, .RP_ADDR [DISK], .CCTLR);                                   ! "CAN'T FIND DISK XXX IN CST X'
;   4733  1         end;                                                                       ! ROUTINE FSET_UPAR
```

```
                        .SBTTL  FSET.UPAR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G FSET.UPAR::
                        JSR     R1,$SAVE4                       ;                       4707
000004  012702  000003          MOV     #3,R2                   ; *,OFFSET              4723
000010  010201          1$:     MOV     R2,R1                   ; OFFSET,*              4725
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,*              4728
000044  004737  000000G         JSR     PC,SET.UPAR
000050  005726                  TST     (SP)+                   ;                       4729
000052  000207                  RTS     PC                      ;                       4727
000054  062702  000012  2$:     ADD     #12,R2                  ; *,OFFSET              4723
000060  020227  000041          CMP     R2,#41                  ; OFFSET,*
000064  003751                  BLE     1$
000066  013746  000000G         MOV     CCTLR,-(SP)             ;                       4732
000072  013700  000000G         MOV     RP.ADDR,R0
000076  016046  000010          MOV     10(R0),-(SP)
000102  012746  000000G         MOV     #DBM19,-(SP)
```

K15


```
000106  012746  000003                    MOV     #3,-(SP)
000112  010600                            MOV     SP,R0                   ; SP,*
000114  104417                            TRAP    17
000116  062706  000010                    ADD     #10,SP                  ;                       4721
000122  000207                            RTS     PC                      ;                       4707

; Routine Size:  42 words,      Routine Base:  $CODE$ + 21612
; Maximum stack depth per invocation:  11 words
```

```
;    4734  1     GLOBAL routine HARD_ERROR : novalue =
;    4735  1
;    4736  1     !+
;    4737  1     !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO INCREMENT THE HARD
;    4738  1     !       ERROR STATISTIC FIELD FOR THE CURRENT UNIT. IF THE HARD ERROR COUNT
;    4739  1     !       HAS EXCEEDED THE OPERATOR-SPECIFIED LIMIT, THEN THE UNIT IS DROPPED
;    4740  1     !       FROM TESTING.
;    4741  1     !
;    4742  1     !       IMPLICIT INPUTS:
;    4743  1     !               L$LUN - CURRENT UNIT NUMBER
;    4744  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    4745  1     !               CUOFF - CST OFFSET FOR CURRENT UNIT
;    4746  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    4747  1     !
;    4748  1
;    4749  2         begin
;    4750  2         T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;                     ! INCREMENT UNIT'S HARD ERROR COUNT
;    4751  2     if .RP_ADDR [CONID] EQL CID_MSCP                                    !FOR MSCP ERRORS   ZZZ
;    4752  2     THEN                                                                !                  ZZZ
;    4753  2
;    4754  2         selectoneu .ST_CODE of
;    4755  2             set
;    4756  2
;    4757  2             [ST SUC].       if .SB_CODE neq 0                           ! SUCCESS WITH NON-ZERO SUB CODE
;    4758  2                             then
;    4759  3                                 begin
;    4760  3
;    4761  3                                 if .SB_CODE eql 4
;    4762  3                                 then
;    4763  3                                     T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1
;    4764  3                                 else
;    4765  3                                     T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD HST] + 1;
;    4766  3
;    4767  3                                 if .APT_MODE
;    4768  3                                 then
;    4769  3                                     ERR_HRD_RTNE_APT (44)
;    4770  3                                 else
;    4771  3                                     ERR_HRD_RTNE (44);
;    4772  3
;    4773  2                                 end;
;    4774  2
;    4775  3                 [ST_CMD]:       begin                                   ! INVALID COMMAND
;    4776  3                 T ADDR [ERR HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    4777  3
;    4778  3                 if .APT_MODE
;    4779  3                 then
;    4780  3                     ERR_HRD_RTNE_APT (31)
;    4781  3                 else
;    4782  3                     ERR_HRD_RTNE (31);
;    4783  3
;    4784  2                 end;
;    4785  2
;    4786  3                 [ST_ABO]:       begin                                   ! COMMMAND ABORTED
```

M15


```
;  4787  3                          T_ADDR [ERR_HRD_DRV] = .T ADDR [ERR HRD_DRV] + 1;
;  4788  3
;  4789  3                          if .APT_MODE
;  4790  3                          then
;  4791  3                              ERR_HRD_RTNE APT (32)
;  4792  3                          else
;  4793  3                              ERR_HRD_RTNE (32);
;  4794  3
;  4795  2                          end;
;  4796  2
;  4797  3              [ST_OFL] :  begin                                           ! OFFLINE
;  4798  3                          T ADDR [ERR_HRD_DRV] = .T ADDR [ERR_HRD DRV] + 1;
;  4799  3
;  4800  3                          if .APT MODE
;  4801  3                          then
;  4802  4                              begin
;  4803  4                              .MAIL_BOX_TESTNUM = 1;
;  4804  4                              .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF DATA. D DISK NUM];
;  4805  3                              end;
;  4806  3
;  4807  3                          ERRDF (18, EGD_18, EMS_18);                     !
;  4808  3                          DUR [.L$LUN] = DU_DFATAL;                       ! DEVICE FATAL ERROR
;  4809  3                          DODU (.L$LUN);                                  ! DROP UNIT
;  4810  2                          end;
;  4811  2
;  4812  3              [ST_AVL]:   begin                                           ! WENT TO AVAILABLE STATE
;  4813  3                          T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR HRD DRV] + 1;
;  4814  3
;  4815  3                          if .APT_MODE
;  4816  3                          then
;  4817  4                              begin
;  4818  4                              .MAIL_BOX_TESTNUM1 = 1;
;  4819  4                              .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D DISK NUM1;
;  4820  3                              end;
;  4821  3
;  4822  3                          ERRDF (24, EGD_18, EMS_24);                     !
;  4823  3                          DUR [.L$LUN] = DU_DFATAL;                       ! DEVICE FATAL ERROR
;  4824  3                          DODU (.L$LUN);                                  ! DROP UNIT
;  4825  2                          end;
;  4826  2
;  4827  3              [ST MFE]:   begin                                           ! MEDIA FORMAT ERROR
;  4828  3                          T ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1;
;  4829  3
;  4830  3                          if .APT_MODE
;  4831  3                          then
;  4832  3                              ERR_HRD_RTNE_APT (35)
;  4833  3                          else
;  4834  3                              ERR_HRD RTNE (35);
;  4835  3
;  4836  2                          end;
;  4837  2
;  4838  3              [ST_WPT]:   begin                                           ! DEVICE WRITE PROTECTED
;  4839  3
```

```
;  4840  3                          if .SB_CODE eql 128
;  4841  3                          then
;  4842  3                              T_ADDR [ERR_HRD HST] = .T ADDR [ERR HRD HST] + 1
;  4843  3                          else
;  4844  3                              T ADDR [ERR_HRD DRV] = .T ADDR [ERR HRD DRV] + 1;
;  4845  3
;  4846  3                          if .APT MODE
;  4847  3                          then
;  4848  3                              ERR_HRD_RTNE APT (36)
;  4849  3                          else
;  4850  3                              ERR_HRD_RTNE (36);
;  4851  3
;  4852  2                          end;
;  4853  2
;  4854  3              [ST CMP]:   begin                                          ! COMPARE ERROR
;  4855  3                          T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  4856  3
;  4857  3                          if .APT_MODE
;  4858  3                          then
;  4859  3                              ERR_HRD_RTNE_APT (37)
;  4860  3                          else
;  4861  3                              ERR_HRD_RTNE (37);
;  4862  3
;  4863  2                          end;
;  4864  2
;  4865  3              [ST_DAT]:   begin                                          ! DATA ERROR
;  4866  3
;  4867  3                          if .SB_CODE eql 2
;  4868  3                          then
;  4869  3                              T_ADDR [ERR HRD_SEK] = .T ADDR [ERR_HRD SEK] + 1
;  4870  3                          else
;  4871  3                              T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  4872  3
;  4873  3                          if (.SB_CODE eql 0) and
;  4874  3                             (not .FORCED_ERROR) and
;  4875  4                             (BIT_TST (SWP_FLAGS, SWF_FER))
;  4876  3                          then
;  4877  4                              begin
;  4878  4                              FORCED_ERROR = TRUE;                       ! BLOCK WITH "FORCED ERROR' FOUND
;  4879  4                              FER_LBN = .RP_ADDR [LBN_LO];               !
;  4880  4                              FER_BC = .RP_ADDR [CBCNT LO];              !
;  4881  3                              end;
;  4882  3
;  4883  3                          if .APT_MODE
;  4884  3                          then
;  4885  3                              ERR_HRD_RTNE_APT (38)
;  4886  3                          else
;  4887  3                              ERR_HRD_RTNE (38);
;  4888  3
;  4889  2                          end;
;  4890  2
;  4891  3              [ST_HST]:   begin                                          ! HOST ACCESS ERROR
;  4892  3                          T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
```

```
;  4893  3
;  4894  3                          if .APT_MODE
;  4895  3                          then
;  4896  3                              ERR_HRD_RTNE_APT (39)
;  4897  3                          else
;  4898  3                              ERR_HRD_RTNE (39);
;  4899  3
;  4900  2                          end;
;  4901  2
;  4902  3                  [ST_CNT]:       begin                                   ! CONTROLLER ERROR
;  4903  3                  T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4904  3
;  4905  3                          if .APT_MODE
;  4906  3                          then
;  4907  3                              ERR_HRD_RTNE_APT (40)
;  4908  3                          else
;  4909  3                              ERR_HRD_RTNE (40);
;  4910  3
;  4911  2                          end;
;  4912  2
;  4913  3                  [ST_DRV]:       begin                                   ! DRIVE ERROR
;  4914  3
;  4915  3                          if .SB_CODE eql 3
;  4916  3                          then
;  4917  3                              T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;  4918  3                          else
;  4919  3                              T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4920  3
;  4921  3                          if .APT_MODE
;  4922  3                          then
;  4923  3                              ERR_HRD_RTNE_APT (41)
;  4924  3                          else
;  4925  3                              ERR_HRD_RTNE (41);
;  4926  3
;  4927  2                          end;
;  4928  2
;  4929  3                  [ST_DIA]:       begin                                   ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  4930  3                  T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4931  3
;  4932  3                          if .APT_MODE
;  4933  3                          then
;  4934  3                              ERR_HRD_RTNE_APT (43)
;  4935  3                          else
;  4936  3                              ERR_HRD_RTNE (43);
;  4937  3
;  4938  2                          end;
;  4939  2
;  4940  3                  [otherwise]:    begin                                   ! PRINT STATUS CODE IF NO MATCH
;  4941  3                  C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  4942  3
;  4943  3                          if .APT_MODE
;  4944  3                          then
;  4945  3                              ERR_HRD_RTNE_APT (45)
```

```
;  4946  3                          else
;  4947  3                              ERR_HRD RTNE (45);
;  4948  3
;  4949  2                          end;
;  4950  2
;  4951  2          tes;
;  4952  2
;  4953  2      if .RP_ADDR [CONID] EQL CID_DUP                                  !FOR DUP ERRORS                          ZZZ
;  4954  2      OR .D FAIL EQL 1                                                 !EVEN IF UNRECOGNIZABLE AS SUCH          ZZZ
;  4955  2      THEN                                                             !                                        ZZZ
;  4956  2
;  4957  2          selectoneu .RP_ADDR [STSCOD] of
;  4958  2                  SET
;  4959  3                  [%o 0 ]             : begin                          ! if status code succesful
;  4960  3                                        if .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END) and ! IF ENDCODE IS GET DUST STATUS
;  4961  3                                           .RP_ADDR [9,9,1,0] NEQ 1                     ! TEST TO SEE IF CONTROLLER LOCAL PR
OGRAMS
;  4962  3                                        then                                            ! (PG 18 OF DUP DOC)
;  4963  4                                                BEGIN
;  4964  4                                                ERR_HRD_RTNE (60);                      !UNABLE TO LOAD LOCAL CONTROLLER DUP
 MEDIA !ZZZ
;  4965  4                                                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4966  4                                                END
;  4967  3                                        else
;  4968  4                                                begin
;  4969  5                                                if (.DUPPKT [DUPTYPE] eql 5)            ! if fatal error
;  4970  4                                                then
;  4971  4      !ZZZ                                              begin                           !DON'T DROP DEVICE ON DUP ERROR
        ZZZ
;  4972  4      !ZZZ                                              DUR [.L$LUN] = DU_DFATAL;       !GIVE F.E. A CHANCE TO SEE ERRORS
        ZZZ
;  4973  4      !ZZZ                                              DODU (.L$LUN);                  ! FATAL DEVICE ERROR DROP UNIT);
        ZZZ
;  4974  4      !ZZZ                                              end;                            ! SET REASON FOR DROPPING UNIT
        ZZZ
;  4975  4                                                selectoneu .DUPPKT [DUPMESG] of
;  4976  4                                                SET
;  4977  5                                                [%o'1' ] : begin
;  4978  5                                                        ERR_HRD_RTNE (62);          ! illegal unit number    !ZZZ
;  4979  5                                                        T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  4980  4                                                        end;
;  4981  5                                                [%o'2' ] : begin
;  4982  5                                                        ERR_HRD_RTNE (63);          ! illegal relative or physical b
lock # !ZZZ
;  4983  5                                                        T_ADDR [ERR_HRD_HST] = .T ADDR [ERR_HRD_HST] + 1;
;  4984  4                                                        end;
;  4985  5                                                [%o'3' ] : begin
;  4986  5                                                        ERR_HRD_RTNE (64);          ! device error    !ZZZ
;  4987  5                                                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4988  4                                                        end;
;  4989  5                                                [%o'4' ] : begin
;  4990  5                                                        ERR_HRD_RTNE (65);          ! zero lenght message   !ZZZ
;  4991  5                                                        T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  4992  4                                                        end;
;  4993  5                                                [OTHERWISE] : begin
;  4994  5                                                        ERR_HRD_RTNE (66);          ! DUP UNKNOWN STATUS CODE   !ZZZ
;  4995  5                                                        C_ERR_TBL [.CCTLR, C ERR HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HR
D] + 1;
;  4996  4                                                        end;
;  4997  4                                                tes;
;  4998  3                          end;
```

```
;  4999  3
;  5000  2                          end;
;  5001  3                [%o 1 ]     : begin
;  5002  3                            ERR_HRD_RTNE (67);      ! INVALID COMMAND           !ZZZ
;  5003  3                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5004  2                            end;
;  5005  3                [%o 2 ]     : begin
;  5006  3                            ERR_HRD_RTNE (68);      ! NO REGION AVAILABLE       !ZZZ
;  5007  3                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5008  2                            end;
;  5009  3                [%o 3 ]     : begin
;  5010  3                            ERR_HRD_RTNE (69);          ! NO REGION SUITABLE        !ZZZ
;  5011  3                            T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  5012  2                            end;
;  5013  3                [%o'4 ]     : begin
;  5014  3                            ERR_HRD_RTNE (70);      ! PROGRAM NOT KNOWN         !ZZZ
;  5015  3                            T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  5016  2                            end;
;  5017  3                [%o 5 ]     : begin
;  5018  3                            ERR_HRD_RTNE (71);          ! LOAD FAILURE                !ZZZ
;  5019  3                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5020  2                            end;
;  5021  3                [%o'6' ]    : begin
;  5022  3                            ERR_HRD_RTNE (72);          ! STANDALONE                  !ZZZ
;  5023  3                            T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  5024  2                            end;
;  5025  3                [OTHERWISE] : begin
;  5026  3                            ERR_HRD_RTNE (73);          ! DUP UNKNOWN STATUS CODE   !ZZZ
;  5027  3                            C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  5028  2                            end;
;  5029  2                    TES;
;  5030  2
;  5031  1          end;                                                    ! ROUTINE HARD_ERROR

                                    .SBTTL  HARD.ERROR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G    HARD.ERROR::
                                    JSR     R1,$SAVE4                   ;                   4734
000004  013701  000000G             MOV     T.ADDR,R1                   ;                   4750
000010  005261  000014              INC     14(R1)
000014  013703  000000G             MOV     RP.ADDR,R3                  ;                   4751
000020  105763  000003              TSTB    3(R3)
000024  001171                      BNE     12$
000026  013702  000000G             MOV     ST.CODE,R2                  ;                   4754
000032  001027                      BNE     4$                          ;                   4757
000034  013704  000000G             MOV     SB.CODE,R4
000040  001563                      BEQ     12$
000042  012700  000050              MOV     #50,R0                      ;                   4763
000046  060100                      ADD     R1,R0
000050  020427  000004              CMP     R4,#4                       ;                   4761
000054  001002                      BNE     1$
000056  105210                      INCB    (R0)                        ;                   4763
000060  000402                      BR      2$                          ;                   4761
```

```
000062  105260  000001            1$:   INCB    1(R0)                       ;                4765
000066  032737  000001  001162    2$:   BIT     #1,APT.MODE                 ;                4767
000074  001403                          BEQ     3$
000076  012746  000054                  MOV     #54,(SP)                    ;                4769
000102  000557                          BR      14$
000104  012746  000054            3$:   MOV     #54,(SP)                    ;                4771
000110  000557                          BR      16$
000112  020227  000001            4$:   CMP     R2,#1                       ;                4775
000116  001014                          BNE     6$
000120  105261  000051                  INCB    51(R1)                      ;                4776
000124  032737  000001  001162'        BIT     #1,APT.MODE                 ;                4778
000132  001403                          BEQ     5$
000134  012746  000037                  MOV     #37,-(SP)                   ;                4780
000140  000570                          BR      20$
000142  012746  000037            5$:   MOV     #37,(SP)                    ;                4782
000146  000570                          BR      22$
000150  020227  000002            6$:   CMP     R2,#2                       ;                4786
000154  001014                          BNE     8$
000156  105261  000050                  INCB    50(R1)                      ;                4787
000162  032737  000001  001162'        BIT     #1,APT.MODE                 ;                4789
000170  001403                          BEQ     7$
000172  012746  000040                  MOV     #40,(SP)                    ;                4791
000176  000570                          BR      24$
000200  012746  000040            7$:   MOV     #40,-(SP)                   ;                4793
000204  000570                          BR      26$
000206  020227  000003            8$:   CMP     R2,#3                       ;                4797
000212  001036                          BNE     10$
000214  105261  000050                  INCB    50(R1)                      ;                4798
000220  032737  000001  001162'        BIT     #1,APT.MODE                 ;                4800
000226  001415                          BEQ     9$
000230  012777  000001  001164'        MOV     #1,@MAIL.BOX.TESTNUM        ;                4803
000236  013700  000000G                 MOV     CUOFF,R0                    ;                4804
000242  006300                          ASL     R0
000244  063700  000000G                 ADD     CST.ADDR,R0
000250  111077  001166'                 MOVB    (R0),@MAIL.BOX.SUBTST
000254  042777  177760  001166'        BIC     #177760,@MAIL.BOX.SUBTST
000262  104455                    9$:   TRAP    55                          .                4807
000264  000022                          .WORD   22
000266  000000G                         .WORD   EGD.18
000270  000000G                         .WORD   EMS.18
000272  013700  000000G                 MOV     L$LUN,R0                    ;                4808
000276  112760  000005  000000G        MOVB    #5,DUR(R0)
000304  104451                          TRAP    51                          ;                4809
000306  000440                          BR      12$                         ;                4754
000310  020227  000004            10$:  CMP     R2,#4                       ;                4812
000314  001037                          BNE     13$
000316  105261  000050                  INCB    50(R1)                      ;                4813
000322  032737  000001  001162'        BIT     #1,APT.MODE                 ;                4815
000330  001415                          BEQ     11$
000332  012777  000001  001164'        MOV     #1,@MAIL.BOX.TESTNUM        ;                4818
000340  013700  000000G                 MOV     CUOFF,R0                    ;                4819
000344  006300                          ASL     R0
000346  063700  000000G                 ADD     CST.ADDR,R0
```

```
000352  111077  001166                        MOVB    (R0),@MAIL.BOX.SUBTST
000356  042777  177760  001166                BIC     #177760,@MAIL.BOX.SUBTST
000364  104455                          11$:   TRAP    55                              ;                       4822
000366  000030                                 .WORD   30
000370  000000G                                .WORD   EGD.18
000372  000000G                                .WORD   EMS.24
000374  013700  000000G                        MOV     L$LUN,R0                        ;                       4823
000400  112760  000005  000000G               MOVB    #5,DUR(R0)
000406  104451                                 TRAP    51                              ;                       4824
000410  000137  023132'                  12$:   JMP     51$                             ;                       4754
000414  020227  000005                  13$:   CMP     R2,#5                           ;                       4827
000420  001014                                 BNE     17$
000422  105261  000046                        INCB    46(R1)                          ;                       4828
000426  032737  000001  001162                BIT     #1,APT.MODE                     ;                       4830
000434  001403                                 BEQ     15$
000436  012746  000043                        MOV     #43,-(SP)                       ;                       4832
000442  000560                          14$:   BR      35$
000444  012746  000043                  15$:   MOV     #43,-(SP)                       ;                       4834
000450  000560                          16$:   BR      37$
000452  020227  000006                  17$:   CMP     R2,#6                           ;                       4838
000456  001025                                 BNE     23$
000460  012700  000050                        MOV     #50,R0                          ;                       4842
000464  060100                                 ADD     R1,R0
000466  023727  000000G 000200                CMP     SB.CODE,#200                    ;                       4840
000474  001003                                 BNE     18$
000476  105260  000001                        INCB    1(R0)                           ;                       4842
000502  000401                                 BR      19$                             ;                       4840
000504  105210                          18$:   INCB    (R0)                            ;                       4844
000506  032737  000001  001162'        19$:   BIT     #1,APT.MODE                     ;                       4846
000514  001403                                 BEQ     21$
000516  012746  000044                        MOV     #44,-(SP)                       ;                       4848
000522  000575                          20$:   BR      43$
000524  012746  000044                  21$:   MOV     #44,-(SP)                       ;                       4850
000530  000575                          22$:   BR      45$
000532  020227  000007                  23$:   CMP     R2,#7                           ;                       4854
000536  001014                                 BNE     27$
000540  105261  000047                        INCB    47(R1)                          ;                       4855
000544  032737  000001  001162                BIT     #1,APT.MODE                     ;                       4857
000552  001403                                 BEQ     25$
000554  012746  000045                        MOV     #45,-(SP)                       ;                       4859
000560  000575                          24$:   BR      47$
000562  012746  000045                  25$:   MOV     #45,-(SP)                       ;                       4861
000566  000577                          26$:   BR      49$
000570  020227  000010                  27$:   CMP     R2,#10                          ;                       4865
000574  001051                                 BNE     32$
000576  012700  000046                        MOV     #46,R0                          ;                       4869
000602  060100                                 ADD     R1,R0
000604  023727  000000G 000002                CMP     SB.CODE,#2                      ;                       4867
000612  001002                                 BNE     28$
000614  105210                                 INCB    (R0)                            ;                       4869
000616  000402                                 BR      29$                             ;                       4867
000620  105260  000001                  28$:   INCB    1(R0)                           ;                       4871
000624  005737  000000G                 29$:   TST     SB.CODE                         ;                       4873
```

```
000630  001021                          BNE     30$
000632  132737  000001  000000G         BITB    #1,FORCED.ERROR         ;                       4874
000640  001015                          BNE     30$
000642  032737  004000  000000G         BIT     #4000,SWP.FLAGS         ;                       4875
000650  001411                          BEQ     30$
000652  112737  00C001  000000G         MOVB    #1,FORCED.ERROR         ;                       4878
000660  016337  000050  000000G         MOV     50(R3),FER.LBN          ;                       4879
000666  016337  000044  000000G         MOV     44(R3),FER.BC           ;                       4880
000674  032737  000001  001162'  30$:   BIT     #1,APT.MODE             ;                       4883
000702  001403                          BEQ     31$
000704  012746  000046                  MOV     #46,-(SP)               ;                       4885
000710  000521                          BR      47$
000712  012746  000046          31$:    MOV     #46,-(SP)               ;                       4887
000716  000523                          BR      49$
000720  020227  000011          32$:    CMP     R2,#11                  ;                       4891
000724  001014                          BNE     34$
000726  105261  000051                  INCB    51(R1)                  ;                       4892
000732  032737  000001  001162'         BIT     #1,APT.MODE             ;                       4894
000740  001403                          BEQ     33$
000742  012746  000047                  MOV     #47,-(SP)               ;                       4896
000746  000502                          BR      47$
000750  012746  000047          33$:    MOV     #47,-(SP)               ;                       4898
000754  000504                          BR      49$
000756  020227  000012          34$:    CMP     R2,#12                  ;                       4902
000762  001014                          BNE     38$
000764  105261  000050                  INCB    50(R1)                  ;                       4903
000770  032737  000001  001162'         BIT     #1,APT.MODE             ;                       4905
000776  001403                          BEQ     36$
001000  012746  000050                  MOV     #50,-(SP)               ;                       4907
001004  000463                  35$:    BR      47$
001006  012746  000050          36$:    MOV     #50,-(SP)               ;                       4909
001012  000465                  37$:    BR      49$
001014  020227  000013          38$:    CMP     R2,#13                  ;                       4913
001020  001023                          BNE     42$
001022  023727  000000G 000003          CMP     SB.CODE,#3              ;                       4915
001030  001003                          BNE     39$
001032  105261  000046                  INCB    46(R1)                  ;                       4917
001036  000402                          BR      40$                     ;                       4915
001040  105261  000050          39$:    INCB    50(R1)                  ;                       4919
001044  032737  000001  001162'  40$:   BIT     #1,APT.MODE             ;                       4921
001052  001403                          BEQ     41$
001054  012746  000051                  MOV     #51,-(SP)               ;                       4923
001060  000435                          BR      47$
001062  012746  000051          41$:    MOV     #51,-(SP)               ;                       4925
001066  000437                          BR      49$
001070  020227  000037          42$:    CMP     R2,#37                  ;                       4929
001074  001014                          BNE     46$
001076  105261  000050                  INCB    50(R1)                  ;                       4930
001102  032737  000001  001162'         BIT     #1,APT.MODE             ;                       4932
001110  001403                          BEQ     44$
001112  012746  000053                  MOV     #53,-(SP)               ;                       4934
001116  000416                  43$:    BR      47$
001120  012746  000053          44$:    MOV     #53,-(SP)               ;                       4936
```

```
001124  000420                          45$:    BR      49$
001126  013700  000000G                 46$:    MOV     CCTLR,R0                ;                       4941
001132  006300                                  ASL     R0
001134  105260  00000* G                         INCB    C.ERR.TBL(R0)
001140  032737  00000* 001162                    BIT     #1,APT.MODE             ;                       4943
001146  001405                                  BEQ     48$
001150  012746  000055                          MOV     #55,-(SP)               ;                       4945
001154  004737  000000V                 47$:    JSR     PC,ERR.HRD.RTNE.APT
001160  000404                                  BR      50$                     ;                       4943
001162  012746  000055                 48$:    MOV     #55,-(SP)               ;                       4947
001166  004737  000000V                 49$:    JSR     PC,ERR.HRD.RTNE
001172  005726                          50$:    TST     (SP)+                   ;                       4940
001174  013700  000000G                 51$:    MOV     RP.ADDR,R0              ;                       4953
001200  126027  000003  000002                  CMPB    3(R0),#2
001206  001404                                  BEQ     52$
001210  123727  000000G 000001                  CMPB    D.FAIL,#1               ;                       4954
001216  001160                                  BNE     69$
001220  116001  000016                 52$:    MOVB    16(R0),R1               ;                       4957
001224  042701  177740                          BIC     #177740,R1
001230  001067                                  BNE     59$                     ;                       4959
001232  126027  000014  000201                  CMPB    14(R0),#201             ;                       4960
001240  001015                                  BNE     54$
001242  032760  001000  000022                  BIT     #1000,22(R0)            ;                       4961
001250  001011                                  BNE     54$
001252  012746  000074                          MOV     #74,-(SP)               ;                       4964
001256  004737  000000V                 53$:    JSR     PC,ERR.HRD.RTNE
001262  013700  000000G                          MOV     T.ADDR,R0               ;                       4965
001266  105260  000050                          INCB    50(R0)
001272  000531                                  BR      68$                     ;                       4963
001274  013700  000000G                 54$:    MOV     DUPPKT,R0               ;                       4969
001300  042700  007777                          BIC     #7777,R0
001304  020027  050000                          CMP     R0,#50000
001310  001123                                  BNE     69$
001312  013701  000000G                          MOV     DUPPKT,R1               ;                       4975
001316  042701  170000                          BIC     #170000,R1
001322  020127  000001                          CMP     R1,#1                   ;                       4977
001326  001003                                  BNE     55$
001330  012746  000076                          MOV     #76,-(SP)               ;                       4978
001334  000470                                  BR      65$
001336  020127  000002                 55$:    CMP     R1,#2                   ;                       4981
001342  001003                                  BNE     56$
001344  012746  000077                          MOV     #77,-(SP)               ;                       4982
001350  000462                                  BR      65$
001352  020127  000003                 56$:    CMP     R1,#3                   ;                       4985
001356  001003                                  BNE     57$
001360  012746  000100                          MOV     #100,-(SP)              ;                       4986
001364  000734                                  BR      53$
001366  020127  000004                 57$:    CMP     R1,#4                   ;                       4989
001372  001003                                  BNE     58$
001374  012746  000101                          MOV     #101,-(SP)              ;                       4990
001400  000446                                  BR      65$
001402  012746  000102                 58$:    MOV     #102,-(SP)              ;                       4994
001406  000454                                  BR      67$
```

```
001410  020127  000001          59$:    CMP     R1,#1                           ;                       5001
001414  001003                          BNE     60$
001416  012746  000103                  MOV     #103,-(SP)                      ;                       5002
001422  000715                          BR      53$
001424  020127  000002          60$:    CMP     R1,#2                           ;                       5005
001430  001003                          BNE     61$
001432  012746  000104                  MOV     #104,-(SP)                      ;                       5006
001436  000707                          BR      53$
001440  020127  000003          61$:    CMP     R1,#3                           ;                       5009
001444  001003                          BNE     62$
001446  012746  000105                  MOV     #105, (SP)                      ;                       5010
001452  000421                          BR      65$
001454  020127  000004          62$:    CMP     R1,#4                           ;                       5013
001460  001003                          BNE     63$
001462  012746  000106                  MOV     #106,-(SP)                      ;                       5014
001466  000413                          BR      65$
001470  020127  000005          63$:    CMP     R1,#5                           ;                       5017
001474  001003                          BNE     64$
001476  012746  000107                  MOV     #107,-(SP)                      ;                       5018
001502  000665                          BR      53$
001504  020127  000006          64$:    CMP     R1,#6                           ;                       5021
001510  001011                          BNE     66$
001512  012746  000110                  MOV     #110,-(SP)                      ;                       5022
001516  004737  000000V         65$:    JSR     PC,ERR.HRD.RTNE
001522  013700  000000G                 MOV     T.ADDR,R0                       ;                       5023
001526  105260  000051                  INCB    51(R0)
001532  000411                          BR      68$                             ;                       5021
001534  012746  000111          66$:    MOV     #111,-(SP)                      ;                       5026
001540  004737  000000V         67$:    JSR     PC,ERR.HRD.RTNE
001544  013700  000000G                 MOV     CCTLR,R0                        ;                       5027
001550  006300                          ASL     R0
001552  105260  000000G                 INCB    C.ERR.TBL(R0)
001556  005726                  68$:    TST     (SP)+                           ;                       5025
001560  000207                  69$:    RTS     PC                              ;                       4734

; Routine Size:  441 words,    Routine Base:  $CODE$ + 21736
; Maximum stack depth per invocation:  7 words


;   5032  1
```

```
;  5033  1     GLOBAL routine UPD_IO TALLY : novalue =
;  5034  1
;  5035  1     !+
;  5036  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;  5037  1     !       PACKETS WITH "SUCCESS" STATUS CODES. ITS PURPOSE IS TO UPDATE ALL THE
;  5038  1     !       APPROPRIATE STATISTICAL FIELDS FOR THE CURRENT UNIT. A CHECK IS ALSO
;  5039  1     !       MADE ON THE TOTAL NUMBER OF BYTES TRANSFERRED THUS FAR; IF THE
;  5040  1     !       OPERATOR-SPECIFIED LIMIT HAS BEEN REACHED, THEN THE UNIT IS DROPPED.
;  5041  1     !
;  5042  1     !       IMPLICIT INPUTS:
;  5043  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  5044  1     !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  5045  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;  5046  1     !               CUOFF - CST OFFSET FOR THE CURRENT UNIT
;  5047  1     !               L$LUN - CURRENT UNIT NUMBER
;  5048  1     !
;  5049  1
;  5050  2         begin
;  5051  2
;  5052  2         local
;  5053  2             THOUSANDS : word,
;  5054  2             MILLIONS : word;                                        ! TOTAL NO. OF BYTES XFERRED TO/FROM A UNIT
;  5055  2
;  5056  3         if .RP_ADDR [ENDCOD] eql (OP_RD or OP_END)                  ! IF ENDCODE IS READ
;  5057  2         then
;  5058  3             begin
;  5059  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1;     ! INCREMENT NO. OF READS AND ADD BYTE COUNT
;  5060  3             T_ADDR [BYTES_READ_LO] = .T_ADDR [BYTES_READ_LO] + .RP_ADDR [BCNT_LO];
;  5061  3             T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + .RP_ADDR [BCNT_LO];
;  5062  3             OVF_CHK (T_ADDR [TOT_READS_LO]);                        ! CHECK FOR FIELD OVERFLOW
;  5063  3             OVF_CHK (T_ADDR [BYTES_READ_LO]);                       !
;  5064  3             OVF_CHK (T_ADDR [TOT_BYT_RED_LO]);                      !
;  5065  3             end
;  5066  2         else
;  5067  2
;  5068  3             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)             ! IF ENDCODE IS WRITE
;  5069  2             then
;  5070  3                 begin
;  5071  3                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1;   ! INCREMENT NO. OF WRITES, ADD BYTE COUNT
;  5072  3                 T_ADDR [BYTES_WRIT_LO] = .T_ADDR [BYTES_WRIT_LO] + .RP_ADDR [BCNT_LO];
;  5073  3                 T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + .RP_ADDR [BCNT_LO];
;  5074  3                 OVF_CHK (T_ADDR [TOT_WRITES_LO]);                   ! CHECK FOR FIELD OVERFLOW
;  5075  3                 OVF_CHK (T_ADDR [BYTES_WRIT_LO]);                   !
;  5076  3                 OVF_CHK (T_ADDR [TOT_BYT_WRT_LO]);                  !
;  5077  2                 end;
;  5078  2
;  5079  2         if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) or
;  5080  3            (.RP_ADDR [ENDCOD] eql (OP_WRT or OP_END))
;  5081  2         then
;  5082  3             begin
;  5083  3             MILLIONS = .T_ADDR [MBYTES_READ] + .T_ADDR [MBYTES_WRIT];   ! TOTAL BYTES TRANSFERRED
;  5084  3             THOUSANDS = .T_ADDR [BYTES_READ_HI] + .T_ADDR [BYTES_WRIT HI];
;  5085  3
```

```
;    5086  3              if .THOUSANDS gequ 1000
;    5087  3              then
;    5088  4                  begin
;    5089  4                  MILLIONS = .MILLIONS + 1;                          ! COUNT THE LOWER OVERFLOW TOO!
;    5090  4                  THOUSANDS = .THOUSANDS   1000;
;    5091  3                  end;
;    5092  3
;    5093  3      !
;    5094  3      ! THIS ADDED BECAUSE IT WILL TAKE FOREVER TO TRANSFER ON THE ORDER OF A MEGABYTE TO A FLOPPY
;    5095  3      ! BUT IT IS A MUCH MORE REASONABLE MEASURE FOR THE RD51/52 WINCHESTER.  THE QUESTION NOW REFERS TO
;    5096  3      ! THE TOTAL DATA TRANSFER TO THE CONTROLLER AND THIS IS PRETTY CLOSE SINCE THE FLOPPIES GET
;    5097  3      ! ABOUT 1/1000 THE DATA THE HARD DISK(S) GET.
;    5098  3      !
;    5099  3
;    5100  3              if .SWP_XFER eql 0                                     ! IF THERE IS A TRANSFER LIMIT
;    5101  3              then
;    5102  4                  begin
;    5103  4
;    5104  4                  if .THOUSANDS gtru 50                              !ZZZ
;    5105  4                  then
;    5106  4                      EOP_FLAG = TRUE;                               ! SET END OF-PASS FLAG
;    5107  4
;    5108  4                  end
;    5109  3              else
;    5110  3
;    5111  3                  if .MILLIONS gequ .SWP_XFER                        ! IF TRANSFER LIMIT IS REACHED
;    5112  3                  then
;    5113  3                      EOP FLAG = TRUE;                               ! SET END-OF-PASS FLAG
;    5114  3
;    5115  2                  end;                                               ! IF UNIT IS STILL ALIVE
;    5116  2
;    5117  2          !.....................................................................
;    5118  2          !                                                                    .
;    5119  2          ! THE FOLLOWING IS ADDED TO MAKE THE RUN TIME ABOUT 1.5 MINUTES FOR A  .
;    5120  2          ! QUICK PASS IF ALL UNITS UNDER TEST ARE FLOPPIES.                     .
;    5121  2          !.....................................................................
;    5122  2
;    5123  2      !!ZZZ    IF .RD_COUNT EQL 0                                     !IF THERE ARE NO WINCHESTERS   ZZZ
;    5124  2      !!ZZZ       THEN                                                !                              ZZZ
;    5125  2      !!ZZZ         BEGIN                                             !                              ZZZ
;    5126  2      !!ZZZ         IF .THOUSANDS GTRU 44                             !IF ABOUT 1.5 MINUTES GONE BY  ZZZ
;    5127  2      !!ZZZ         THEN                                              !                              ZZZ
;    5128  2      !!ZZZ            EOP_FLAG = TRUE;                               !SET THE END OF PASS FLAG      ZZZ
;    5129  2      !!ZZZ         END;                                              !                              ZZZ
;    5130  2
;    5131  2
;    5132  2          ROUND_OUTPUT ();                                           ! ROUND TOTALS TO FIT PRINT POSITIONS
;    5133  1          end;                                                       ! ROUTINE UPD_IO_TALLY

                                        .SBTTL  UPD.IO.TALLY MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          UPD.IO.TALLY::
                                        JSR     R1,$SAVE2                      ;                                  5033
```

```
000004  013701  000000G                        MOV     RP.ADDR,R1              ;                               5056
000010  126127  000014  000241                 CMPB    14(R1),#241
000016  001027                                  BNE     1$
000020  013700  000000G                        MOV     T.ADDR,R0               ;                               5059
000024  005260  000016                          INC     16(R0)
000030  066110  000020                          ADD     20(R1),(R0)             ;                               5060
000034  066160  000020  000032                 ADD     20(R1),32(R0)           ;                               5061
000042  012746  000016                          MOV     #16,-(SP)               ;                               5062
000046  060016                                  ADD     R0,(SP)
000050  004737  000000V                        JSR     PC,OVF.CHK
000054  013716  000000G                        MOV     T.ADDR,(SP)             ;                               5063
000060  004737  000000V                        JSR     PC,OVF.CHK
000064  013716  000000G                        MOV     T.ADDR,(SP)             ;                               5064
000070  062716  000032                          ADD     #32,(SP)
000074  000435                                  BR      2$
000076  126127  000014  000242          1$:     CMPB    14(R1),#242             ;                               5068
000104  001034                                  BNE     3$
000106  013700  000000G                        MOV     T.ADDR,R0               ;                               5071
000112  005260  000024                          INC     24(R0)
000116  066160  000020  000006                 ADD     20(R1),6(R0)            ;                               5072
000124  066160  000020  000040                 ADD     20(R1),40(R0)           ;                               5073
000132  012746  000024                          MOV     #24,-(SP)               ;                               5074
000136  060016                                  ADD     R0,(SP)
000140  004737  000000V                        JSR     PC,OVF.CHK
000144  013716  000000G                        MOV     T.ADDR,(SP)             ;                               5075
000150  062716  000006                          ADD     #6,(SP)
000154  004737  000000V                        JSR     PC,OVF.CHK
000160  013716  000000G                        MOV     T.ADDR,(SP)             ;                               5076
000164  062716  000040                          ADD     #40,(SP)
000170  004737  000000V                2$:     JSR     PC,OVF.CHK
000174  005726                                  TST     (SP)+                   ;                               5070
000176  013700  000000G                3$:     MOV     RP.ADDR,R0              ;                               5079
000202  126027  000014  000241                 CMPB    14(R0),#241
000210  001404                                  BEQ     4$
000212  126027  000014  000242                 CMPB    14(R0),#242             ;                               5080
000220  001034                                  BNE     8$
000222  013700  000000G                4$:     MOV     T.ADDR,R0               ;                               5083
000226  016002  0000C4                          MOV     4(R0),R2                ; *,MILLIONS
000232  066002  000012                          ADD     12(R0),R2               ; *,MILLIONS
000236  016001  000002                          MOV     2(R0),R1                ; *,THOUSANDS                   5084
000242  066001  000010                          ADD     10(R0),R1               ; *,THOUSANDS
000246  020127  001750                          CMP     R1,#1750                ; THOUSANDS,*                   5086
000252  103403                                  BLO     5$
000254  005202                                  INC     R2                      ; MILLIONS                      5089
000256  162701  001750                          SUB     #1750,R1                ; *,THOUSANDS                   5090
000262  013700  000000G                5$:     MOV     SWP.XFER,R0             ;                               5100
000266  001004                                  BNE     6$
000270  020127  000062                          CMP     R1,#62                  ; THOUSANDS,*                   5104
000274  101406                                  BLOS    8$
000276  000402                                  BR      7$                      ;                               5106
000300  020200                        6$:     CMP     R2,R0                   ; MILLIONS,*                    5111
000302  103403                                  BLO     8$
000304  112737  000001  000000G        7$:     MOVB    #1,EOP.FLAG             ;                               5113
```

M16


```
000312  004737  C00000V                8$:     JSR     PC,ROUND.OUTPUT         ;                       5132
000316  000207                                  RTS     PC                      ;                       5033

; Routine Size:  104 words,     Routine Base:  $CODE$ + 23520
; Maximum stack depth per invocation:  5 words
```

```
;  5134  1     GLOBAL routine OVF_CHK (ADDR) : novalue =
;  5135  1     !+
;  5136  1     !       THIS ROUTINE IS CALLED FROM UPD_IO_TALLY TO CHECK FOR OVERFLOW IN
;  5137  1     !       CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
;  5138  1     !       LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
;  5139  1     !       EXCEEDING 1000. IF TRUE, THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
;  5140  1     !       THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IN INCREMENTED.
;  5141  1     !
;  5142  1     !       INPUTS:
;  5143  1     !               ADDR - ADDRESS OF THE BYTES_READ_LO OR BYTES_WRIT_LO FIELD FOR
;  5144  1     !                       THE CURRENT UNIT (SEE STATISTIC TABLE (TALLY) LAYOUT)
;  5145  1     !
;  5146  2         begin
;  5147  2
;  5148  2         while ..ADDR gequ 1000 do                   ! IF LO-ORDER OVERFLOW
;  5149  3             begin
;  5150  3             .ADDR = ..ADDR   1000;                  ! SUBTRACT 1000
;  5151  3             (.ADDR + 2) = .(.ADDR + 2) + 1;         ! INCR HI-ORDER
;  5152  2             end;
;  5153  2
;  5154  2         if .(.ADDR + 2) gequ 1000                   ! IF HI-ORDER OVERFLOW
;  5155  2         then
;  5156  3             begin
;  5157  3             (.ADDR + 2) = .(.ADDR + 2) - 1000;      ! SUBTRACT 1000
;  5158  3             (.ADDR + 4) = .(.ADDR + 4) + 1;         ! INCREMENT MBYTES
;  5159  2             end;
;  5160  2
;  5161  1         end;                                        ! ROUTINE OVF_CHK

                                        .SBTTL  OVF.CHK MULTI-DRIVE TEST ROUTINES
000000  010146                 OVF.CHK::
                                        MOV     R1,-(SP)                ;                       5134
000002  016600  000004                  MOV     4(SP),R0                ; ADDR,*                5148
000006  012701  000002                  MOV     #2,R1                   ;                       5151
000012  060001                          ADD     R0,R1
000014  021027  001750          1$:     CMP     (R0),#1750              ;                       5148
000020  103404                          BLO     2$
000022  162710  001750                  SUB     #1750,(R0)              ;                       5150
000026  005211                          INC     (R1)                    ;                       5151
000030  000771                          BR      1$                      ;                       5148
000032  021127  001750          2$:     CMP     (R1),#1750              ;                       5154
000036  103404                          BLO     3$
000040  162711  001750                  SUB     #1750,(R1)              ;                       5157
000044  005260  000004                  INC     4(R0)                   ;                       5158
000050  012601                  3$:     MOV     (SP)+,R1                ;                       5134
000052  000207                          RTS     PC

; Routine Size:  22 words,      Routine Base:  $CODE$ + 24040
; Maximum stack depth per invocation:  2 words
```

```
;  5162  1     GLOBAL routine ROUND_OUTPUT : novalue =
;  5163  1
;  5164  1     !+
;  5165  1     !       THIS ROUTINE ROUNDS THE TOTALS TO FIT PRINT POSITIONS.
;  5166  1     !
;  5167  1
;  5168  2         begin
;  5169  2
;  5170  2         if .T_ADDR [TOT_READS_HI] gtru 9999
;  5171  2         then
;  5172  3             begin
;  5173  3
;  5174  3             if .T_ADDR [TOT_READS_LO] lssu 999
;  5175  3             then
;  5176  4                 begin
;  5177  4                 T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI]   1;
;  5178  4                 T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1000;
;  5179  3                 end;
;  5180  3
;  5181  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO]   999;
;  5182  3             T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI]   9999;
;  5183  2             end;
;  5184  2
;  5185  2         if .T_ADDR [TOT_WRITES_HI] gtru 9999
;  5186  2         then
;  5187  3             begin
;  5188  3
;  5189  3             if .T_ADDR [TOT_WRITES_LO] lssu 999
;  5190  3             then
;  5191  4                 begin
;  5192  4                 T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI]   1;
;  5193  4                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1000;
;  5194  3                 end;
;  5195  3
;  5196  3             T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO]   999;
;  5197  3             T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI]   9999;
;  5198  2             end;
;  5199  2
;  5200  2         if .T_ADDR [MTOT_BYT_RED] gtru 999
;  5201  2         then
;  5202  3             begin
;  5203  3
;  5204  3             if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;  5205  3             then
;  5206  4                 begin
;  5207  4                 T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;  5208  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;  5209  3                 end;
;  5210  3
;  5211  3             if .T_ADDR [TOT_BYT_RED_LO] lssu 999
;  5212  3             then
;  5213  4                 begin
;  5214  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI]   1;
```

```
;    5215  4                  T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT BYT_RED LO] + 1000;
;    5216  4
;    5217  4                  if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;    5218  4                  then
;    5219  5                      begin
;    5220  5                      T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED]   1;
;    5221  5                      T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;    5222  4                      end;
;    5223  3                  end;
;    5224  3
;    5225  3              T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO]   999;
;    5226  3              T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI]   999;
;    5227  3              T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED]   999;
;    5228  2              end;
;    5229  2
;    5230  2          if .T_ADDR [MTOT_BYT_WRT] gtru 999
;    5231  2          then
;    5232  3              begin
;    5233  3
;    5234  3              if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;    5235  3              then
;    5236  4                  begin
;    5237  4                  T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    5238  4                  T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;    5239  3                  end;
;    5240  3
;    5241  3                  if .T_ADDR [TOT_BYT_WRT_LO] lssu 999
;    5242  3                  then
;    5243  4                      begin
;    5244  4                      T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI]   1;
;    5245  4                      T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + 1000;
;    5246  4
;    5247  4                      if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;    5248  4                      then
;    5249  5                          begin
;    5250  5                          T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    5251  5                          T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;    5252  4                          end;
;    5253  3                      end;
;    5254  3
;    5255  3                  T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] - 999;
;    5256  3                  T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 999;
;    5257  3                  T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 999;
;    5258  2                  end;
;    5259  2
;    5260  1          end;


                                        .SBTTL  ROUND.OUTPUT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          ROUND.OUTPUT::
                                        JSR     R1,$SAVE3                       ;                                       5162
000004  013700  000000G                 MOV     T.ADDR,R0                       ;                                       5170
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                  ;                5174
000030  060001                          ADD     R0,R1
000032  021127  001747                  CMP     (R1),#1747
000036  103003                          BHIS    1$
000040  005312                          DEC     (R2)                    ;                5177
000042  062711  001750                  ADD     #1750,(R1)              ;                5178
000046  162711  001747          1$:     SUB     #1747,(R1)              ;                5181
000052  162712  023417                  SUB     #23417,(R2)             ;                5182
000056  012702  000026          2$:     MOV     #26,R2                  ;                5185
000062  060002                          ADD     R0,R2
000064  021227  023417                  CMP     (R2),#23417
000070  101415                          BLOS    4$
000072  012701  000024                  MOV     #24,R1                  ;                5189
000076  060001                          ADD     R0,R1
000100  021127  001747                  CMP     (R1),#1747
000104  103003                          BHIS    3$
000106  005312                          DEC     (R2)                    ;                5192
000110  062711  001750                  ADD     #1750,(R1)              ;                5193
000114  162711  001747          3$:     SUB     #1747,(R1)              ;                5196
000120  162712  023417                  SUB     #23417,(R2)             ;                5197
000124  012703  000036          4$:     MOV     #36,R3                  ;                5200
000130  060003                          ADD     R0,R3
000132  021327  001747                  CMP     (R3),#1747
000136  101436                          BLOS    7$
000140  012701  000034                  MOV     #34,R1                  ;                5204
000144  060001                          ADD     R0,R1
000146  021127  001747                  CMP     (R1),#1747
000152  103003                          BHIS    5$
000154  005313                          DEC     (R3)                    ;                5207
000156  062711  001750                  ADD     #1750,(R1)              ;                5208
000162  012702  000032          5$:     MOV     #32,R2                  ;                5211
000166  060002                          ADD     R0,R2
000170  021227  001747                  CMP     (R2),#1747
000174  103011                          BHIS    6$
000176  005311                          DEC     (R1)                    ;                5214
000200  062712  001750                  ADD     #1750,(R2)              ;                5215
000204  021127  001747                  CMP     (R1),#1747              ;                5217
000210  103003                          BHIS    6$
000212  005313                          DEC     (R3)                    ;                5220
000214  062711  001750                  ADD     #1750,(R1)              ;                5221
000220  162712  001747          6$:     SUB     #1747,(R2)              ;                5225
000224  162711  001747                  SUB     #1747,(R1)              ;                5226
000230  162713  001747                  SUB     #1747,(R3)              ;                5227
000234  012702  000044          7$:     MOV     #44,R2                  ;                5230
000240  060002                          ADD     R0,R2
000242  021227  001747                  CMP     (R2),#1747
000246  101435                          BLOS    10$
000250  012701  000042                  MOV     #42,R1                  ;                5234
000254  060001                          ADD     R0,R1
000256  021127  001747                  CMP     (R1),#1747
```

```
000262  103003                          BHIS    8$
000264  005312                          DEC     (R2)                    ;                       5237
000266  062711  001750                  ADD     #1750,(R1)              ;                       5238
000272  062700  000040          8$:     ADD     #40,R0                  ;                       5241
000276  021027  001747                  CMP     (R0),#1747
000302  103011                          BHIS    9$
000304  005311                          DEC     (R1)                    ;                       5244
000306  062710  001750                  ADD     #1750,(R0)              ;                       5245
000312  021127  001747                  CMP     (R1),#1747              ;                       5247
000316  103003                          BHIS    9$
000320  005312                          DEC     (R2)                    ;                       5250
000322  062711  001750                  ADD     #1750,(R1)              ;                       5251
000326  162710  001747          9$:     SUB     #1747,(R0)              ;                       5255
000332  162711  001747                  SUB     #1747,(R1)              ;                       5256
000336  162712  001747                  SUB     #1747,(R2)              ;                       5257
000342  000207                  10$:    RTS     PC                      ;                       5162

; Routine Size:  114 words,    Routine Base:  $CODE$ + 24114
; Maximum stack depth per invocation:  5 words
```

```
;  5261  1     GLOBAL routine HOST_WRT_CHK =
;  5262  1
;  5263  1     !+
;  5264  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;  5265  1     !       PACKETS WITH "SUCCESS" STATUS CODES, BUT ONLY IF THE HOST WRITE-COMPARE
;  5266  1     !       OPTION WAS SELECTED BY THE OPERATOR.
;  5267  1     !
;  5268  1     !       IF THE CURRENT RETPKT BEING PROCESSED IS A WRITE FUNCTION, THEN THE
;  5269  1     !       PACKET INDEX (RP_INDX) IS SAVED IN THE CONTROLLER'S RETURN PACKET SAVE
;  5270  1     !       AREA (RP_SAVE). OTHERWISE, THE PACKET IS A READ, SO ITS ASSOCIATED
;  5271  1     !       WRITE PACKET IS REMOVED FROM THE SAVE AREA, AND A BYTE-BY-BYTE
;  5272  1     !       COMPARISON IS PERFORMED ON THE TWO I/O BUFFERS. ANY DIFFERENCES
;  5273  1     !       ENCOUNTERED RESULTS IN THE DECLARATION OF A HARD ERROR.
;  5274  1     !
;  5275  1     !       IMPLICIT INPUTS:
;  5276  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  5277  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;  5278  1     !-
;  5279  1
;  5280  2         begin
;  5281  2
;  5282  2         local
;  5283  2             BUFF1 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;  5284  2             BUFF2 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;  5285  2             BUFFW,                                            ! I/O BUFFER ADDRESS
;  5286  2             COUNT : word,                                     ! BYTE COUNT
;  5287  2             FLAG : byte initial (byte (TRUE)),
;  5288  2             index : signed word;
;  5289  2
;  5290  3         if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)           ! IF WRITE OPERATION
;  5291  2         then
;  5292  2             FLAG = FALSE                                      ! DON'T CALL SWEEP FROM IO_RETPKT
;  5293  2         else
;  5294  2
;  5295  2             if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) and
;  5296  3                ((index = RPS_REM ()) geq 0)                   ! IF ASSOCIATED WRITE PACKET IS FOUND
;  5297  2             then
;  5298  3                 begin
;  5299  3                 BUFFW = RETPKT [.index, BUFF_0];              ! ADDR OF ADDR OF WRITE I/O BUFFER
;  5300  3                 BUFF1 = ..BUFFW;                              ! ADDR OF WRITE I/O BUFFER
;  5301  3                 BUFF2 = .RP_ADDR [BUFF_0];                    ! ADDR OF READ I/O BUFFER
;  5302  3                 COUNT = .RP_ADDR [BCNT_LO];                   ! BYTE COUNT
;  5303  3
;  5304  3                 incr I from 1 to .COUNT do                    ! FOR EACH BYTE IN BUFFERS
;  5305  3
;  5306  3                     if .(.BUFF1)<0, 8, 0> eql .(.BUFF2)<0, 8, 0>   ! IF BYTES COMPARE O.K.
;  5307  3                     then
;  5308  4                         begin
;  5309  4                         BUFF1 = .BUFF1 + 1;                   ! ADVANCE WRITE BUFFER ADDR
;  5310  4                         BUFF2 = .BUFF2 + 1;                   ! ADVANCE READ BUFFER ADDR
;  5311  4                         end
;  5312  3                     else
;  5313  4                         begin                                 ! ELSE - COMPARE ERROR
```

```
;  5314  4                    T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;
;  5315  4                    T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  5316  4
;  5317  4                    if .APT_MODE
;  5318  4                    then
;  5319  4                        ERR_HRD_RTNE_APT (42)                   ! I/O REQUEST FAILED
;  5320  4                    else
;  5321  4                        ERR_HRD_RTNE (42);
;  5322  4
;  5323  4                    EMS_CMP (RETPKT + (.index * RP_LEN * 2));
;  5324  4
;  5325  4                    if .T_ADDR [ERR_HARD] gequ .SWP_ERROR
;  5326  4                    then
;  5327  5                        begin
;  5328  5                        DUR [.L$LUN] = DU_HERR;                 ! IF ERROR COUNT EXCEEDED
;  5329  5                        DODU (.L$LUN);                          ! DROP UNIT
;  5330  4                        end;
;  5331  4
;  5332  4                    exitloop;                                   ! NO NEED TO CONTINUE
;  5333  3                    end;                                        ! IF COMPARE ERROR
;  5334  3
;  5335  2                end;                                            ! IF ASSOCIATED WRITE RETPKT WAS FOUND
;  5336  2
;  5337  2            return (.FLAG);
;  5338  1            end;                                                ! ROUTINE HOST_WRT_CHK


                                .SBTTL  HOST.WRT.CHK MULTI-DRIVE TEST ROUTINES
000000  004137  000000G         HOST.WRT.CHK::
                                JSR     R1,$SAVE5                       ;                               5261
000004  005746                  TST     -(SP)
000006  112705  000001          MOVB    #1,R5                           ; *,FLAG                        5280
000012  013700  000000G         MOV     RP.ADDR,R0                      ;                               5290
000016  126027  000014  000242  CMPB    14(R0),#242
000024  001002                  BNE     1$
000026  105005                  CLRB    R5                              ; FLAG                          5292
000030  000511                  BR      8$                              ;                               5290
000032  126027  000014  000241  1$:     CMPB    14(R0),#241             ;                               5295
000040  001105                  BNE     8$
000042  004737  000000V         JSR     PC,RPS.REM                      ;                               5296
000046  005700                  TST     R0                              ; INDEX
000050  002501                  BLT     8$
000052  010046                  MOV     R0,(SP)                         ; INDEX,*                       5299
000054  012746  000054          MOV     #54,-(SP)
000060  004737  000000G         JSR     PC,BL$MUL
000064  010066  000004          MOV     R0,4(SP)
000070  062700  000024G         ADD     #RETPKT+24,R0                   ; *,BUFFW
000074  011001                  MOV     (R0),R1                         ; BUFFW,BUFF1                   5300
000076  013700  000000G         MOV     RP.ADDR,R0                      ;                               5301
000102  016002  000024          MOV     24(R0),R2                       ; *,BUFF2
000106  016004  000020          MOV     20(R0),R4                       ; *,COUNT                       5302
000112  005003                  CLR     R3                              ; I                             5304
000114  000453                  BR      6$
```

```
000116  121112                          2$:     CMPB    (R1),(R2)                       ; BUFF1,BUFF2                   5306
000120  001003                                  BNE     3$
000122  005201                                  INC     R1                              ; BUFF1                         5309
000124  005202                                  INC     R2                              ; BUFF2                         5310
000126  000446                                  BR      6$                              ;                               5306
000130  013700  000000G                  3$:     MOV     T.ADDR,R0                       ;                               5314
000134  005260  000014                          INC     14(R0)
000140  105260  000051                          INCB    51(R0)                          ;                               5315
000144  032737  000001  001162                  BIT     #1,APT.MODE                     ;                               5317
000152  001405                                  BEQ     4$
000154  012716  000052                          MOV     #52,(SP)                        ;                               5319
000160  004737  000000V                          JSR     PC,ERR.HRD.RTNE.APT
000164  000404                                  BR      5$                              ;                               5317
000166  012716  000052                  4$:     MOV     #52,(SP)                        ;                               5321
000172  004737  000000V                          JSR     PC,ERR.HRD.RTNE
000176  016616  000004                  5$:     MOV     4(SP),(SP)                      ;                               5323
000202  062716  000000G                          ADD     #RETPKT,(SP)
000206  004737  000000G                          JSR     PC,EMS.CMP
000212  013700  000000G                          MOV     T.ADDR,R0                       ;                               5325
000216  026037  000014  000000G                  CMP     14(R0),SWP.ERROR
000224  103412                                  BLO     7$
000226  013700  000000G                          MOV     L$LUN,R0                        ;                               5328
000232  112760  000004  000000G                  MOVB    #4,DUR(R0)
000240  104451                                  TRAP    51                              ;                               5329
000242  000403                                  BR      7$                              ;                               5313
000244  005203                          6$:     INC     R3                              ; I                             5304
000246  020304                                  CMP     R3,R4                           ; I,COUNT
000250  003722                                  BLE     2$
000252  022626                          7$:     CMP     (SP)+,(SP)+                     ;                               5298
000254  005000                          8$:     CLR     R0                              ;                               5337
000256  150500                                  BISB    R5,R0                           ; FLAG,*
000260  005726                                  TST     (SP)+                           ;                               5261
000262  000207                                  RTS     PC

; Routine Size:  90 words,      Routine Base:  $CODE$ + 24460
; Maximum stack depth per invocation:  11 words
```

```
;    5339  1     GLOBAL routine SWEEP : novalue =
;    5340  1
;    5341  1     !+
;    5342  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT AND OTHERS TO DEALLOCATE THE
;    5343  1     !       RESOURCES ASSOCIATED WITH THE CURRENT RETURN PACKET. THIS INCLUDES THE
;    5344  1     !       PACKET ITSELF AND THE I/O BUFFER. IN ADDITION, IF THE HOST IS
;    5345  1     !       PERFORMING WRITE-COMPARES, AND IF THE CURRENT RETURN PACKET IS A READ
;    5346  1     !       FUNCTION, THEN THE CURRENT CONTROLLER'S RP_SAVE AREA IS SEARCHED FOR
;    5347  1     !       THE ASSOCIATED WRITE RETPKT SO THAT ITS RESOURCES CAN ALSO BE
;    5348  1     !       DEALLOCATED.
;    5349  1     !
;    5350  1     !       IMPLICIT INPUTS:
;    5351  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    5352  1     !               RP_INDX - INDEX OF CURRENT RETURN PACKET
;    5353  1     !
;    5354  1
;    5355  2         begin
;    5356  2
;    5357  2         local
;    5358  2             index : signed word;
;    5359  2
;    5360  2         if (.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD          ! IF READ OPCODE OR ENDCODE
;    5361  2         then
;    5362  2
;    5363  3             if BIT_TST (SWP_FLAGS, SWF_HWC)                  ! IF HOST IS DOING WRITE COMPARES
;    5364  2             then
;    5365  2
;    5366  2                 if (index = RPS_REM ()) geq 0                ! IF ASSOCIATED WRITE RETPKT IS FOUND
;    5367  2                 then
;    5368  3                     begin
;    5369  3                     PUT_IO_BUFF (RETPKT [.index, BUFF_0]);   ! RETURN WRITE I/O BUFFER TO POOL
;    5370  3                     PUT_RETPKT (.index);                     ! RETURN WRITE PACKET TO POOL
;    5371  2                     end;
;    5372  2
;    5373  2             PUT_IO_BUFF (RP_ADDR [BUFF_0]);                  ! RETURN CURRENT I/O BUFFER TO POOL
;    5374  2             PUT_RETPKT (.RP_INDX);                           ! RETURN CURRENT RETPKT TO POOL
;    5375  1             end;                                             ! ROUTINE SWEEP


                                        .SBTTL  SWEEP MULTI-DRIVE TEST ROUTINES
000000  010146                 SWEEP:: MOV     R1,-(SP)                        ;                                5339
000002  013700  000000G                MOV     RP.ADDR,R0                      ;                                5360
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                   ;                                5363
000032  001422                        BEQ     1$
000034  004737  000000V                JSR     PC,RPS.REM                      ;                                5366
000040  010001                        MOV     R0,R1                           ; *,INDEX
000042  002416                        BLT     1$
000044  010146                        MOV     R1,-(SP)                        ; INDEX,*                        5369
000046  012746  000054                MOV     #54, (SP)
```

```
000052  004737  000000G                        JSR     PC,BL$MUL
000056  062700  000024G                        ADD     #RETPKT+24,R0
000062  010016                                 MOV     R0,(SP)
000064  004737  000000G                        JSR     PC,PUT.IO.BUFF
000070  010116                                 MOV     R1,(SP)                 ; INDEX,*                       5370
000072  004737  000000G                        JSR     PC,PUT.RETPKT
000076  022626                                 CMP     (SP)+,(SP)+             ;                               5368
000100  013746  000000G             1$:        MOV     RP.ADDR, (SP)           ;                               5373
000104  062716  000024                         ADD     #24,(SP)
000110  004737  000000G                        JSR     PC,PUT.IO.BUFF
000114  013716  000000G                        MOV     RP.INDX,(SP)            ;                               5374
000120  004737  000000G                        JSR     PC,PUT.RETPKT
000124  005726                                 TST     (SP)+                   ;                               5355
000126  012601                                 MOV     (SP)+,R1                ;                               5339
000130  000207                                 RTS     PC

; Routine Size:  45 words,      Routine Base:  $CODE$ + 24744
; Maximum stack depth per invocation:  4 words
```

```
;    5376  1     GLOBAL routine RPS_REM =
;    5377  1
;    5378  1     !+
;    5379  1     !       THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP_SAVE AREA FOR A
;    5380  1     !       RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
;    5381  1     !       CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
;    5382  1     !       OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
;    5383  1     !       RP_SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
;    5384  1     !       OPERATION IS RETURNED TO THE CALLER.
;    5385  1     !
;    5386  1     !       IMPLICIT INPUTS:
;    5387  1     !               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
;    5388  1     !
;    5389  1     !       OUTPUTS:
;    5390  1     !               INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
;    5391  1     !                       A CRN WHICH IS ONE LESS THAN THE CURRENT
;    5392  1     !
;    5393  1
;    5394  2         begin
;    5395  2
;    5396  2         local
;    5397  2             index : signed word initial (-1);                          ! ASSUME NOT FOUND
;    5398  2
;    5399  2         incr COUNT from 0 to RP_CNT - 1 do                             ! FOR EACH ENTRY IN RP_SAVE
;    5400  2
;    5401  2             if (.RP_USE [.COUNT] eql .CCTLR) and                       ! IF THIS IS A VALID RETPKT INDEX
;    5402  3                (.RETPKT [.COUNT, ENDCOD] eql (OP_WRT or OP_END))
;    5403  2             then
;    5404  2
;    5405  3                 if ((.RETPKT [.COUNT, CRF_LO] eql (.RP_ADDR [CRF_LO]   1)) and     ! IF CORRECT CRN
;    5406  2                     (.RETPKT [.COUNT, CRF_HI] eql .RP_ADDR [CRF_HI])) or
;    5407  3                    ((.RETPKT [.COUNT, CRF_HI] eql (.RP_ADDR [CRF_HI]   1)) and
;    5408  3                     (.RETPKT [.COUNT, CRF_LO] eql %o'177777') and
;    5409  3                     (.RP_ADDR [CRF_LO] eql 0))
;    5410  2                 then
;    5411  3                     begin
;    5412  3                     index = .COUNT;                                    ! INDEX TO BE RETURNED
;    5413  3                     exitloop;                                          ! DONE
;    5414  2                     end;
;    5415  2
;    5416  2         return .index;
;    5417  1         end;                                                           ! ROUTINE RPS REM


                                .SBTTL  RPS.REM MULTI-DRIVE TEST ROUTINES
000000  004137  000000G    RPS.REM::
                            JSR     R1,$SAVE4                       ;                       5376
000004  012704  177777      MOV     #-1,R4                          ; #,INDEX               5394
000010  005003              CLR     R3                              ; COUNT                 5399
000012  116300  000000G     1$:     MOVB    RP.USE(R3),R0           ; #(COUNT),#            5401
000016  020037  000000G             CMP     R0,CCTLR
000022  001053                      BNE     4$
000024  010346                      MOV     R3,-(SP)                ; COUNT,#               5402
```

```
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,*                       5405
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)      ;                               5406
000112  001415                                BEQ     3$
000114  016102  000006              2$:       MOV     6(R1),R2                ;                               5407
000120  005302                                DEC     R2
000122  026002  000006G                       CMP     RETPKT+6(R0),R2
000126  001011                                BNE     4$
000130  026027  000004G 177777                CMP     RETPKT+4(R0),#-1        ;                               5408
000136  001005                                BNE     4$
000140  005761  000004                        TST     4(R1)                   ;                               5409
000144  001002                                BNE     4$
000146  010304                      3$:       MOV     R3,R4                   ; COUNT,INDEX                   5412
000150  000404                                BR      5$                      ;                               5411
000152  005203                      4$:       INC     R3                      ; COUNT                         5399
000154  020327  000007                        CMP     R3,#7                   ; COUNT,*
000160  003714                                BLE     1$
000162  010400                      5$:       MOV     R4,R0                   ; INDEX,*                       5394
000164  000207                                RTS     PC                      ;                               5376

; Routine Size:  59 words,      Routine Base:  $CODE$ + 25076
; Maximum stack depth per invocation:  8 words
```

```
;    5418  1     GLOBAL routine DR_RETPKT : novalue =
;    5419  1
;    5420  1     !+
;    5421  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT FOR ALL PACKETS ORIGINATING AT
;    5422  1     !       THE "DRIVER" PORTION OF THE PROGRAM. THIS INCLUDES PACKETS DESCRIBING
;    5423  1     !       FATAL DEVICE ERRORS.
;    5424  1     !
;    5425  1     !       FOR FATAL DEVICE ERRORS, THIS ROUTINE RELEASES ALL RESOURCES HELD BY
;    5426  1     !       THE CONTROLLER. THE CONTROLLER IS MARKED OFFLINE IN ITS CST, AND ALL
;    5427  1     !       UNITS ATTACHED TO THE CONTROLLER ARE DROPPED.
;    5428  1     !
;    5429  1     !       IMPLICIT INPUTS:
;    5430  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;    5431  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5432  1     !               CST_ADDR   ADDRESS OF THE CURRENT CONTROLLER S CST
;    5433  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    5434  1     !
;    5435  1
;    5436  2         begin
;    5437  2
;    5438  2
;    5439  2         PUTA_BUFF ();                           ! RELEASE ALL I/O BUFFERS HELD BY CONTROLLER
;    5440  2
;    5441  2         incr index from 0 to RP_CNT   1 do      ! FOR EACH ENTRY IN CONTROLLER S RP_SAVE
;    5442  2
;    5443  2             if .RP_USE [.index] eql .CCTLR      ! IF VALID RETPKT INDEX
;    5444  2             then
;    5445  2                 PUT_RETPKT (. ndex);            ! RETURN RETPKT TO POOL
;    5446  2
;    5447  2         QIO [.CCTLR] = 0;                       ! CLEAR NO. OF OUTSTANDING QIOs
;    5448  2         CST_ADDR [STATE] = OFFLINE;             ! MARK CST OFFLINE
;    5449  2         DROP_CTLR (.CCTLR, DU_CFATAL);          ! DROP CONTROLLER'S UNITS
;    5450  2         PUT_RETPKT (.RP_INDX);                  ! PUT BACK RETPKT
;    5451  1         end;                                    ! ROUTINE DR_RETPKT


                                .SBTTL  DR.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146          DR.RETPKT::
                                MOV     R1,-(SP)                        ;                       5418
000002  004737  000000G         JSR     PC,PUTA.BUFF                    ;                       5439
000006  005001                  CLR     R1                              ; INDEX                 5441
000010  116100  000000G 1$:     MOVB    RP.USE(R1),R0                   ; *(INDEX),*            5443
000014  020037  000000G         CMP     R0,CCTLR
000020  001004                  BNE     2$
000022  010146                  MOV     R1,-(SP)                        ; INDEX,*               5445
000024  004737  000000G         JSR     PC,PUT.RETPKT
000030  005726                  TST     (SP)+
000032  005201          2$:     INC     R1                              ; INDEX                 5441
000034  020127  000007          CMP     R1,#7                           ; INDEX,*
000040  003763                  BLE     1$
000042  013701  000000G         MOV     CCTLR,R1                        ;                       5447
000046  105061  000000G         CLRB    QIO(R1)
000052  013700  000000G         MOV     CST.ADDR,R0                     ;                       5448
```

```
000056  042760  100000  000          BIC     #100000,2(R0)
000064  010146                       MOV     R1,-(SP)                ;                5449
000066  012746  000006               MOV     #6,-(SP)
000072  004737  0000006              JSR     PC,DROP.CTLR
000076  013716  0000006              MOV     RP.INDX,(SP)            ;                5450
000102  004737  0000006              JSR     PC,PUT.RETPKT
000106  022626                       CMP     (SP)+,(SP)+             ;                5436
000110  012601                       MOV     (SP)+,R1                ;                5418
000112  000207                       RTS     PC

; Routine Size:  38 words,      Routine Base:  $CODE$ + 25264
; Maximum stack depth per invocation:  4 words
```

```
.    5452  1      %sbttl 'RDRx INTERRUPT SERVICE ROUTINES
;    5453  1
;    5454  1      !+
;    5455  1      !       THERE EXISTS AN RDRX INTERRUPT SERVICE ROUTINE FOR EACH DEVICE
;    5456  1      !       CONTROLLER. EACH SERVICE ROUTINE BEGINS BY SIMPLY SETTING THE
;    5457  1      !       APPROPRIATE CONTROLLER NUMBER INTO "ICTLR". ALL SERVICE ROUTINES THEN
;    5458  1      !       BRANCH TO A COMMON INTERRUPT PROCESSING ROUTINE.
;    5459  1      !
;    5460  1
;    5461  2      BGNSRV (AZINTO);
;    5462  2      ICTLR = 0;
;    5463  2      AZINT ();
;    5464  1      ENDSRV;


                                        .SBTTL  AZINTO RDRx INTERRUPT SERVICE ROUTINES
000000  010046                  AZINTO::MOV     R0,-(SP)                        ;  5461
000002  005037  000104                 CLR     ICTLR                           ;  5462
000006  004737  000000V                JSR     PC,AZINT                        ;  5463
000012  012600                         MOV     (SP)+,R0                        ;  5464
000014  000002                         RTI

; Routine Size:  7 words,       Routine Base:  $CODE$ + 25400
; Maximum stack depth per invocation:  2 words
```

```
;  5465  1     GLOBAL routine AZINT : novalue =
;  5466  1
;  5467  1     !+
;  5468  1     !       THIS IS THE COMMON INTERRUPT SERVICE ROUTINE FOR ALL RDRX CONTROLLERS.
;  5469  1     !       AFTER CALCULATING THE DCT ADDRESS FOR THE INTERRUPTING DEVICE, THIS
;  5470  1     !       ROUTINE WILL SAVE THE CURRENT CONTENTS OF THE SA REGISTER IN THE DCT.
;  5471  1     !       THEN, IF THE "IGNORE INTERRUPT" BIT IS SET, NO FURTHER ACTION IS TAKEN.
;  5472  1     !       OTHERWISE, THE SA VALUE IS CHECKED FOR A FATAL ERROR, AND THE COMMAND
;  5473  1     !       AND RESPONSE RINGS ARE POLLED.
;  5474  1     !
;  5475  1
;  5476  2         begin
;  5477  2         IDCT_ADDR = DCT + (.ICTLR * DCT_LEN * 2);           ! GET DCT ADDRESS
;  5478  2         ICST_ADDR = CST + (.ICTLR * CST_LEN * 2);           ! GET CST ADDRESS
;  5479  2         IRDRX_ADDR = .ICST_ADDR [IP_ADDR];                  ! GET RDRX ADDRESS
;  5480  2         ICOM_ADDR = COMM_AREA + (.ICTLR * COMM_LEN * 2);    ! GET COMM_AREA ADDR
;  5481  2         IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];   ! SAVE SA REGISTER
;  5482  2
;  5483  2         if .IDCT_ADDR [IG_INT]                              ! IGNORE INTERRUPT?
;  5484  2         then
;  5485  2             return;                                         ! RETURN IF INTERRUPTS IGNORED
;  5486  2
;  5487  3         if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)            ! IF FATAL ERROR
;  5488  2         then
;  5489  2             FATAL_ERROR ()
;  5490  2         else
;  5491  3             begin
;  5492  3             POLL_CRING ();                                  ! POLL COMMAND RING
;  5493  3             POLL_RRING ();                                  ! POLL RESPONSE RING
;  5494  2             end;
;  5495  2
;  5496  1         end;


                                        .SBTTL  AZINT RDRX INTERRUPT SERVICE ROUTINES
000000  010146                 AZINT:: MOV     R1,-(SP)                        ;                       5465
000002  005746                         TST     -(SP)
000004  013701  000104'                MOV     ICTLR,R1                        ;                       5477
000010  010146                         MOV     R1,-(SP)
000012  012746  000022                 MOV     #22,-(SP)
000016  004737  000000G                JSR     PC,BL$MUL
000022  062700  000000G                ADD     #DCT,R0
000026  010037  000100'                MOV     R0,IDCT.ADDR
000032  010116                         MOV     R1,(SP)                         ;                       5478
000034  012746  000126                 MOV     #126,-(SP)
000040  004737  000000G                JSR     PC,BL$MUL
000044  062700  000000G                ADD     #CST,R0
000050  010037  000076'                MOV     R0,ICST.ADDR
000054  011037  000000G                MOV     @R0),IRDRX.ADDR                 ; ICST.ADDR,*           5479
000060  010116                         MOV     R1,(SP)                         ;                       5480
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.ADDP
000102  013701  000100'                        MOV     IDCT.ADDR,R1            ;                       5481
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           5483
000132  001016                                 BNE     2$                      ;                       5465
000134  016601  000010                         MOV     10(SP),R1               ;                       5487
000140  042701  077777                         BIC     #77777,R1
000144  020127  100000                         CMP     R1,#-100000
000150  001003                                 BNE     1$
000152  004737  000000V                        JSR     PC,FATAL.ERROR          ;                       5489
000156  000404                                 BR      2$                      ;                       5487
000160  004737  000000V                1$:     JSR     PC,POLL.CRING           ;                       5492
000164  004737  000000V                        JSR     PC,POLL.RRING           ;                       5493
000170  062706  000012                2$:     ADD     #12,SP                  ;                       5465
000174  012601                                 MOV     (SP)+,R1
000176  000207                                 RTS     PC

; Routine Size:  64 words,      Routine Base:  $CODE$ + 25416
; Maximum stack depth per invocation:  7 words


;   5497  1
```

```
;  5498  1
;  5499  1     !+
;  5500  1     GLOBAL ROUTINE DUP_RSP : NOVALUE =                                  !ZZZ
;  5501  1
;  5502  1     !*
;  5503  1     !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH DUP RESPONSE
;  5504  1     !       ITS GENERAL PURPOSE IS TO ACT ON A DATAGRAM OR SEQUENTIAL MESSAGE.
;  5505  1     !       IF THE MESSAGE TYPE IS SEQUENTIAL, THE ROUTINE COPIES THE
;  5506  1     !       CONTENTS OF THE MESSAGE ENVELOPE INTO A RETURN PACKET SO THAT THE
;  5507  1     !       ENVELOPE CAN BE RETURNED TO THE CONTROLLER.
;  5508  1     !
;  5509  1     !       IMPLICIT INPUTS:
;  5510  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;  5511  1     !               IPKT_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING RESPONSE
;  5512  1     !
;  5513  2         begin
;  5514  2
;  5515  2         local
;  5516  2             R_INDEX : signed word,
;  5517  2             DEBUG,                                                  !ZZZ
;  5518  2             SRC_ADDR,
;  5519  2             DST_ADDR,
;  5520  2             R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;  5521  2         !PRINTX (DER34);
;  5522  2
;  5523  2         incr COUNT from 0 to PKT_CNT - 1 do
;  5524  2
;  5525  2             if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and     ! IF THIS IS THE ASSOC CMD
;  5526  2                (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  5527  2                (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;  5528  2                ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;  5529  2                (.MSCP_PKT [.COUNT, CONNID] eql CID_DUP) and
;  5530  3                ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END)
;  5531  2             then
;  5532  3                 begin
;  5533  3                 P_INDEX = .COUNT;                                   ! SET PKT NUMBER
;  5534  3                 exitloop;
;  5535  2                 end;
;  5536  2
;  5537  2         if .P_INDEX lss 0                                           ! IF COMMAND NOT FOUND
;  5538  2         then
;  5539  3             begin
;  5540  3             PRINTF (DBM108, .IPKT_ADDR [CRN_LO]);                   ! UNKNOWN COMMAND REF. NUMBER
;  5541  3             return;
;  5542  2             end;
;  5543  2
;  5544  2         if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0                    ! IF RETPKT IS NOT AVAILABLE
;  5545  2         then
;  5546  2             DEBUG = TRUE                                            !TO SEE IF THIS PATH TAKEN          ZZZ
;  5547  2     !       PRINTF (DBM112)                                         ! "DUP-RSP: RETPKT NOT AVAILABLE"   ZZZ
;  5548  2         else
;  5549  3             begin
;  5550  3             SRC_ADDR = .IPKT_ADDR + 6;                              ! SET UP COPY (SKIP OVER PKT DESC)
```

```
;    5551  3          R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);    ! START OF ALLOCATED RETPKT
;    5552  3
;    5553  3          incr COUNT from 1 to RP_LEN do
;    5554  4              begin
;    5555  4              .DST_ADDR = ..SRC_ADDR;              ! COPY 1 WORD
;    5556  4              DST_ADDR = .DST_ADDR + 2;            ! ADVANCE DESTINATION ADDR
;    5557  4              SRC_ADDR = .SRC_ADDR + 2;            ! ADVANCE SOURCE ADDR
;    5558  3              end;                                 ! COPY LOOP
;    5559  3
;    5560  3          IN_IODQ (.R_INDEX);                      ! PUT RETPKT INDEX INTO IODQ
;    5561  2          end;                                     ! IF RETPKT WAS ALLOCATED
;    5562  2
;    5563  2
;    5564  2      if .P_INDEX geq 0                            ! IF ASSOC CMD PKT WAS FOUND
;    5565  2      then
;    5566  2          PUT PKT (.P_INDEX);                      ! RETURN COMMAND PACKET TO POOL
;    5567  2
;    5568  1      end;                                         ! ROUTINE DUP-RSP


                                        .SBTTL  DUP.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          DUP.RSP::
                                        JSR     R1,$SAVE3                       ;                       5500
000004  013701  000000G                 MOV     IPKT.ADDR,R1                    ;                       5525
000010  005002                          CLR     R2                              ; COUNT                 5523
000012  010246                  1$:     MOV     R2,-(SP)                        ; COUNT,*               5525
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)          ;                       5526
000044  001020                          BNE     2$
000046  026011  000000G                 CMP     MSCP.PKT(R0),(R1)               ;                       5527
000052  001415                          BEQ     2$
000054  105760  000022G                 TSTB    MSCP.PKT+22(R0)                 ;                       5528
000060  100412                          BMI     2$
000062  126027  000011G 000002          CMPB    MSCP.PKT+11(R0),#2              ;                       5529
000070  001006                          BNE     2$
000072  105761  000022                  TSTB    22(R1)                          ;                       5530
000076  100003                          BPL     2$
000100  010237  000000G                 MOV     R2,P.INDEX                      ; COUNT,*               5533
000104  000406                          BR      3$                              ;                       5532
000106  005202                  2$:     INC     R2                              ; COUNT                 5523
000110  020227  000013                  CMP     R2,#13                          ; COUNT,*
000114  003736                          BLE     1$
000116  005737  000000G                 TST     P.INDEX                         ;                       5537
000122  002013                  3$:     BGE     4$
000124  016146  000012                  MOV     12(R1),-(SP)                    ;                       5540
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                           ;                               5541
000150  000207                                  RTS     PC                              ;                               5539
000152  013746  000104          4$:             MOV     ICTLR, (SP)                     ;                               5544
000156  004737  000000G                         JSR     PC,GET.RETPKT
000162  010001                                  MOV     R0,R1                           ; *,R.INDEX
000164  005726                                  TST     (SP)+
000166  005701                                  TST     R1                              ; R.INDEX
000170  002003                                  BGE     5$
000172  012700  000001                          MOV     #1,R0                           ; *,DEBUG                       5546
000176  000425                                  BR      7$                              ;                               5544
000200  013702  000000G         5$:             MOV     IPKT.ADDR,R2                    ; *,SRC.ADDR                    5550
000204  062702  000006                          ADD     #6,R2                           ; *,SRC.ADDR
000210  010146                                  MOV     R1,-(SP)                        ; R.INDEX,*                     5551
000212  012746  000054                          MOV     #54,-(SP)
000216  004737  000000G                         JSR     PC,BL$MUL
000222  062700  000000G                         ADD     #RETPKT,R0
000226  010003                                  MOV     R0,R3                           ; *,DST.ADDR
000230  012700  000026                          MOV     #26,R0                          ; *,COUNT                       5553
000234  012223          6$:             MOV     (R2)+,(R3)+                     ; SRC.ADDR,DST.ADDR             5555
000236  005300                                  DEC     R0                              ; COUNT                         5553
000240  001375                                  BNE     6$
000242  010116                                  MOV     R1,(SP)                         ; R.INDEX,*                     5560
000244  004737  000000G                         JSR     PC,IN.IODQ
000250  022626                                  CMP     (SP)+,(SP)+                     ;                               5549
000252  013700  000000G         7$:             MOV     P.INDEX,R0                      ;                               5564
000256  002404                                  BLT     8$
000260  010046                                  MOV     R0,-(SP)                        ;                               5566
000262  004737  000000G                         JSR     PC,PUT.PKT
000266  005726                                  TST     (SP)+
000270  000207          8$:             RTS     PC                              ;                               5500

; Routine Size:  93 words,      Routine Base:  $CODE$ + 25616
; Maximum stack depth per invocation:  9 words


;   5569  1
```

```
;    5570  1     GLOBAL routine FATAL_ERROR : novalue =
;    5571  1
;    5572  1     !*
;    5573  1     !       THIS ROUTINE IS CALLED BY THE INTERRUPT SERVICE ROUTINE (AZINT) UPON
;    5574  1     !       DETECTING AN UNRECOVERABLE ERROR THROUGH THE DEVICE'S SA REGISTER.
;    5575  1     !       ITS PURPOSE IS TO CLEAN UP DEVICE DATA IN THE "DRIVER" PORTION OF
;    5576  1     !       THE EXERCISER, AND TO INFORM THE "PROGRAM" PORTION OF THE EVENT VIA
;    5577  1     !       RETURN PACKET.
;    5578  1     !
;    5579  1     !       IMPLICIT INPUTS:
;    5580  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    5581  1     !               IDCT_ADDR   ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    5582  1     !               ICST_ADDR - ADDRESS OF INTERRUPTING CONTROLLER S CST
;    5583  1     !               IRDRX ADDR - ADDRESS OF INTERRUPTING CONTROLLER S IP REGISTER
;    5584  1     !
;    5585  1
;    5586  2         begin
;    5587  2
;    5588  2         local
;    5589  2             index : signed word,
;    5590  2             U_SAVE : word;
;    5591  2
;    5592  2         SA_REG = .IDCT_ADDR [SA_SAVE];
;    5593  2         U_SAVE = .L$LUN;                                    ! SAVE PRE-INTERRUPT CURRENT UNIT NUMBER
;    5594  2         C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR TBL [.ICTLR, C ERR_HRD] + 1;
;    5595  2
;    5596  2         if .APT_MODE
;    5597  2         then
;    5598  3             begin
;    5599  3             .MAIL_BOX_TESTNUM = 1;
;    5600  3             .MAIL_BOX_SUBTST = 0;
;    5601  2             end;
;    5602  2
;    5603  2         L$LUN = .ICST_ADDR [OF_UN + OF_DATA, D_UNIT];       ! SET CURRENT UNIT TO FIRST IN CONTROLLER
;    5604  2         ERRDF (14, EGD_14, EMS_14);                         ! FATAL CONTROLLER ERROR
;    5605  2         L$LUN = .U_SAVE;                                    ! RESTORE PRE-INTERRUPT CURRENT UNIT
;    5606  2         DRV_CTLERR (.ICTLR);                                ! CLEAN UP DRIVER DATA FOR CONTROLLER
;    5607  2
;    5608  2         if (index = GET_RETPKT (.ICTLR)) lss 0              ! TRY TO GET A RETPKT; IF FAILURE
;    5609  2         then
;    5610  3             PRINTF (DBM18)                                  ! "FATAL_ERROR: RETPKT NOT AVAILABLE'
;    5611  2         else
;    5612  3             begin                                           ! IF RETPKT WAS ALLOCATED
;    5613  3             RETPKT [.index, CONID] = CID_DRIVER;            ! SET CONNECTION ID TO "DRIVER'
;    5614  3             RETPKT [.index, MESTYP] = MT_FATAL;             ! FATAL ERROR
;    5615  3             RETPKT [.index, CTLR] = .ICTLR;                 ! CONTROLLER NUMBER
;    5616  3             IN_IODQ (.index);                               ! LOAD RETPKT INDEX INTO IODQ
;    5617  2             end;                                            ! IF RETPKT WAS ALLOCATED
;    5618  2
;    5619  1         end;                                                ! ROUTINE FATAL_ERR

                                .SBTTL  FATAL.ERROR RDRX INTERRUPT SERVICE ROUTINES
```

```
000000  004137  000000G                FATAL.ERROR::
                                        JSR     R1,$SAVE2                       ;                               5570
000004  013700  000100                  MOV     IDCT.ADDR,R0                    ;                               5592
000010  016037  000002  000000G         MOV     2(R0),SA.REG
000016  013701  000000G                 MOV     L$LUN,R1                        ; *,U.SAVE                      5593
000022  013700  000104'                 MOV     ICTLR,R0                        ;                               5594
000026  006300                          ASL     R0
000030  105260  000000G                 INCB    C.ERR.TBL(R0)
000034  032737  000001  001162'         BIT     #1,APT.MODE                     ;                               5596
000042  001405                          BEQ     1$
000044  012777  000001  001164          MOV     #1,@MAIL.BOX.TESTNUM            ;                               5599
000052  005077  001166                  CLR     @MAIL.BOX.SUBTST                ;                               5600
000056  013700  000076'        1$:     MOV     ICST.ADDR,R0                    ;                               5603
000062  016002  000006                  MOV     6(R0),R2
000066  000302                          SWAB    R2
000070  042702  177760                  BIC     #177760,R2
000074  010237  000000G                 MOV     R2,L$LUN
000100  104455                          TRAP    55                              ;                               5604
000102  000016                          .WORD   16
000104  000000G                         .WORD   EGD.14
000106  000000G                         .WORD   EMS.14
000110  010137  000000G                 MOV     R1,L$LUN                        ; U.SAVE,*                      5605
000114  013746  000104'                 MOV     ICTLR,-(SP)                     ;                               5606
000120  004737  000000G                 JSR     PC,DRV.CTLERR
000124  013716  000104'                 MOV     ICTLR,(SP)                      ;                               5608
000130  004737  000000G                 JSR     PC,GET.RETPKT
000134  010001                          MOV     R0,R1                           ; *,INDEX
000136  002007                          BGE     2$
000140  012716  000000G                 MOV     #DBM18,(SP)                     ;                               5610
000144  012746  000001                  MOV     #1,-(SP)
000150  010600                          MOV     SP,R0                           ; SP,*
000152  104417                          TRAP    17
000154  000424                          BR      3$                              ;                               5608
000156  010116                2$:     MOV     R1,(SP)                         ; INDEX,*                       5613
000160  012746  000054                  MOV     #54,-(SP)
000164  004737  000000G                 JSR     PC,BL$MUL
000170  062700  000002G                 ADD     #RETPKT+2,R0
000174  112760  000003  000001         MOVB    #3,1(R0)
000202  013702  000104'                 MOV     ICTLR,R2                        ;                               5615
000206  042702  177760                  BIC     #177760,R2
000212  112710  000060                  MOVB    #60,(R0)
000216  150210                          BISB    R2,(R0)
000220  010116                          MOV     R1,(SP)                         ; INDEX,*                       5616
000222  004737  000000G                 JSR     PC,IN.IODQ
000226  022626                3$:     CMP     (SP)+,(SP)+                     ;                               5586
000230  000207                          RTS     PC                              ;                               5570

; Routine Size:  77 words,      Routine Base:  $CODE$ + 26110
; Maximum stack depth per invocation:  7 words
```

```
;    5620  1     GLOBAL routine POLL_CRING : novalue =
;    5621  1
;    5622  1     !+
;    5623  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    5624  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    5625  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S COMMAND RING AND CHECK FOR ANY
;    5626  1     !       COMMAND SLOTS THAT HAVE BEEN "TAKEN" BY THE CONTROLLER. SUCH SLOTS
;    5627  1     !       HAVE BEEN RETURNED TO THE HOST, INDICATED BY A ZERO OWNERSHIP BIT. FOR
;    5628  1     !       EACH SLOT THAT HAS BEEN RETURNED TO THE HOST, THE CRING COUNT IS
;    5629  1     !       DECREMENTED, AND THE CR_POLL ADDRESS IS ADVANCED TO THE NEXT SLOT IN
;    5630  1     !       THE COMMAND RING.
;    5631  1     !
;    5632  1     !       IMPLICIT INPUTS:
;    5633  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    5634  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    5635  1     !               ICOM_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S COMM_AREA
;    5636  1     !
;    5637  1
;    5638  2         begin
;    5639  2
;    5640  3         while ((.IDCT_ADDR [CRING_CNT] gtru 0) and                  ! WHILE # OF COMMANDS IN CRING > 0 AND
;    5641  2             not (BIT_TST ((.IDCT_ADDR [CR_POLL] + 2), ED_OWN))) do  ! CURRENT SLOT IS HOST-OWNED
;    5642  3             begin
;    5643  3             IDCT_ADDR [CRING_CNT] = .IDCT_ADDR [CRING_CNT] - 1;     ! DECREMENT # CMDs IN CRING
;    5644  3             IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_POLL] + 4;         ! ADVANCE TO NEXT SLOT TO POLL
;    5645  3
;    5646  3             if .IDCT_ADDR [CR_POLL] gtra .IDCT_ADDR [CR_END]        ! IF BEYOND END OF RING
;    5647  3             then
;    5648  3                 IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_BEG];          ! SET POINTER TO TOP OF CRING
;    5649  3
;    5650  2             end;
;    5651  2
;    5652  2         ICOM_ADDR [CMD_INT] = 0;                                    ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER
;    5653  1         end;


                                        .SBTTL  POLL.CRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          POLL.CRING::
                                        JSR     R1,$SAVE2                       ;                       5620
000004  013701  000100'                 MOV     IDCT.ADDR,R1                    ;                       5640
000010  012702  000016                  MOV     #16,R2                          ;                       5644
000014  060102                          ADD     R1,R2
000016  105711                  1$:     TSTB    (R1)                            ;                       5640
000020  001422                          BEQ     2$
000022  016100  000016                  MOV     16(R1),R0                       ;                       5641
000026  016000  000002                  MOV     2(R0),R0
000032  042700  077777                  BIC     #77777,R0
000036  020027  100000                  CMP     R0,#-100000
000042  001411                          BEQ     2$
000044  105311                          DECB    (R1)                            ;                       5643
000046  062712  000004                  ADD     #4,(R2)                         ;                       5644
000052  021261  000012                  CMP     (R2),12(R1)                     ;                       5646
000056  101757                          BLOS    1$
```

L 2


```
000060  016112  000010                        MOV     10(R1),(R2)                     ;                       5648
000064  000754                                BR      1$                              ;                       5640
000066  013700  000074              2$:       MOV     ICOM.ADDR,R0                    ;                       5652
000072  005060  000004                        CLR     4(R0)
000076  000207                                RTS     PC                              ;                       5620

; Routine Size:  32 words,      Routine Base:  $CODE$ + 26342
; Maximum stack depth per invocation:  4 words
```

```
;    5654  1     GLOBAL routine POLL_RRING : novalue =
;    5655  1
;    5656  1
;    5657  1     !*
;    5658  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    5659  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    5660  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S RESPONSE RING AND CHECK FOR ANY
;    5661  1     !       SLOTS WHICH HAVE BEEN RETURNED TO THE HOST (OWNERSHIP BIT = 0). FOR
;    5662  1     !       EACH SUCH SLOT, THE ASSOCIATED MESSAGE IS PROCESSED BASED ON ITS
;    5663  1     !       CONNECTION ID (DISK OR DUP). AFTER PROCESSING, THE MESSAGE PACKET
;    5664  1     !       IS RE-INITIALIZED AND RETURNED TO THE CONTROLLER (OWNERSHIP BIT SET
;    5665  1     !       TO 1).
;    5666  1     !
;    5667  1     !       IMPLICIT INPUTS:
;    5668  1     !               ICTLR - NUMBER OF INTERRUPTING CONTROLLER
;    5669  1     !               IDCT_ADDR   ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    5670  1     !
;    5671  1
;    5672  1
;    5673  2         begin
;    5674  2
;    5675  2
;    5676  2         while not (BIT_TST ((.IDCT_ADDR [RR_POLL] + 2), ED_OWN)) do          ! WHILE O = 0
;    5677  3             begin
;    5678  3             IPKT_ADDR = ..IDCT_ADDR [RR_POLL]   10;                          ! ADDRESS OF RESPONSE PACKET
;    5679  3
;    5680  3
;    5681  4     IF NOT (.IPKT_ADDR [CONNID] EQL CID_DUP)                 !               ZZZ
;    5682  3             THEN                                             !               ZZZ
;    5683  3             (CREDIT_BAL = .CREDIT_BAL + .IPKT_ADDR [CREDITS];);          !ZZZ
;    5684  3     !IT WAS NOTICE THAT DUP WAS SENDING BACK CREDITS WHICH IT SHOULD NOT.   !ZZZ
;    5685  3             selectoneu .IPKT_ADDR [CONNID] of
;    5686  3                 set
;    5687  3
;    5688  3                 [CID_DISK] :        DISK_RSP ();
;    5689  3
;    5690  3                 [CID_DUP]  :        DUP_RSP  ();                         !ZZZ
;    5691  3
;    5692  3                 [otherwise] :       PRINTF (D8M20, .IPKT ADDR [CONNID], .IRDRX_ADDR);
;    5693  3                                                                          ! "BAD CONN ID = XXXXX FROM XXXXXX'
;    5694  3                 tes;
;    5695  3
;    5696  3
;    5697  3             IPKT_ADDR [MSGLEN] = MSG_LEN + 2;                            ! RE-INIT PKT FIELDS; MESSAGE LENGTH
;    5698  3             IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;              ! ADVANCE TO HI ORDER WORD OF RING SLOT
;    5699  3             .IDCT_ADDR [RR_POLL] = .IPKT_ADDR [PKT_HI];                  ! RETURN SLOT TO CONTROLLER
;    5700  3             .IDCT_ADDR [RR_POLL] = ..IDCT_ADDR [RR_POLL] or ED OWN or ED FLAG; ! OWNERSHIP TOO
;    5701  3             IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;              ! ADVANCE TO NEXT RRING SLOT
;    5702  3
;    5703  3
;    5704  3             if .IDCT_ADDR [RR_POLL] gtra .IDCT ADDR [RR END]             ! IF BEYOND END OF RING
;    5705  3
;    5706  3             then
```

```
;  5707  3                IDCT_ADDR [RR POLL] = .IDCT_ADDR [RR BEG];                  ! CYCLE TO TOP OF RING
;  5708  3
;  5709  2            end;                                                           ! WHILE LOOP
;  5710  2
;  5711  2        ICOM_ADDR [RSP_INT] = 0;                                           ! CLR RESPONSE INTERRUPT WRD IN RING HEADER
;  5712  1        end;
```

```
                                        .SBTTL  POLL.RRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          POLL_RRING::
                                        JSR     R1,$SAVE3                       ;                                       5654
000004  013701  000100                  MOV     IDCT.ADDR,R1                    ;                                       5676
000010  062701  000014                  ADD     #14,R1
000014  011100                  1$      MOV     (R1),R0
000016  016000  000002                  MOV     2(R0),R0
000022  042700  077777                  BIC     #77777,R0
000026  020027  100000                  CMP     R0,#-100000
000032  001504                          BEQ     6$
000034  017137  000000  000000G        MOV     @0(R1),IPKT.ADDR                ;                                       5678
000042  162737  000012  000000G        SUB     #12,IPKT.ADDR
000050  013700  000000G                MOV     IPKT.ADDR,R0                    ;                                       5681
000054  005002                          CLR     R2
000056  156002  000011                  BISB    11(R0),R2
000062  020227  000002                  CMP     R2,#2
000066  001406                          BEQ     2$
000070  116003  000010                  MOVB    10(R0),R3                       ;                                       5683
000074  042703  177760                  BIC     #177760,R3
000100  060337  000000G                ADD     R3,CREDIT.BAL
000104  005702                  2$:     TST     R2                              ;                                       5688
000106  001003                          BNE     3$
000110  004737  000000V                JSR     PC,DISK.RSP
000114  000421                          BR      5$                              ;                                       5685
000116  020227  000002          3$:     CMP     R2,#2                           ;                                       5690
000122  001003                          BNE     4$
000124  004737  025616                  JSR     PC,DUP.RSP
000130  000413                          BR      5$                              ;                                       5685
000132  013746  000000G          4$:     MOV     IRDRX.ADDR,-(SP)                ;                                       5692
000136  010246                          MOV     R2,-(SP)
000140  012746  000000G                MOV     #DBM20,(SP)
000144  012746  000003                  MOV     #3,-(SP)
000150  010600                          MOV     SP,R0                           ; SP,*
000152  104417                          TRAP    17
000154  062706  000010                  ADD     #10,SP
000160  013700  000000G          5$:     MOV     IPKT.ADDR,R0                    ;                                       5697
000164  012760  000074  000006        MOV     #74,6(R0)
000172  013702  000100'                MOV     IDCT.ADDR,R2                    ;                                       5698
000176  010201                          MOV     R2,R1
000200  062701  000014                  ADD     #14,R1
000204  062711  000002                  ADD     #2,(R1)
000210  016071  000002  000000        MOV     2(R0),@0(R1)                    ;                                       5699
000216  052771  140000  000000        BIS     #-40000,@0(R1)                  ;                                       5700
000224  062711  000002                  ADD     #2,(R1)                         ;                                       5701
000230  021162  000006                  CMP     (R1),6(R2)                      ;                                       5704
```

```
000234  101667                          BLOS    1$
000236  016211  000004                  MOV     4(R2),(R1)              ;                       5707
000242  000664                          BR      1$                      ;                       5676
000244  013700  000074          6$:     MOV     ICOM.ADDR,R0            ;                       5711
000250  005060  000006                  CLR     6(R0)
000254  000207                          RTS     PC                      ;                       5654

; Routine Size:  87 words.      Routine Base:  $CODE$ + 26442
; Maximum stack depth per invocation:  10 words


;   5713  1
```

```
;    5714  1
;    5715  1
;    5716  1     GLOBAL routine DISK_RSP : novalue =
;    5717  1
;    5718  1
;    5719  1     !+
;    5720  1     !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH RESPONSE MESSAGE
;    5721  1     !       WHICH HAS A CONNECTION ID INDICATING A DISK MSCP ORIGINATOR
;    5722  1     !       (I.E., ALL EXCEPT DUP RESPONSES). ITS PURPOSE IS TO PASS
;    5723  1     !       CONTROL TO THE APPROPRIATE ROUTINE BASED ON THE MESSAGE TYPE
;    5724  1     !       FIELD (SEQUENTIAL, DATAGRAM, OR CREDIT NOTIFICATION).
;    5725  1     !
;    5726  1     !       IMPLICIT INPUTS:
;    5727  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    5728  1     !                           MESSAGE
;    5729  1     !
;    5730  1
;    5731  1
;    5732  1
;    5733  1         selectoneu .IPKT_ADDR [MSGTYP] of
;    5734  1
;    5735  1
;    5736  1             set
;    5737  1
;    5738  1             [MT_SEQ] :      SEQUEN ();
;    5739  1
;    5740  1             [MT_DG] :       DATAGM ();
;    5741  1
;    5742  1             [otherwise] :   PRINTF (DBM21, .IPKT_ADDR [MSGTYP]);    ! "MESSAGE TYPE XX RECEIVED"
;    5743  1             tes;


                                        .SBTTL  DISK.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  010146                 DISK.RSP::
                                        MOV     R1,-(SP)                        ;                       5716
000002  013700  000000G                 MOV     IPKT.ADDR,R0                    ;                       5733
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$                              ;                       5738
000030  004737  000000V                 JSR     PC,SEQUEN
000034  000417                          BR      3$                              ;                       5733
000036  020127  000001          1$:     CMP     R1,#1                           ;                       5740
000042  001003                          BNE     2$
000044  004737  000000V                 JSR     PC,DATAGM
000050  000411                          BR      3$                              ;                       5733
000052  010146                  2$:     MOV     R1,-(SP)                        ;                       5742
000054  012746  000000G                 MOV     #DBM21,-(SP)
000060  012746  000002                  MOV     #2,-(SP)
000064  010600                          MOV     SP,R0                           ; SP,*
```

```
000066  104417                          TRAP    17
000070  062706  000006                  ADD     #6,SP
000074  012601                  3$:     MOV     (SP)+,R1                ;                       5716
000076  000207                          RTS     PC

; Routine Size:  32 words,      Routine Base:  $CODE$ + 26720
; Maximum stack depth per invocation:  6 words


;   5744  1
;   5745  1
;   5746  1
```

```
;   5747  1     GLOBAL routine SEQUEN : novalue =
;   5748  1
;   5749  1     !+
;   5750  1     !       THIS ROUTINE IS CALLED BY DISK_RSP FOR EACH DISK MSCP RESPONSE MESSAGE
;   5751  1     !       WITH THE "SEQUENTIAL" MESSAGE TYPE. ITS GENERAL PURPOSE IS TO COPY THE
;   5752  1     !       CONTENTS OF THE MESSAGE PACKET INTO A RETURN PACKET SO THAT THE
;   5753  1     !       PACKET CAN BE RETURNED TO THE CONTROLLER.   IN ADDITION,
;   5754  1     !       IF THE COMMAND WAS AN I/O TRANSFER (READ, WRITE, OR ACCESS), THEN SOME
;   5755  1     !       FIELDS OF THE COMMAND PACKET ARE COPIED INTO THE RETURN PACKET.
;   5756  1     !
;   5757  1     !       IMPLICIT INPUTS:
;   5758  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;   5759  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;   5760  1     !
;   5761  1
;   5762  2         begin
;   5763  2
;   5764  2         local
;   5765  2             P_INDEX : signed word initial (-1),                          ! ASSUME NO ASSOCIATED COMMAND PKT
;   5766  2             R_INDEX : signed word,
;   5767  2             SRC_ADDR,
;   5768  2             DST_ADDR,
;   5769  2             R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;   5770  2
;   5771  2         incr COUNT from 0 to PKT_CNT - 1 do
;   5772  2
;   5773  2             if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and      ! IF THIS IS THE ASSOC CMD
;   5774  2                (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;   5775  2                (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;   5776  2                ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;   5777  2                (.MSCP_PKT [.COUNT, MSGTYP] eql MT_SEQ) and
;   5778  2                ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END) and
;   5779  3                (.PKT_USE [.COUNT] eql .ICTLR)
;   5780  2             then
;   5781  3                 begin
;   5782  3                 P_INDEX = .COUNT;                                       ! SET PKT NUMBER
;   5783  3                 exitloop;
;   5784  2                 end;
;   5785  2
;   5786  2         if .P_INDEX lss 0                                               ! IF COMMAND NOT FOUND
;   5787  2         then
;   5788  3             begin
;   5789  3             PRINTF (DBM108, .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);  ! UNKNOWN COMMAND REF. NUMBER
;   5790  3             return;
;   5791  2             end;
;   5792  2
;   5793  3         if .MSCP_PKT [.P_INDEX, OPCODE] neq (.IPKT_ADDR [OPCODE] and (not OP_END))  ! IF OPCODE MISMATCH
;   5794  2         then
;   5795  2             PRINTF (DBM111, .MSCP_PKT [.P_INDEX, OPCODE], .IPKT_ADDR [OPCODE], .IPKT_ADDR [CRN_HI], .IPKT ADDR [CRN_LO])
;
;   5796  2
;   5797  3         if ((.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or
;   5798  2             (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END))) and
;   5799  3            ((.IPKT_ADDR [STATUS_CODE] eql ST_SUC) and
```

```
;   5800  2              (.IPKT_ADDR [STATUS_SUBCODE] eql 0)) and
;   5801  3             ((.MSCP_PKT [.P_INDEX, BC_LO] neq .IPKT_ADDR [BC_LO]) or
;   5802  3              (.MSCP_PKT [.P_INDEX, BC_HI] neq .IPKT_ADDR [BC_HI]))
;   5803  2          then
; P 5804  2              PRINTF (DBM112,
; P 5805  2                      .MSCP_PKT [.P_INDEX, BC_HI], .MSCP_PKT [.P_INDEX, BC LO], .IPKT_ADDR [BC_HI], .IPKT ADDR [BC LO],
;   5806  2                      .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);
;   5807  2
;   5808  2          if .MSCP_PKT [.P_INDEX, RSP_RECEIVED]
;   5809  2          then
;   5810  3              begin
;   5811  3              PRINTF (DBM120, .MSCP_PKT [.P_INDEX, CRN_HI], .MSCP_PKT [.P_INDEX, CRN_LO]);
;   5812  3              PUT_PKT (.P_INDEX);
;   5813  3              return;
;   5814  3              end
;   5815  2          else
;   5816  2              MSCP_PKT [.P_INDEX, RSP_RECEIVED] = TRUE;                          ! MARK RESPONSE RECEIVED
;   5817  2
;   5818  2          if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0                               ! IF RETPKT IS NOT AVAILABLE
;   5819  2          then
;   5820  3              begin
;   5821  3              PRINTF (DBM22);                                                    ! "SEQUEN: RETPKT NOT AVAILABLE"
;   5822  3              PUT_PKT (.P_INDEX);
;   5823  3              return;
;   5824  3              end
;   5825  2          else
;   5826  3              begin
;   5827  3              SRC_ADDR = .IPKT_ADDR + 6;                                         ! SET UP COPY (SKIP OVER PKT DESC)
;   5828  3              R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);              ! START OF ALLOCATED RETPKT
;   5829  3
;   5830  3              incr COUNT from 1 to RP_LEN do
;   5831  4                  begin
;   5832  4                  .DST_ADDR = ..SRC_ADDR;                                        ! COPY 1 WORD
;   5833  4                  DST_ADDR = .DST_ADDR + 2;                                      ! ADVANCE DESTINATION ADDR
;   5834  4                  SRC_ADDR = .SRC_ADDR + 2;                                      ! ADVANCE SOURCE ADDR
;   5835  4
;   5836  5                  if .IPKT_ADDR [OPCODE] eql (OP_ONL or OP_END)                  ! IF THIS IS THE ONLINE END MESSAGE
;   5837  4                  then
;   5838  4                      if .COUNT eql 10                                           ! SKIP OVER RESERVED WORDS
;   5839  4                      then
;   5840  4                          SRC_ADDR = .SRC_ADDR + 4;                              !       IN ONLINE END - MESSAGE
;   5841  3                  end;                                                           ! COPY LOOP
;   5842  3
;   5843  3              R_ADDR [CTLR] = .ICTLR;                                            ! LOAD CONTROLLER NUMBER INTO PKT
;   5844  3
;   5845  3              if .P_INDEX geq 0                                                  ! IF ASSOC. CMD PKT WAS FOUND
;   5846  3              then
;   5847  3
;   5848  3                  if (.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or              ! IF END MESSAGE IS
;   5849  3                     (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END)) or             !       READ, WRITE, OR
;   5850  4                     (.IPKT_ADDR [OPCODE] eql (OP_ACC or OP_END))                !       ACCESS
;   5851  3                  then
;   5852  4                      begin
```

```
;   5853  4                   R_ADDR [CMDMOD] = .MSCP_PKT [.P_INDEX, MODIFY];          ! COPY
;   5854  4                   R_ADDR [CBCNT_LO] = .MSCP_PKT [.P_INDEX, BC_LO];         ! RELEVANT
;   5855  4                   R_ADDR [CBCNT_HI] = .MSCP_PKT [.P_INDEX, BC_HI];         ! FIELDS
;   5856  4                   R_ADDR [LBN_LO] = .MSCP_PKT [.P_INDEX, LBN_L];           ! FROM
;   5857  4                   R_ADDR [LBN_HI] = .MSCP_PKT [.P_INDEX, LBN_H];           ! COMMAND
;   5858  4                   R_ADDR [BUFF_0] = .MSCP_PKT [.P_INDEX, BUF_0];           ! PACKET
;   5859  4                   R_ADDR [BUFF_1] = .MSCP_PKT [.P_INDEX, BUF_1];           ! TO RETPKT
;   5860  3                   end;                                                     ! IF ENDCODE WAS READ/WRITE/ACCESS
;   5861  3
;   5862  3               IN_IODQ (.R_INDEX);                                          ! PUT RETPKT INDEX INTO IODQ
;   5863  2               end;                                                         ! IF RETPKT WAS ALLOCATED
;   5864  2
;   5865  2           if (.IPKT_ADDR [STATUS_CODE] neq  ST_SUC) or
;   5866  3              (.IPKT_ADDR [STATUS_SUBCODE] neq 0)
;   5867  2           then
;   5868  2               LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_OCCURED                ! SAVE ERROR CONDITION
;   5869  2           else
;   5870  2               LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_NOT_OCCURED;           !
;   5871  2
;   5872  2           LAST_PKT [.ICTLR, LAST_CRN_LO] = .IPKT_ADDR [CRN_LO];            ! SAVE COMMAND REFERENCE NUMBER
;   5873  2           LAST_PKT [.ICTLR, LAST_CRN_HI] = .IPKT_ADDR [CRN_HI];            !
;   5874  2           SCAN_ERRLOG ();                                                  ! PRINT ANY ASSOCIATED ERROR-LOGS
;   5875  2
;   5876  2           if .P_INDEX geq 0                                                ! IF ASSOC CMD PKT WAS FOUND
;   5877  2           then
;   5878  2               PUT_PKT (.P_INDEX);                                          ! RETURN COMMAND PACKET TO POOL
;   5879  2
;   5880  1           end;                                                             ! ROUTINE DISK_RSP


                                        .SBTTL  SEQUEN RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          SEQUEN::JSR     R1,$SAVE5                       ;                                5747
000004  005746                          TST     -(SP)
000006  012746  177777                  MOV     #-1,-(SP)                       ; *,P.INDEX                      5762
000012  013701  000000G                  MOV     IPKT.ADDR,R1                    ;                                5773
000016  005002                          CLR     R2                              ; COUNT                          5771
000020  010246                  1$:     MOV     R2,-(SP)                        ; COUNT,*                        5773
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)          ;                                5774
000052  001024                          BNE     2$
000054  026011  000000G                  CMP     MSCP.PKT(R0),(R1)               ;                                5775
000060  001421                          BEQ     2$
000062  105760  000022G                  TSTB    MSCP.PKT+22(R0)                 ;                                5776
000066  100416                          BMI     2$
000070  132760  000360  000010G         BITB    #360,MSCP.PKT+10(R0)            ;                                5777
000076  001012                          BNE     2$
000100  105761  000022                  TSTB    22(R1)                          ;                                5778
000104  100007                          BPL     2$
000106  116200  000000G                  MOVB    PKT.USE(R2),R0                  ; *(COUNT),*                     5779
```

```
000112  020037  000104                        CMP     R0,ICTLR
000116  001002                                BNE     2$
000120  010216                                MOV     R2,(SP)                 ; COUNT,P.INDEX                 5782
000122  000405                                BR      3$                      ;                               5781
000124  005202                        2$:     INC     R2                      ; COUNT                         5771
000126  020227  000013                        CMP     R2,#13                  ; COUNT,*
000132  003732                                BLE     1$
000134  005716                                TST     (SP)                    ; P.INDEX                       5786
000136  002013                        3$:     BGE     4$
000140  016146  000012                        MOV     12(R1),-(SP)            ;                               5789
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$                      ;                               5790
000166  011646                        4$:     MOV     (SP),-(SP)              ; P.INDEX,*                     5793
000170  012746  000106                        MOV     #106,-(SP)
000174  004737  000000G                       JSR     PC,BL$MUL
000200  010001                                MOV     R0,R1
000202  022626                                CMP     (SP)+,(SP)+
000204  013700  000000G                       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)            ;                               5795
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$:     MOV     IPKT.ADDR,R0            ;                               5797
000302  126027  000022  000241                CMPB    22(R0),#241
000310  001404                                BEQ     6$
000312  126027  000022  000242                CMPB    22(R0),#242             ;                               5798
000320  001045                                BNE     8$
000322  012702  000024                6$:     MOV     #24,R2                  ;                               5799
000326  060002                                ADD     R0,R2
000330  132712  000037                        BITB    #37,(R2)
000334  001037                                BNE     8$
000336  032712  177740                        BIT     #177740,(R2)            ;                               5800
000342  001034                                BNE     8$
000344  026160  000026G 000026                CMP     MSCP.PKT+26(R1),26(R0)  ;                               5801
000352  001004                                BNE     7$
000354  026160  000030G 000030                CMP     MSCP.PKT+30(R1),30(R0)  ;                               5802
```

```
000362  001424                          BEQ     8$
000364  016046  000012          7$:     MOV     12(R0),-(SP)            ;                               5806
000370  016046  000014                  MOV     14(R0),-(SP)
000374  016046  000026                  MOV     26(R0),-(SP)
000400  016046  000030                  MOV     30(R0),-(SP)
000404  016146  000026G                 MOV     MSCP.PKT+26(R1), (SP)
000410  016146  000030G                 MOV     MSCP.PKT+30(R1),-(SP)
000414  012746  000000G                 MOV     #DBM112, (SP)
000420  012746  000007                  MOV     #7,-(SP)
000424  010600                          MOV     SP,R0                   ; SP,*
000426  104417                          TRAP    17
000430  062706  000020                  ADD     #20,SP
000434  132761  000400  000005G  8$:     BITB    #400,MSCP.PKT+5(R1)     ;                               5808
000442  001422                          BEQ     10$
000444  016146  000012G                 MOV     MSCP.PKT+12(R1),-(SP)   ;                               5811
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,*                     5812
000474  004737  000000G                 JSR     PC,PUT.PKT
000500  062706  000010          9$:     ADD     #10,SP                  ;                               5813
000504  000137  030160'                 JMP     21$                     ;                               5810
000510  112761  000001  000005G  10$:    MOVB    #1,MSCP.PKT+5(R1)       ;                               5816
000516  013746  000104'                 MOV     ICTLR,-(SP)             ;                               5818
000522  004737  000000G                 JSR     PC,GET.RETPKT
000526  010066  000004                  MOV     R0,4(SP)                ; *,R.INDEX
000532  005726                          TST     (SP)+
000534  005766  000002                  TST     2(SP)                   ; R.INDEX
000540  002007                          BGE     11$
000542  012746  000000G                 MOV     #DBM22,-(SP)            ;                               5821
000546  012746  000001                  MOV     #1,-(SP)
000552  010600                          MOV     SP,R0                   ; SP,*
000554  104417                          TRAP    17
000556  000563                          BR      19$                     ;                               5822
000560  013704  000000G          11$:    MOV     IPKT.ADDR,R4            ; *,SRC.ADDR                    5827
000564  062704  000006                  ADD     #6,R4                   ; *,SRC.ADDR
000570  016646  000002                  MOV     2(SP),-(SP)             ; R.INDEX,*                     5828
000574  012746  000054                  MOV     #54,-(SP)
000600  004737  000000G                 JSR     PC,BL$MUL
000604  062700  000000G                 ADD     #RETPKT,R0
000610  010005                          MOV     R0,R5                   ; *,DST.ADDR
000612  013702  000000G                 MOV     IPKT.ADDR,R2            ;                               5836
000616  012703  000001                  MOV     #1,R3                   ; *,COUNT                       5830
000622  012425                  12$:    MOV     (R4)+,(R5)+             ; SRC.ADDR,DST.ADDR             5832
000624  126227  000022  000211          CMPB    22(R2),#211             ;                               5836
000632  001005                          BNE     13$
000634  020327  000012                  CMP     R3,#12                  ; COUNT,*                       5838
000640  001002                          BNE     13$
000642  062704  000004                  ADD     #4,R4                   ; *,SRC.ADDR                    5840
000646  005203                  13$:    INC     R3                      ; COUNT                         5830
000650  020327  000026                  CMP     R3,#26                  ; COUNT,*
```

```
000654  003762                          BLE     12$
000656  013703  000104'                 MOV     ICTLR,R3                ;                                       5843
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                               5845
000704  002441                          BLT     15$
000706  005003                          CLR     R3                      ;                                       5848
000710  156203  000022                  BISB    22(R2),R3
000714  020327  000241                  CMP     R3,#241
000720  001406                          BEQ     14$
000722  020327  000242                  CMP     R3,#242                 ;                                       5849
000726  001403                          BEQ     14$
000730  020327  000220                  CMP     R3,#220                 ;                                       5850
000734  001025                          BNE     15$
000736  016160  000024G 000012  14$:    MOV     MSCP.PKT+24(R1),12(R0)  ; *,*(R.ADDR)                           5853
000744  016160  000026G 000044          MOV     MSCP.PKT+26(R1),44(R0)  ; *,*(R.ADDR)                           5854
000752  016160  000030G 000046          MOV     MSCP.PKT+30(R1),46(R0)  ; *,*(R.ADDR)                           5855
000760  016160  000046G 000050          MOV     MSCP.PKT+46(R1),50(R0)  ; *,*(R.ADDR)                           5856
000766  016160  000050G 000052          MOV     MSCP.PKT+50(R1),52(R0)  ; *,*(R.ADDR)                           5857
000774  016160  000032G 000024          MOV     MSCP.PKT+32(R1),24(R0)  ; *,*(R.ADDR)                           5858
001002  016160  000034G 000026          MOV     MSCP.PKT+34(R1),26(R0)  ; *,*(R.ADDR)                           5859
001010  016616  000006          15$:    MOV     6(SP),(SP)              ; R.INDEX,*                             5862
001014  004737  000000G                 JSR     PC,IN.IODQ
001020  005726                          TST     (SP)+                   ;                                       5826
001022  013716  000104'                 MOV     ICTLR,(SP)              ;                                       5868
001026  012746  000006                  MOV     #6,-(SP)
001032  004737  000000G                 JSR     PC,BL$MUL
001036  013701  000000G                 MOV     IPKT.ADDR,R1            ;                                       5865
001042  012703  000024                  MOV     #24,R3
001046  060103                          ADD     R1,R3
001050  132713  000037                  BITB    #37,(R3)
001054  001003                          BNE     16$
001056  032713  177740                  BIT     #177740,(R3)            ;                                       5866
001062  001404                          BEQ     17$
001064  012760  000001  000120'  16$:    MOV     #1,LAST.PKT(R0)         ;                                       5868
001072  000402                          BR      18$                     ;                                       5865
001074  005060  000120'          17$:    CLR     LAST.PKT(R0)            ;                                       5870
001100  016160  000012  000122'  18$:    MOV     12(R1),LAST.PKT+2(R0)   ;                                       5872
001106  016160  000014  000124'          MOV     14(R1),LAST.PKT+4(R0)   ;                                       5873
001114  004737  000000V                 JSR     PC,SCAN.ERRLOG          ;                                       5874
001120  005766  000004                  TST     4(SP)                   ; P.INDEX                               5876
001124  002404                          BLT     20$
001126  016616  000004          19$:    MOV     4(SP),(SP)              ; P.INDEX,*                             5878
001132  004737  000000G                 JSR     PC,PUT.PKT
001136  022626                  20$:    CMP     (SP)+,(SP)+             ;                                       5762
001140  022626                  21$:    CMP     (SP)+,(SP)+             ;                                       5747
001142  000207                          RTS     PC

; Routine Size:  306 words,     Routine Base:  $CODE$ + 27020
; Maximum stack depth per invocat'on:  18 words
```

```
;  5881  1     GLOBAL routine SCAN_ERRLOG : novalue =
;  5882  1
;  5883  1     !+
;  5884  1     ! THIS ROUTINE SCANS THE ERROR-LOG SAVE AREA AND PRINTS ANY ERROR-LOGS RECEIVED FOR THE ASSOCIATED RESPONSE
;  5885  1     !
;  5886  1
;  5887  2         begin
;  5888  2
;  5889  2         local
;  5890  2             TEMP_UNIT,
;  5891  2             SFT_ERR_PRINTED : byte initial (byte (FALSE));
;  5892  2
;  5893  2         incr index from 0 to EP_CNT do                                          ! SCAN ERROR LOG PACKET SAVE AREA
;  5894  3             begin
;  5895  3
;  5896  3             if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;  5897  3                (.ELOG_PKT [.index, EL_CRN_LO] eql .IPKT_ADDR [CRN_LO]) and
;  5898  3                (.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  5899  4                (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;  5900  3             then
;  5901  4                 begin                                                        ! ERROR-LOG PENDING THIS RESPONSE
;  5902  4
;  5903  4                 if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED      ! IF SOFT ERROR OCCURED
;  5904  4                 then
;  5905  4
;  5906  4                     if .ELOG_PKT [.index, EL_FORMAT] lequ 4
;  5907  4                     then
;  5908  5                         begin
;  5909  5                         SOFT_ERROR (.index);                               ! UPAIE SOFT ERROR COUNT
;  5910  5                         TEMP_UNIT = .L$LUN;                                ! SAVE UNIT NUMBER AS KNOWN TO DRS
;  5911  5
;  5912  5                         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR   1) * UNIT SIZE + OF_UN) by UNIT_SIZE do
;  5913  5
;  5914  5                             if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_PKT [.index, EL_DK_NUM]) and
;  5915  6                                (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;  5916  5                             then
;  5917  6                                 begin
;  5918  6                                 L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D UNIT];    ! CORECT UNIT NO. FOR ERROR MESSAGE
;  5919  6                                 exitloop;
;  5920  5                                 end;
;  5921  5
;  5922  5                         case .ELOG_PKT [.index, EL_FORMAT] from 0 to 4 of
;  5923  5                             set
;  5924  5
;  5925  5                             [0]:    if .APT_MODE                           ! CONTROLLER ERROR
;  5926  5                                     then
;  5927  5                                         ERR_SOFT_RTNE_APT (50, .index)
;  5928  5                                     else
;  5929  5                                         ERR_SOFT_RTNE (50);
;  5930  5
;  5931  5                             [1]:    if .APT_MODE                           ! HOST MEMORY ACCESS ERROR
;  5932  5                                     then
;  5933  5                                         ERR_SOFT_RTNE_APT (51, .index)
```

```
;  5934  5                                    else
;  5935  5                                        ERR_SOFT_RTNE (51);
;  5936  5
;  5937  5                            [2]:    if .APT_MODE                                    ! DISK TRANSFER ERROR
;  5938  5                                    then
;  5939  5                                        ERR_SOFT_RTNE_APT (52, .index)
;  5940  5                                    else
;  5941  5                                        ERR_SOFT_RTNE (52);
;  5942  5
;  5943  5                            [3]:    if .APT_MODE                                    ! SDI ERROR
;  5944  5                                    then
;  5945  5                                        ERR_SOFT_RTNE_APT (53, .index)
;  5946  5                                    else
;  5947  5                                        ERR_SOFT_RTNE (53);
;  5948  5
;  5949  5                            [4]:    if .APT_MODE                                    ! SMALL DISK ERROR
;  5950  5                                    then
;  5951  5                                        ERR_SOFT_RTNE_APT (54, .index)
;  5952  5                                    else
;  5953  5                                        ERR_SOFT_RTNE (54);
;  5954  5                            tes;
;  5955  5
;  5956  5                        L$LUN = .TEMP_UNIT;                                         ! RESTORE UNIT NUMBER
;  5957  5                        SFT_ERR_PRINTED = TRUE;                                     ! SOFT ERROR PRINTOUT OCCURED
;  5958  5                        end
;  5959  4                    else
;  5960  4                        PRINTF (DBM109, .ELOG_PKT [.index, EL_FORMAT]);             ! UNKNOWN ERROR-LOG FORMAT
;  5961  4
;  5962  5                if not (.SFT_ERR_PRINTED)
;  5963  4                then
;  5964  4                    PRINTB (CRLF);                                                  ! EXTRA CARRIAGE-RETURN/LINE FEED
;  5965  4
;  5966  4                    EMS_EL (.index);                                                ! PRINT ERROR-LOG CONTENTS
;  5967  4                    end
;  5968  3            else
;  5969  3
;  5970  3                if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;  5971  4                   ((.ELOG_PKT [.index, EL_CRN_HI] lssu .IPKT_ADDR [CRN_HI]) or
;  5972  5                    ((.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  5973  3                     (.ELOG_PKT [.index, EL_CRN_LO] lssu .IPKT_ADDR [CRN_LO]))) and
;  5974  4                   (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;  5975  3                then
;  5976  4                    begin
;  5977  4                    PRINTB (CRLF);                                                  ! CARRIAGE-RETURN/LINE-FEED
;  5978  4                    EMS_EL (.index);                                                ! PRINT ERROR-LOG CONTENTS
;  5979  3                    end;
;  5980  3
;  5981  2            end;                                                                    ! ERROR-LOG SAVE AREA SCAN
;  5982  2
;  5983  1        end;

                                    .SBTTL  SCAN.ERRLOG RDRX INTERRUPT SERVICE ROUTINES
```

M3


```
000000  004137  000000G                SCAN.ERRLOG::
                                                JSR     R1,$SAVE5                       ;                               5881
000004  005746                                    TST     (SP)
000006  105005                                    CLRB    R5                              ; SFT.ERR.PRINTED               5887
000010  005002                                    CLR     R2                              ; INDEX                         5893
000012  010246                            1$:     MOV     R2,-(SP)                        ; INDEX,*                       5896
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                    ;                               5897
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)          ;                               5898
000076  001003                                    BNE     2$
000100  126327  000001  000001                    CMPB    1(R3),#1                        ;                               5899
000106  001402                            2$:     BEQ     3$
000110  000137  030724'                           JMP     25$
000114  013746  000104                    3$:     MOV     ICTLR,-(SP)                     ;                               5903
000120  012746  000006                            MOV     #6,-(SP)
000124  004737  000000G                           JSR     PC,BL$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              ;                               5906
000146  101142                                    BHI     21$
000150  010246                                    MOV     R2,-(SP)                        ; INDEX,*                       5909
000152  004737  000000V                           JSR     PC,SOFT.ERROR
000156  013766  000000G 000002                    MOV     L$LUN,2(SP)                     ; *,TEMP.UNIT                   5910
000164  012703  000006                            MOV     #6,R3                           ; *,OFFSET                      5912
000170  010300                            4$:     MOV     R3,R0                           ; OFFSET,*                      5914
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)                     ;                               5915
000220  001407                                    BEQ     5$
000222  011004                                    MOV     (R0),R4                         ;                               5918
000224  000304                                    SWAB    R4
000226  042704  177760                            BIC     #177760,R4
000232  010437  000000G                           MOV     R4,L$LUN
000236  000405                                    BR      6$                              ;                               5917
000240  062703  000024                    5$:     ADD     #24,R3                          ; *,OFFSET                      5912
```

N7


```
000244  020327  000102                    CMP     R3,#102                         ; OFFSET,*
000250  003747                            BLE     4$
000252  005000                    6$      CLR     R0                              .                       5925
000254  153700  001162                    BISB    APT.MODE,R0
000260  116101  000016G                   MOVB    ELOG.PKT+16(R1),R1              ;                       5922
000264  042701  177400                    BIC     #177400,R1
000270  006301                            ASL     R1
000272  066107  000000                    ADD     P.AAA(R1),PC                    ; Case d spatch
000276  032700  000001            8$:     BIT     #1,R0                           ;                       5925
000302  001403                            BEQ     9$
000304  012716  000062                    MOV     #62,(SP)                        ;                       5927
000310  000442                            BR      17$
000312  012716  000062            9$:     MOV     #62,(SP)                        ;                       5929
000316  000446                            BR      19$
000320  032700  000001            10$:    BIT     #1,R0                           ;                       5931
000324  001403                            BEQ     11$
000326  012716  000063                    MOV     #63,(SP)                        ;                       5933
000332  000431                            BR      17$
000334  012716  000063            11$:    MOV     #63,(SP)                        ;                       5935
000340  000435                            BR      19$
000342  032700  000001            12$:    BIT     #1,R0                           ;                       5937
000346  001403                            BEQ     13$
000350  012716  000064                    MOV     #64,(SP)                        ;                       5939
000354  000420                            BR      17$
000356  012716  000064            13$:    MOV     #64,(SP)                        ;                       5941
000362  000424                            BR      19$
000364  032700  000001            14$.    BIT     #1,R0                           ;                       5943
000370  001403                            BEQ     15$
000372  012716  000065                    MOV     #65,(SP)                        ;                       5945
000376  000407                            BR      17$
000400  012716  000065            15$:    MOV     #65,(SP)                        .                       5947
000404  000413                            BR      19$
000406  006000                    16$:    ROR     R0                              ;                       5949
000410  103007                            BCC     18$
000412  012716  000066                    MOV     #66,(SP)                        ;                       5951
000416  010246                    17$:    MOV     R2,-(SP)                        ; INDEX,*
000420  004737  000000V                   JSR     PC,ERR.SOFT.RTNE.APT
000424  005726                            TST     (SP)+
000426  000404                            BR      20$                             ;                       5949
000430  012716  000066            18$:    MOV     #66,(SP)                        ;                       5953
000434  004737  000000V            19$:    JSR     PC,ERR.SOFT.RTNE
000440  016637  000002  000000G    20$:    MOV     2(SP),L$LUN                     ; TEMP.UNIT,*           5956
000446  112705  000001                    MOVB    #1,R5                           ; *,SFT.ERR.PRINTED     5957
000452  000412                            BR      22$                             ;                       5906
000454  005046                    21$:    CLR     -(SP)                           ;                       5960
000456  116116  000016G                   MOVB    ELOG.PKT+16(R1),(SP)
000462  012746  000000G                   MOV     #DBM109,-(SP)
000466  012746  000002                    MOV     #2,-(SP)
000472  010600                            MOV     SP,R0                           ; SP,*
000474  104417                            TRAP    17
000476  022626                            CMP     (SP)+,(SP)+
000500  005726                    22$:    TST     (SP)+                           ;                       5906
000502  032705  000001            23$:    BIT     #1,R5                           ; *,SFT.ERR.PRINTED     5962
```

```
000506  001007                          BNE     24$
000510  012746  000000G                MOV     #CRLF,-(SP)                     ;                               5964
000514  012746  000001                  MOV     #1,-(SP)
000520  010600                          MOV     SP,R0                           ; SP,#
000522  104414                          TRAP    14
000524  022626                          CMP     (SP)+,(SP)+
000526  010246                  24$:    MOV     R2,-(SP)                        ; INDEX,#                       5966
000530  004737  000000G                JSR     PC,EMS.EL
000534  005726                          TST     (SP)+                           ;                               5901
000536  000433                          BR      27$                             ;                               5896
000540  006004                  25$:    ROR     R4                              ;                               5970
000542  103031                          BCC     27$
000544  013700  000000G                MOV     IPKT.ADDR,R0                    ;                               5971
000550  026160  000010G 000014          CMP     ELOG.PKT+10(R1),14(R0)
000556  103405                          BLO     26$
000560  001022                          BNE     27$                             ;                               5972
000562  026160  000006G 000012          CMP     ELOG.PKT+6(R1),12(R0)           ;                               5973
000570  103016                          BHIS    27$
000572  126327  000001  000001  26$:    CMPB    1(R3),#1                        ;                               5974
000600  001012                          BNE     27$
000602  012746  000000G                MOV     #CRLF,-(SP)                     ;                               5977
000606  012746  000001                  MOV     #1,-(SP)
000612  010600                          MOV     SP,R0                           ; SP,#
000614  104414                          TRAP    14
000616  010216                          MOV     R2,(SP)                         ; INDEX,#                       5978
000620  004737  000000G                JSR     PC,EMS.EL
000624  022626                          CMP     (SP)+,(SP)+                     ;                               5976
000626  005202                  27$:    INC     R2                              ; INDEX                         5893
000630  020227  000014                  CMP     R2,#14                          ; INDEX,#
000634  003002                          BGT     28$
000636  000137  030176'                 JMP     1$
000642  005726                  28$:    TST     (SP)+                           ;                               5881
000644  000207                          RTS     PC

; Routine Size:  211 words,    Routine Base:  $CODE$ + 30164
; Maximum stack depth per invocation:  12 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:                                          ; CASE Table for SCAN.ERRLOG+0272       5922
000000  000000                  7$:     .WORD   0                               ; [8$]
000002  000022                          .WORD   22                              ; [10$]
000004  000044                          .WORD   44                              ; [12$]
000006  000066                          .WORD   66                              ; [14$]
000010  000110                          .WORD   110                             ; [16$]
```

```
;    5984  1     GLOBAL routine DATAGM : novalue =
;    5985  1
;    5986  1     !+
;    5987  1     !       THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
;    5988  1     !       THE RDRX
;    5989  1     !
;    5990  1     !       IMPLICIT INPUTS:
;    5991  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING ERROR LOG
;    5992  1     !                       MESSAGE
;    5993  1     !               ICST_ADDR   ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;    5994  1     !
;    5995  1
;    5996  2         begin
;    5997  2
;    5998  2         local
;    5999  2             index : signed word initial (-1),
;    6000  2             SAVE_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;    6001  2             SRC_ADDR,
;    6002  2             DST_ADDR,
;    6003  2             TEMP_UNIT,
;    6004  2             SFT_ERR_PRINTED : byte initial (byte (FALSE)),
;    6005  2             PACKET_LEN : word;
;    6006  2
;    6007  2     !
;    6008  2     ! FIND AN EMPTY SLOT IN THE ERROR-LOG PACKET SAVE AREA
;    6009  2     !
;    6010  2
;    6011  2         incr COUNT from 0 to EP_CNT - 1 do
;    6012  2
;    6013  2             if .ELOG_PKT [.COUNT, EL_CONTENTS] eql EMPTY                ! IF EMPTY SLOT FOUND
;    6014  2             then
;    6015  3                 begin
;    6016  3                 index = .COUNT;                                         ! SAVE INDEX INTO THE SAVE AREA
;    6017  3                 exitloop;
;    6018  2                 end;
;    6019  2
;    6020  2         if .index lss 0
;    6021  2         then
;    6022  2             index = EP_CNT;                                             ! IF NO SLOT FOUND, USE LAST SPARE SLOT
;    6023  2
;    6024  2     !
;    6025  2     ! SAVE THE PACKET CONTENTS
;    6026  2     !
;    6027  2
;    6028  2         SAVE_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                   ! ADDRESS OF THE SAVE AREA
;    6029  2         SAVE_ADDR [EL_CONTENTS ] = FULL;                                ! MARK IT FULL
;    6030  2         SAVE_ADDR [EL_CNTR] = .ICTLR;                                   ! OWNERSHIP
;    6031  2         SRC_ADDR = .IPKT_ADDR + 6;                                      ! SETUP COPY ADDRESSES
;    6032  2         DST_ADDR = .SAVE_ADDR + 2;                                      !
;    6033  2         PACKET_LEN = ((.IPKT_ADDR [MSGLEN] + 1) / 2) * 2;               ! LENGTH OF ERROR-LOG INCLUDING ENVELOPE
;    6034  2
;    6035  2         if .PACKET_LEN gtru EP_LEN - 1
;    6036  2         then
```

```
;  6037  2              PACKET_LEN = EP_LEN   1;                                        ! ADJUST LENGTH, IF TOO LONG
;  6038  2
;  6039  2          incr COUNT from 1 to .PACKET_LEN do
;  6040  3              begin
;  6041  3              .DST_ADDR = ..SRC_ADDR;                                         ! COPY A WORD
;  6042  3              SRC_ADDR = .SRC_ADDR + 2;                                       ! UPDATE ADDRESS POINTERS
;  6043  3              DST_ADDR = .DST_ADDR + 2;                                       !
;  6044  3              end;
;  6045  2
;  6046  2      !
;  6047  2      ! CHECK IF THE CORRESPONDING RESPONSE HAS ALREADY BEEN RECEIVED
;  6048  2      !
;  6049  2
;  6050  2          if (.SAVE_ADDR [EL_CRN_LO] eql .LAST_PKT [.ICTLR, LAST_CRN_LO]) and
;  6051  3             (.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI])
;  6052  2          then
;  6053  3              begin                                                           ! LOG REFERS TO THE LAST RESPONSE RECEIVED
;  6054  3
;  6055  3              if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED         ! IF SOFT ERROR HAD OCCURED
;  6056  3              then
;  6057  3
;  6058  3                  if .SAVE_ADDR [EL_FORMAT] lequ 4
;  6059  3                  then
;  6060  4                      begin
;  6061  4                      SOFT_ERROR (.index);                                    ! UPDATE SOFT ERROR COUNT
;  6062  4                      TEMP_UNIT = .L$LUN;                                     ! SAVE UNIT NUMBER AS KNOWN TO DRS
;  6063  4
;  6064  4                      incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  6065  4
;  6066  4                      if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .SAVE_ADDR [EL_DK_NUM]) and
;  6067  5                         (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;  6068  4                      then
;  6069  5                          begin
;  6070  5                          L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT]; ! CORRECT UNIT NUMBER FOR ERROR MESSAGE
;  6071  5                          exitloop;
;  6072  4                          end;
;  6073  4
;  6074  4                      case .SAVE_ADDR [EL_FORMAT] from 0 to 4 of
;  6075  4                          set
;  6076  4
;  6077  4
;  6078  4                          [0] :       if .APT_MODE                        ! CONTROLLER ERROR
;  6079  4                                      then
;  6080  4                                          ERR_SOFT_RTNE_APT (50, .index)
;  6081  4                                      else
;  6082  4                                          ERR_SOFT_RTNE (50);
;  6083  4
;  6084  4                          [1] :       if .APT_MODE                        ! HOST MEMORY ACCESS ERROR
;  6085  4                                      then
;  6086  4                                          ERR_SOFT_RTNE_APT (51, .index)
;  6087  4                                      else
;  6088  4                                          ERR SOFT_RTNE (51);
;  6089  4
```

```
;  6090  4                        [2] :       f .APT_MODE                                  ! DISK TRANSFER ERROR
;  6091  4                                    then
;  6092  4                                        ERR_SOFT_RTNE_APT (52, .index)
;  6093  4                                    else
;  6094  4                                        ERR_SOFT_RTNE (52);
;  6095  4
;  6096  4                        [3] :       if .APT_MODE                                 ! SDI ERROR
;  6097  4                                    then
;  6098  4                                        ERR_SOFT_RTNE_APT (53,  index)
;  6099  4                                    else
;  6100  4                                        ERR_SOFT_RTNE (53);
;  6101  4
;  6102  4                        [4] :       if .APT_MODE                                 ! SMALL DISK ERROR
;  6103  4                                    then
;  6104  4                                        ERR_SOFT_RTNE_APT (54, .index)
;  6105  4                                    else
;  6106  4                                        ERR_SOFT_RTNE (54);
;  6107  4                        tes;
;  6108  4
;  6109  4
;  6110  4                    L$LUN = .TEMP_UNIT;                                          ! RESTORE UNIT NUMBER
;  6111  4                    SFT_ERR_PRINTED = TRUE;                                      ! SOFT ERROR PRINTOUT OCCURED
;  6112  4                    end
;  6113  4
;  6114  3                else
;  6115  3                    PRINTF (D8M109, .SAVE_ADDR [EL_FORMAT]);                     ! ERROR LOG FORMAT UNKNOWN
;  6116  3
;  6117  4            if not (.SFT_ERR_PRINTED)
;  6118  4
;  6119  3            then
;  6120  3                PRINTB (CRLF);                                                   ! EXTRA CARRIEGE-RETURN/LINE-FEED
;  6121  3
;  6122  3                EMS_EL (.index);                                                 ! PRINT PACKET CONTENTS
;  6123  3                end                                                              ! CORRESPONDING RESPONSE RECEIVED
;  6124  3
;  6125  2        else
;  6126  2
;  6127  2            if (.SAVE_ADDR [EL_CRN_HI] lssu .LAST_PKT [.ICTLR, LAST_CRN_HI]) or
;  6128  3               ((.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI]) and
;  6129  3                (.SAVE_ADDR [EL_CRN_LO] lssu .LAST_PKT [.ICTLR, LAST_CRN_LO]))
;  6130  3
;  6131  2            then
;  6132  3                begin                                                            ! LOG REFERS TO SOME PREVIOUS RESPONSE
;  6133  3                PRINTB (CRLF);                                                   ! CARRIAGE-RETURN/LINE-FEED
;  6134  3                EMS_EL (.index);                                                 ! PRINT PACKET CONTENTS
;  6135  2                end;
;  6136  2
;  6137  1        end;


                                        .SBTTL  DATAGM RDRX INTERRUPT SERVICE ROUTINES
031032                                  .PSECT  $CODE$,  RO
```

```
000000  004137  000000G            DATAGM::JSR     R1,$SAVE5                       ;                                       5984
000004  012704  177777                     MOV     #-1,R4                          ; *,INDEX                               5996
000010  105046                       CLRB    -(SP)                           ; SFT.ERR.PRINTED
000012  005001                       CLR     R1                              ; COUNT                                 6011
000014  010146                 1$:   MOV     R1,-(SP)                        ; COUNT,*                               6013
000016  012746  000102               MOV     #102,-(SP)
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                           6016
000040  000405                       BR      3$                              ;                                       6015
000042  005201                 2$:   INC     R1                              ; COUNT                                 6011
000044  020127  000013               CMP     R1,#13                          ; COUNT,*
000050  003761                       BLE     1$
000052  005704                       TST     R4                              ; INDEX                                 6020
000054  002002                 3$:   BGE     4$
000056  012704  000014               MOV     #14,R4                          ; *,INDEX                               6022
000062  010446                 4$:   MOV     R4,-(SP)                        ; INDEX,*                               6028
000064  012746  000102               MOV     #102,-(SP)
000070  004737  000000G              JSR     PC,BL$MUL
000074  062700  000000G              ADD     #ELOG.PKT,R0
000100  010001                       MOV     R0,R1                           ; *,SAVE.ADDR
000102  111761  000001               MOVB    (PC),1(R1)                      ; *,*(SAVE.ADDR)                        6029
000106  113711  000104'              MOVB    ICTLR,(R1)                      ; *,SAVE.ADDR                           6030
000112  013700  000000G              MOV     IPKT.ADDR,R0                    ;                                       6031
000116  012705  000006               MOV     #6,R5                           ; *,SRC.ADDR
000122  060005                       ADD     R0,R5                           ; *,SRC.ADDR
000124  012703  000002               MOV     #2,R3                           ; *,DST.ADDR                            6032
000130  060103                       ADD     R1,R3                           ; SAVE.ADDR,DST.ADDR
000132  016016  000006               MOV     6(R0),(SP)                      ;                                       6033
000136  005216                       INC     (SP)
000140  012746  000002               MOV     #2,-(SP)
000144  004737  000000G              JSR     PC,BL$DIV
000150  062700  000002               ADD     #2,R0
000154  020027  000040               CMP     R0,#40                          ; PACKET.LEN,*                          6035
000160  101402                       BLOS    5$
000162  012700  000040               MOV     #40,R0                          ; *,PACKET.LEN                          6037
000166  005002                 5$:   CLR     R2                              ; COUNT                                 6039
000170  000401                       BR      7$
000172  012523                 6$:   MOV     (R5)+,(R3)+                     ; SRC.ADDR,DST.ADDR                     6041
000174  005202                 7$:   INC     R2                              ; COUNT                                 6039
000176  020200                       CMP     R2,R0                           ; COUNT,PACKET.LEN
000200  003774                       BLE     6$
000202  013716  000104'              MOV     ICTLR,(SP)                      ;                                       6050
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  031674'              JMP     30$
000234  026160  000010  000124'  8$:  CMP     10(R1),LAST.PKT+4(R0)           ; *(SAVE.ADDR),*                        6051
000242  001402                       BEQ     9$
```

```
000244  000137  031702                    JMP     31$
000250  005760  000120'          9$:      TST     LAST.PKT(R0)                ;                                   6055
000254  001153                            BNE     28$
000256  005003                            CLR     R3                          ;                                   6058
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,*                           6061
000274  004737  000000V                   JSR     PC,SOFT.ERROR
000300  013705  000000G                   MOV     L$LUN,R5                    ; *,TEMP.UNIT                       6062
000304  012702  000006                    MOV     #6,R2                       ; *,OFFSET                          6064
000310  010200                  10$:      MOV     R2,R0                       ; OFFSET,*                          6066
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)                 ;                                   6067
000340  001407                            BEQ     11$
000342  011046                            MOV     (R0),-(SP)                  ;                                   6070
000344  000316                            SWAB    (SP)
000346  042716  177760                    BIC     #177760,(SP)
000352  012637  000000G                   MOV     (SP)+,L$LUN
000356  000405                            BR      12$                         ;                                   6069
000360  062702  000024          11$:      ADD     #24,R2                      ; *,OFFSET                          6064
000364  020227  000102                    CMP     R2,#102                     ; OFFSET,*
000370  003747                            BLE     10$
000372  005000                  12$:      CLR     R0                          ;                                   6078
000374  153700  001162'                   BISB    APT.MODE,R0
000400  006303                            ASL     R3                          ;                                   6074
000402  066307  000012'                   ADD     P.AAB(R3),PC                ; Case dispatch
000406  032700  000001          14$:      BIT     #1,R0                       ;                                   6078
000412  001403                            BEQ     15$
000414  012716  000062                    MOV     #62,(SP)                    ;                                   6080
000420  000442                            BR      23$
000422  012716  000062          15$:      MOV     #62,(SP)                    ;                                   6082
000426  000446                            BR      25$
000430  032700  000001          16$:      BIT     #1,R0                       ;                                   6084
000434  001403                            BEQ     17$
000436  012716  000063                    MOV     #63,(SP)                    ;                                   6086
000442  000431                            BR      23$
000444  012716  000063          17$:      MOV     #63,(SP)                    ;                                   6088
000450  000435                            BR      25$
000452  032700  000001          18$:      BIT     #1,R0                       ;                                   6090
000456  001403                            BEQ     19$
000460  012716  000064                    MOV     #64,(SP)                    ;                                   6092
000464  000420                            BR      23$
000466  012716  000064          19$:      MOV     #64,(SP)                    ;                                   6094
000472  000424                            BR      25$
000474  032700  000001          20$:      BIT     #1,R0                       ;                                   6096
000500  001403                            BEQ     21$
000502  012716  000065                    MOV     #65,(SP)                    ;                                   6098
```

```
000506  000407                          BR      23$
000510  012716  000065         21$:     MOV     #65,(SP)                ;                               6100
000514  000413                          BR      25$
000516  006000                 22$:     ROR     R0                      ;                               6102
000520  103007                          BCC     24$
000522  012716  000066                  MOV     #66,(SP)                ;                               6104
000526  010446                 23$:     MOV     R4,-(SP)                ; INDEX,*
000530  004737  000000V                 JSR     PC,ERR.SOFT.RTNE.A^T
000534  005726                          TST     (SP)+
000536  000404                          BR      26$                     ;                               6102
000540  012716  000066         24$:     MOV     #66,(SP)                ;                               6106
000544  004737  000000V         25$:     JSR     PC,ERR.SOFT.RTNE
000550  010537  000000G         26$:     MOV     R5,L$LUN                ; TEMP.UNIT,*                   6110
000554  112766  000001  000006          MOVB    #1,6(SP)                ; *.SFT.ERR.PRINTED             6111
000562  000410                          BR      28$                     ;                               6058
000564  010316                 27$:     MOV     R3,(SP)                 ;                               6115
000566  012746  000000G                 MOV     #OBM10$,-(SP)
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$:     BIT     #1,6(SP)                ; *.SFT.ERR.PRINTED             6117
000612  001007                          BNE     29$
000614  012716  000000G                 MOV     #CRLF,(SP)              ;                               6120
000620  012746  000001                  MOV     #1,-(SP)
000624  010600                          MOV     SP,R0                   ; SP,*
000626  104414                          TRAP    14
000630  005726                          TST     (SP)+
000632  010416                 29$:     MOV     R4,(SP)                 ; INDEX,*                       6122
000634  004737  000000G                 JSR     PC,EMS.EL
000640  000426                          BR      33$                     ;                               6050
000642  026160  000010  000124'  30$:     CMP     10(R1),LAST.PKT+4(R0)   ; *(SAVE.ADDR),*                6127
000650  103410                 31$:     BLO     32$                                                     6128
000652  026160  000010  000124'          CMP     10(R1),LAST.PKT+4(R0)   ; *(SAVE.ADDR),*
000660  001016                          BNE     33$                                                     6129
000662  026160  000006  000122'          CMP     6(R1),LAST.PKT+2(R0)    ; *(SAVE.ADDR),*
000670  103012                          BHIS    33$                                                     6133
000672  012716  000000G         32$:     MOV     #CRLF,(SP)              ;
000676  012746  000001                  MOV     #1,-(SP)
000702  010600                          MOV     SP,R0                   ; SP,*
000704  104414                          TRAP    14
000706  010416                          MOV     R4,(SP)                 ; INDEX,*                       6134
000710  004737  000000G                 JSR     PC,EMS.EL
000714  005726                          TST     (SP)+                   ;                               6132
000716  062706  000010         33$:     ADD     #10,SP                  ;                               5984
000722  000207                          RTS     PC

; Routine Size:  234 words,    Routine Base:  $CODE$ + 31032
; Maximum stack depth per invocation:  14 words


000012                                  .PSECT  $PLIT$,  RO ,  D
```

```
                                P.AAB:
000012  000000                  13$:    .WORD   0               ; CASE Table for DATAGM+0402    6074
000014  000022                          .WORD   22              ; [14$]
000016  000044                          .WORD   44              ; [16$]
000020  000066                          .WORD   66              ; [18$]
000022  000110                          .WORD   110             ; [20$]
                                                                ; [22$]


;    6138  1
;    6139  1
;    6140  1
```

```
;   6141  1     GLOBAL routine SOFT_ERROR (index) : novalue =
;   6142  1
;   6143  1     !+
;   6144  1     !       THIS ROUTINE UPDATES THE SOFT ERROR COUNT IN THE TALLY TABLE FOR EACH
;   6145  1     !       ERROR LOG MESSAGE RECEIVED
;   6146  1     !
;   6147  1     !       IMPLICIT INPUTS:
;   6148  1     !               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;   6149  1     !-
;   6150  1
;   6151  2         begin
;   6152  2
;   6153  2         local
;   6154  2             FOUND: byte initial (byte (FALSE)),
;   6155  2             SOFT_OCCURED : byte initial (byte (FALSE)),
;   6156  2             UNIT: word,
;   6157  2             ERROR_CODE : byte,
;   6158  2             ERROR_SUB : word,
;   6159  2             RETRIES : word,
;   6160  2             TALLY_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;   6161  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;   6162  2
;   6163  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                       ! ADDR OF ERROR PKT
;   6164  2         ERROR_CODE = .ELOG_ADDR [EL_CODE];                                  ! ERROR CODE
;   6165  2         ERROR_SUB = .ELOG_ADDR [EL_SUBCODE];                                ! ERROR SUBCODE
;   6166  2
;   6167  2         if (BIT_TST (SWP_FLAGS, SWF_TRY)) and
;   6168  3            (.ELOG_ADDR [EL_FORMAT] eql 2)
;   6169  2         then
;   6170  2             RETRIES = .ELOG_ADDR [EL_RETRY]                                 ! COUNT EACH RETRY
;   6171  2         else
;   6172  2             RETRIES = 1;                                                    ! IGNORE RETRIES
;   6173  2
;   6174  2         if .RETRIES eql 0                                                   ! IN CASE OF A BUG
;   6175  2         then
;   6176  2             RETRIES = 1;
;   6177  2
;   6178  2         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;   6179  2
;   6180  2             if (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and     ! DISK TO UNIT NO.
;   6181  3                (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_ADDR [EL_DK_NUM])
;   6182  2             then
;   6183  3                 begin
;   6184  3                 FOUND = TRUE;
;   6185  3                 UNIT = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];          ! DISK'S UNIT NO.
;   6186  3                 exitloop;
;   6187  2                 end;
;   6188  2
;   6189  2     !   if (.ELOG_ADDR [EL_SUCCESS]) or
;   6190  2     !      (.ELOG_ADDR [EL_CONTINUE])
;   6191  2     !   then
;   6192  2             SOFT_OCCURED = TRUE;                                        ! SOFT ERROR FLAG
;   6193  2
```

```
;  6194  2          if .FOUND                                                                   ! IF UNIT FOUND
;  6195  2          then
;  6196  3              begin
;  6197  3              TALLY_ADDR = TALLY + (.UNIT * TALLY_LEM * 2);                           ! ADDR OF TALLY TBL
;  6198  3
;  6199  3              if .SOFT_OCCURED                                                        ! FOR SOFT ERRORRS
;  6200  3              then
;  6201  3                  selectoneu .ERROR_CODE of
;  6202  3                      set
;  6203  3
;  6204  3                      [ST_MFE]:       TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES;! SOFT-MEDIA FORMAT
;  6205  3
;  6206  3                      [ST_DAT]:       if .ERROR_SUB eql 2                             ! SOFT DATA
;  6207  3                                      then
;  6208  3                                          TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
;  6209  3                                      else
;  6210  3                                          TALLY_ADDR [ERR_SFT_DAT] = .TALLY_ADDR [ERR_SFT_DAT] + .RETRIES;
;  6211  3
;  6212  3                      [ST_HST]:       TALLY_ADDR [ERR_SFT_HST] = .TALLY_ADDR [ERR_SFT_HST] + .RETRIES;! SOFT-HOST ACCESS
;  6213  3
;  6214  3                      [ST_CNT]:       C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + .RETRIES;
;  6215  3                                                                                      ! SOFT-CONTROLLER
;  6216  3
;  6217  3                      [ST_DRV]:       if .ERROR_SUB eql 3                             ! SOFT_DRIVE
;  6218  3                                      then
;  6219  3                                          TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR SFT_SEK] + .RETRIES
;  6220  3                                      else
;  6221  3                                          TALLY_ADDR [ERR_SFT_DRV] = .TALLY_ADDR [ERR_SFT DRV] + .RETRIES;
;  6222  3                  tes
;  6223  3          else
;  6224  3
;  6225  3              if (.ELOG_ADDR [EL_CRN_LO] eql 0) and
;  6226  4                 (.ELOG_ADDR [EL_CRN_HI] eql 0)
;  6227  3              then
;  6228  3                  selectoneu .ERROR_CODE of
;  6229  3                      set
;  6230  3
;  6231  3                      [ST_MFE]:   TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD SEK] + 1;      ! HARD-MEDIA FORMAT
;  6232  3
;  6233  3                      [ST_DAT]:   if .ERROR_SUB eql 2                                 ! HARD DATA
;  6234  3                                  then
;  6235  3                                      TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
;  6236  3                                  else
;  6237  3                                      TALLY_ADDR [ERR_HRD_DAT] = .TALLY_ADDR [ERR_HRD_DAT] + 1;
;  6238  3
;  6239  3                      [ST_HST]:   TALLY_ADDR [ERR_HRD_HST] = .TALLY_ADDR [ERR_HRD_HST] + 1;      ! HARD-HOST ACCESS
;  6240  3
;  6241  3                      [ST_CNT]:   C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR TBL [.ICTLR, C_ERR HRD] + 1;
;  6242  3                                                                                      ! HARD CONTROLLER
;  6243  3
;  6244  3                      [ST_DRV]:   if .ERROR_SUB eql 3                                 ! HARD-DRIVE
;  6245  3                                  then
;  6246  3                                      TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR HRD_SEK] + 1
```

```
;   6247  3                                   else
;   6248  3                                       TALLY_ADDR [ERR_HRD DRV] = .TALLY_ADDR [ERR HRD_DRV] + 1;
;   6249  3                           tes;
;   6250  3
;   6251  3               end
;   6252  2           else                                                          ! UNIT NOT FOUND
;   6253  2
;   6254  2               if .SOFT_OCCURED
;   6255  2               then
;   6256  2                   C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR TBL [.ICTLR, C_ERR SFT] + 1
;   6257  2               else
;   6258  2                   C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;   6259  2
;   6260  1           end;                                                          ! RTNE S WFT ERROR


                                    .SBTTL  SOFT.ERROR RDRX INTERRUPT SERVICE ROUTINES
031756                              .PSECT  $CODE$,  RO

000000  004137  000000G       SOFT.ERROR::
                                    JSR     R1,$SAVE5                   ;                                   6141
000004  005746                      TST     -(SP)
000006  105046                      CLRB    -(SP)                       ; FOUND                             6151
000010  105046                      CLRB    -(SP)                       ; SOFT.OCCURED
000012  016646  000024              MOV     24(SP),-(SP)                ; INDEX,*                           6163
000016  012746  000102              MOV     #102,-(SP)
000022  004737  000000G             JSR     PC,BL$MUL
000026  062700  000000G             ADD     #ELOG.PKT,R0
000032  010001                      MOV     R0,R1                       ; *,ELOG.ADDR
000034  116100  000020              MOVB    20(R1),R0                   ; *(ELOG.ADDR),*                    6164
000040  042700  177740              BIC     #177740,R0
000044  105003                      CLRB    R3                          ; ERROR.CODE
000046  050003                      BIS     R0,R3                       ; *,ERROR.CODE
000050  016105  000020              MOV     20(R1),R5                   ; *(ELOG.ADDR),ERROR.SUB            6165
000054  006205                      ASR     R5                          ; ERROR.SUB
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,R0                ;                                   6167
000076  042700  077777              BIC     #77777,R0
000102  020027  100000              CMP     R0,#-100000
000106  001010                      BNE     1$
000110  126127  000016  000002      CMPB    16(R1),#2                   ; *(ELOG.ADDR),*                    6168
000116  001004                      BNE     1$
000120  005004                      CLR     R4                          ; RETRIES                           6170
000122  156104  000051              BISB    51(R1),R4                   ; *(ELOG.ADDR),RETRIES
000126  000402                      BR      2$                          ;                                   6167
000130  012704  000001      1$:     MOV     #1,R4                       ; *,RETRIES                         6172
000134  005704              2$:     TST     R4                          ; RETRIES                           6174
000136  001002                      BNE     3$
000140  012704  000001              MOV     #1,R4                       ; *,RETRIES                         6176
```

```
000144  012702  000006              3$:     MOV     #6,R2                   ; *,OFFSET                      6178
000150  010200                      4$:     MOV     R2,R0                   ; OFFSET,*                      6180
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),*                6181
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                       6184
000210  011046                              MOV     (R0),-(SP)              ;                               6185
000212  000316                              SWAB    (SP)
000214  042716  177760                      BIC     #177760,(SP)
000220  012666  000010                      MOV     (SP)+,10(SP)            ; *,UNIT
000224  000405                              BR      6$                      ;                               6183
000226  062702  000024              5$:     ADD     #24,R2                  ; *,OFFSET                      6178
000232  020227  000102                      CMP     R2,#102                 ; OFFSET,*
000236  003744                              BLE     4$
000240  112766  000001  000004      6$:     MOVB    #1,4(SP)                ; *,SOFT.OCCURED                6192
000246  032766  000001  000006              BIT     #1,6(SP)                ; *,FOUND                       5194
000254  001002                              BNE     7$
000256  000137  032644'                      JMP     22$
000262  016616  000010              7$:     MOV     10(SP),(SP)             ; UNIT,*                        6197
000266  012746  000066                      MOV     #66,-(SP)
000272  004737  000000G                      JSR     PC,BL$MUL
000276  062700  000000G                      ADD     #TALLY,R0
000302  032766  000001  000006              BIT     #1,6(SP)                ; *,SOFT.OCCURED                6199
000310  001503                              BEQ     14$
000312  120327  000005                      CMPB    R3,#5                   , ERROR.CODE,*                  6204
000316  001462                              BEQ     12$
000320  120327  000010                      CMPB    R3,#10                  ; ERROR.CODE,*                  6206
000324  001022                              BNE     9$
000326  012702  000052                      MOV     #52,R2                  ;                               6208
000332  060002                              ADD     R0,R2                   ; TALLY.ADDR,*
000334  020527  000002                      CMP     R5,#2                   ; ERROR.SUB,*                   6206
000340  001005                              BNE     8$
000342  005001                              CLR     R1                      ;                               6208
000344  151201                              BISB    (R2),R1
000346  060401                              ADD     R4,R1                   ; RETRIES,*
000350  110112                              MOVB    R1,(R2)
000352  000543                              BR      21$                     ;                               6206
000354  005001              8$:     CLR     R1                      ;                               6210
000356  156201  000001                      BISB    1(R2),R1
000362  060401                              ADD     R4,R1                   ; RETRIES,*
000364  110162  000001                      MOVB    R1,1(R2)
000370  000534                              BR      21$                     ;                               6201
000372  120327  000011              9$:     CMPB    R3,#11                  ; ERROR.CODE,*                  6212
000376  001007                              BNE     10$
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$                             ;                               6201
000416  120327  000012        10$:      CMPB    R3,#12                          ; ERROR.CODE,*                  6214
000422  001012                          BNE     11$
000424  013702  000104                  MOV     ICTLR,R2
000430  006302                          ASL     R2
000432  005001                          CLR     R1
000434  156201  000001G                 BISB    C.ERR.TBL+1(P2),R1
000440  060401                          ADD     R4,R1                           ; RETRIES,*
000442  110162  000001G                 MOVB    R1,C.ERR.TBL+1(R2)
000446  000505                          BR      21$                             ;                               6201
000450  120327  000013        11$:      CMPB    R3,#13                          ; ERROR.CODE,*                  6217
000454  001102                          BNE     21$
000456  020527  000003                  CMP     R5,#3                           ; ERROR SUB,*
000462  001007                          BNE     13$
000464  005001                  12$:    CLR     R1                              ;                               6219
000466  156001  000052                  BISB    52(R0),R1                       ; *(TALLY.ADDR),*
000472  060401                          ADD     R4,R1                           ; RETRIES,*
000474  110160  000052                  MOVB    R1,52(R0)                       ; *,*(TALLY.ADDR)
000500  000470                          BR      21$                             ;                               6217
000502  005001                  13$:    CLR     R1                              ;                               6221
000504  156001  000054                  BISB    54(R0),R1                       ; *(TALLY.ADDR),*
000510  060401                          ADD     R4,R1                           ; RETRIES,*
000512  110160  000054                  MOVB    R1,54(R0)                       ; *,*(TALLY.ADDR)
000516  000461                          BR      21$                             ;                               6201
000520  005761  000006        14$:      TST     6(R1                            ; *(ELOG.ADDR)                  6225
000524  001056                          BNE     21$
000526  005761  000010                  TST     10(R1)                          ; *(ELOG.ADDR)                  6226
000532  001053                          BNE     21$
000534  120327  000005                  CMPB    R3,#5                           ; ERROR.CODE,*                  6231
000540  001443                          BEQ     19$
000542  120327  000010                  CMPB    R3,#10                          ; ERROR.CODE,*                  6233
000546  001013                          BNE     16$
000550  012704  000046                  MOV     #46,R4                          ;                               6235
000554  060004                          ADD     R0,R4                           ; TALLY.ADDR,*
000556  020527  000002                  CMP     R5,#2                           ; ERROR.SUB,*                   6233
000562  001002                          BNE     15$
000564  105214                          INCB    (R4)                            ;                               6235
000566  000435                          BR      21$                             ;                               6233
000570  105264  000001        15$:      INCB    1(R4)                           ;                               6237
000574  000432                          BR      21$                             ;                               6228
000576  120327  000011        16$:      CMPB    R3,#11                          ; ERROR.CODE,*                  6239
000602  001003                          BNE     17$
000604  105260  000051                  INCB    51(R0)                          ; *(TALLY.ADDR)
000610  000424                          BR      21$                             ;                               6228
000612  120327  000012        17$:      CMPB    R3,#12                          ; ERROR.CODE,*                  6241
000616  001006                          BNE     18$
000620  013702  000104'                 MOV     ICTLR,R2
000624  006302                          ASL     R2
000626  105262  000000G                 INCB    C.ERR.TBL(R2)
000632  000413                          BR      21$                             ;                               6228
000634  120327  000013        18$:      CMPB    R3,#13                          ; ERROR.CODE,*                  6244
000640  001010                          BNE     21$
000642  020527  000003                  CMP     R5,#3                           ; ERROR.SUB,*
```

```
000646  001003                          BNE     20$
000650  105260  000046         19$:     INCB    46(R0)          ; .(TALLY.ADDR)         6246
000654  000402                          BR      21$             ;                       6244
000656  105260  000050         20$:     INCB    50(R0)          ; .(TALLY.ADDR)         6248
000662  005726                 21$:     TST     (SP)+           ;                       6196
000664  000415                          BR      24$             ;                       6194
000666  013700  000104         22$:     MOV     ICTLR,R0        ;                       6256
000672  006300                          ASL     R0
000674  062700  000000G                 ADD     #C.ERR.TBL,R0
000700  032766  000001  000004          BIT     #1,4(SP)        ; .,SOFT.OCCURED        6254
000706  001403                          BEQ     23$
000710  105260  000001                  INCB    1(R0)           ;                       6256
000714  000401                          BR      24$             ;                       6254
000716  105210                 23$:     INCB    (R0)            ;                       6258
000720  062706  000012         24$:     ADD     #12,SP          ;                       6141
000724  000207                          RTS     PC

; Routine Size:  235 words,     Routine Base:  $CODE$ + 31756
; Maximum stack depth per invocation:  13 words
```

```
;  6261  1     routine ERR_HRD_RTNE (ERRNUM) : novalue =
;  6262  1
;  6263  1     !+
;  6264  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD  MACRO CALL TO DRS OR TO FAKE
;  6265  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;  6266  1     !
;  6267  1
;  6268  2         begin
;  6269  2
;  6270  2         local
;  6271  2             CUR_PRIORITY : word;
;  6272  2
;  6273  2         builtin
;  6274  2             PC;
;  6275  2
;  6276  2         GETPRI (CUR_PRIORITY);
;  6277  2     !ZZZ    SETPRI (PRI04);                     ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;  6278  2         SETPRI (.BRLEVEL);                      ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW    ZZZ
;  6279  2
;  6280  2         if (.ERRNUM lequ 34) or                 ! FOR NON-BAD BLOCK TYPE ERRORS
;  6281  2            (.ERRNUM gtru 38) or
;  6282  2            (.ERRNUM eql 36) or
;  6283  3            (.ERRNUM eql 37)
;  6284  2         then
;  6285  2
;  6286  3             if BIT_TST (SWP_FLAGS, SWF_HRD)     ! IF ERRORS TO BE TREATED NORMALLY
;  6287  2             then
;  6288  2
;  6289  2     !ZZZ        case .ERRNUM from 31 to 45 of
;  6290  2                 case .ERRNUM from 31 to 73 of   !INCLUDE DUP NUMBERS (60 73)        ZZZ
;  6291  2             set
;  6292  2
;  6293  2                 [31]:   ERRHRD (31, EGH_30, EMS_30);    ! INVALID COMMAND
;  6294  2
;  6295  2                 [32]:   ERRHRD (32, EGH_30, EMS_30);    ! COMMAND ABORTED
;  6296  2
;  6297  2                 [33]:   ;                               !
;  6298  2
;  6299  2                 [34]:   ;                               !
;  6300  2
;  6301  2                 [35]:   ;                               ! MEDIA FORMAT ERROR
;  6302  2
;  6303  2                 [36]:   ERRHRD (36, EGH_30, EMS_30);    ! WRITE PROTECTED
;  6304  2
;  6305  2                 [37]:   ERRHRD (37, EGH_30, EMS_30);    ! COMPARE ERROR
;  6306  2
;  6307  2                 [38]:   ;                               ! DATA ERROR
;  6308  2
;  6309  2                 [39]:   ERRHRD (39, EGH_30, EMS_30);    ! HOST BUFFER ACCESS ERROR
;  6310  2
;  6311  2                 [40]:   ERRHRD (40, EGH_30, EMS_30);    ! CONTROLLER ERROR
;  6312  2
;  6313  2                 [41]:   ERRHRD (41, EGH_30, EMS_30);    ! DRIVE ERROR
```

```
;  6314  2
;  6315  2                    [42]:   ERRHRD (42, EGH 30, 0);          ! HOST WRITE COMPARE ERROR
;  6316  2
;  6317  2                    [43]:   ERRHRD (43, EGH_30, EMS 30);     ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  6318  2
;  6319  2                    [44]:   ERRHRD (44, EGH_30, EMS_30);     ! DUPLICATE UNIT NUMBER
;  6320  2
;  6321  2                    [45]:   ERRHRD (45, EGH_30, EMS_30);     ! INVALID END CODE
;  6322  2
;  6323  2                    [46]:   ;                                !LEAVE ROOM FOR SOFT ERROR NUMBERS AND SOME PADDING  ZZZ
;  6324  2                    [47]:   ;                                !
;  6325  2                    [48]:   ;                                !
;  6326  2                    [49]:   ;                                !
;  6327  2                    [50]:   ;                                !
;  6328  2                    [51]:   ;                                !
;  6329  2                    [52]:   ;                                !
;  6330  2                    [53]:   ;                                !
;  6331  2                    [54]:   ;                                !
;  6332  2                    [55]:   ;                                !
;  6333  2                    [56]:   ;                                !
;  6334  2                    [57]:   ;                                !
;  6335  2                    [58]:   ;                                !
;  6336  2                    [59]:   ;                                !
;  6337  2
;  6338  2                    [60]:  ERRHRD (60, EH_12, EMS_30);       !NOT USED                                    ZZZ
;  6339  2                    [61]:  ERRHRD (61, EH_13, EMS_30);       !SUCCESSFUL MESSAGE                          ZZZ
;  6340  2                    [62]:  ERRHRD (62, EH_13, EMS_30);       !ILLEGAL UNIT NUMBER                         ZZZ
;  6341  2                    [63]:  ERRHRD (63, EH_13, EMS_30);       !ILLEGAL RELATIVE OR PHYSICAL BLOCK          ZZZ
;  6342  2                    [64]:  ERRHRD (64, EH_13, EMS_30);       !DEVICE ERROR                                ZZZ
;  6343  2                    [65]:  ERRHRD (65, EH_13, EMS_30);       !ZERO LENGTH MESSAGE                         ZZZ
;  6344  2                    [66]:  ERRHRD (66, EH_8,  EMS_30);       !DUP UNKNOWN STATUS CODE                     ZZZ
;  6345  2                    [67]:  ERRHRD (67, EH_7,  EMS_30);       !INVALID COMMAND                             ZZZ
;  6346  2                    [68]:  ERRHRD (68, EH_7,  EMS_30);       !NO REGION AVAILABLE                         ZZZ
;  6347  2                    [69]:  ERRHRD (69, EH_7,  EMS_30);       !NO REGION SUITABLE                          ZZZ
;  6348  2                    [70]:  ERRHRD (70, EH_7,  EMS_30);       !PROGRAM NOT KNOWN                           ZZZ
;  6349  2                    [71]:  ERRHRD (71, EH_7,  EMS_30);       !LOAD FAILURE                                ZZZ
;  6350  2                    [72]:  ERRHRD (72, EH_7,  EMS_30);       !STANDALONE                                  ZZZ
;  6351  2                    [73]:  ERRHRD (73, EH_8,  EMS_30);       !DUP UNKNOWN STATUS CODE                     ZZZ
;  6352  2
;  6353  2                        tes
;  6354  2                else
;  6355  3                    begin
;  6356  3                    !****increment error count               ! INCREMENT TOTAL ERROR COUNT
;  6357  3                    PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);  ! PRINT ERROR MESSAGE JUST LIKE DRS
;  6358  3
;  6359  3                    if .ERRNUM neq 42
;  6360  3                    then
;  6361  4                        begin
;  6362  4                        PRINTB (HRD_SUB);                    ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;  6363  4                        EMS_ERR ();                          ! PRINT REST OF THE INFORMATION
;  6364  3                        end;
;  6365  2                    end;
;  6366  2
```

```
;    6367  2          if (.ERRNUM eql 35) or                          ! FOR BAD-BLOCK TYPE ERRORS
;    6368  3             (.ERRNUM eql 38)
;    6369  2          then
;    6370  2
;    6371  3              if BIT_TST (SWP_FLAGS, SWF_BLK)             ! IF ERRORS TO BE TREATED NORMALLY
;    6372  2              then
;    6373  2
;    6374  2                  selectoneu .ERRNUM of
;    6375  2                      set
;    6376  2
;    6377  2                      [35]:   ERRHRD (35, EGH_30, EMS_30);    ! MEDIA FORMAT ERROR
;    6378  2
;    6379  2                      [38]:   ERRHRD (38, EGH_30, EMS_30);    ! DATA ERROR
;    6380  2                      tes
;    6381  2              else
;    6382  3                  begin
;    6383  3                  !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;    6384  3                  PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);     ! PRINT ERROR LINE JUST LIKE DRS
;    6385  3                  PRINTB (HRD_SUB);                           ! PRINT NEXT LINE TOO
;    6386  3                  EMS_ERR ();                                 ! PRINT REST OF THE INFORMATION
;    6387  2                  end;
;    6388  2
;    6389  2          SETPRI (.CUR_PRIORITY);                         ! PRIORITY BACK TO NORMAL
;    6390  2
;    6391  1          end;


                                        .SBTTL  ERR.HRD.RTNE RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          ERR.HRD.RTNE:
                                        JSR     R1,$SAVE2                       ;                       6261
000004  104440                          TRAP    40                              ;                       6276
000006  010002                          MOV     R0,R2                           ; *.CUR.PRIORITY
000010  013700  000000G                 MOV     BRLEVEL,R0                      ;                       6278
000014  104441                          TRAP    41
000016  016601  000010                  MOV     10(SP),R1                       ; ERRNUM,*              6280
000022  020127  000042                  CMP     R1,#42
000026  101411                          BLOS    1$
000030  020127  000046                  CMP     R1,#46                          ;                       6281
000034  101006                          BHI     1$
000036  020127  000044                  CMP     R1,#44                          ;                       6282
000042  001403                          BEQ     1$
000044  020127  000045                  CMP     R1,#45                          ;                       6283
000050  001176                          BNE     27$
000052  032737  010000  000000G  1$:     BIT     #10000,SWP.FLAGS                ;                       6286
000060  001002                          BNE     2$
000062  000137  033400'                 JMP     31$
000066  010100                  2$:     MOV     R1,R0                           ;                       6290
000070  162700  000037                  SUB     #37,R0
000074  006300                          ASL     R0
000076  066007  000024'                 ADD     P.AAC(R0),PC                    ; Case dispatch
000102  104456                  4$:     TRAP    56                              ;                       6293
000104  000037                          .WORD   37
000106  000000G                         .WORD   EGH.30
```

```
000110  000000G                         .WORD   EMS.30
000112  000567                          BR      30$                             ;       6290
000114  104456              5$:       TRAP    56                              ;       6295
000116  000040                          .WORD   40
000120  000000G                         .WORD   EGH.30
000122  000000G                         .WORD   EMS.30
000124  000562                          BR      30$                             ;       6290
000126  104456              6$:       TRAP    56                              ;       6303
000130  000044                          .WORD   44
000132  000000G                         .WORD   EGH.30
000134  000000G                         .WORD   EMS.30
000136  000555                          BR      30$                             ;       6290
000140  104456              7$:       TRAP    56                              ;       6305
000142  000045                          .WORD   45
000144  000000G                         .WORD   EGH.30
000146  000000G                         .WORD   EMS.30
000150  000550                          BR      30$                             ;       6290
000152  104456              8$:       TRAP    56                              ;       6309
000154  000047                          .WORD   47
000156  000000G                         .WORD   EGH.30
000160  000000G                         .WORD   EMS.30
000162  000574                          BR      33$                             ;       6290
000164  104456              9$:       TRAP    56                              ;       6311
000166  000050                          .WORD   50
000170  000000G                         .WORD   EGH.30
000172  000000G                         .WORD   EMS.30
000174  000567                          BR      33$                             ;       6290
000176  104456              10$:      TRAP    56                              ;       6313
000200  000051                          .WORD   51
000202  000000G                         .WORD   EGH.30
000204  000000G                         .WORD   EMS.30
000206  000562                          BR      33$                             ;       6290
000210  104456              11$:      TRAP    56                              ;       6315
000212  000052                          .WORD   52
000214  000000G                         .WORD   EGH.30
000216  000000                          .WORD   0
000220  000555                          BR      33$                             ;       6290
000222  104456              12$:      TRAP    56                              ;       6317
000224  000053                          .WORD   53
000226  000000G                         .WORD   EGH.30
000230  000000G                         .WORD   EMS.30
000232  000550                          BR      33$                             ;       6290
000234  104456              13$:      TRAP    56                              ;       6319
000236  000054                          .WORD   54
000240  000000G                         .WORD   EGH.30
000242  000000G                         .WORD   EMS.30
000244  000543                          BR      33$                             ;       6290
000246  104456              14$:      TRAP    56                              ;       6321
000250  000055                          .WORD   55
000252  000000G                         .WORD   EGH.30
000254  000000G                         .WORD   EMS.30
000256  000536                          BR      33$                             ;       6290
000260  104456              15$:      TRAP    56                              ;       6339
```

```
000262  000074                          .WORD   74
000264  000000G                         .WORD   EH.12
000266  000000G                         .WORD   EMS.30
000270  000531                          BR      33$                     ;                       6290
000272  104456                  16$:    TRAP    56                      ;                       6339
000274  000075                          .WORD   75
000276  000000G                         .WORD   EH.13
000300  000000G                         .WORD   EMS.30
000302  000524                          BR      33$                     ;                       6290
000304  104456                  17$:    TRAP    56                      ;                       6340
000306  000076                          .WORD   76
000310  000000G                         .WORD   EH.13
000312  000000G                         .WORD   EMS.30
000314  000517                          BR      33$                     ;                       6290
000316  104456                  18$:    TRAP    56                      ;                       6341
000320  000077                          .WORD   77
000322  000000G                         .WORD   EH.13
000324  000000G                         .WORD   EMS.30
000326  000512                          BR      33$                     ;                       6290
000330  104456                  19$:    TRAP    56                      ;                       6342
000332  000100                          .WORD   100
000334  000000G                         .WORD   EH.13
000336  000000G                         .WORD   EMS.30
000340  000505                          BR      33$                     ;                       6290
000342  104456                  20$:    TRAP    56                      ;                       6343
000344  000101                          .WORD   101
000346  000000G                         .WORD   EH.13
000350  000000G                         .WORD   EMS.30
000352  000500                          BR      33$                     ;                       6290
000354  104456                  21$:    TRAP    56                      ;                       6344
000356  000102                          .WORD   102
000360  000000G                         .WORD   EH.8
000362  000000G                         .WORD   EMS.30
000364  000473                          BR      33$                     ;                       6290
000366  104456                  22$:    TRAP    56                      ;                       6345
000370  000103                          .WORD   103
000372  000000G                         .WORD   EH.7
000374  000000G                         .WORD   EMS.30
000376  000466                          BR      33$                     ;                       6290
000400  104456                  23$:    TRAP    56                      ;                       6346
000402  000104                          .WORD   104
000404  000000G                         .WORD   EH.7
000406  000000G                         .WORD   EMS.30
000410  000461                          BR      33$                     ;                       6290
000412  104456                  24$:    TRAP    56                      ;                       6347
000414  000105                          .WORD   105
000416  000000G                         .WORD   EH.7
000420  000000G                         .WORD   EMS.30
000422  000454                          BR      33$                     ;                       6290
000424  104456                  25$:    TRAP    56                      ;                       6348
000426  000106                          .WORD   106
000430  000000G                         .WORD   EH.7
000432  000000G                         .WORD   EMS.30
```

```
000434  000447                          BR      33$                             ;                       6290
000436  104456                  26$:    TRAP    56                              ;                       6349
000440  000107                          .WORD   107
000442  000000G                         .WORD   EH.7
000444  000000G                         .WORD   EMS.30
000446  000442                  27$:    BR      33$                             ;                       6290
000450  104456                  28$:    TRAP    56                              ;                       6350
000452  000110                          .WORD   110
000454  000000G                         .WORD   EH.7
000456  000000G                         .WORD   EMS.30
000460  000435                          BR      33$                             ;                       6290
000462  104456                  29$:    TRAP    56                              ;                       6351
000464  000111                          .WORD   111
000466  000000G                         .WORD   EH.8
000470  000000G                         .WORD   EMS.30
000472  000430                  30$:    BR      33$                             ;                       6286
000474  010746                  31$:    MOV     PC,-(SP)                        ; PC,*                  6357
000476  013746  000000G                 MOV     L$LUN,-(SP)
000502  010146                          MOV     R1,-(SP)
000504  012746  000000G                 MOV     #HRD.MSG,-(SP)
000510  012746  000004                  MOV     #4,-(SP)
000514  010600                          MOV     SP,R0                           ; SP,*
000516  104414                          TRAP    14
000520  020127  000052                  CMP     R1,#52                          ;                       6359
000524  001411                          BEQ     32$
000526  012716  000000G                 MOV     #HRD.SUB,(SP)                   ;                       6362
000532  012746  000001                  MOV     #1,-(SP)
000536  010600                          MOV     SP,R0                           ; SP,*
000540  104414                          TRAP    14
000542  004737  000000G                 JSR     PC,EMS.ERR                      ;                       6363
000546  005726                          TST     (SP)+                           ;                       6361
000550  062706  000012          32$:    ADD     #12,SP                          ;                       6355
000554  020127  000043          33$:    CMP     R1,#43                          ;                       6367
000560  001403                          BEQ     34$
000562  020127  000046                  CMP     R1,#46                          ;                       6368
000566  001050                          BNE     37$
000570  032737  040000  000000G  34$:    BIT     #40000,SWP.FLAGS                ;                       6371
000576  001420                          BEQ     36$
000600  020127  000043                  CMP     R1,#43                          ;                       6377
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$                             ;                       6374
000620  020127  000046          35$:    CMP     R1,#46                          ;                       6379
000624  001031                          BNE     37$
000626  104456                          TRAP    56
000630  000046                          .WORD   46
000632  000000G                         .WORD   EGH.30
000634  000000G                         .WORD   EMS.30
000636  000424                          BR      37$                             ;                       6371
000640  010746                  36$:    MOV     PC,(SP)                         ; PC,*                  6384
```

```
000642  013746  000000G                        MOV     L$LUN,-(SP)
000646  010146                                  MOV     R1,(SP)
000650  012746  000000G                        MOV     #HRD.MSG,(SP)
000654  012746  000004                        MOV     #4,-(SP)
000660  010600                                  MOV     SP,R0                   ; SP,*
000662  104414                                  TRAP    14
000664  012716  000000G                        MOV     #HRD.SUB,(SP)           ;                       6385
000670  012746  000001                        MOV     #1,(SP)
000674  010600                                  MOV     SP,R0                   ; SP,*
000676  104414                                  TRAP    14
000700  004737  000000G                        JSR     PC,EMS.ERR              ;                       6386
000704  062706  000014                        ADD     #14,SP                  ;                       6382
000710  010200                        37$:     MOV     R2,R0                   ; CUR.PRIORITY,*        6389
000712  104441                                  TRAP    41
000714  000207                                  RTS     PC                      ;                       6261

; Routine Size:  231 words,    Routine Base:  $CODE$ + 32704
; Maximum stack depth per invocation:  11 words


000024                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAC:                                          ; CASE Table for ERR.HRD.RTNE+0076      6290
000024  000000                  3$:     .WORD   0                               ; [4$]
000026  000012                          .WORD   12                              ; [5$]
000030  000452                          .WORD   452                             ; [33$]
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$]
```

```
;   6392  1     routine ERR_SOFT_RTNE (ERRNUM) : novalue =
;   6393  1
;   6394  1     !+
;   6395  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT  MACRO CALL TO DRS OR TO FAKE
;   6396  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;   6397  1     !
;   6398  1
;   6399  2         begin
;   6400  2
;   6401  2         builtin
;   6402  2             PC;
;   6403  2
;   6404  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                 ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;   6405  2         then
;   6406  2
;   6407  2             case .ERRNUM from 50 to 54 of
;   6408  2                 set
;   6409  2
;   6410  2                 [50]:       ERRSOFT (50, 0, 0);         ! CONTROLLER ERROR
;   6411  2
;   6412  2                 [51]:       ERRSOFT (51, 0, 0);         ! HOST MEMORY ACCESS ERROR
;   6413  2
;   6414  2                 [52]:       ERRSOFT (52, 0, 0);         ! DISK TRANSFER ERROR
;   6415  2
;   6416  2                 [53]:       ERRSOFT (53, 0, 0);         ! SDI ERROR
;   6417  2
;   6418  2                 [54]:       ERRSOFT (54, 0, 0);         ! SMALL DISK ERROR
;   6419  2                 tes
;   6420  2         else
;   6421  3             begin
;   6422  3             !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;   6423  3             PRINTB (SFT_MSG, .ERRNUM, .L$LUN, .PC);     ! PRINT ERROR LINE JUST LIKE DRS
;   6424  2             end;
;   6425  2
;   6426  1         end;


                                        .SBTTL  ERR.SOFT.RTNE RDRX INTERRUPT SERVICE ROUTINES
033622                                  .PSECT  $CODE$,  RO

000000  032737  020000  000000G  ERR.SOFT.RTNE:
                                        BIT     #20000,SWP.FLAGS        ;                       6404
000006  001440                          BEQ     7$
000010  016600  000002                  MOV     2(SP),R0                ; ERRNUM,*              6407
000014  162700  000062                  SUB     #62,R0
000020  006300                          ASL     R0
000022  066007  000152'                  ADD     P.AAD(R0),PC            ; Case dispatch
000026  104457                  2$:     TRAP    57                      ;                       6410
000030  000062                          .WORD   62
000032  000000                          .WORD   0
000034  000000                          .WORD   0
000036  000207                          RTS     PC                      ;                       6407
000040  104457                  3$:     TRAP    57                      ;                       6412
```

```
000042  000063                          .WORD   63
000044  000000                          .WORD   0
000046  000000                          .WORD   0
000050  000207                          RTS     PC                              ;                               6407
000052  104457                  4$:     TRAP    57                              ;                               6414
000054  000064                          .WORD   64
000056  000000                          .WORD   0
000060  000000                          .WORD   0
000062  000207                          RTS     PC                              ;                               6407
000064  104457                  5$:     TRAP    57                              ;                               6416
000066  000065                          .WORD   65
000070  000000                          .WORD   0
000072  000000                          .WORD   0
000074  000207                          RTS     PC                              ;                               6407
000076  104457                  6$:     TRAP    57                              ;                               6418
000100  000066                          .WORD   66
000102  000000                          .WORD   0
000104  000000                          .WORD   0
000106  000207                          RTS     PC                              ;                               6404
000110  010746                  7$:     MOV     PC,-(SP)                        ; PC,*                          6423
000112  013746  000000G                 MOV     L$LUN,-(SP)
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                          ;                               6421
000142  000207                          RTS     PC                              ;                               6392

; Routine Size:  50 words,      Routine Base:  $CODE$ + 33622
; Maximum stack depth per invocation:  7 words


000152                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAD:                                          ; CASE Table for ERR.SOFT.RTNE+0022     6407
000152  000000                 1$:     .WORD   0                               ; [2$]
000154  000012                          .WORD   12                              ; [3$]
000156  000024                          .WORD   24                              ; [4$]
000160  000036                          .WORD   36                              ; [5$]
000162  000050                          .WORD   50                              ; [6$]
```

```
;   6427  1     routine ERR_HRD RTNE APT (ERRNUM) : novalue =
;   6428  1
;   6429  1
;   6430  1     !+
;   6431  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;   6432  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;   6433  1     !
;   6434  1
;   6435  1
;   6436  2         begin
;   6437  2
;   6438  2
;   6439  2         local
;   6440  2             CUR_PRIORITY;
;   6441  2
;   6442  2
;   6443  2         builtin
;   6444  2             PC;
;   6445  2
;   6446  2         GETPRI (CUR_PRIORITY);
;   6447  2     !ZZZ    SETPRI (PRI04);                         ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;   6448  2         SETPRI (.BRLEVEL);                          ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW     ZZZ
;   6449  2
;   6450  2
;   6451  2         if .APT_MODE
;   6452  2         then
;   6453  2
;   6454  3             begin
;   6455  3             .MAIL_BOX_TESTNUM = .RP_ADDR [LBN_LO];  ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;   6456  3             .MAIL_BOX_SUBTST = .RP_ADDR [DISK];     ! CHANGE SUB TEST NUMBER TO SHOW DISK NUMBER UNDER APT ONLY
;   6457  2             end;
;   6458  2
;   6459  2
;   6460  2         if (.ERRNUM lequ 34) or                     ! FOR NON-BAD BLOCK TYPE ERRORS
;   6461  2            (.ERRNUM gtru 38) or
;   6462  2            (.ERRNUM eql 36) or
;   6463  3            (.ERRNUM eql 37)
;   6464  3
;   6465  2         then
;   6466  2
;   6467  3             if BIT_TST (SWP_FLAGS, SWF_HRD)         ! IF ERRORS TO BE TREATED NORMALLY
;   6468  2             then
;   6469  2
;   6470  2                 case .ERRNUM from 31 to 45 of
;   6471  2                     set
;   6472  2
;   6473  2                     [31]:   ERRDF (31, EGH_30, EMS_30);     ! INVALID COMMAND
;   6474  2
;   6475  2                     [32]:   ERRDF (32, EGH_30, EMS_30);     ! COMMAND ABORTED
;   6476  2
;   6477  2                     [33]:   ;                               !
;   6478  2
;   6479  2                     [34]:   ;                               !
```

```
;   6480  2
;   6481  2                    [35]:   ;                                   ! MEDIA FORMAT ERROR
;   6482  2
;   6483  2                    [36]:   ERRDF (36, EGH_30, EMS_30);         ! WRITE PROTECTED
;   6484  2
;   6485  2                    [37]:   ERRDF (37, EGH_30, EMS_30);         ! COMPARE ERROR
;   6486  2
;   6487  2                    [38]:   ;                                   ! DATA ERROR
;   6488  2
;   6489  2                    [39]:   ERRDF (39, EGH_30, EMS_30);         ! HOST BUFFER ACCESS ERROR
;   6490  2
;   6491  2                    [40]:   ERRDF (40, EGH_30, EMS_30);         ! CONTROLLER ERROR
;   6492  2
;   6493  2                    [41]:   ERRDF (41, EGH_30, EMS_30);         ! DRIVE ERROR
;   6494  2
;   6495  2                    [42]:   ERRDF (42, EGH_30, 0);              ! HOST WRITE COMPARE ERROR
;   6496  2
;   6497  2                    [43]:   ERRDF (43, EGH_30, EMS_30);         ! MESSAGE FROM INTERNAL DIAGNOSTICS
;   6498  2
;   6499  2                    [44]:   ERRDF (44, EGH_30, EMS_30);         ! DUPLICATE UNIT NUMBER
;   6500  2
;   6501  2                    [45]:   ERRDF (45, EGH_30, EMS_30);         ! INVALID END CODE
;   6502  2                    tes
;   6503  2
;   6504  2            else
;   6505  2
;   6506  3                begin
;   6507  3                !****increment error count                      ! INCREMENT TOTAL ERROR COUNT
;   6508  3                PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);         ! PRINT ERROR MESSAGE JUST LIKE DRS
;   6509  3
;   6510  3
;   6511  3                if .ERRNUM neq 42
;   6512  3
;   6513  3                then
;   6514  4                    begin
;   6515  4                    PRINTB (HRD SUB);                          . NEXT LINE FOR NON-HOST COMPARE ERRORS
;   6516  4                    EMS_ERR ();                                ! PRINT REST OF THE INFORMATION
;   6517  3                    end;
;   6518  2                end;
;   6519  2
;   6520  2            if (.ERRNUM eql 35) or                             ! FOR BAD BLOCK TYPE ERRORS
;   6521  3               (.ERRNUM eql 38)
;   6522  3
;   6523  2            then
;   6524  2
;   6525  3                if BIT_TST (SWP FLAGS, SWF_BLK)                ! IF ERRORS TO BE TREATED NORMALLY
;   6526  2                then
;   6527  2
;   6528  2                    selectoneu .ERRNUM of
;   6529  2                        set
;   6530  2
;   6531  2                        [35]:   ERRDF (35, EGH_30, EMS_30);    ! MEDIA FORMAT ERROR
;   6532  2
```

```
:    6533  2                    [38]:   ERRDF (38, EGH 30, EMS 30);     ! DATA ERROR
:    6534  2                    tes
:    6535  2
:    6536  2                else
:    6537  2
:    6538  3                    begin
:    6539  3                    !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
:    6540  3                    PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);      ! PRINT ERROR LINE JUST LIKE DRS
:    6541  3                    PRINTB (MRD_SUB);                           ! PRINT NEXT LINE TOO
:    6542  3                    EMS_ERR ();                                 ! PRINT REST OF THE INFORMATION
:    6543  2                    end;
:    6544  2
:    6545  2
:    6546  2            SETPRI (.CUR_PRIORITY);                             ! PRIORITY BACK TO NORMAL
:    6547  2
:    6548  2
:    6549  1            end;


                                        .SBTTL  ERR.MRD.RTNE.APT RDRx INTERRUPT SERVICE ROUTINES
033766                                  .PSECT  $CODE$,  RO

000000  004137  000000G          ERR.MRD.RTNE.APT:
                                        JSR     R1,$SAVE2                   ;                                   6427
000004  104440                          TRAP    40                          ;                                   6446
000006  010002                          MOV     R0,R2                       ; #.CUR.PRIORITY
000010  013700  000000G                 MOV     @#LEVEL,R0                  ;                                   6448
000014  104441                          TRAP    41
000016  032737  000001  001162'         BIT     #1,APT.MODE                 ;                                   6451
000024  001412                          BEQ     1$
000026  013700  000000G                 MOV     RP.ADDR,R0                  ;                                   6455
000032  016077  000050  001164         MOV     50(R0),@MAIL.BOX.TESTNUM
000040  013700  000000G                 MOV     RP.ADDR,R0                  ;                                   6456
000044  016077  000010  001166'         MOV     10(R0),@MAIL.BOX.SUBTST
000052  016601  000010          1$:     MOV     10(SP),R1                   ; ERRNUM,#                          6460
000056  020127  000042                  CMP     R1,#42
000062  101411                          BLOS    2$
000064  020127  000046                  CMP     R1,#46                      ;                                   6461
000070  101006                          BHI     2$
000072  020127  000044                  CMP     R1,#44                      ;                                   6462
000076  001403                          BEQ     2$
000100  020127  000045                  CMP     R1,#45                      ;                                   6463
000104  001131                          BNE     17$
000106  032737  010000  000000G   2$:     BIT     #10000,SWP.FLAGS            ;                                   6467
000114  001475                          BEQ     15$
000116  010100                          MOV     R1,R0                       ;                                   6470
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                          ;                                   6473
000134  000037                  .WORD   37
000136  000000G                 .WORD   EGH.30
000140  000000G                 .WORD   EMS.30
```

```
000142  000512                          BR      17$                             ;                               6470
000144  104455                  5$:     TRAP    55                              ;                               6475
000146  000040                          .WORD   40
000150  000000G                         .WORD   EGM.30
000152  000000G                         .WORD   EMS.30
000154  000505                          BR      17$                             ;                               6470
000156  104455                  6$:     TRAP    55                              ;                               6483
000160  000044                          .WORD   44
000162  000000G                         .WORD   EGM.30
000164  000000G                         .WORD   EMS.30
000166  000500                          BR      17$                             ;                               6470
000170  104455                  7$:     TRAP    55                              ;                               6485
000172  000045                          .WORD   45
000174  000000G                         .WORD   EGM.30
000176  000000G                         .WORD   EMS.30
000200  000473                          BR      17$                             ;                               6470
000202  104455                  8$:     TRAP    55                              ;                               6489
000204  000047                          .WORD   47
000206  000000G                         .WORD   EGM.30
000210  000000G                         .WORD   EMS.30
000212  000466                          BR      17$                             ;                               6470
000214  104455                  9$:     TRAP    55                              ;                               6491
000216  000050                          .WORD   50
000220  000000G                         .WORD   EGM.30
000222  000000G                         .WORD   EMS.30
000224  000461                          BR      17$                             ;                               6470
000226  104455                  10$:    TRAP    55                              ;                               6493
000230  000051                          .WORD   51
000232  000000G                         .WORD   EGM.30
000234  000000G                         .WORD   EMS.30
000236  000454                          BR      17$                             ;                               6470
000240  104455                  11$:    TRAP    55                              ;                               6495
000242  000052                          .WORD   52
000244  000000G                         .WORD   EGM.30
000246  000000                          .WORD   0
000250  000447                          BR      17$                             ;                               6470
000252  104455                  12$:    TRAP    55                              ;                               6497
000254  000053                          .WORD   53
000256  000000G                         .WORD   EGM.30
000260  000000G                         .WORD   EMS.30
000262  000442                          BR      17$                             ;                               6470
000264  104455                  13$:    TRAP    55                              ;                               6499
000266  000054                          .WORD   54
000270  000000G                         .WORD   EGM.30
000272  000000G                         .WORD   EMS.30
000274  000435                          BR      17$                             ;                               6470
000276  104455                  14$:    TRAP    55                              ;                               6501
000300  000055                          .WORD   55
000302  000000G                         .WORD   EGM.30
000304  000000G                         .WORD   EMS.30
000306  000430                          BR      17$                             ;                               6467
000310  010746                  15$:    MOV     PC,-(SP)                        ; PC,*                          6508
000312  013746  000000G                 MOV     L$LUN,-(SP)
```

```
000316  010146                                  MOV     P1,-(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                  ;                       6511
000340  001411                                  BEQ     16$
000342  012716  000000G                         MOV     #WRD.SUB,(SP)           ;                       6515
000346  012746  000001                          MOV     #1,-(SP)
000352  010600                                  MOV     SP,R0                   ; SP,*
000354  104414                                  TRAP    14
000356  004737  000000G                         JSR     PC,EMS.ERR              ;                       6516
000362  005726                                  TST     (SP)+                   ;                       6514
000364  062706  000012              16$:        ADD     #12,SP                  ;                       6506
000370  020127  000043              17$:        CMP     R1,#43                  ;                       6520
000374  001403                                  BEQ     18$
000376  020127  000046                          CMP     R1,#46                  ;                       6521
000402  001050                                  BNE     21$
000404  032737  040000  000000G     18$:        BIT     #40000,SWP.FLAGS        ;                       6525
000412  001420                                  BEQ     20$
000414  020127  000043                          CMP     R1,#43                  ;                       6531
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$                     ;                       6528
000434  020127  000046              19$:        CMP     R1,#46                  ;                       6533
000440  001031                                  BNE     21$
000442  104455                                  TRAP    55
000444  000046                                  .WORD   46
000446  000000G                                 .WORD   EGH.30
000450  000000G                                 .WORD   EMS.30
000452  000424                                  BR      21$                     ;                       6525
000454  010746                      20$:        MOV     PC,-(SP)                ; PC,*                  6540
000456  013746  000000G                         MOV     L$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     #WRD.SUB,(SP)           ;                       6541
000504  012746  000001                          MOV     #1,-(SP)
000510  010600                                  MOV     SP,R0                   ; SP,*
000512  104414                                  TRAP    14
000514  004737  000000G                         JSR     PC,EMS.ERR              ;                       6542
000520  062706  000014                          ADD     #14,SP                  ;                       6538
000524  010200                      21$:        MOV     R2,R0                   ; CUR.PRIORITY,*        6546
000526  104441                                  TRAP    41
000530  000207                                  RTS     PC                      ;                       6427

; Routine Size:  173 words,     Routine Base:  $CODE$ + 33766
; Maximum stack depth per invocation:  11 words
```

```
000164                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAE:
000164  000000                3$:     .WORD   0                       ; CASE Table for ERR.HRD.RTNE.AP+0126   6470
000166  000012                        .WORD   12                      ; [4$]
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$]


;   6550  1
;   6551  1
```

```
;  6552  1     routine ERR_SOFT_RTNE_APT (ERRNUM, index) : novalue =
;  6553  1
;  6554  1     !+
;  6555  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;  6556  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;  6557  1     !
;  6558  1
;  6559  2         begin
;  6560  2
;  6561  2         local
;  6562  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP FIELDS);
;  6563  2
;  6564  2         builtin
;  6565  2             PC;
;  6566  2
;  6567  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);       ! ADDRESS OF THE SAVED ERROR LOG INFORMATION
;  6568  2
;  6569  2         if  APT_MODE
;  6570  2         then
;  6571  3             begin
;  6572  3             .MAIL_BOX_TESTNUM = .ELOG_ADDR [EL_BLOCK];      ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;  6573  3             .MAIL_BOX_SUBTST = .ELOG_ADDR [EL_DK_NUM];      ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER IN APT ONLY
;  6574  2             end;
;  6575  2
;  6576  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                     ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;  6577  2         then
;  6578  2
;  6579  2             case .ERRNUM from 50 to 54 of
;  6580  2                 set
;  6581  2
;  6582  2                 [50]:       ERRDF (50, 0, 0);               ! CONTROLLER ERROR
;  6583  2
;  6584  2                 [51]:       ERRDF (51, 0, 0);               ! HOST MEMORY ACCESS ERROR
;  6585  2
;  6586  2                 [52]:       ERRDF (52, 0, 0);               ! DISK TRANSFER ERROR
;  6587  2
;  6588  2                 [53]:       ERRDF (53, 0, 0);               ! SDI ERROR
;  6589  2
;  6590  2                 [54]:       ERRDF (54, 0, 0);               ! SMALL DISK ERROR
;  6591  2                 tes
;  6592  2         else
;  6593  3             begin
;  6594  3             !****increment error count                       ! INCREMENT TOTAL ERROR COUNT
;  6595  3             PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);          ! PRINT ERROR LINE JUST LIKE DRS
;  6596  2             end;
;  6597  2
;  6598  1         end;


                                        .SBTTL  ERR.SOFT.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
034520                                  .PSECT  $CODE$,  RO

000000  016646  000002          ERR.SOFT.RTNE.APT:
```

```
                                        MOV     2(SP),-(SP)                     ; INDEX,*                       6567
000004  012746  000102                  MOV     #102,-(SP)
000010  004737  000000G                 JSR     PC,BL$MUL
000014  062700  000000G                 ADD     #ELOG.PKT,R0
000020  032737  000001  001162          BIT     #1,APT.MODE                     ;                               6569
000026  001406                          BEQ     1$
000030  016077  000056  001164'         MOV     56(R0),@MAIL.BOX.TESTNUM        ; *(ELOG.ADDR),*                6572
000036  016077  000012  001166'         MOV     12(R0),@MAIL.BOX.SUBTST         ; *(ELOG.ADDR),*                6573
000044  032737  020000  000000G  1$:    BIT     #20000,SWP.FLAGS                ;                               6576
000052  001440                          BEQ     8$
000054  016600  000010                  MOV     10(SP),R0                       ; ERRNUM,*                      6579
000060  162700  000062                  SUB     #62,R0
000064  006300                          ASL     R0
000066  066007  000222'                 ADD     P.AAF(R0),PC                    ; Case dispatch
000072  104455                  3$:     TRAP    55                              ;                               6582
000074  000062                          .WORD   62
000076  000000                          .WORD   0
000100  000000                          .WORD   0
000102  000441                          BR      9$                              ;                               6579
000104  104455                  4$:     TRAP    55                              ;                               6584
000106  000063                          .WORD   63
000110  000000                          .WORD   0
000112  000000                          .WORD   0
000114  000434                          BR      9$                              ;                               6579
000116  104455                  5$:     TRAP    55                              ;                               6586
000120  000064                          .WORD   64
000122  000000                          .WORD   0
000124  000000                          .WORD   0
000126  000427                          BR      9$                              ;                               6579
000130  104455                  6$:     TRAP    55                              ;                               6588
000132  000065                          .WORD   65
000134  000000                          .WORD   0
000136  000000                          .WORD   0
000140  000422                          BR      9$                              ;                               6579
000142  104455                  7$:     TRAP    55                              ;                               6590
000144  000066                          .WORD   66
000146  000000                          .WORD   0
000150  000000                          .WORD   0
000152  000415                          BR      9$                              ;                               6576
000154  010716                  8$:     MOV     PC,(SP)                         ; PC,*                          6595
000156  013746  000000G                 MOV     L$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                          ;                               6593
000206  022626                  9$:     CMP     (SP)+,(SP)+                     ;                               6559
000210  000207                          RTS     PC                              ;                               6552

; Routine Size:  69 words,      Routine Base:  $CODE$ + 34520
; Maximum stack depth per invocation:  8 words
```

```
000222                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAF:                                          ; CASE Table for ERR.SOFT.RTNE.A+0066   6579
000222  000000                  2$:     .WORD   0                               ; [3$]
000224  000012                          .WORD   12                              ; [4$]
000226  000024                          .WORD   24                              ; [5$]
000230  000036                          .WORD   36                              ; [6$]
000232  000050                          .WORD   50                              ; [7$]


;   6599  1
;   6600  1
;   6601  1     end
;   6602  1
;   6603  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
;         $GGG$                           324       RO ,  I  ,  LCL,  REL,  CON
;         $CODE$                         7405       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]ZRQAF0.L16;11       406      332        81        21        00:00.1


;                                       COMMAND QUALIFIERS

;       BLISS/FJP11 ZRQAF0.BL2/LIST=ZRQAF0.LS2/OBJECT=ZRQAF0.OB2/SOURCE=PAGE:53
```

```
;   6604  0     module ZRQAM4 (
;   6605  0
;   6606  0     %title  RD/RX EXERCISER
;   6607  0                     ident =  V01.9 ,
;   6608  0                     addressing mode (absolute),
;   6609  0                     environment (noeis)
;   6610  0                     ) =
;   6611  0
;   6612  1     begin
;   6613  1
;   6614  1     %sbttl 'LASTAD AND SETUP
;   6615  1
;   6616  1     library 'ZRQAF0.L16 ;
;   6617  1
;   6618  1     require 'BLSMAC.REQ ;                          ! DIAGNOSTIC SUPERVISOR LIBRARY
;   8109  1
;   8110  2     LASTAD
;   8111  2
;   8112  2     BGNSETUP (4)                                                        !ZZZ
;   8113  2
; P 8114  2             BGNPTAB
; P 8115  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000020', 0, 0, RD52_MAX_LBN, 0  !ZZZ
; P 8116  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
; P 8117  2
;   8118  2             ENDPTAB
;   8119  2
; P 8120  2             BGNPTAB
; P 8121  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000001', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P 8122  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;  .8123  2             ENDPTAB
; . 8124  2
; P 8125  2             BGNPTAB
; P 8126  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000002', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P 8127  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;   8128  2             ENDPTAB
; P 8129  2             BGNPTAB
; P 8130  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000003', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P 8131  2                             !HERE'S ONE FOR THE 4TH DRIVE                                           !ZZZ
;   8132  2             ENDPTAB                                                                                   !ZZZ
;   8133  2
;   8134  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
        000004'                 $LAS5=          P.AAP
        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                                      ;                                       8108

; Routine Size:  1 word,        Routine Base:  $XYZ$ + 0126
; Maximum stack depth per invocation:  0 words


;   8135  1     end
;   8136  1
;   8137  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]ZRQAF0.L16;11   406        7         1        21          00:00.1


;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAF0.BL2/LIST=ZRQAF0.LS2/OBJECT=ZRQAF0.OB2/SOURCE=PAGE:53

; Size:          7406 code + 445 data words
; Run Time:        03:53.1
; Elapsed Time:    07:57.0
; Lines/CPU Min:    2094
; Lexemes/CPU-Min: 19768
; Memory Used:  550 pages
; Compilation Complete
```

```
Partition name : DUMMY
Identification : V02.1
Task  UIC      : [202,24]
Task attributes: -HD
Total address windows: 1.
Task  image  size  : 17792. words
Task address limits: 002000 107347
R-W disk blk limits: 000002 000107 000106 00070.

*** Root segment: ZRQAF0


R/W mem  limits: 002000 107347 105350 35560.
Disk blk limits: 000002 000107 000106 00070.


Memory allocation synopsis:

Section                                          Title  Ident  File
-------                                          -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON)      002000 000000 00000.
$CODE$:(RO,I,LCL,REL,CON)      002000 075262 31410.
                               002000 022704 09668.  ZRQAM1 V02.1  ZRQAF0.OB1;1
                               024704 015000 06656.  ZRQAM2 V01.9  ZRQAF0.OB1;1
                               041704 034732 14810.  ZRQAM3 V02.1  ZRQAF0.OB2;1
                               076636 000316 00206.  B16MUL 2.8    NOEIS.OLB;1
                               077154 000106 00070.  B16SAV 2.4    NOEIS.OLB;1
$FFF$ :(RW,D GBL,REL,CON)      077262 006014 03084.
                               077262 006014 03084.  ZRQAM1 V02.1  ZRQAF0.OB1;1
$GGG$ :(RO,I,LCL,REL,CON)      105276 001210 00648.
                               105276 001210 00648.  ZRQAM3 V02.1  ZRQAF0.OB2;1
$OWN$ :(RW,D,LCL,REL,CON)      106506 000224 00148.
                               106506 000224 00148.  ZRQAM2 V01.9  ZRQAF0.OB1;1
$PLIT$:(RO,D,LCL,REL,CON)      106732 000264 00180.
                               106732 000030 00024.  ZRQAM2 V01.9  ZRQAF0.OB1;1
                               106762 000234 00156.  ZRQAM3 V02.1  ZRQAF0.OB2;1
$XYZ$ :(RO,I,LCL,REL,CON)      107216 000130 00088.
                               107216 000130 00088.  ZRQAM4 V01.9  ZRQAF0.OB2;1


Global symbols:

ADDR.V 105275-R  BIT06  000100  BIT3   000010  BOE    000400  CMD.TI 105226-R  DASH   024546-R  DBM121 006576-R
ADR    000020    BIT07  000200  BIT4   000020  BRLEVE 105264-R  CNTR.E 023452-R  DATAGM 072736-R  DBM15  004724-R
ASTERI 024554-R  BIT08  000400  BIT5   000040  BST    077442-R  CREDIT 105242-R  DBM101 005746-R  DBM18  004754-R
AZINT  067322-R  BIT09  001000  BIT6   000100  BUFF.A 105122-R  CRLF   024542-R  DBM104 005774-R  DBM19  005026-R
AZINT0 067304-R  BIT1   000002  BIT7   000200  BUFF.0 105142-R  CRN.HI 105234-R  DBM105 006036-R  DBM20  005112-R
BIT0   000001    BIT10  002000  BIT8   000400  BYTS.P 105212-R  CRN.LO 105232-R  DBM107 006074-R  DBM21  005166-R
BIT00  000001    BIT11  004000  BIT9   001000  CCTLR  105172-R  CST    077262-R  DBM108 006132-R  DBM22  005250-R
BIT01  000002    BIT12  010000  BL$DIV 077062-R  CDISK  105174-R  CST.AD 077410-R  DBM109 006222-R  DBM23  005314-R
BIT02  000004    BIT13  020000  BL$LAS 107216-R  CER.01 016416-R  CTLR.C 105200-R  DBM111 006302-R  DBM25  005352-R
BIT03  000010    BIT14  040000  BL$MOD 077074-R  CER.02 016462-R  CTLR.I 043530-R  DBM112 006402-R  DBM26  005420-R
BIT04  000020    BIT15  100000  BL$MUL 076636-R  CLK.PR 105252-R  CUOFF  105176-R  DBM12  004650-R  DBM27  005452-R
BIT05  000040    BIT2   000004  BL$SHF 077106-R  CLK.TI 105250-R  C.ERR. 101064-R  DBM120 006504-R  DBM28A 005524-R
```

```
DBM288 005564-R  EGD.22 011714-R  EX.CBC 015344-R  GP$3   024770-R  IRDRX. 077440-R  L$RPT  026522-R  QIO.UN 053312-R
DBM29  005624-R  EGD.23 011754-R  EX.CBR 015410-R  GP$30  025350-R  ISR    000100    L$SFTL 025142-R  RANDOM 101024-R
DBM32  005672-R  EGD.24 012016-R  EX.CBW 015462-R  GP$31  025364-R  IXE    004000    L$SOFT 025144-R  RDM.CN 101022-R
DBM5   004622-R  EGH.30 012062-R  EX.CMD 014474-R  GP$32  025402-R  LOE    040000    L$SPC  002056-R  RDRX.A 077436-R
DCT    077412-R  EGS.01 011046-R  EX.CMP 014534-R  GP$33  025414-R  LOT    000010    L$SPCP 002020-R  RDRX.E 024172-R
DCT.AD 077434-R  EGS.02 011066-R  EX.LBN 015054-R  GP$4   025002-R  L$ACP  002110-R  L$SPTP 002024-R  RD.COU 105262-R
DFPTBL 024564-R  EH.0   012540-R  EX.LBR 015150-R  GP$5   025014-R  L$APT  002036-R  L$STA  002030-R  REG.EX 044156-R
DF.MSG 024210-R  EH.1   012576-R  EX.LBW 015216-R  GP$6   025024-R  L$AU   032274-R  L$SW   024614-R  RETPKT 102614-R
DIO.RE 062616-R  EH.10  013152-R  EX.ONL 014550-R  GP$7   025034-R  L$AUT  002070-R  L$SWLE 024612-R  ROUND. 066020-R
DISK.R 070624-R  EH.12  013202-R  EX.OP  014574-R  GP$8   025044-R  L$AUTO 031302-R  L$TEST 002114-R  RPS.RE 067002-R
DRIVER 043176-R  EH.13  013236-R  EX.PBN 015112-R  GP$9   025056-R  L$CCP  002106-R  L$TIML 002014-R  RPT1   007554-R
DROP.C 033746-R  EH.2   012634-R  EX.RBN 015264-R  HARD.E 063642-R  L$CLEA 031576-R  L$UNIT 002012-R  RPT10  010264-R
DRV.CT 034054-R  EH.3   012674-R  EX.RD  014514-R  HARD.I 045174-R  L$CO   002032-R  MD.INI 052070-R  RPT11  010352-R
DR.ERR 050620-R  EH.4   012732-R  EX.RP  016046-R  HOE    100000    L$DEPO 002011-R  MINUTE 105246-R  RPT12  010420-R
DR.RET 067170-R  EH.5   012756-R  EX.SA  014400-R  HOE.FL 105253-R  L$DESC 024724-R  MODULA 034606-R  RPT13  010466-R
DUP    055576-R  EH.6   013006-R  EX.SB  014452-R  HOST.W 066364-R  L$DESP 002076-R  MSCP.P 101066-R  RPT14  010566-R
DUPCOM 060436-R  EH.7   013036-R  EX.SBO 014446-R  HOURS  105245-R  L$DEVP 002060-R  MSG.01 007434-R  RPT15  010664-R
DUPIDL 060604-R  EH.8   013066-R  EX.SC  014416-R  HRD.MS 024306-R  L$DISP 002124-R  MSG.02 007466-R  RPT16  010764-R
DUPPKT 100014-R  EH.9   013122-R  EX.TIM 016144-R  HRD.SU 024504-R  L$DLY  002116-R  MSG.03 007522-R  RPT2   007640-R
DUPRED 057302-R  ELG.FM 014366-R  EX.WRD 016134-R  HWPTE0 024600-R  L$DTP  002040-R  MULTI. 051414-R  RPT3   007704-R
DUPROU 024630-R  ELG.00 014034-R  EX.WRT 014524-R  HWPTE1 024602-R  L$DTYP 002034-R  NAME.H 024606-R  RPT4   007770-R
DUPWRT 056132-R  ELOG.P 103370-R  FATAL. 070014-R  HWPTS0 024574-R  L$DU   032202-R  NAME.L 024604-R  RPT5   010034-R
DUP.CO 063150-R  EMS.BL 036502-R  FER.BC 105272-R  HWPTS1 024576-R  L$DUT  002072-R  NEX    105230-R  RPT6   010122-R
DUP.FL 105170-R  EMS.CM 040532-R  FER.LB 105270-R  HWPT.B 024570-R  L$DVTY 024704-R  NEXT.P 105244-R  RPT7   010166-R
DUP.RS 067522-R  EMS.DB 036304-R  FILL.B 062170-R  HWPT.D 024572-R  L$EF   002052-R  NEX.TR 032304-R  RPT8   010204-R
DUR    105202-R  EMS.EL 037410-R  FORCED 105267-R  HWPT.I 024564-R  L$ENVI 002044-R  NULL   004620-R  RPT9   010232-R
DU.MSG 006666-R  EMS.ER 041060-R  FREE.M 105210-R  HWPT.V 024566-R  L$ERRT 002126-R  OF.RC  105222-R  RP.ADD 103366-R
DU.RSN 007406-R  EMS.RP 037352-R  FSET.U 063516-R  HWQ1   002136-R  L$ETP  002102-R  OUT.IO 033622-R  RP.IND 103364-R
D$PCNT 002122-R  EMS.01 041264-R  GET.IO 033442-R  HWQ10  002740-R  L$EXP1 002046-R  OVF.CH 065744-R  RP.USE 103354-R
D.FAIL 105266-R  EMS.10 041322-R  GET.PK 032572-R  HWQ11  002770-R  L$EXP4 002064-R  PKT.US 102600-R  SA.REG 105224-R
EBD.10 012150-R  EMS.12 041364-R  GET.RA 053202-R  HWQ2   002152-R  L$EXP5 002066-R  PNT    001000    SB.COD 105216-R
EBD.12 012210-R  EMS.13 041422-R  GET.RE 033320-R  HWQ3   002162-R  L$HARD 024750-R  POLL.C 070246-R  SCAN.E 072070-R
EBD.13 012256-R  EMS.14 041464-R  GP$DIS 025320-R  HWQ4   002226-R  L$HIME 002120-R  POLL.R 070346-R  SC.CLK 021550-R
EBD.14 012310-R  EMS.18 041526-R  GP$1   024750-R  HWQ5   002244-R  L$HPCP 002016-R  PRI    002000    SC.CON 016562-R
EBD.18 012350-R  EMS.21 041570-R  GP$10  025070-R  HWQ6A  002314-R  L$HPTP 002022-R  PRI00  000000    SC.CTO 021012-R
EBD.19 012404-R  EMS.22 041606-R  GP$11  025104-R  HWQ6B  002366-R  L$HRDL 024746-R  PRI01  000040    SC.DIS 017110-R
EBD.24 012464-R  EMS.24 041624-R  GP$12  025120-R  HWQ7A  002442-R  L$HW   024564-R  PRI02  000100    SC.DST 017520-R
EBS.01 012106-R  EMS.30 041666-R  GP$13  025130-R  HWQ7B  002512-R  L$HWLE 024562-R  PRI03  000140    SC.DS2 017574-R
EF.CON 000036    ENTRY. 105166-R  GP$14  025144-R  HWQ8   002562-R  L$ICP  002104-R  PRI04  000200    SC.DUP 016604-R
EF.NEW 000035    EOP.FL 105167-R  GP$15  025156-R  HWQ9   002640-R  L$INIT 031270-R  PRI05  000240    SC.ECC 017646-R
EF.PWR 000034    ERRBLK 002134-R  GP$16  025170-R  IBE    010000    L$LADP 002026-R  PRI06  000300    SC.ECD 017730-R
EF.RES 000037    ERRMSG 002132-R  GP$17  025202-R  IDU    000040    L$LAST 107222-R  PRI07  000340    SC.EC1 020200-R
EF.STA 000040    ERRNBR 002130-R  GP$18  025214-R  IER    020000    L$LOAD 002100-R  PROC.R 062422-R  SC.EC2 020230-R
EGD.10 011160-R  ERRTYP 002126-R  GP$19  025222-R  INIT.I 052230-R  L$LUN  002074-R  PUTA.B 033562-R  SC.EC3 020260-R
EGD.11 011220-R  ERR.CO 014000-R  GP$2   024760-R  INIT.O 105274-R  L$MREV 002050-R  PUT.IO 033516-R  SC.EC4 020312-R
EGD.12 011244-R  ERR.00 013310-R  GP$20  025230-R  INIT.T 043040-R  L$NAME 002000-R  PUT.PK 033160-R  SC.EC5 020342-R
EGD.13 011272-R  EVL    000004    GP$21  025236-R  INI.CT 044032-R  L$NDHR 025140-R  PUT.RE 033426-R  SC.EC6 020372-R
EGD.14 011320-R  EX.ACC 014562-R  GP$22  025244-R  INI.RR 046044-R  L$NDHW 024610-R  P.INDE 105260-R  SC.EC7 020422-R
EGD.15 011350-R  EX.BB  014600-R  GP$23  025252-R  INT.GE 044714-R  L$NDSF 025432-R  QIO    105206-R  SC.EC8 020454-R
EGD.16 011366-R  EX.BBU 014764-R  GP$24  025260-R  IN.IOD 033660-R  L$NDSW 024674-R  QIO.FU 054716-R  SC.EC9 020506-R
EGD.17 011416-R  EX.BB1 014670-R  GP$25  025274-R  IODQ   105152-R  L$PRIO 002042-R  QIO.GE 052602-R  SC.EDC 021142-R
EGD.18 011434-R  EX.BC  015534-R  GP$26  025304-R  IODQ.I 105162-R  L$PROT 024676-R  QIO.LB 061030-R  SC.FCT 020132-R
EGD.19 011454-R  EX.BD  015610-R  GP$27  025312-R  IODQ.O 105164-R  L$PRT  002112-R  QIO.OK 052434-R  SC.FER 017202-R
EGD.20 011514-R  EX.BDR 015666-R  GP$28  025324-R  IO.RET 063246-R  L$REPP 002062-R  QIO.OU 052530-R  SC.FE2 017270-R
EGD.21 011602-R  EX.BDW 015756-R  GP$29  025340-R  IPKT.A 102576-R  L$REV  002010-R  QIO.SI 061760-R  SC.FUL 020002-R
```

NF.

SC.HWP	020610-R	SC.RDY	021466-R	SEQUEN	070724-R	SWP.ER	024614-R	SWQ15	003620-R	SWQ9	003220-R	UPD.IO	065424-R
SC.IDS	021202-R	SC.RSP	021576-R	SET.CP	032376-R	SWP.FL	024620-R	SWQ17	003670-R	S.DUPP	105256-R	VEC.BR	044370-R
SC.IOP	017036-R	SC.SDI	016536-R	SET.CT	046146-R	SWP.RA	024624-R	SWQ19	003766-R	S.PATT	105254-R	WAIT	034570 R
SC.ISH	017360-R	SC.SDS	021064-R	SET.UP	032472-R	SWP.TI	024626-R	SWQ2	003044-R	TALLY	077462-R	XX13	016204-R
SC.IS2	017440 R	SC.SDN	016656-R	SFPTBL	024614-R	SWP.UC	024632-R	SWQ20	004056-R	TEMP1	105236-R	XX23	016226 R
SC.NXM	020722 R	SC.SRI	021344-R	SFT.MS	024404 R	SWP.UD	024634-R	SWQ21	004144-R	TEMP2	105240-R	XX32	016262-R
SC.ODA	020650-R	SC.SRT	021252-R	SOFT.E	073662-R	SWP.XF	024616-R	SWQ22	004230-R	TIME	032314-R	XX33	016310 R
SC.ODB	020700-R	SC.SUR	021644-R	SPACE4	024536-R	SWQ1	003022-R	SWQ23	004270-R	TRK.SG	101016-R	XX34	016346 R
SC.ONL	016634-R	SC.SWP	020550-R	STEP	105220-R	SWQ10	003274-R	SWQ24	004342-R	T$FREE	107342-R	$END.L	107344 R
SC.PAR	020756-R	SC.UNK	016676-R	ST.COD	105214-R	SWQ11	003340-R	SWQ25	004406-R	T$PTHV	000004	$SAVE2	077154-R
SC.POE	021432-R	SC.VOL	016756-R	SWEEP	066650-R	SWQ12	003372-R	SWQ26	004460-R	T.ADDR	100012-R	$SAVE3	077170-R
SC.PSP	021674-R	SC.576	020056-R	SWM1	004522-R	SWQ13	003470-R	SWQ4	003124-R	T1	043024-R	$SAVE4	077206 R
SC.RCT	017762-R	SEND	034132-R	SWP.DP	024622-R	SWQ14	003546-R	SWQ7	003146-R	UAM	000200	$SAVE5	077226 R

*** Task builder statistics:

Total work file references: 155891.
Work file reads: 0.
Work file writes: 0.
Size of core pool: 5616. words (21. pages)
Size of work file: 5120. words (20. pages)

Elapsed time:00:01:14

SYMBOL	VALUE	REFERENCES...		
ADDR.V	105275-R	# ZRQAM1	ZRQAM2	ZRQAM3
ADR	000020	# ZRQAM1	# ZRQAM2	# ZRQAM3
ASTERI	024554-R	# ZRQAM1	ZRQAM2	
AZINT	067322-R	# ZRQAM3		
AZINTO	067304-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$DIV	077062-R	# B16MUL	ZRQAM2	ZRQAM3
BL$LAS	107216-R	# ZRQAM4		
BL$MOD	077074-R	# B16MUL	ZRQAM2	ZRQAM3
BL$MUL	076636-R	# B16MUL	ZRQAM2	ZRQAM3
BL$SHF	077106-R	# B16MUL	ZRQAM3	
BOE	000400	# ZRQAM1	# ZRQAM2	# ZRQAM3
BRLEVE	105264-R	# ZRQAM1	ZRQAM2	ZRQAM3
BST	077442-R	# ZRQAM1	ZRQAM2	ZRQAM3
BUFF.A	105122-R	# ZRQAM1	ZRQAM2	ZRQAM3
BUFF.O	105142-R	# ZRQAM1	ZRQAM2	ZRQAM3
BYTS.P	105212-R	# ZRQAM1	ZRQAM2	ZRQAM3
CCTLR	105172-R	# ZRQAM1	ZRQAM2	ZRQAM3
CDISK	105174-R	# ZRQAM1	ZRQAM2	ZRQAM3
CER.01	016416-R	# ZRQAM1	ZRQAM2	
CER.02	016462-R	# ZRQAM1	ZRQAM2	
CLK.PR	105252-R	# ZRQAM1	ZRQAM2	ZRQAM3
CLK.TI	105250-R	# ZRQAM1	ZRQAM2	ZRQAM3
CMD.TI	105226-R	# ZRQAM1	ZRQAM2	ZRQAM3
CNTR.E	023452-R	# ZRQAM1	ZRQAM2	
CREDIT	105242-R	# ZRQAM1	ZRQAM2	ZRQAM3
CRLF	024542-R	# ZRQAM1	ZRQAM2	ZRQAM3

SYMBOL	VALUE	REFERENCES...		
CRN.HI	105234-R	# ZRQAM1	ZRQAM2	ZRQAM3
CRN.LO	105232-R	# ZRQAM1	ZRQAM2	ZRQAM3
CST	077262-R	# ZRQAM1	ZRQAM2	ZRQAM3
CST.AD	077410-R	# ZRQAM1	ZRQAM2	ZRQAM3
CTLR.C	105200-R	# ZRQAM1	ZRQAM2	ZRQAM3
CTLR.I	043530-R	# ZRQAM3		
CUOFF	105176-R	# ZRQAM1	ZRQAM2	ZRQAM3
C.ERR.	101064-R	# ZRQAM1	ZRQAM2	ZRQAM3
DASH	024546-R	# ZRQAM1	ZRQAM2	
DATAGM	072736-R	# ZRQAM3		
DBM101	005746-R	# ZRQAM1		
DBM104	005774-R	# ZRQAM1		
DBM105	006036-R	# ZRQAM1		
DBM107	006074-R	# ZRQAM1	ZRQAM2	
DBM108	006132-R	# ZRQAM1	ZRQAM3	
DBM109	006222-R	# ZRQAM1	ZRQAM3	
DBM111	006302-R	# ZRQAM1	ZRQAM3	
DBM112	006402-R	# ZRQAM1	ZRQAM3	
DBM12	004650-R	# ZRQAM1	ZRQAM3	
DBM120	006504-R	# ZRQAM1	ZRQAM3	
DBM121	006576-R	# ZRQAM1	ZRQAM3	
DBM15	004724-R	# ZRQAM1		
DBM18	004754-R	# ZRQAM1	ZRQAM3	
DBM19	005026-R	# ZRQAM1	ZRQAM3	
DBM20	005112-R	# ZRQAM1	ZRQAM3	
DBM21	005166-R	# ZRQAM1	ZRQAM3	
DBM22	005250-R	# ZRQAM1	ZRQAM3	
DBM23	005314-R	# ZRQAM1	ZRQAM3	
DBM25	005352-R	# ZRQAM1	ZRQAM3	
DBM26	005420-R	# ZRQAM1	ZRQAM3	
DBM27	005452-R	# ZRQAM1	ZRQAM3	
DBM28A	005524-R	# ZRQAM1		
DBM28B	005564-R	# ZRQAM1		
DBM29	005624-R	# ZRQAM1	ZRQAM3	
DBM32	005672-R	# ZRQAM1		
DBM5	004622-R	# ZRQAM1	ZRQAM2	
DCT	077412-R	# ZRQAM1	ZRQAM2	ZRQAM3
DCT.AD	077434-R	# ZRQAM1	ZRQAM2	ZRQAM3
DFPTBL	024564-R	# ZRQAM1		
DF.MSG	024210-R	# ZRQAM1	ZRQAM3	
DIO.RE	062516-R	# ZRQAM3		
DISK.R	070624-R	# ZRQAM3		
DRIVER	043176-R	# ZRQAM3		
DROP.C	033746-R	# ZRQAM2	ZRQAM3	
DRV.CT	034054-R	# ZRQAM2	ZRQAM3	
DR.ERR	050620-R	# ZRQAM3		
DR.RET	067170-R	# ZRQAM3		
DUP	055576-R	# ZRQAM3		
DUPCOM	060436-R	# ZRQAM3		
DUPIDL	060604-R	# ZRQAM3		
DUPPKT	100014-R	# ZRQAM1	ZRQAM2	ZRQAM3
DUPRED	0573C2-R	# ZRQAM3		

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
DUPROU	024630-R	# ZRQAM1	ZRQAM3	
DUPWRT	056132-R	# ZRQAM3		
DUP.CO	063150-R	# ZRQAM3		
DUP.FL	105170-R	# ZRQAM1	ZRQAM2	ZRQAM3
DUP.RS	067522-R	# ZRQAM3		
DUR	105202-R	# ZRQAM1	ZRQAM2	ZRQAM3
DU.MSG	006666-R	# ZRQAM1	ZRQAM2	
DU.RSN	007406-R	# ZRQAM1	ZRQAM2	
D$PCNT	002122-R	# ZRQAM1		
D.FAIL	105266-R	# ZRQAM1	ZRQAM2	ZRQAM3
EBD.10	012150-R	# ZRQAM1	ZRQAM2	
EBD.12	012210-R	# ZRQAM1	ZRQAM2	
EBD.13	012256-R	# ZRQAM1	ZRQAM2	
EBD.14	012310-R	# ZRQAM1	ZRQAM2	
EBD.18	012350-R	# ZRQAM1	ZRQAM2	
EBD.19	012404-R	# ZRQAM1	ZRQAM2	
EBD.24	012464-R	# ZRQAM1	ZRQAM2	
EBS.01	012106-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	011160-R	# ZRQAM1	ZRQAM3	
EGD.11	011220-R	# ZRQAM1	ZRQAM3	
EGD.12	011244-R	# ZRQAM1	ZRQAM3	
EGD.13	011272-R	# ZRQAM1	ZRQAM3	
EGD.14	011320-R	# ZRQAM1	ZRQAM3	
EGD.15	011350-R	# ZRQAM1	ZRQAM3	
EGD.16	011366-R	# ZRQAM1	ZRQAM3	
EGD.17	011416-R	# ZRQAM1	ZRQAM3	
EGD.18	011434-R	# ZRQAM1	ZRQAM3	
EGD.19	011454-R	# ZRQAM1	ZRQAM3	
EGD.20	011514-R	# ZRQAM1	ZRQAM3	
EGD.21	011602-R	# ZRQAM1	ZRQAM3	
EGD.22	011714-R	# ZRQAM1	ZRQAM3	
EGD.23	011754-R	# ZRQAM1	ZRQAM3	
EGD.24	012016-R	# ZRQAM1	ZRQAM3	
EGH.30	012062-R	# ZRQAM1	ZRQAM3	
EGS.01	011046-R	# ZRQAM1	ZRQAM2	
EGS.02	011066-R	# ZRQAM1	ZRQAM3	
EH.0	012540-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.1	012576-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.10	013152-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.12	013202-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.13	013236-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.2	012634-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.3	012674-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.4	012732-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.5	012756-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.6	013006-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.7	013036-R	# ZRQAM1	ZRQAM2	ZRQAM3

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
EM.8	013066-R	# ZRQAM1	ZRQAM2	ZRQAM3
EM.9	013122-R	# ZRQAM1	ZRQAM2	ZRQAM3
ELG.FM	014366-R	# ZRQAM1	ZRQAM2	
ELG.OO	014034-R	# ZRQAM1	ZRQAM2	
ELOG.P	103370-R	# ZRQAM1	ZRQAM2	ZRQAM3
EMS.BL	036502-R	# ZRQAM2		
EMS.CM	040532-R	# ZRQAM2	ZRQAM3	
EMS.DB	036304-R	# ZRQAM2		
EMS.EL	037410-R	# ZRQAM2	ZRQAM3	
EMS.ER	041060-R	# ZRQAM2	ZRQAM3	
EMS.RP	037352-R	# ZRQAM2	ZRQAM3	
EMS.01	041264-R	# ZRQAM2		
EMS.10	041322-R	# ZRQAM2	ZRQAM3	
EMS.12	041364-R	# ZRQAM2	ZRQAM3	
EMS.13	041422-R	# ZRQAM2	ZRQAM3	
EMS.14	041464-R	# ZRQAM2	ZRQAM3	
EMS.18	041526-R	# ZRQAM2	ZRQAM3	
EMS.21	041570-R	# ZRQAM2	ZRQAM3	
EMS.22	041606-R	# ZRQAM2	ZRQAM3	
EMS.24	041624-R	# ZRQAM2	ZRQAM3	
EMS.30	041666-R	# ZRQAM2	ZRQAM3	
ENTRY.	105166-R	# ZRQAM1	ZRQAM2	ZRQAM3
EOP.FL	105167-R	# ZRQAM1	ZRQAM2	ZRQAM3
ERRBLK	002134-R	# ZRQAM1		
ERRMSG	002132-R	# ZRQAM1		
ERRNBR	002130-R	# ZRQAM1		
ERRTYP	002126-R	# ZRQAM1		
ERR.CO	014000-R	# ZRQAM1	ZRQAM2	
ERR.OO	013310-R	# ZRQAM1	ZRQAM2	
EVL	000004	# ZRQAM1	# ZRQAM2	# ZRQAM3
EX.ACC	014562-R	# ZRQAM1	ZRQAM2	
EX.BB	014600-R	# ZRQAM1	ZRQAM2	
EX.BBU	014764-R	# ZRQAM1	ZRQAM2	
EX.BB1	014670-R	# ZRQAM1	ZRQAM2	
EX.BC	015534-R	# ZRQAM1	ZRQAM2	
EX.BD	015610-R	# ZRQAM1	ZRQAM2	
EX.BDR	015666-R	# ZRQAM1	ZRQAM2	
EX.BDW	015756-R	# ZRQAM1	ZRQAM2	
EX.CBC	015344-R	# ZRQAM1	ZRQAM2	
EX.CBR	015410-R	# ZRQAM1	ZRQAM2	
EX.CBW	015462-R	# ZRQAM1	ZRQAM2	
EX.CMD	014474-R	# ZRQAM1	ZRQAM2	
EX.CMP	014534-R	# ZRQAM1	ZRQAM2	
EX.LBN	015054-R	# ZRQAM1	ZRQAM2	
EX.LBR	015150-R	# ZRQAM1	ZRQAM2	
EX.LBW	015216-R	# ZRQAM1	ZRQAM2	
EX.ONL	014550-R	# ZRQAM1	ZRQAM2	
EX.OP	014574-R	# ZRQAM1	ZRQAM2	
EX.PBN	015112-R	# ZRQAM1	ZRQAM2	
EX.RBN	015264-R	# ZRQAM1	ZRQAM2	
EX.RD	014514-R	# ZRQAM1	ZRQAM2	
EX.RP	016046-R	# ZRQAM1	ZRQAM2	

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
EX.SA	014400-R	# ZRQAM1	ZRQAM2	
EX.SB	014452-R	# ZRQAM1	ZRQAM2	
EX.SBO	014446-R	# ZRQAM1	ZRQAM2	
EX.SC	014416-R	# ZRQAM1	ZRQAM2	
EX.TIM	016144-R	# ZRQAM1	ZRQAM2	
EX.WRD	016134-R	# ZRQAM1	ZRQAM2	
EX.WRT	014524-R	# ZRQAM1	ZRQAM2	
FATAL.	070014-R	# ZRQAM3		
FER.BC	105272-R	# ZRQAM1	ZRQAM2	ZRQAM3
FER.LB	105270-R	# ZRQAM1	ZRQAM2	ZRQAM3
FILL.B	062170-R	# ZRQAM3		
FORCED	105267-R	# ZRQAM1	ZRQAM2	ZRQAM3
FREE.M	105210-R	# ZRQAM1	ZRQAM2	ZRQAM3
FSET.U	063516-R	# ZRQAM3		
GET.IO	033442-R	# ZRQAM2	ZRQAM3	
GET.PK	032572-R	# ZRQAM2	ZRQAM3	
GET.RA	053202-R	# ZRQAM3		
GET.RE	033320-R	# ZRQAM2	ZRQAM3	
GP$DIS	025320-R	# ZRQAM2		
GP$	024750-R	# ZRQAM2		
GP$10	025070-R	# ZRQAM2		
GP$11	025104-R	# ZRQAM2		
GP$12	025120-R	# ZRQAM2		
GP$13	025130 R	# ZRQAM2		
GP$14	025144-R	# ZRQAM2		
GP$15	025156-R	# ZRQAM2		
GP$16	025170-R	# ZRQAM2		
GP$17	025202-R	# ZRQAM2		
GP$18	025214-R	# ZRQAM2		
GP$19	025222-R	# ZRQAM2		
GP$2	024760-R	# ZRQAM2		
GP$20	025230-R	# ZRQAM2		
GP$21	025236-R	# ZRQAM2		
GP$22	025244-R	# ZRQAM2		
GP$23	025252-R	# ZRQAM2		
GP$24	025260-R	# ZRQAM2		
GP$25	025274-R	# ZRQAM2		
GP$26	025304-R	# ZRQAM2		
GP$27	025312-R	# ZRQAM2		
GP$28	025324-R	# ZRQAM2		
GP$29	025340-R	# ZRQAM2		
GP$3	024770-R	# ZRQAM2		
GP$30	025350-R	# ZRQAM2		
GP$31	025364-R	# ZRQAM2		
GP$32	025402-R	# ZRQAM2		
GP$33	025414-R	# ZRQAM2		
GP$4	025002-R	# ZRQAM2		
GP$5	025014-R	# ZRQAM2		
GP$6	025024-R	# ZRQAM2		
GP$7	025034-R	# ZRQAM2		
GP$8	025044-R	# ZRQAM2		
GP$9	025056-R	# ZRQAM2		

GLOBAL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES...		
HARD.E	063642-R	# ZRQAM3		
HARD.I	045174-R	# ZRQAM3		
HOE	100000	# ZRQAM1	# ZRQAM2	# ZRQAM3
HOE.FL	105253-R	# ZRQAM1	ZRQAM2	ZRQAM3
HOST.W	066364-R	# ZRQAM3		
HOURS	105245-R	# ZRQAM1	ZRQAM2	ZRQAM3
HRD.MS	024306-R	# ZRQAM1	ZRQAM3	
HRD.SU	024504-R	# ZRQAM1	ZRQAM3	
HWPTE0	024600-R	# ZRQAM1		
HWPTE1	024602-R	# ZRQAM1		
HWPTS0	024574-R	# ZRQAM1		
HWPTS1	024576-R	# ZRQAM1		
HWPT.B	024570-R	# ZRQAM1		
HWPT.D	024572-R	# ZRQAM1		
HWPT.I	024564-R	# ZRQAM1		
HWPT.V	024566-R	# ZRQAM1		
HWQ1	002136-R	# ZRQAM1	ZRQAM2	
HWQ10	002740-R	# ZRQAM1	ZRQAM2	
HWQ11	002770-R	# ZRQAM1	ZRQAM2	
HWQ2	002152-R	# ZRQAM1	ZRQAM2	
HWQ3	002162-R	# ZRQAM1	ZRQAM2	
HWQ4	002226-R	# ZRQAM1	ZRQAM2	
HWQ5	002244-R	# ZRQAM1	ZRQAM2	
HWQ6A	002314-R	# ZRQAM1	ZRQAM2	
HWQ6B	002366-R	# ZRQAM1	ZRQAM2	
HWQ7A	002442-R	# ZRQAM1	ZRQAM2	
HWQ7B	002512-R	# ZRQAM1	ZRQAM2	
HWQ8	002562-R	# ZRQAM1	ZRQAM2	
HWQ9	002640-R	# ZRQAM1	ZRQAM2	
IBE	010000	# ZRQAM1	# ZRQAM2	# ZRQAM3
IDU	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
IER	020000	# ZRQAM1	# ZRQAM2	# ZRQAM3
INIT.I	052230-R	# ZRQAM3		
INIT.O	105274-R	# ZRQAM1	ZRQAM2	ZRQAM3
INIT.T	043040-R	# ZRQAM3		
INI.CT	044032-R	# ZRQAM3		
INI.RR	046044-R	# ZRQAM3		
INT.GE	044714-R	# ZRQAM3		
IN.IOD	033660-R	# ZRQAM2	ZRQAM3	
IODQ	105152-R	# ZRQAM1	ZRQAM2	ZRQAM3
IODQ.I	105162-R	# ZRQAM1	ZRQAM2	ZRQAM3
IODQ.O	105164-R	# ZRQAM1	ZRQAM2	ZRQAM3
IO.RET	063246-R	# ZRQAM3		
IPKT.A	102576-R	# ZRQAM1	ZRQAM2	ZRQAM3
IRDRX.	077440-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$ACP	002110-R	# ZRQAM1		
L$APT	002036-R	# ZRQAM1		
L$AU	032274-R	ZRQAM1	# ZRQAM2	

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
L$AUT	002070-R	# ZRQAM1		
L$AUTO	031302-R	ZRQAM1	# ZRQAM2	
L$CCP	002106-R	# ZRQAM1		
L$CLEA	031576-R	ZRQAM1	# ZRQAM2	
L$CO	002032-R	# ZRQAM1		
L$DEPO	002011-R	# ZRQAM1		
L$DESC	024724-R	ZRQAM1	# ZRQAM2	
L$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	032202-R	ZRQAM1	# ZRQAM2	
L$DUT	002072-R	# ZRQAM1		
L$DVTY	024704-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	024750-R	ZRQAM1	# ZRQAM2	
L$HIME	002120-R	# ZRQAM1	ZRQAM2	
L$HPCP	002016-R	# ZRQAM1		
L$HPTP	002022-R	# ZRQAM1		
L$HRDL	024746-R	# ZRQAM2		
L$HW	024564-R	# ZRQAM1		
L$HWLE	024562-R	# ZRQAM1		
L$ICP	002104-R	# ZRQAM1		
L$INIT	031270-R	ZRQAM1	# ZRQAM2	
L$LADP	002026-R	# ZRQAM1		
L$LAST	107222-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$NDHR	025140-R	# ZRQAM2		
L$NDHW	024610-R	# ZRQAM1		
L$NDSF	025432-R	# ZRQAM2		
L$NDSW	024674-R	# ZRQAM1		
L$PRIO	002042-R	# ZRQAM1		
L$PROT	024676-R	# ZRQAM1		
L$PRT	002112-R	# ZRQAM1		
L$REPP	002062-R	# ZRQAM1		
L$REV	002010-R	# ZRQAM1		
L$RPT	026522-R	ZRQAM1	# ZRQAM2	
L$SFTL	025142-R	# ZRQAM2		
L$SOFT	025144-R	ZRQAM1	# ZRQAM2	
L$SPC	002056-R	# ZRQAM1		
L$SPCP	002020-R	# ZRQAM1		

GLOBAL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES...		
L$SPTP	002024-R	# ZRQAM1		
L$STA	002030-R	# ZRQAM1		
L$SW	024614-R	# ZRQAM1		
L$SWLE	024612-R	# ZRQAM1		
L$TEST	002114-R	# ZRQAM1		
L$TIML	002014-R	# ZRQAM1		
L$UNIT	002012-R	# ZRQAM1	ZRQAM2	ZRQAM3
MD.INI	052070-R	# ZRQAM3		
MINUTE	105246-R	# ZRQAM1	ZRQAM2	ZRQAM3
MODULA	034606-R	# ZRQAM2	ZRQAM3	
MSCP.P	101066-R	# ZRQAM1	ZRQAM2	ZRQAM3
MSG.01	007434-R	# ZRQAM1	ZRQAM2	
MSG.02	007466-R	# ZRQAM1	ZRQAM3	
MSG.03	007522-R	# ZRQAM1	ZRQAM3	
MULTI.	051414-R	# ZRQAM3		
NAME.H	024606-R	# ZRQAM1		
NAME.L	024604-R	# ZRQAM1		
NEX	105230-R	# ZRQAM1	ZRQAM2	ZRQAM3
NEXT.P	105244-R	# ZRQAM1	ZRQAM2	ZRQAM3
NEX.TR	032304-R	# ZRQAM2	ZRQAM3	
NULL	004620-R	# ZRQAM1	ZRQAM2	
OF.RC	105222-R	# ZRQAM1	ZRQAM2	ZRQAM3
OUT.IO	033622-R	# ZRQAM2	ZRQAM3	
OVF.CH	065744-R	# ZRQAM3		
PKT.US	102600-R	# ZRQAM1	ZRQAM2	ZRQAM3
PNT	001000	# ZRQAM1	# ZRQAM2	# ZRQAM3
POLL.C	070246-R	# ZRQAM3		
POLL.R	070346-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	062422-R	# ZRQAM3		
PUTA.B	033562-R	# ZRQAM2	ZRQAM3	
PUT.IO	033516-R	# ZRQAM2	ZRQAM3	
PUT.PK	033160-R	# ZRQAM2	ZRQAM3	
PUT.RE	033426-R	# ZRQAM2	ZRQAM3	
P.INDE	105260-R	# ZRQAM1	ZRQAM2	ZRQAM3
QIO	105206-R	# ZRQAM1	ZRQAM2	ZRQAM3
QIO.FU	054716-R	# ZRQAM3		
QIO.GE	052602-R	# ZRQAM3		
QIO.LB	061030-R	# ZRQAM3		
QIO.OK	052434-R	# ZRQAM3		
QIO.OU	052530-R	# ZRQAM3		
QIO.SI	061760-R	# ZRQAM3		
QIO.UN	053312-R	# ZRQAM3		
RANDOM	101024-R	# ZRQAM1	ZRQAM2	ZRQAM3

GLOBAL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES...		
RDM.CN	101022-R	# ZRQAM1	ZRQAM2	ZRQAM3
RDRX.A	077436-R	# ZRQAM1	ZRQAM2	ZRQAM3
RDRX.E	024172-R	# ZRQAM1	ZRQAM2	
RD.COU	105262-R	# ZRQAM1	7RQAM2	ZRQAM3
REG.EX	044156-R	# ZRQAM3		
RETPKT	102614-R	# ZRQAM1	ZRQAM2	ZRQAM3
ROUND.	066020-R	# ZRQAM3		
RPS.RE	067002-R	# ZRQAM3		
RPT1	007554-R	# ZRQAM1	ZRQAM2	
RPT10	010264-R	# ZRQAM1	ZRQAM2	
RPT11	010352-R	# ZRQAM1	ZRQAM2	
RPT12	010420-R	# ZRQAM1	ZRQAM2	
RPT13	010466-R	# ZRQAM1	ZRQAM2	
RPT14	010566-R	# ZRQAM1	ZRQAM2	
RPT15	010664-R	# ZRQAM1	ZRQAM2	
RPT16	010764-R	# ZRQAM1	ZRQAM2	
RPT2	007640-R	# ZRQAM1	ZRQAM2	
RPT3	007704-R	# ZRQAM1	ZRQAM2	
RPT4	007770-R	# ZRQAM1	ZRQAM2	
RPT5	010034-R	# ZRQAM1	ZRQAM2	
RPT6	010122-R	# ZRQAM1	ZRQAM2	
RPT7	010166-R	# ZRQAM1	ZRQAM2	
RPT8	010204-R	# ZRQAM1	ZRQAM2	
RPT9	010232-R	# ZRQAM1	ZRQAM2	
RP.ADD	103366-R	# ZRQAM1	ZRQAM2	ZRQAM3
RP.IND	103364-R	# ZRQAM1	ZRQAM2	ZRQAM3
RP.USE	103354-R	# ZRQAM1	ZRQAM2	ZRQAM3
SA.REG	105224-R	# ZRQAM1	ZRQAM2	ZRQAM3
SB.COD	105216-R	# ZRQAM1	ZRQAM2	ZRQAM3
SCAN.E	072070-R	# ZRQAM3		
SC.CLK	021550-R	# ZRQAM1	ZRQAM2	
SC.CON	016562-R	# ZRQAM1	ZRQAM2	
SC.CTO	021012-R	# ZRQAM1	ZRQAM2	
SC.DIS	017110-R	# ZRQAM1	ZRQAM2	
SC.DST	017520-R	# ZRQAM1	ZRQAM2	
SC.DS2	017574-R	# ZRQAM1	ZRQAM2	
SC.DUP	016604-R	# ZRQAM1	ZRQAM2	
SC.ECC	017646-R	# ZRQAM1	ZRQAM2	
SC.ECD	017730-R	# ZRQAM1	ZRQAM2	
SC.EC1	020200-R	# ZRQAM1	ZRQAM2	
SC.EC2	020230-R	# ZRQAM1	ZRQAM2	
SC.EC3	020260-R	# ZRQAM1	ZRQAM2	
SC.EC4	020312-R	# ZRQAM1	ZRQAM2	
SC.EC5	020342-R	# ZRQAM1	ZRQAM2	
SC.EC6	020372-R	# ZRQAM1	ZRQAM2	
SC.EC7	020422-R	# ZRQAM1	ZRQAM2	
SC.EC8	020454-R	# ZRQAM1	ZRQAM2	
SC.EC9	020506-R	# ZRQAM1	ZRQAM2	
SC.EDC	021142-R	# ZRQAM1	ZRQAM2	
SC.FCT	020132-R	# ZRQAM1	ZRQAM2	
SC.FER	017202-R	# ZRQAM1	ZRQAM2	
SC.FE2	017270-R	# ZRQAM1	ZRQAM2	

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
SC.FUL	020002-R	# ZRQAM1	ZRQAM2	
SC.HWP	020610-R	# ZRQAM1	ZRQAM2	
SC.IDS	021202-R	# ZRQAM1	ZRQAM2	
SC.IDP	017036-R	# ZRQAM1	ZRQAM2	
SC.ISH	017360-R	# ZRQAM1	ZRQAM2	
SC.IS2	017440-R	# ZRQAM1	ZRQAM2	
SC.NXM	020722-R	# ZRQAM1	ZRQAM2	
SC.ODA	020650-R	# ZRQAM1	ZRQAM2	
SC.ODB	020700-R	# ZRQAM1	ZRQAM2	
SC.ONL	016634-R	# ZRQAM1	ZRQAM2	
SC.PAR	020756-R	# ZRQAM1	ZRQAM2	
SC.PDE	021432-R	# ZRQAM1	ZRQAM2	
SC.PSP	021674-R	# ZRQAM1	ZRQAM2	
SC.RCT	017762-R	# ZRQAM1	ZRQAM2	
SC.RDY	021466-R	# ZRQAM1	ZRQAM2	
SC.RSP	021576-R	# ZRQAM1	ZRQAM2	
SC.SDI	016536-R	# ZRQAM1	ZRQAM2	
SC.SDS	021064-R	# ZRQAM1	ZRQAM2	
SC.SON	016656-R	# ZRQAM1	ZRQAM2	
SC.SRI	021344-R	# ZRQAM1	ZRQAM2	
SC.SRT	021252-R	# ZRQAM1	ZRQAM2	
SC.SUR	021644-R	# ZRQAM1	ZRQAM2	
SC.SWP	020550-R	# ZRQAM1	ZRQAM2	
SC.UNK	016676-R	# ZRQAM1	ZRQAM2	
SC.VOL	016756-R	# ZRQAM1	ZRQAM2	
SC.576	020056-R	# ZRQAM1	ZRQAM2	
SEND	034132-R	# ZRQAM2	ZRQAM3	
SEQUEN	070724-R	# ZRQAM3		
SET.CP	032376-R	# ZRQAM2	ZRQAM3	
SET.CT	046146-R	# ZRQAM3		
SET.UP	032472-R	# ZRQAM2	ZRQAM3	
SFPTBL	024614-R	# ZRQAM1		
SFT.MS	024404-R	# ZRQAM1	ZRQAM3	
SOFT.E	073662-R	# ZRQAM3		
SPACE4	024536-R	# ZRQAM1	ZRQAM2	
STEP	105220-R	# ZRQAM1	ZRQAM2	ZRQAM3
ST.COD	105214-R	# ZRQAM1	ZRQAM2	ZRQAM3
SWEEP	066650-R	# ZRQAM3		
SWM1	004522-R	# ZRQAM1	ZRQAM2	
SWP.DP	024622-R	# ZRQAM1	ZRQAM3	
SWP.ER	024614-R	# ZRQAM1	ZRQAM3	
SWP.FL	024620-R	# ZRQAM1	ZRQAM2	ZRQAM3
SWP.RA	024624-R	# ZRQAM1	ZRQAM3	
SWP.TI	024626-R	# ZRQAM1	ZRQAM3	
SWP.UC	024632-R	# ZRQAM1	ZRQAM3	
SWP.UD	024634-R	# ZRQAM1	ZRQAM3	
SWP.XF	024616-R	# ZRQAM1	ZRQAM3	
SWQ1	003022-R	# ZRQAM1	ZRQAM2	
SWQ10	003274-R	# ZRQAM1	ZRQAM2	
SWQ11	003340-R	# ZRQAM1	ZRQAM2	
SWQ12	003372-R	# ZRQAM1	ZRQAM2	
SWQ13	003470-R	# ZRQAM1	ZRQAM2	

```
$       ex t                                                                    SEQ 0503
ZRQAF0     CREATED BY  TKB     ON 27 DEC 84 AT 13:19     PAGE 11

GLOBAL CROSS REFERENCE                                          CREF   V02

SYMBOL  VALUE       REFERENCES...

SWQ14   003546-R  # ZRQAM1     ZRQAM2
SWQ15   003620-R  # ZRQAM1     ZRQAM2
SWQ17   003670-R  # ZRQAM1     ZRQAM2
SWQ19   003766-R  # ZRQAM1     ZRQAM2
SWQ2    003044-R  # ZRQAM1     ZRQAM2
SWQ20   004056-R  # ZRQAM1     ZRQAM2
SWQ21   004144-R  # ZRQAM1     ZRQAM2
SWQ22   004230-R  # ZRQAM1     ZRQAM2
SWQ23   004270-R  # ZRQAM1     ZRQAM2
SWQ24   004342-R  # ZRQAM1     ZRQAM2
SWQ25   004406-R  # ZRQAM1     ZRQAM2
SWQ26   004460-R  # ZRQAM1     ZRQAM2
SWQ4    003124-R  # ZRQAM1     ZRQAM2
SWQ7    003146-R  # ZRQAM1     ZRQAM2
SWQ9    003220-R  # ZRQAM1     ZRQAM2
S.DUPP  105256-R  # ZRQAM1     ZRQAM2     ZRQAM3
S.PATT  105254-R  # ZRQAM1     ZRQAM2     ZRQAM3
TALLY   077462-R  # ZRQAM1     ZRQAM2     ZRQAM3
TEMP1   105236-R  # ZRQAM1     ZRQAM2     ZRQAM3
TEMP2   105240-R  # ZRQAM1     ZRQAM2     ZRQAM3
TIME    032314-R  # ZRQAM2     ZRQAM3
TRK.SG  101016-R  # ZRQAM1     ZRQAM2     ZRQAM3
T$FREE  107342-R  # ZRQAM4
T$PTHV  000004      ZRQAM1   # ZRQAM4
T.ADDR  100012-R  # ZRQAM1     ZRQAM2     ZRQAM3
T1      043024-R    ZRQAM1   # ZRQAM3
UAM     000200    # ZRQAM1   # ZRQAM2   # ZRQAM3
UPD.IO  065424-R  # ZRQAM3
VEC.BR  044370-R  # ZRQAM3
WAIT    034570-R  # ZRQAM2     ZRQAM3
XX13    016204-R  # ZRQAM1     ZRQAM2
XX23    016226-R  # ZRQAM1     ZRQAM2
XX32    016262-R  # ZRQAM1     ZRQAM2
XX33    016310-R  # ZRQAM1     ZRQAM2
XX34    016346-R  # ZRQAM1     ZRQAM2
$END.L  107344-R  # ZRQAM4
$SAVE2  077154-R    B16MUL   # B16SAV     ZRQAM2     ZRQAM3
$SAVE3  077170-R  # B16SAV     ZRQAM2     ZRQAM3
$SAVE4  077206-R  # B16SAV     ZRQAM2     ZRQAM3
$SAVE5  077226-R    B16MUL   # B16SAV     ZRQAM2     ZRQAM3
$       VERIFY_CODE = f$verify(0)
$       on control_y then                   $ goto END
$       set verify
$       inquire ANSWER "*       Compile ZRQAF0 library? [Y/N]"
$       if ANSWER .nes. "Y" then            $ goto L1
$       bliss/pdp11 ZRQAF0.REQ/list=ZRQAF0.LIS/library=ZRQAF0.L1o/source=page:53
$L1:    inquire ANSWER "*       Compile Module 1? [Y/N]"
$       if ANSWER .nes. "Y" then            $ goto L2
$       bliss/pdp11 ZRQAF0.BL1/list=ZRQAF0.LS1/object=ZRQAF0.O81/source=page:53
$L2:    inquire ANSWER "*       Compile Module 2? [Y/N]"
$       if ANSWER .nes. "Y" then            $ goto L3
$       bliss/pdp11 ZRQAF0.BL2/list=ZRQAF0.LS2/object=ZRQAF0.O82/source=page:53
$L3:    inquire ANSWER "*       Task-Build? [Y/N]"
$       if ANSWER .nes. "Y" then            $ goto L4
$       mcr tkb
ZRQAF0.EXE/nohd/nomm,ZRQAF0/CR/-sp=ZRQAF0.O81,ZRQAF0.O82,NOEIS.OLB/lb
/
par=DUMMY:2000:176000
stack=0
//
$!
$L4
```

```
$!
$L4:
$!      TKBBIN
$!      INPUT FILENAME?    ZRQAF0.EXE
$!
$!      ZRQAF0.EXE -> ZRQAF0.BIN
$       inquire ANSWER "$       Purge Directory? [Y/N]'
$       if ANSWER .nes. "Y" then        $ goto END
$       pur
$END:   set noverify
        if VERIFY_CODE then             $ set verify
```