RQDX1/2/3 | RQDX/RUX50 Exerciser
CZRQAH0

AH-T399H-MC
1 of 3 Jan. 1986

RQDX|RUX50 Exerciser AH-T399 H-MC
RQDX1/2/3 CZRQAHO 2 of 3 Jan 1986

RQDX 1/2/3 RQDX/Rux50 Exerciser AH-T399H-MC

CZRQAHO 3 of 3 Jan. 1986

```
module ZRQAM1 (

%title 'RD/RX EXERCISER'
                ident = 'V02.3',
                addressing_mode (absolute),
                environment (noeis)
                ) =

begin

%(
                                IDENTIFICATION
                                --------------

        PRODUCT CODE:           AC-T398H-MC

        PRODUCT NAME:           CZRQAHO RQDX/RUX50 EXERCISER

        PRODUCT DATE:           10-OCT-85

        MAINTAINER:             DIAGNOSTIC ENGINEERING

        AUTHOR:                 RAVINDER K. KARWAN
                                BOB POWERS


        Copyright (C) 1983, 1985

        Digital Equipment Corporation, Maynard, Massachusetts  01754

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

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

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


        The following are trademarks of Digital Equipment Corporation:

        DIGITAL         PDP             UNIBUS          MASSBUS
        DEC             DECUS           DECTAPE
```

```
!*****************************************************************************
!                                                                            *
!                          L I T E R A L S                                   *
!                                                                            *
!*****************************************************************************
!
LITERAL
!
!****** ODT TRAP VECTOR LOCATION
!
        O_TVEC           = %o'14',
!
!****** HARDWARE ADDRESSES ETC.
!
        INIT_INTR_VECT   = %o'154',                         ! VECTOR ADDRESS
        INIT_IP_ADDR     = %o'172150',                      ! IP REGISTER ADDRESS
        INIT_BR_LEVEL    = %o'4',                           ! BUS REQUEST LEVEL
!
        LINE_CLOCK       = %o'177546',                      ! LINE-CLOCK ADDRESS
!
!****** HARDWARE LIMITS
!
        MAX_CTLR         = 1,                               ! MAXIMUM NUMBER OF LCP CONTROLLERS ALLOWED
        UNITS_PER_CNTR   = 4,                               ! MAXIMUM UNITS PER CONTROLLER
        MAX_UNITS        = MAX_CTLR * UNITS_PER_CNTR,       ! MAXIMUM NUMBER OF UNITS TO TEST

!                                                           !HIGHEST RD51 LBN = 52137 OCT
        RD51_MAX_TRACK   = 1200,                            ! MAXIMUM NUMBER OF TRACKS FOR RD51
        RD51_SEC_PER_TRK= 18,                               ! NUMBER OF SECTORS PER TRACK FOR RD51
        RD51_MAX_LBN     = RD51_MAX_TRACK * RD51_SEC_PER_TRK - 1,! MAX LBN FOR RD51

        RD52_MAX_TRACK   = 2976,                            ! MAXIMUM NUMBER OF TRACKS FOR RD52
        RD52_SEC_PER_TRK= 18,                               ! NUMBER OF SECTORS PER TRACK FOR RD52
        RD52_MAX_LBN     = RD52_MAX_TRACK * RD52_SEC_PER_TRK - 1,! MAX LBN FOR RD52

        RX50_MAX_TRACK   = 80,                              ! MAXIMUM NUMBER OF TRACKS FOR RX50
        RX50_SEC_PER_TRK= 10,                               ! NUMBER OF SECTORS PER TRACK FOR RX50
        RX50_MAX_LBN     = RX50_MAX_TRACK * RX50_SEC_PER_TRK - 1,! MAX LBN FOR RX50

!                                                           !HIGHEST RD53 LBN = 416660 OCT (2/016660)
!                                                           !RD51  SEAGATE 4 HEADS
!                                                           !RD52A QUANTUM 8 HEADS 98%
!                                                           !RD52B ATASI   7 HEADS
!                                                           !RD53  MICROPOLIS 8 HEADS
!                                                           !RQDX2 18 BKS/TK
!                                                           !RQDX2 17 BKS/TK

        BYTES_PER_SECT   = 512,                             ! BYTES/SECTOR (AT PRESENT SAME FOR RDs AND RXs)
!       MAX_XFER_SIZE    = 2 * BYTES_PER_SECT,              ! ARTBITRARY MAX SIZE OF EACH DISK I/O
        MAX_XFER_SIZE    = BYTES_PER_SECT * 3 / 2,
!
! NOTE - BOTH OF THESE NUMBERS ARE NOW ARBITRARILY CHOSEN AS THE NUMBER OF LBNS CONTAINED PER UNIT/10 .
```

```
;      0      !
;      0      !****** RING SIZES
;      0      !
;      0              CR_LOG          = 2,                                      ! LOG2 LENGTH OF COMMAND RING
;      0              RR_LOG          = 2,                                      ! LOG2 LENGTH OF RESPONSE RING
;      0              CRING_LEN       = 1 ↑ CR_LOG,                             ! COMMAND RING LENGTH
;      0              RRING_LEN       = 1 ↑ RR_LOG,                             ! RESPONSE RING LENGTH
;      0      !
;      0      !****** OFFSETS (IN WORDS)
;      0      !
;      0              OF_UN           = 3,                                      ! OFFSET FROM START OF CST TO FIRST UNIT
;      0              OF_DATA         = 0,                                      ! OFFSET TO DISK UNIT FLAGS   WITHIN UNIT'S CST
;      0              OF_BEG          = 1,                                      ! OFFSET TO BEGINNING BLK NO. WITHIN UNIT'S CST
;      0              OF_BEG1         = 2,       !OFFSET TO START BK HI         ZZZ
;      0              OF_END          = 3,       !OFFSET TO END BLOCK LO        ZZZ
;      0              OF_END1         = 4,       !OFFSET TO END BK HI           ZZZ
;      0              OF_NAME_0       = 5,       !OFFSET TO 1st 2 CHARS OF NAME ZZZ
;      0              OF_NAME_2       = 6,       !OFFSET TO 2nd 2 CHARS OF NAME ZZZ
;      0              OF_DUPFLAGS     = 8,       !OFFSET TO DUP FLAGS           ZZZ
;      0              OF_COUNT        = 9,       !OFFSET TO MSCP FUNCTION COUNTER ZZZ
;      0              OF_DBN          = 8,       !OFFSET TO RELATIVE DBN        ZZZ
;      0      !
;      0      !
;      0      !****** TABLE AND OTHER STRUCTURE SIZES
;      0      !
;      0              LBNADR_LEN      = 2,                                      !MAX_LBN'S ARE 2 WD ADDRESSES
 ZZZ
;      0              HWPT_LEN        = 8,                                      ! SIZE (WORDS) OF HW P-TABLE
 ZZZ
;      0              COMM_LEN        = (RRING_LEN * 2) + (CRING_LEN * 2) + 4,! SIZE (WORDS) OF COMMUNICATION AREA PER CONTROLLER
;      0              UNIT_SIZE       = 10,                                     ! SIZE (WORDS) OF CST UNIT ENTRY
 ZZZ
;      0              CST_LEN         = UNITS_PER_CNTR * UNIT_SIZE + OF_UN,     ! SIZE (WORDS) OF A CONTROLLER STATUS TABLE
;      0              TALLY_CLEAR     = 7,                                      ! SIZE (WORDS) OF STATISTICS TBL CLEARED EVERY PASS
;      0              TALLY_TOTALS    = 20,                                     ! SIZE (WORDS) OF STATISTICS TABLE FOR TOTALS
 ZZZ
;      0              TALLY_LEN       = TALLY_CLEAR + TALLY_TOTALS,             ! SIZE (WORDS) OF A STATISTICS TABLE
;      0              C_ERR_LEN       = 1,                                      ! SIZE (WORDS) OF CONTROLLER ERROR TABLE
;      0              RP_LEN          = 22,                                     ! SIZE (WORDS) OF A RETURN PACKET
;      0              MSG_LEN         = 30,                                     ! SIZE (WORDS) OF AN MSCP MESSAGE (TEXT PORTION)
;      0              PKT_LEN         = MSG_LEN + 5,                            ! SIZE (WORDS) OF AN MSCP PACKET
;      0              DCT_LEN         = 9,                                      ! SIZE (WORDS) OF A DRIVER CONTROLLER TABLE
;      0              RDM_LEN         = 16,                                     ! SIZE (WORDS) OF THE RANDOM NUMBER TABLE
;      0              MAX_UDP_CNT     = 16,                                     ! MAX SIZE OF USER DATA PATTERN
;      0              MAX_BUF_CNT     = (CRING_LEN * 2) * MAX_CTLR,             ! MAX NO. OF I/O BUFFERS (BUFF_ADDR & BUFF_OWN)
;      0              PKT_CNT         = ((CRING_LEN * 2) + RRING_LEN) * MAX_CTLR,
;      0                                                                        ! NO. OF MSCP PACKETS IN POOL
;      0              RP_CNT          = PKT_CNT - (RRING_LEN * MAX_CTLR),       ! NO. OF RETURN PACKETS IN POOL
;      0              IODQ_LEN        = RP_CNT,                                 ! NO. OF ENTRIES IN I/O DONE QUEUE (IODQ)
;      0              OUTC_CNT        = CRING_LEN * 2,                          ! NO. OF ENTRIES/CONTROLLER'S OUTSTANDING CMD LIST
;      0              DP_CNT          = 21,                                     ! NO. OF PRE-DEFINED DATA PATTERNS
;      0              EP_CNT          = MAX_CTLR * RRING_LEN * 3,               ! NO. OF ERROR-LOG PACKET SAVE BUFFERS
;      0              EP_LEN          = PKT_LEN - 3 + 1,                        ! LENGTH OF EACH ERROR-LOG SAVE BUFFER
;      0              LAST_PKT_LEN    = 3,                                      ! BUFFER LENGTH TO SAVE INFO. ABOUT LAST RESPONSE
;      0              DESC_SIZ        = 4,                                      !NO. OF BYTES IN A PACKET DESCRIPTOR          ZZZ
;      0      !
;      0      !****** SW P-TABLE FLAGS (SWP_FLAGS)
```

```
!
!ZZZ    SWF_TRC         = %o'000001',                   ! DIAGNOSTIC TRACE
        SWF_APT         = %o'000001',                   !RUNNING UNDER A.P.T. MONITOR             ZZZ
        SWF_RDM         = %o'000002',                   ! RANDOM SEEK MODE
        SWF_CRC         = %o'000004',                   ! READ-COMPARE AT CONTROLLER
!       SWF_DCC         = %o'000010',                   ! DRIVE COMPLEMENT COMPLETE
        SWF_CWC         = %o'000020',                   ! WRITE-COMPARE AT CONTROLLER
        SWF_HWC         = %o'000040',                   ! WRITE-COMPARE AT HOST
        SWF_UDP         = %o'000100',                   ! USER-DEFINED DATA PATTERN
        SWF_CST         = %o'000200',                   ! CLEAR STATISTICAL TABLES
!       SWF_DIA         = %o'000400',                   ! DIAGNOSTIC PACKAGE, WHEN THIS IS SELECTED
                                                        ! ALL INTERRUPTS ARE WAITED FOR, E.G. ONLY
                                                        ! ONE MSCP PACKET IS OUTSTANDING AT A TIME
        SWF_SEQ         = %o'001000',                   ! RANDOM OR FIXED SEQUENTIAL STEPPING
!       SWF_DUP         = %o'002000',                   ! RUN DUP DIAGNOSTIC
        SWF_FER         = %o'004000',                   ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT DETECTED
        SWF_HRD         = %o'010000',                   ! HALT ON HARD ERRORS ALSO WITH 'HOE' DRS FLAG?
        SWF_SFT         = %o'020000',                   ! HALT ON SOFT ERRORS ALSO WITH 'HOE' DRS FLAG?
        SWF_BLK         = %o'040000',                   ! HALT ON BAD-BLOCK ERRORS ALSO WITH 'HOE' DRS FLAG?
        SWF_TRY         = %o'100000',                   ! COUNT EACH RETRY AS ANOTHER EXTRA SOFT-ERROR
!
!****** FLAGS FOR DUP EXERCISER (DUP_FLAGS)                       ZZZ
!                                                                 ZZZ
        SWP_DINT        = %o'2',         !DUP CAUSED INIT         ZZZ

!
!****** ENTRY_REASON VALUES
!               (HOW PROGRAM WAS INVOKED)
!
        START           = 1,                            ! START
        RESTART         = 2,                            ! RESTART
        CONT            = 3,                            ! CONTINUE
        PWR_FAIL        = 4,                            ! POWER FAIL
        NEW_PASS        = 5,                            ! NEW PASS
!
!****** DROP UNIT REASONS
!               (LOADED INTO DUR VECTOR)
!
        DU_USER         = 0,                            ! USER COMMAND
        DU_CONF         = 1,                            ! CONFIGURATION ERROR
        DU_INIT         = 2,                            ! INITIALIZATION ERROR
        DU_XFER         = 3,                            ! TRANSFER LIMIT REACHED
        DU_HERR         = 4,                            ! HARD ERROR LIMIT REACHED
        DU_DFATAL       = 5,                            ! UNRECOVERABLE DEVICE ERROR
        DU_CFATAL       = 6,                            ! UNRECOVERABLE CONTROLLER ERROR
        DU_ONLINE       = 7,                            ! ONLINE FAILED
        DU_ACCESS       = 8,                            ! ACCESS TO LAST TRACK FAILED
        DU_PROTECT      = 9,                            ! WRITE PROTECT CONFLICT
        DU_TIME         = 10,                           ! COMMAND TIME OUT
!
!****** MISCELLANEOUS LITERALS
!
        MAX_DBN         = 63,                           !HIGHEST RELATIVE DBN NUMBER              ZZ
```

```
;  0          INI_ATT          = 2,                                        ! NO. OF HW INIT ATTEMPTS BEFORE FAILURE IS ASSUMED
;  0          WR_RING          = ((%o'200') or (CR_LOG ↑ 3) or (RR_LOG)),
;  0                                                                       ! WR-BIT-AND-RING-LENGTH (STEP 1 WRITE/STEP 2 READ)
;  0          QIO_PER_CTLR     = CRING_LEN * 2,                            ! MAXIMUM NUMBER OF OUTSTANDING QIOS PER CONTROLLER
;  0    !     MAX_XFER         = 256,                                      ! MAXIMUM SIZE (WORDS) OF AN I/O TRANSFER
;  0          REMOVABLE_BIT    = %o'0',                                    ! BIT IN HARDWARE TABLES MARKING A REMOVABLE DISK
;  0          FIXED_BIT        = %o'20',                                   ! BIT IN HARDWARE TABLES MARKING A FIXED     DISK
;  0          REMOVABLE        = 0,                                        ! NUMBER FOR REMOVABLE DISK WHEN SHIFTED RIGHT
;  0          FIXED            = 1,                                        ! NUMBER FOR FIXED     DISK WHEN SHIFTED RIGHT
;  0          RX_50            = 0,             !D_TYPE FLAG = 0 FOR RX50  (THESE FLAGS AREN'T USED. INSTEAD,)   ZZZ
;  0          RD_51            = 1,             !D_TYPE FLAG = 1 FOR RD51  (D_TYPE = 1 FOR FIXED, 0 FOR REMOV)   ZZZ
;  0          RD_52            = 2,             !              2 FOR RD52                                        ZZZ
;  0
;  0    !
;  0    !****** MSCP PACKET DESCRIPTOR
;  0    !
;  0          ED_OWN           = %o'100000',                               ! OWNERSHIP BIT
;  0          ED_FLAG          = %o'040000',                               ! FLAG BIT
;  0    !
;  0    !****** MSCP COMMAND PACKET OPCODES
;  0    !
;  0          OP_MSK           = %o'177',                                  ! OPCODE MASK
;  0          OP_END           = %o'200',                                  ! ENDCODE DESIGNATOR
;  0          OP_ACC           = %o'20',                                   ! ACCESS COMMAND
;  0          OP_ONL           = %o'11',                                   ! ONLINE COMMAND
;  0          OP_RD            = %o'41',                                   ! READ COMMAND
;  0          OP_SCC           = %o'4',                                    ! SET CONTROLLER CHARACTERISTICS COMMAND
;  0          OP_WRT           = %o'42',                                   ! WRITE COMMAND
;  0          OP_GDS           = %o'1',         !get dust status         ZZZ
;  0          OP_ESP           = %o'2',         !execute supplied prog   ZZZ
;  0          OP_ELP           = %o'3',         !execute local program   ZZZ
;  0          OP_SDD           = %o'4',         !send data               ZZZ
;  0          OP_RCD           = %o'5',         !receive data            ZZZ
;  0          OP_ABT           = %o'6',         !abort program           ZZZ
;  0
;  0    !
;  0    !****** PACKET SIZES
;  0    !
;  0          SZ_ACC = %decimal '32',                                      ! ACCESS
;  0          SZ_ONL = %decimal '36',                                      ! ON LINE COMMAND
;  0          SZ_RD  = %decimal '32',                                      ! READ
;  0          SZ_SCC = %decimal '32',                                      ! SET CONTROLLER CHARACTERISTICS
;  0          SZ_WRT = %decimal '32',                                      ! WRITE
;  0          SZ_GEN = %decimal '32',                                      ! GENERAL PACKET SIZE
;  0          SZ_REC = %DECIMAL '28',               !                        ZZZ
;  0          SZ_SEN = %DECIMAL '28',               !                        ZZZ
;  0          SZ_ELP = %DECIMAL '18',               !                        ZZZ
;  0          SZ_ABT = %DECIMAL '12',               !                        ZZZ
;  0          SZ_GDS = %DECIMAL '12',               !                        ZZZ
;  0    !
;  0    !****** MSCP COMMAND MODIFIERS
;  0    !
;  0          MD_CMP           = %o'040000',                               ! COMPARE
```

```
        MD_EXP          = %o'100000',                   ! EXPRESS REQUEST
!
!****** CONNECTION ID VALUES (MSCP_PKT, RETPKT)
!               (SERVE AS SOURCES AND DESTINATIONS OF MSCP MESSAGES)
!
        CID_DISK        = 0,                            ! DISK MSCP
        CID_MSCP        = 0,                            !DISK MSCP
!ZZZ
        CID_TAPE        = 1,                            ! TAPE MSCP
        CID_DUP         = 2,                            ! DIAGNOSTIC AND UTILITIES PROTOCOL
        CID_DRIVER      = 3,                            ! EXERCISER "DRIVER"
!
!****** MESSAGE TYPE VALUES
!
        MT_SEQ          = 0,                            ! SEQUENTIAL (FROM PORT)
        MT_DG           = 1,                            ! DATAGRAM (FROM PORT)
        MT_CRD          = 2,                            ! CREDIT NOTIFICATION (FROM PORT)
        MT_FATAL        = 3,                            ! FATAL DEVICE ERROR (FROM "DRIVER")
        MT_TIMEOUT      = 4,                            ! COMMAND TIMEOUT (FROM "DRIVER")
!
!****** CONTROLLER FLAGS
!               (IN SET CONTROLLER CHARACTERISTICS COMMAND AND RESPONSE)
!
        CF_ATN          = %o'000200',                   ! ENABLE ATTENTION MESSAGES
        CF_MSC          = %o'000100',                   ! ENABLE MISCELLANEOUS ERROR LOG MESSAGES
        CF_OTH          = %o'000040',                   ! ENABLE OTHER HOST'S ERROR LOG MESSAGES
        CF_THS          = %o'000020',                   ! ENABLE THIS HOST'S ERROR LOG MESSAGES
!       CF_MASK         = CF_ATN or CF_MSC or CF_THS,
        CF_MASK         = CF_MSC or CF_THS,             ! RELEVENT BITS IN CTLR FLAGS WORD
!
!****** UNIT FLAGS
!               (IN ONLINE COMMAND AND RESPONSE)
!
        UF_REMOVABLE    = %o'000200',                   ! REMOVABLE MEDIA
        UF_WPH          = %o'020000',                   ! WRITE PROTECT (HARDWARE)
!
!****** STATUS / EVENT CODE DEFINITIONS
!
        ST_SUC          = %o'0',                        ! SUCCESS
        ST_CMD          = %o'1',                        ! INVALID COMMAND
        ST_ABO          = %o'2',                        ! COMMAND ABORTED
        ST_OFL          = %o'3',                        ! UNIT OFFLINE
        ST_AVL          = %o'4',                        ! DRIVE AVAILABLE
        ST_MFE          = %o'5',                        ! MEDIA FORMAT ERROR
        ST_WPT          = %o'6',                        ! WRITE PROTECTED
        ST_CMP          = %o'7',                        ! COMPARE ERROR
        ST_DAT          = %o'10',                       ! DATA ERROR
        ST_HST          = %o'11',                       ! HOST BUFFER ACCESS ERROR
        ST_CNT          = %o'12',                       ! CONTROLLER ERROR
        ST_DRV          = %o'13',                       ! DRIVE ERROR
        ST_BRC          = %o'24',                       ! BAD BLOCK REPLACEMENT COMPLETION MMM
        ST_DIA          = %o'37',                       ! MESSAGE FROM INTERNAL DIAGNOSTICS
!
!****** END MESSAGE FLAGS
```

```
!
        EF_BBR          = %o'200',                               ! BAD BLOCK REPORTED
        EF_BBU          = %o'100',                               ! BAD BLOCK NOT REPORTED
!
!****** RDRX LITERALS
!
        RCIP            = 0,                                     ! IP REGISTER
        RCSA            = 1,                                     ! SA REGISTER
!
!****** COMMON SA REGISTER BIT DEFINITIONS
!
        SA_S1           = %o'004000',                            ! STEP 1 STATUS BIT
        SA_S2           = %o'010000',                            !   :  2
        SA_S3           = %o'020000',                            !   :  3
        SA_S4           = %o'040000',                            !   V  4
        SA_ERR          = %o'100000',                            ! ERROR INDICATOR
        SA_INT          = %o'000200',                            ! INTERRUPT ENABLE DURING INITIALIZATION
        SA_GO           = %o'000001',                            ! GO BIT TO START FIRMWARE
!
!****** INITIALIZATION STEP READ MASKS
!
        S1_MASK         = %o'176000',                            ! STEP 1 READ BITS
        S2_MASK         = %o'174377',                            !   :  2
        S3_MASK         = %o'174377',                            !   :  3
        S4_MASK         = %o'174000',                            !   V  4
!
!****** COMMAND TYPES
!
        IMM_CMD         = 0,                                     ! IMMEDIATE COMMAND
        SEQ_CMD         = 1,                                     ! SEQUENTIAL COMMAND
        NON_SEQ_CMD     = 2,                                     ! NON-SEQUENTIAL COMMAND
!
!****** ERROR-LOG FORMAT TYPES
!
        FORMAT_CNTR     = %o'0',                                 ! CONTROLLER ERROR
        FORMAT_HOST     = %o'1',                                 ! HOST MEMORY ACCESS ERROR
        FORMAT_XFER     = %o'2',                                 ! DISK TRANSFER ERROR
        FORMAT_SDI      = %o'3',                                 ! 'STANDARD DISK INTECONNECT' ERROR
        FORMAT_SDE      = %o'4',                                 ! SMALL DISK ERROR
        FORMAT_BRA      = %o'11',                                ! BAD BLOCK REPLACEMENT ATTEMPT MMM
!
!****** ERROR-LOG BLOCK NUMBER INFORMATION
!
        TYPE_LBN        = %o'0000',                              ! LOGICAL BLOCK NUMBER
        TYPE_RBN        = %o'0110',                              ! REPLACEMENT BLOCK NUMBER
!
!****** MSCP DISK MODEL CODES
!
!       MODEL_RX50      = 7,                                     ! RX50  THESE ARE NO LONGER USED. THE
!       MODEL_RD51      = 6,                                     ! RD51  MODEL IS DETERMINED ANOTHER WAY.
!       MODEL_RD52      = 8,                                     ! RD52
!
!****** LITERALS FOR READABILITY
```

```
;        0      !
;        0                  YES              = 1,
;        0                  NO               = 0,
;        0                  TRUE             = 1,
;        0                  FALSE            = 0,
;        0                  SUCCESS          = 1,
;        0                  FAILURE          = 0,
;        0                  FOUND            = 1,
;        0                  NOT_FOUND        = 0,
;        0                  PRESENT          = 1,                  ! DISK IS PRESENT IN CONTROLLER
;        0                  NCT_PRESENT      = 0,                  ! DISK IS NOT PRESENT IN CONTROLLER
;        0                  UNPROTECTED      = 1,                  ! DISK HAS UNPROTECTED CUSTOMER LBN'S
;        0                  PROTECTED        = 0,                  ! DISK HAS PROTECTED CUSTOMER LBN'S
;        0                  ONLINE           = 1,
;        0                  OFFLINE          = 0,
;        0                  IDLE             = 0,                  !IDLE
!ZZZ
;        0                  ACTIVE           = 1,                  !ACTIVE
!ZZZ
;        0                  FULL             = 1,                  ! ERROR-LOG SAVE PACKET FILLED
;        0                  EMPTY            = 0,                  ! ERROR-LOG SAVE PACKET PRINTED
;        0                  HRD_OCCURED      = 1,                  ! HARD ERROR DETECTED IN RESPONSE PACKET
;        0                  HRD_NOT_OCCURED  = 0,                  ! HARD ERROR NOT DETECTED
;        0                  ALL_ONES         = %o'177777';
```

```
!*****************************************************************************
!                                                                            *
!                        F I E L D S                                         *
!                                                                            *
!*****************************************************************************
!
FIELD
!
!****** HARDWARE P-TABLE FIELDS
!
HWP_FIELDS =
        set
        HWP_IP_ADDR     = [0, 0, 16, 0],                ! IP ADDRESS
        HWP_VECTOR      = [1, 0, 16, 0],                ! VECTOR ADDRESS
        HWP_BR_LEVEL    = [2, 0, 16, 0],                ! BUS REQUEST LEVEL
        HWP_DISK        = [3, 0, 16, 0],                ! DISK (ALL FIELDS)
        HWP_DISK_NUM    = [3, 0, 4, 0],                 ! DISK NUMBER
        HWP_DISK_TYPE   = [3, 4, 1, 0],                 ! DISK TYPE
        HWP_DISK_DUPEX  = [3, 5, 1, 0],                 !RUN DUP EXERCISER          !ZZZ
        HWP_DISK_DUPWT  = [3, 6, 1, 0],                 !DUP WRITE FLAG             !ZZZ
        HWP_ENTIRE      = [3, 7, 1, 0],                 !TEST ENTIRE DISK           !ZZZ
        HWP_DISK_CP     = [3, 15, 1, 0],                ! PROTECT CUSTOMER DATA BIT
        HWP_BEG_TRK     = [4, 0, 16, 0],                ! BEGINNING TRACK LO        !ZZZ
        HWP_BEG_TRK1    = [5, 0, 16, 0],                ! BEGINNING TRACK HI        !ZZZ
        HWP_END_TRK     = [6, 0, 16, 0],                ! ENDING TRACK LO           !ZZZ
        HWP_END_TRK1    = [7, 0, 16, 0]                 ! ENDING TRACK HI           !ZZZ
        tes,
!
!****** COMMUNICATION AREA HEADER FIELDS
!
COM_FIELDS =
        set
        ADAP_CH         = [1, 8, 8, 0],                 ! ADAPTER CHANNEL NUMBER FOR PURGES
        CMD_INT         = [2, 0, 16, 0],                ! COMMAND RING INTERRUPT
        RSP_INT         = [3, 0, 16, 0]                 ! RESPONSE RING INTERRUPT
        tes,

!                                                       ZZZ
!       DUP BUFFER FIELD                                ZZZ
!                                                       ZZZ
DP_FIELDS =                                             !ZZZ
        SET                                             !ZZZ
        DUPBF0          = [0, 0, 16, 0],                !ZZZ
        DUPBF1          = [1, 0, 16, 0],                !ZZZ
        DUPBF2          = [2, 0, 16, 0],                !ZZZ
        DUPTYPE         = [0, 12, 4, 0],                !ZZZ
        DUPMESG         = [0, 0, 12, 0]                 !ZZZ
        TεS,                                            !ZZZ


!****** CONTROLLER STATUS TABLE (CST) FIELDS
!
CST_FIELDS =
```

```
0               set
0               IP_ADDR         = [0, 0, 16, 0],                        ! IP ADDRESS
0               VEC_ADDR        = [1, 0, 9, 0],                         ! VECTOR ADDRESS
0               STATE           = [1, 15, 1, 0],                        ! CONTROLLER STATUS
0               BR_LEV          = [2, 0, 8, 0],                         ! BUS REQUEST LEVEL
0               U_CNT           = [2, 8, 8, 0]                          ! NUMBER OF UNITS (DISKS) FOR THIS CONTROLLER
0       !
0       !       DO_ALL          = [3, 0, 16, 0],                        ! DISK 0 (ALL FIELDS)
0       !       DO_DISK_NUM     = [3, 0, 4, 0],                         ! DISK NUMBER
0       !       DO_TYPE         = [3, 4, 1, 0],                         ! DISK TYPE
0       !       DO_UNIT         = [3, 8, 4, 0],                         ! DISK 0 UNIT NUMBER (DRS UNIT)
0       !       DO_FATAL        = [3, 12, 1, 0],                        ! DISK 0 FATAL ERROR BIT
0       !       DO_STAT         = [3, 13, 1, 0],                        ! DISK 0 STATUS BIT
0       !       DO_PRES         = [3, 14, 1, 0],                        ! DISK 0 PRESENT BIT
0       !       DO_PROT         = [3, 15, 1, 0],                        ! DK 0 PROTECT CUSTOMER DATA
0       !       DO_BEG0         = [4, 0, 16, 0],        !DK 0 BEGIN TK LO       ZZZ
0       !       DO_BEG1         = [5, 0, 16, 0],        !DK 0 BEGIN TK HI       ZZZ
0       !       DO_END0         = [6, 0, 16, 0],        !DK 0 END TK LO         ZZZ
0       !       DO_END1         = [7, 0, 16, 0],        !DK 0 END TK HI         ZZZ
0       !       DO_NAME0        = [8, 0, 8, 0],         !DK 0 MANE BYTE 0       ZZZ
0       !       DO_NAME1        = [8, 8, 8, 0],         !DK 0 NAME BYTE 1       ZZZ
0       !       DO_NAME2        = [9, 0, 8, 0],         !DK 0 NAME BYTE 2       ZZZ
0       !       DO_NAME3        = [9, 8, 8, 0],         !DK 0 NAME BYTE 3       ZZZ
0       !       DO_NUL          = [10, 0, 16, 0]        !NUL AFTER NAME         ZZZ
0       !       DO_DBN          = [11, 0, 8, 0],        !DK 0 RELATIVE DBN      ZZZ
0       !       DO_WRITE        = [11, 12, 1, 0],       !DK 0 DUP WRITE FLAG    ZZZ
0       !       DO_ACTIVE       = [11, 13, 1, 0],       !DK 0 ACTIVE FLAG       ZZZ
0       !       DO_DUPERROR     = [11, 14, 1, 0],       !DK 0 DUP ERROR FLAG    ZZZ
0       !       DONODUPMED      = [11, 15, 1, 0],       !DK 0 NO DUP MEDIA FLAG ZZZ
0       !       DO_COUNT        = [12, 0, 16, 0],       !DK 0 RELATIVE MSCP FUN- ZZZ
0       !                                                     CTION COUNTER     ZZZ
0       !                                                                       ZZZ
0       !       ! REPEAT WORDS 3 THROUGH 12 ABOVE AS: !                         ZZZ
0       !       ! WORDS 13 THROUGH 21 FOR DRIVE 1     !                         ZZZ
0       !       ! WORDS 22 THROUGH 30 FOR DRIVE 2     !                         ZZZ
0       !       ! WORDS 31 THROUGH 39 FOR DRIVE 3     !                         ZZZ
0       !       !                                     !                         ZZZ
0       !                                                                       ZZZ
0               tes.
0       !
0       !****** MSCP PACKET FIELDS
0       !               (NOTE: BASE ADDRESS OF PACKET REFERENCES THE PACKET'S OWN
0       !               BUFFER DESCRIPTOR, RATHER THAN THE MESSAGE BODY (TEXT + 0).
0       !               SEE DOCUMENTATION FOR LAYOUT OF MSCP PACKETS.)
0       !
0       PKT_FIELDS =
0               set
0       !
0       !       HEADER FIELDS
0       !
0               PKT_LO          = [0, 0, 16, 0],                        ! PACKET DESCRIPTOR (LO ORDER)
0               PKT_HI          = [1, 0, 16, 0],                        ! PACKET DESCRIPTOR (HI ORDER - ALL FIELDS)
0               PKT_U           = [1, 0, 2  0],                         ! PACKET DESCRIPTOR (HI ORDER UNIBUS BITS)
```

```
        PKT_Q           = [1, 2, 4, 0],                 ! PACKET DESCRIPTOR (HI ORDER Q-BUS BITS)
        PKT_F           = [1, 14, 1, 0],                ! PACKET DESCRIPTOR FLAG BIT
        PKT_O           = [1, 15, 1, 0],                ! PACKET DESCRIPTOR OWNERSHIP BIT
        CMD_TYPE        = [2, 0, 8, 0],                 ! COMMAND TYPE
        RSP_RECEIVED    = [2, 8, 8, 0],                 ! FLAG SET IF RESPONSE TO COMMAND RECEIVED
        MSGLEN          = [3, 0, 16, 0],                ! MESSAGE LENGTH
        CREDITS         = [4, 0, 4, 0],                 ! CREDITS
        MSGTYP          = [4, 4, 4, 0],                 ! MESSAGE TYPE
        CONNID          = [4, 8, 8, 0],                 ! CONNECTION ID
    !
    !   GENERIC COMMAND PACKET AND END PACKET HEADER FILEDS
    !
        CRN_LO          = [5, 0, 16, 0],                ! COMMAND REF NUMBER (LO ORDER)
        CRN_HI          = [6, 0, 16, 0],                ! COMMAND REF NUMBER (HI ORDER)
        DK_NUM          = [7, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
        OPCODE          = [9, 0, 8, 0],                 ! OPCODE AND ENDCODE
        MODIFY          = [10, 0, 16, 0],               ! COMMAND MODIFIERS
        STATUS_CODE     = [10, 0, 5, 0],                ! STATUS (PART OF RESPONSE PACKET)
        STATUS_SUBCODE  = [10, 5, 11, 0],               ! SUBCODE (PART OF RESPONSE PACKET)
    !
    !   READ, WRITE, AND ACCESS COMMAND FIELDS (FOR COMMAND AND END PACKETS)
    !
        BC_LO           = [11, 0, 16, 0],               ! BYTE COUNT (LO ORDER)
        BC_HI           = [12, 0, 16, 0],               ! BYTE COUNT (HI ORDER)
        BUF_0           = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR
        BUF_1           = [14, 0, 16, 0],               !        :
        BUF_2           = [15, 0, 16, 0],               !        :
        BUF_3           = [16, 0, 16, 0],               !        :
        BUF_4           = [17, 0, 16, 0],               !        :
        BUF_5           = [18, 0, 16, 0],               !        V
        LBN_L           = [19, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
        LBN_H           = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)

    !
    ! DUP PROGRAM LETTER FIELDS (FOR EXECUTE LOCAL PROGRAM CMD)        ZZZ
    !                                                                  ZZZ
        L1              = [11, 0, 8, 0],      !LETTER NO 1             ZZZ
        L2              = [11, 8, 8, 0],      !LETTER NO 2             ZZZ
        L3              = [12, 0, 8, 0],      !LETTER NO 3             ZZZ
        L4              = [12, 8, 8, 0],      !LETTER NO 4             ZZZ
        L5              = [13, 0, 8, 0],      !LETTER NO 5             ZZZ
        L6              = [13, 8, 8, 0],      !LETTER NO 6             ZZZ
    !
    !   SET CONTROLLER CHARACTERISTICS COMMAND FIELDS
    !
        C_FLAGS         = [12, 0, 16, 0],               ! CONTROLLER FLAGS
    !
    !   ONLINE COMMAND FIELDS
    !
        U_FLAGS         = [12, 0, 16, 0],               ! UNIT FLAGS
        DDPAR           = [19, 0, 16, 0]                ! DEVICE-DEPENDENT PARAMETERS
        tes,
    !
```

```
0     !****** RETURN PACKET (RETPKT) FIELDS
0     !               (SIMILAR, BUT NOT IDENTICAL, TO MSCP PACKET FIELDS)
0     !
0     RP_FIELDS =
0             set
0     !
0     !       COMMON TO ALL RETURN PACKETS FROM DISK MSCP
0     !
0             MESLEN          = [0, 0, 16, 0],                ! MESSAGE LENGTH
0             CTLR            = [1, 0, 4, 0],                 ! CONTROLLER NUMBER (CREDITS OVERWRITTEN)
0             MESTYP          = [1, 4, 4, 0],                 ! MESSAGE TYPE
0             CONID           = [1, 8, 8, 0],                 ! CONNECTION ID
0             CRF_LO          = [2, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (LO ORDER)
0             CRF_HI          = [3, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (HI ORDER)
0             DISK            = [4, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
0             CMDMOD          = [5, 0, 16, 0],                ! COMMAND MODIFIERS
0             ENDCOD          = [6, 0, 8, 0],                 ! END CODE
0             FLAGS           = [6, 8, 8, 0],                 ! FLAGS
0             STATUS          = [7, 0, 16, 0],                ! STATUS AND SUB-CODE
0             STSCOD          = [7, 0, 5, 0],                 ! STATUS CODE
0             SUBCOD          = [7, 5, 11, 0],                ! SUB-CODE
0     !
0     !       READ, WRITE, AND ACCESS COMMAND RETURN PACKETS
0     !
0             BCNT_LO         = [8, 0, 16, 0],                ! BYTE COUNT (LO ORDER)
0             BCNT_HI         = [9, 0, 16, 0],                ! BYTE COUNT (HI ORDER)
0             BUFF_0          = [10, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 0)
0             BUFF_1          = [11, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 1)
0             BUFF_2          = [12, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 2)
0             BUFF_3          = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 3)
0             BUFF_4          = [14, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 4)
0             BUFF_5          = [15, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 5)
0             BBLK_LO         = [16, 0, 16, 0],               ! FIRST BAD BLOCK (LO ORDER)
0             BBLK_HI         = [17, 0, 16, 0],               ! FIRST BAD BLOCK (HI ORDER)
0             CBCNT_LO        = [18, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (LO ORDER)
0             CBCNT_HI        = [19, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (HI ORDER)
0             LBN_LO          = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
0             LBN_HI          = [21, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
0     !
0     !       SET CONTROLLER CHARACTERISTICS RETURN PACKET
0     !
0             C_FLGS          = [9, 0, 16, 0],                ! CONTROLLER FLAGS
0             C_TIME          = [10, 0, 16, 0],               ! CONTROLLER TIMEOUT
0     !
0     !       UNIT ONLINE RETURN PACKET
0     !
0             U_FLGS          = [9, 0, 16, 0],                ! UNIT FLAGS
0             R_MODEL         = [13, 0, 8, 0],                ! 2 DIGIT MODEL NUMBER              ZZZ
0             NAME_NUM        = [14, 0, 6, 0],                ! MODEL NAME - 2 DIGIT NUMBER
0             NAME_1_LO       = [14, 12, 4, 0],               ! MODEL NAME - 2ND CHARACTER (LOW  ORDER 4 BITS)
0             NAME_1_HI       = [15, 0, 1, 0],                ! MODEL NAME - 2ND CHARACTER (HIGH ORDER 1 BIT)
0             NAME_0          = [15, 1, 5, 0],                ! MODEL NAME - IST CHARACTER
0     !ZZZ    USIZ_LO         = [18, 0, 16, 0],               ! UNIT SIZF (LO ORDER)
```

```
'ZZZ    USIZ_HI          = [19, 0, 16, 0],            ! UNIT SIZE (HI ORDER)
        SIZE0            = [18, 0, 16, 0],            !LOWER WD OF MAX LBNS OR UNIT SIZE      ZZZ
        SIZE1            = [19, 0, 16, 0]             !UPPER WD    "        "        "        ZZZ
        tes,
!
!****** STATISTICS TABLE (TALLY) FIELDS
!
T_FIELDS =
        set
        BYTES_READ_LO    = [0, 0, 16, 0],             ! NUMBER OF BYTES READ (LO ORDER)
        BYTES_READ_HI    = [1, 0, 16, 0],             ! NUMBER OF BYTES READ (HI ORDER)
        MBYTES_READ      = [2, 0, 16, 0],             ! MEGABYTES READ
        BYTES_WRIT_LO    = [3, 0, 16, 0],             ! NUMBER OF BYTES WRITTEN (LO ORDER)
        BYTES_WRIT_HI    = [4, 0, 16, 0],             ! NUMBER OF BYTES WRITTEN (HI ORDER)
        MBYTES_WRIT      = [5, 0, 16, 0],             ! MEGABYTES WRITTEN
        ERR_HARD         = [6, 0, 16, 0],             ! NUMBER OF HARD ERRORS
!
        TOT_READS_LO     = [7, 0, 16, 0],             ! TOTAL NUMBER OF READS (LO ORDER)
        TOT_READS_HI     = [8, 0, 16, 0],             ! TOTAL NUMBER OF READS (HI ORDER)
        TOT_WRITES_LO    = [10, 0, 16, 0],            ! TOTAL NUMBER OF WRITES (LO ORDER)
        TOT_WRITES_HI    = [11, 0, 16, 0],            ! TOTAL NUMBER OF WRITES (HI ORDER)
        TOT_BYT_RED_LO   = [13, 0, 16, 0],            ! TOTAL BYTES READ (LO ORDER)
        TOT_BYT_RED_HI   = [14, 0, 16, 0],            ! TOTAL BYTES READ (HI ORDER)
        MTOT_BYT_RED     = [15, 0, 16, 0],            ! TOTAL MEGABYTES READ
        TOT_BYT_WRT_LO   = [16, 0, 16, 0],            ! TOTAL BYTES WRITTEN (LO ORDER)
        TOT_BYT_WRT_HI   = [17, 0, 16, 0],            ! TOTAL BYTES WRITTEN (HI ORDER)
        MTOT_BYT_WRT     = [18, 0, 16, 0],            ! TOTAL MEGABYTES WRITTEN
        ERR_HRD_SEK      = [19, 0, 8, 0],             ! TOTAL HARD ERRORS - SEEK
        ERR_HRD_DAT      = [19, 8, 8, 0],             ! TOTAL HARD ERRORS - DATA
        ERR_HRD_DRV      = [20, 0, 8, 0],             ! TOTAL HARD ERRORS - DRIVE
        ERR_HRD_HST      = [20, 8, 8, 0],             ! TOTAL HARD ERRORS - HOST
        ERR_SFT_SEK      = [21, 0, 8, 0],             ! TOTAL SOFT ERRORS - SEEK
        ERR_SFT_DAT      = [21, 8, 8, 0],             ! TOTAL SOFT ERRORS - DATA
        ERR_SFT_DRV      = [22, 0, 8, 0],             ! TOTAL SOFT ERRORS - DRIVE
        ERR_SFT_HST      = [22, 8, 8, 0],             ! TOTAL SOFT ERRORS - HOST
        T_BLK_WT         = [23, 0,16, 0],    !                     ZZZ
        T_DBN_WT         = [24, 0,16, 0],    !DBNS WRITTEN         ZZZ
        T_BLK_RD         = [25, 0,16, 0],    !                     ZZZ
        T_DBN_RD         = [26, 0,16, 0]     !DBNS READ            ZZZ

        tes,
!
!****** CONTROLLER ERROR TALLY FIELDS
!
C_ERR_FIELDS =
        set
        C_ERR_HRD        = [0, 0, 8, 0],              ! HARD ERRORRS
        C_ERR_SFT        = [0, 8, 8, 0]               ! SOFT ERRORS
        tes,
!
!****** DRIVER CONTROLLER TABLE (DCT) FIELDS
!
DCT_FIELDS =
```

```
                set
                WORD0           = [0, 0, 16, 0],                ! ALL FIELDS IN WORD 0
                CRING_CNT       = [0, 0, 8, 0],                 ! NUMBER OF SLOTS IN CRING NOT YET RETURNED TO HOST
                IG_INT          = [0, 14, 1, 0],                ! IGNORE INTERRUPT BIT
                STAT            = [0, 15, 1, 0],                ! ONLINE / OFFLINE STATUS
                SA_SAVE         = [1, 0, 16, 0],                ! SA REGISTER SAVE WORD
                RR_BEG          = [2, 0, 16, 0],                ! FIXED ADDRESSES OF START AND
                RR_END          = [3, 0, 16, 0],                !       END OF EACH RING
                CR_BEG          = [4, 0, 16, 0],                !             :
                CR_END          = [5, 0, 16, 0],                !             V
                RR_POLL         = [6, 0, 16, 0],                ! ADDR OF NEXT RRING SLOT TO BE POLLED
                LR_POLL         = [7, 0, 16, 0],                ! ADDR OF NEXT CRING SLOT TO BE POLLED
                CR_NEXT         = [8, 0, 16, 0]                 ! ADDR OF NEXT AVAIL CRING SLOT
                tes,
        !
        !****** ERROR LOG PACKET SAVE AREA FIELDS
        !
        EP_FIELDS =
                set
                EL_CNTR         = [0, 0, 8, 0],                 ! CONTROLLER NUMBER
                EL_CONTENTS     = [0, 8, 8, 0],                 ! FLAG INDICATES IF PACKET CONTENTS ALREADY PRINTED
                EL_MSGLEN       = [1, 0, 16, 0],                ! PACKET LENGTH
                EL_CRN_LO       = [3, 0, 16, 0],                ! COMMAND REFERENCE NUMBER
                EL_CRN_HI       = [4, 0, 16, 0],                !
                EL_DK_NUM       = [5, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
                EL_FORMAT       = [7, 0, 8, 0],                 ! FORMAT
                EL_EDR          = [7, 12, 1, 0],                ! ERROR DURING REPLACEMENT FLAG MMM
                EL_BRR          = [7, 13, 1, 0],                ! BAD BLOCK REPLACEMENT REQUEST FLAG  MMM
                EL_CONTINUE     = [7, 14, 1, 0],                ! CONTINUE FLAG
                EL_SUCCESS      = [7, 15, 1, 0],                ! SUCCESS FLAG
                EL_CODE         = [8, 0, 5, 0],                 ! ERROR CODE
                EL_SUBCODE      = [8, 5, 11, 0],                ! SUB CODE
                EL_RETRY        = [20, 8, 8, 0],                ! RETRY COUNT
                EL_BLOCK        = [23, 0, 16, 0],               ! BLOCK NUMBER
                EL_BLOCK_HI     = [24, 0, 12, 0],               !HIGH BITS OF BLOCK NUMBER                ZZZ
                EL_BLOCK_TYPE   = [24, 12, 4, 0]                ! TYPE OF BLOCK NUMBER INFO RETURNED
                tes,
        !
        !****** INFORMATION ABOUT LAST RESPONSE PACKET
        !
        LAST_PKT_FIELDS =
                set
                LAST_HRD_ERR    = [0, 0, 16, 0],                ! FLAG INDICATES IF HARD ERROR OCCURED
                LAST_CRN_LO     = [1, 0, 16, 0],                ! COMMAND REFERENCE NUMBER
                LAST_CRN_HI     = [2, 0, 16, 0]                 !
                tes,
        !
        !****** RDRX REGISTER FIELDS
        !
        RC_REG =
                set
                RC_ALL          = [0, 16, 0]                    ! DEFINE ALL BITS
                tes;
```

```
!****************************************************************************
!                                                                           *
!                          M A C R O S                                      *
!                                                                           *
!****************************************************************************
!
macro
!
!****** CST FIELDS. MODEL FOR WDS 3-12, 13-21, 22-30, AND 31-39.        ZZZ
!
        D_ALL           = 0, 16, 0%,                            ! ALL FIELDS
        D_DISK_NUM      = 0, 4, 0%,                             ! DISK ADDRESS
        D_TYPE          = 4, 1, 0%,                             !DISK TYPE - 1 BIT      ZZZ
        D_UNIT          = 8, 4, 0%,                             ! DISK UNIT NUMBER (DRS UNIT)
        D_FATAL         = 12, 1, 0%,                            ! FATAL ERROR BIT
        D_STAT          = 13, 1, 0%,                            ! DISK STATUS BIT
        D_PRES          = 14, 1, 0%,                            ! DISK PRESENT BIT
        D_PROT          = 15, 1, 0%,                            ! DISK PROTECTION BIT
        D_BEG0          = 0, 16, 0%,            !BEGIN TRACK LO         ZZZ
        D_BEG1          = 0, 16, 0%,            !BEGIN TRACK HI         ZZZ
        D_END0          = 0, 16, 0%,            !END TRACK LO           ZZZ
        D_END1          = 0, 16, 0%,            !END TRACK HI           ZZZ
        D_NAME_0        = 0, 8, 0%,             ! NAME (FIRST  CHARACTER)
        D_NAME_1        = 8, 8, 0%,             ! NAME (SECOND CHARACTER)
        D_NAME_2        = 0, 8, 0%,             ! NAME (THIRD  CHARACTER)
        D_NAME_3        = 8, 8, 0%,             ! NAME (FOURTH CHARACTER)
!       D_NUL           = 0, 16, 0%,            !NUL AFTER NAME         ZZZ
        D_DBN           = 0, 8, 0%,             !RELATIVE DBN           ZZZ
        DUPWRITE        = 12, 1, 0%,            !DUP WRITE FLAG         ZZZ
        D_ACTIVE        = 13, 1, 0%,            !ACTIVE STATE           ZZZ
        DUPERROR        = 14, 1, 0%,            !DUP ERROR FLAG         ZZZ
        NODUPMEDIA      = 15, 1, 0%,            !NO DUP MEDIA           ZZZ
        D_COUNT         = 0, 16, 0%,            !MSCP FUNCTION COUNTER  ZZZ


!*********** BST FIELDS *************                                   ZZZ
!                                                                       ZZZ
        HI_WRD          = 1, 0, 16, 0%,         !HI LBN                 ZZZ
        LO_WRD          = 0, 0, 16, 0%,         !LO LBN                 ZZZ
!
!****** BIT TEST
!               (CAUTION: THE FIRST ARGUMENT IS THE ADDRESS AND NOT THE CONTENTS)
!
BIT_TST (ADDR, EXPECTED) =
    (if (.ADDR and EXPECTED) eql EXPECTED
    then
        TRUE
    else
        FALSE )%,
!
!****** RDRX WRITE
!
WRT_RDRX (O, FIELDNAM, IMAGE) =
```

```
begin
local
    RC_REG;
RC_REG <%fieldexpand (FIELDNAM)> = IMAGE;
(.RDRX_ADDR + (%upval * 0)) = .RC_REG;
end%;
```

```
;  0   !***********************************************************************
;  0   !                                                                      *
;  0   !                       S T R U C T U R E S                            *
;  0   !                                                                      *
;  0   !***********************************************************************
;  0   !
;  0   !****** NIBBLE (4-BIT) VECTOR STRUCTURE
;  0   !
;  0   !structure
;  0   !    NIBVECTOR [I; N] =
;  0   !        [(N + 1) / 2]
;  0   !        (NIBVECTOR + I / 2) <(I ^ 2) and 4, 4>;
;  0   !
;  0   !****** RDRX ACCESS ALGORITHM
;  0   !
;  0   structure
;  0       RDRX [O, P, S, E] =
;  1           begin
;  1           local
;  1               RC_REG;
;  1           RC_REG = .(RDRX + %upval * O) <O, %bpval, O>;
;  1           RC_REG
;  1           end
;  0           <P, S, E>;
```

```
;                              COMMAND QUALIFIERS

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

; Run Time:         00:04.7
; Elapsed Time:     00:16.0
; Lines/CPU Min:      9540
; Lexemes/CPU-Min: 50317
; Memory Used:  73 pages
; Library Precompilation Complete
```

```
REVISION HISTORY:

REV 1.6    11-APR-84    MERGED FIELD AND MANUFACTURING VERSIONS OF THE RD/RX EXERCISER.
                        ADDED SUPPORT FOR THE RUX50.

REV 1.7    01-MAY-84    ADDED CODE TO GET DEVICE TYPE FROM CONTROLLER CHARACTERISTICS;
                        ADDED APT BREAKS IN UNIT_INIT ROUTINE; CORRECTED SOFT SEEK ERROR
                        TOTALS; PROTECT MEDIA ON DEFAULT.

REV 1.8    06-JUL-84    ELIMINATE GETTING DISK TYPE FROM ID BLOCK ON A RESTART;

REV 1.9    19-SEP-84    ON END OF PASS, WAIT UNTIL LAST PACKET RETURNED BEFORE WRITING IP.


REV 2.0    09-NOV-84    DON'T OUTPUT DUP STATS HEADER IF NO WINCHESTER.
                        FIXED 'CMD REF NO. NOT SENT BY HOST' PROBLEM BY USING OPERATOR-
                        SPECIFIED BR LEVEL WHEN SENDING PACKETS.

REV 2.1    27-DEC-84    ADDED APT MODE QUESTION; ADDED RETRIES TO DUP TESTS. ADDED CODE TO
                        INT_GEN ROUTINE TO MAKE IT COMPATIBLE WITH MICROCODE VERSION 9.1.

REV 2.2    04-APR-85    IN POLL_RRING AND POLL_CRING ROUTINES, ZERO INTERRUPT COUNTERS ON
                        ENTRY TO PREVENT SLOWDOWN PROBLEM. ADDED 32-BIT LBN ADDRESSING.
                        CHANGED HEADER MACRO AND .REQ FILE FOR COMPATIBILITY WITH XXDP V2.
                        MADE DUP COMMAND TYPE = 0.

REV 2.3    17-JUN-85    ADDED CODE TO HANDLE DUP'S OUT OF RANGE STATUS. REPORT BOTH WORDS
                        OF LBN ADDRESS IN ERROR MESSAGES. ACCESS LBNS 0 AND 100 IN ACCESS
                        ROUTINE.

REV 2.4    18-JUN-85    ADD AND REARRANGE APT BREAKS FOR APT COMPATIBILITY.  BALANCE RATIO
                        OF READS TO WRITES (2:1) WHEN RUNNING WITH FASTER CPU.  PRINT DUP
                        FATAL MESSAGE NUMBER.  INCLUDE DUP FATAL MESSAGE, ERROR CODE 5.
                        COUNT "ZERO FILL" AS DATA TRANSFERED ON PARTIAL BLOCK TRANSFERS
                        FOR DISK ERROR RATE CALCULATION.

REV 2.5    27-JUN-85    QIO_OK WILL BE TRUE WHEN QIO COUNT IS LESS THAN COMMAND RING - 2
                        TO AVOID DISCARDING PACKECTS AND HENCE SEQUENTIAL AND RANDOM
                        NUMBERS DURING OPERATION.  REMOVE DUP AND PLACE INTO ANOTHER
                        EXERCISER.
```

```
; C 0095  1
; C 0096  1                    TABLE OF CONTENTS
; C 0097  1
; C 0098  1
; C 0099  1
; C 0100  1            1.0     GENERAL INFORMATION
; C 0101  1            1.1     PROGRAM ABSTRACT
; C 0102  1            1.2     SYSTEM REQUIREMENTS
; C 0103  1            1.2.1   HARDWARE REQUIREMENTS
; C 0104  1            1.2.2   SOFTWARE REQUIREMENTS
; C 0105  1            1.3     RELATED DOCUMENTS AND STANDARDS
; C 0106  1            1.4     DIAGNOSTIC HIERARCHY PREREQUISITES
; C 0107  1            1.5     ASSUMPTIONS
; C 0108  1            1.6     MEMORY MAP
; C 0109  1
; C 0110  1            2.0     OPERATING INSTRUCTIONS
; C 0111  1            2.1     HARDWARE QUESTIONS
; C 0112  1            2.2     SOFTWARE QUESTIONS
; C 0113  1
; C 0114  1            3.0     ERROR TYPES
; C 0115  1            3.1     ERROR INFORMATION
; C 0116  1            3.2     INITIALIZATION ERRORS
; C 0117  1            3.3     EXERCISER ERRORS
; C 0118  1            3.4     ERROR LOG MESSAGES
; C 0119  1            3.5     MSCP ERRORS
; C 0120  1            3.6     SAMPLE ERROR STATEMENT
; C 0121  1
; C 0122  1            4.0     PERFORMANCE AND PROGRESS REPORTS
; C 0123  1
; C 0124  1            5.0     TEST SUMMARY
; C 0125  1            5.1     INITIALIZATION SUBTEST
; C 0126  1            5.2     EXERCISER
; C 0127  1            5.3     DROP UNIT SUMMARY
; C 0128  1
; C 0129  1            6.0     ERROR CODES
; C 0130  1
; C 0131  1            7.0     DATA PATTERNS
```

```
1.0    GENERAL INFORMATION
       -------------------

1.1    PROGRAM ABSTRACT
       ----------------

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


1.2    SYSTEM REQUIREMENTS
       -------------------

1.2.1  HARDWARE REQUIREMENTS
       ---------------------

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

1.2.2  SOFTWARE REQUIREMENTS
       ---------------------

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


1.3    RELATED DOCUMENTS AND STANDARDS
       -------------------------------

       XXDP+ SUPERVISOR/USERS MANUAL   CHQUS
       UQSSP UNIBUS/Q-BUS STORAGE SYSTEM 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.
```

```
1.6     MEMORY MAP
        ----------


        Memory layout on 28k machine - XXDP environment
        -----------------------------------------------

                                          ADDRESS

                +-------------------------+  160000
                !                         !
                !     XXDP+ MONITOR       !
                !                         !
                +-------------------------+
                !                         !
                !  XXDP+    TTY I/O AREA  !
                !                         !
                +-------------------------+
                !                         !
                !                         !
                !          DRS            !
                !                         !
                !                         !
                +-------------------------+  121000
                !                         !
                !       EXERCISER         !
                !                         !
                +-------------------------+  2000
                !                         !
                !        STACKS           !
                !                         !
                +-------------------------+  400
                !                         !
                !      VECTOR AREA        !
                !                         !
                +-------------------------+  0


        In a machine  with more memory,  free space will occur
        between the exerciser and the DRS.
```

2.0 OPERATING INSTRUCTIONS

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

2.1 HARDWARE QUESTIONS

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

Hardware Configuration Questions

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

1. CHANGE HW (L) Y ?

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

2. NUMBER OF UNITS (D) ?

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

3. IP ADDRESS (O) 172150 ?

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

```
4.  VECTOR ADDRESS (O) 154 ?

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

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

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


6.  DRIVE NUMBER (D) 0 ?

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


7.  ALSO RUN DUP EXERCISER (L) N ?

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



8.  WRITE ON DIAGNOSTIC AREA (L) N ?

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



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

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

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

    Enter in octal the less significant 16-bit word of the lowest
```

```
        LBN address in the test range. The value may be from 000000
        to 177777.

    11. HIGHER OCTAL WORD OF BEGINNING LBN ADDRESS (O) 0?

        Enter in octal the more significant 16-bit word of the
        lowest LBN address in the test range.

    12. LOWER OCTAL WORD OF ENDING LBN ADDRESS (O) 150477?

        Enter in octal the less significant 16-bit word of the
        highest LBN address in the test range. 150477 is the
        highest LBN address for an RD52.

    13. HIGHER OCTAL WORD OF ENDING LBN ADDRESS (O) 0?

        Enter in octal the more significant 16-bit word of the
        highest LBN address in the test range.

         Note:
              The four previous questions are usually software
              Parameter questions, but since three different disk
              drives exist on the subsystem, this becomes a  unit
              by unit question.  It is possible to specify an LBN
              which is too large since we are dealing with different
              drives. The program will check for block number bounds,
              and, if they are exceeded, will assign the maximum
              bounds for that drive.


     14. WRITE ON CUSTOMER DATA AREA ON THIS DISK UNIT (L) ?

        Answering YES will destroy any customer data that is  on
        the  disk; therefore, the following warning message will
        appear, followed by a confirmation prompt:

        ** WARNING - CUSTOMER DATA AREA WILL BE OVERWRITTEN! ...
         CONFIRM (L) ?

        This question will default to NO if the operator has de-
        cided  to  bypass  the  hardware  questions.  Otherwise,
        there is no default.
```

2.2 SOFTWARE QUESTIONS

Software Parameter Questions

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

1. CHANGE SW (L) Y ?

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

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

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

3. HARD ERROR LIMIT (D) 32 ?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

10. HALT ON SOFT ERRORS (#s 50-54) (L) N ?

This question is similar to question 8, but refers to Soft Errors.

11. COUNT EACH RETRY AS A SEPARATE SOFT ERROR (L) N ?

On encountering any error on a read/write, the controller retries the operation a number of times. If the operation is eventually successful, this is reported as a Soft Error. The error log packet contains the number of retries performed before the operation was successful. Normally, the whole sequence of retries is classified as one Soft Error. Answer Yes to the question if it is desired to count each internal retry attempt as a separate Soft Error.

12. RANDOM SEEK MODE (L) Y ?

Answer YES to cause block numbers to be chosen randomly. Answer NO to cause block numbers to be selected sequentially.

13. UNITS TO BE SELECTED AT RANDOM (NO, IMPLIES SEQUENTIAL) (L) N ?

This question is optionally asked if the answer to the previous question is N[o]. The selection of units for sequential operations is affected by the answer to this question. If the default answer is chosen (N[o]), then units shall be selected in a predetermined manner in accordance with the typical seek time margins for each drive. If the alternate answer is chosen (Y[es]), then the units will be chosen at random in accordance with the percentages specified in Software question 4.

14. READ-COMPARES PERFORMED AT THE CONTROLLER (L) Y ?

Answering YES causes all read commands to include the "compare" modifier. This essentially forces the controller to perform two read operations on the same disk address, and to compare the results.

The following message will appear after the operator has answered this question:

15. RUNNING UNDER THE A.P.T. MONITOR (L) N ?

THIS QUESTION SHOULD BE ANSWERED N (DEFAULT) IN THE FIELD. IT ENABLES THE PROGRAM TO KNOW THAT IT IS RUNNING UNDER A SPECIAL (AUTOMATED PRODUCT TEST) MONITOR.

THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED DISK UNITS.

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

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

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

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

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

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

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

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

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

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

```
3.0     ERROR TYPES
        ---- ------

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

        SYSTEM FATAL ERRORS
        --------- --------

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

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


        DRIVE FATAL ERRORS
        ------------------

        Drive fatal errors are a result of:

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


        HARD ERRORS
        -----------

        Hard errors are a result of:

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

                *  Note: Retries are executed in the controller

        SOFT ERRORS
        -----------

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

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

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 add-ess, if applicable. In some cases where a drive-fata_ error applies to a controller, the controller's IP address will be printed.

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

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

3.2 INITIALIZATION ERRORS

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

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

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

3.3 EXERCISER ERRORS

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

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

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

3.4 ERROR LOG MESSAGES

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

3.5 MSCP ERRORS

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

```
3.6   SAMPLE ERROR STATEMENT
      ----------------------


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


(example)                             (comments)

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



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

4.0 PERFORMANCE AND PROGRESS REPORTS

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

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

The sub categories are:

1. disk related errors

2. seek (or format) related errors

3. controller or drive related errors

4. host (the CPU) related errors.

All numeric values are in decimal radix.

```
UNT        # OF BYTS  # OF    BYTES   -- HRD ERS --   -- SFT ERS --
 #   TYPE READS READ WRITES  WRITTEN DAT SEK DRV HST DAT SEK DRV HST
--- ---- ----- ---- ------  ------- -------------- -------------

 X   XXXX  XXXX XXXX  XXXXX  XXXXXX  X   X   X   X   X   X   X   X
 .    ..    ..   ..    ...    ....   .   .   .   .   .   .   .   .
 .    ..    ..   ..    ...    ....   .   .   .   .   .   .   .   .
```

5.0 TEST SUMMARY

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

5.1 INITIALIZATION SUBTEST

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

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

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

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

5.2 EXERCISER

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

If a read/write error occurs during this test, then the controller will initiate a. appropriate number of retries. If all retries fail, then a hard error will be reported to the host, an error message will be displayed on the console terminal and the error will be tallied for the summary report. The unit will be dropped if the hard error count has exceeded the specified limit.

5.3 DROP UNIT SUMMARY

During the Initialization Subtest, individual units will be dropped from the test sequence if they are unable to be brought online or the operator specified drive does not match the hardware.

During the Exercise, the program will drop a unit for one of three reasons. The normal path is for each unit to complete the transfer of N megabytes, where N is specified by the operator during SW questioning and be soft-dropped. Otherwise, a unit will be hard-dropped if the number of hard errors encountered exceeds the operator-specified limit, or if a fatal error is detected. Units hard-dropped may later be added to the test cycle. However, statistics for the hard-added unit will be cleared to zero; if a transfer limit was specified, in which case the unit was soft-dropped, the statistics may or may not be cleared depending on the operators answer to Software question 12.

```
6.0  ERROR CODES
     This section describes the error codes generated by this exerciser.

     SYSTEM FATAL ERRORS
     -------------------

     1    More than 4 units specified

     DRIVE FATAL ERRORS
     ------------------

     10   Controller couldn't be  addressed  Wrong IP address selected
          at the address given.

     11   Controller  didn't  interrupt  at  Wrong vector address sel-
          the interrupt vector given.        ected.

     12   Controller  didn't  interrupt  at  Wrong BR level selected.
          the BR level given.

     13   Init sequence failed.              Either  one of  the  four
                                             initialization  steps did
                                             not receive  the  correct
                                             response  from  the  Con-
                                             troller,  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  Innermost   or   outermost
          the outer track failed.           track's header may be cor-
                                            repeated.

     18   Unit went off-line.               ---

     19   Drive type not known.             The  version of the  Exer-
                                            ciser  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.

41	Drive error.	See #40.
42	Host write compare error.	Error detected when Host CPU compared the data written and read back. May be a problem with the Host or Controller RAM.
43	Message from internal diagnostics	See #40.
44	Duplicate unit number detected by the Controller.	---
45	Unknown end code received.	Problem with the Controller microcode or the port/DMA interface.

SOFT ERRORS

50	Controller error.	See error-log packet for details as the exact cause may not be evident.
51	Host memory access error.	See #50.
52	Disk transfer error.	See #50.
53	'Standard Disk Interconnect' error.	See #50.
54	'Small Disk' error.	See #50.

DUP ERRORS
----

60 Unable to load local controller DUP media.

61 (Not used)

62 Illegal unit number.

63 Illegal relative or physical block.

64 Device error.

```
; C 1012  1
; C 1013  1                   65 Zero length message.
; C 1014  1
; C 1015  1                   66 Unknown DUP status code.
; C 1016  1
; C 1017  1                   67 Invalid command.
```

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

68 No region available.

69 No region suitable.

70 Program not known.

71 Load failure.

72 Standalone.

73 Unknown DUP status code.
```

```
         7.0   DATA PATTERNS
               -------- ----


                        HEX      OCTAL                BINARY
                        ---      -----                ------

       Pattern 1              R A N D O M   N U M B E R S

       Pattern 2        0000     000000         0 000 000 000 000 000

       Pattern 3        FFFF     177777         1 111 111 111 111 111

       Pattern 4        8B8B     105613         1 000 101 110 001 011

       Pattern 5        3333     031463         0 011 001 100 110 011

       Pattern 6        3091     030221         0 011 000 010 010 001

       Pattern 7        0001     000001         0 000 000 000 000 001
                        0003     000003         0 000 000 000 000 011
                        0007     000007         0 000 000 000 000 111
                        000F     000017         0 000 000 000 001 111
                        001F     000037         0 000 000 000 011 111
                        003F     000077         0 000 000 000 111 111
                        007F     000177         0 000 000 001 111 111
                        00FF     000377         0 000 000 011 111 111
                        01FF     000777         0 000 000 111 111 111
                        03FF     001777         0 000 001 111 111 111
                        07FF     003777         0 000 011 111 111 111
                        0FFF     007777         0 000 111 111 111 111
                        1FFF     017777         0 001 111 111 111 111
                        3FFF     037777         0 011 111 111 111 111
                        7FFF     077777         0 111 111 111 111 111
                        FFFF     177777         1 111 111 111 111 111

       Pattern 8        FFFE     177776         1 111 111 111 111 110
                        FFFC     177774         1 111 111 111 111 100
                        FFF8     177770         1 111 111 111 111 000
                        FFF0     177760         1 111 111 111 110 000
                        FFE0     177740         1 111 111 111 100 000
                        FFC0     177700         1 111 111 111 000 000
                        FF80     177600         1 111 111 110 000 000
                        FF00     177400         1 111 111 100 000 000
                        FE00     177000         1 111 111 000 000 000
                        FC00     176000         1 111 110 000 000 000
                        F800     174000         1 111 100 000 000 000
                        F000     170000         1 111 000 000 000 000
                        E000     160000         1 110 000 000 000 000
                        C000     140000         1 100 000 000 000 000
                        8000     100000         1 000 000 000 000 000
                        0000     000000         0 000 000 000 000 000
```

```
; C 1090  1          Pattern 9     0000    000000        0 000 000 000 000 000
; C 1091  1                        0000    000000        0 000 000 000 000 000
; C 1092  1                        0000    000000        0 000 000 000 000 000
; C 1093  1                        FFFF    177777        1 111 111 111 111 111
; C 1094  1                        FFFF    177777        1 111 111 111 111 111
; C 1095  1                        FFFF    177777        1 111 111 111 111 111
; C 1096  1                        0000    000000        0 000 000 000 000 000
; C 1097  1                        0000    000000        0 000 000 000 000 000
; C 1098  1                        FFFF    177777        1 111 111 111 111 111
; C 1099  1                        FFFF    177777        1 111 111 111 111 111
; C 1100  1                        0000    000000        0 000 000 000 000 000
; C 1101  1                        FFFF    177777        1 111 111 111 111 111
; C 1102  1                        0000    000000        0 000 000 000 000 000
; C 1103  1                        FFFF    177777        1 111 111 111 111 111
; C 1104  1                        0000    000000        0 000 000 000 000 000
; C 1105  1                        FFFF    177777        1 111 111 111 111 111
; C 1106  1
; C 1107  1          Pattern 10    B6D9    133331        1 011 011 011 011 001
; C 1108  1
; C 1109  1          Pattern 11    5555    052525        0 101 010 101 010 101
; C 1110  1                        5555    052525        0 101 010 101 010 101
; C 1111  1                        5555    052525        0 101 010 101 010 101
; C 1112  1                        AAAA    125252        1 010 101 010 101 010
; C 1113  1                        AAAA    125252        1 010 101 010 101 010
; C 1114  1                        AAAA    125252        1 010 101 010 101 010
; C 1115  1                        5555    052525        0 101 010 101 010 101
; C 1116  1                        5555    052525        0 101 010 101 010 101
; C 1117  1                        AAAA    125252        1 010 101 010 101 010
; C 1118  1                        AAAA    125252        1 010 101 010 101 010
; C 1119  1                        5555    052525        0 101 010 101 010 101
; C 1120  1                        AAAA    125252        1 010 101 010 101 010
; C 1121  1                        5555    052525        0 101 010 101 010 101
; C 1122  1                        AAAA    125252        1 010 101 010 101 010
; C 1123  1                        5555    052525        0 101 010 101 010 101
; C 1124  1                        AAAA    125252        1 010 101 010 101 010
```

```
; C 1125  1       Pattern 12    2D2D    026455    0 010 110 100 101 101
; C 1126  1                     2D2D    026455    0 010 110 100 101 101
; C 1127  1                     2D2D    026455    0 010 110 100 101 101
; C 1128  1                     D2D2    151322    1 101 001 011 010 010
; C 1129  1                     D2D2    151322    1 101 001 011 010 010
; C 1130  1                     D2D2    151322    1 101 001 011 010 010
; C 1131  1                     2D2D    026455    0 010 110 100 101 101
; C 1132  1                     2D2D    026455    0 010 110 100 101 101
; C 1133  1                     D2D2    151322    1 101 001 011 010 010
; C 1134  1                     D2D2    151322    1 101 001 011 010 010
; C 1135  1                     2D2D    026455    0 010 110 100 101 101
; C 1136  1                     2D2D    026455    0 010 110 100 101 101
; C 1137  1                     D2D2    151322    1 101 001 011 010 010
; C 1138  1                     2D2D    026455    0 010 110 100 101 101
; C 1139  1                     D2D2    151322    1 101 001 011 010 010
; C 1140  1                     2D2D    026455    0 010 110 100 101 101
; C 1141  1                     D2D2    151322    1 101 001 011 010 010
; C 1142  1                     2D2D    026455    0 010 110 100 101 101
; C 1143  1                     D2D2    151322    1 101 001 011 010 010
; C 1144  1                     2D2D    026455    0 010 110 100 101 101
; C 1145  1
; C 1146  1       Pattern 13    6DB6    066666    0 110 110 110 110 110
; C 1147  1
; C 1148  1       Pattern 14    0001    000001    0 000 000 000 000 001
; C 1149  1                     0002    000002    0 000 000 000 000 010
; C 1150  1                     0004    000004    0 000 000 000 000 100
; C 1151  1                     0008    000010    0 000 000 000 001 000
; C 1152  1                     0010    000020    0 000 000 000 010 000
; C 1153  1                     0020    000040    0 000 000 000 100 000
; C 1154  1                     0040    000100    0 000 000 001 000 000
; C 1155  1                     0080    000200    0 000 000 010 000 000
; C 1156  1                     0100    000400    0 000 000 100 000 000
; C 1157  1                     0200    001000    0 000 001 000 000 000
; C 1158  1                     0400    002000    0 000 010 000 000 000
; C 1159  1                     0800    004000    0 000 100 000 000 000
; C 1160  1                     1000    010000    0 001 000 000 000 000
; C 1161  1                     2000    020000    0 010 000 000 000 000
; C 1162  1                     4000    040000    0 100 000 000 000 000
; C 1163  1                     8000    100000    1 000 000 000 000 000
```

```
; C 1164  1        Pattern 15      FFFE     177776        1 111 111 111 111 110
; C 1165  1                        FFFD     177775        1 111 111 111 111 101
; C 1166  1                        FFFB     177773        1 111 111 111 111 011
; C 1167  1                        FFF7     177767        1 111 111 111 110 111
; C 1168  1                        FFEF     177757        1 111 111 111 101 111
; C 1169  1                        FFDF     177737        1 111 111 111 011 111
; C 1170  1                        FFBF     177677        1 111 111 110 111 111
; C 1171  1                        FF7F     177577        1 111 111 101 111 111
; C 1172  1                        FEFF     177377        1 111 111 011 111 111
; C 1173  1                        FDFF     176777        1 111 110 111 111 111
; C 1174  1                        FBFF     175777        1 111 101 111 111 111
; C 1175  1                        F7FF     173777        1 111 011 111 111 111
; C 1176  1                        EFFF     167777        1 110 111 111 111 111
; C 1177  1                        DFFF     157777        1 101 111 111 111 111
; C 1178  1                        BFFF     137777        1 011 111 111 111 111
; C 1179  1                        7FFF     077777        0 111 111 111 111 111
; C 1180  1
; C 1181  1        Pattern 16      B6D9     133331        1 011 011 011 011 001
; C 1182  1                        B6D9     133331        1 011 011 011 011 001
; C 1183  1                        B6D9     133331        1 011 011 011 011 001
; C 1184  1                        DB6C     155554        1 101 101 101 101 100
; C 1185  1                        DB6C     155554        1 101 101 101 101 100
; C 1186  1                        DB6C     155554        1 101 101 101 101 100
; C 1187  1                        B6D9     133331        1 011 011 011 011 001
; C 1188  1                        B6D9     133331        1 011 011 011 011 001
; C 1189  1                        DB6C     155554        1 101 101 101 101 100
; C 1190  1                        DB6C     155554        1 101 101 101 101 100
; C 1191  1                        B6D9     133331        1 011 011 011 011 001
; C 1192  1                        DB6C     155554        1 101 101 101 101 100
; C 1193  1                        B6D9     133331        1 011 011 011 011 001
; C 1194  1                        DB6C     155554        1 101 101 101 101 100
; C 1195  1                        B6D9     133331        1 011 011 011 011 001
; C 1196  1                        DB6C     155554        1 101 101 101 101 100
```

```
; C 1197   1          Pattern 17        (LBN)*    (LBN)                  (LBN)
; C 1198   1                            8D36      106466         1 000 110 100 110 110
; C 1199   1                            8D36      106466         1 000 110 100 110 110
; C 1200   1                            72C9      071311         0 111 001 011 001 001
; C 1201   1                            72C9      071311         0 111 001 011 001 001
; C 1202   1                            72C9      071311         0 111 001 011 001 001
; C 1203   1                            8D36      106466         1 000 110 100 110 110
; C 1204   1                            8D36      106466         1 000 110 100 110 110
; 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                            72C9      071311         0 111 001 011 001 001
; 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                            8D36      106466         1 000 110 100 110 110
; C 1218   1
; C 1219   1                    * This word position contains the number of the locical
; C 1220   1                      block to be written.
; C 1221   1
; C 1222   1
; C 1223   1
; C 1224   1          Pattern 18        8D36      106466         1 000 110 100 110 110
; C 1225   1                            (LBN)     (LBN)                (LBN)
; C 1226   1                            72C9      071311         0 111 001 011 001 001
; C 1227   1                            8D36      106466         1 000 110 100 110 110
; C 1228   1                            8D36      106466         1 000 110 100 110 110
; C 1229   1                            8D36      106466         1 000 110 100 110 110
; C 1230   1                            72C9      071311         0 111 001 011 001 001
; C 1231   1                            72C9      071311         0 111 001 011 001 001
; C 1232   1                            72C9      071311         0 111 001 011 001 001
; C 1233   1                            72C9      071311         0 111 001 011 001 001
; C 1234   1                            8D36      106466         1 000 110 100 110 110
; C 1235   1                            8D36      106466         1 000 110 100 110 110
; C 1236   1                            8D36      106466         1 000 110 100 110 110
; C 1237   1                            8D36      106466         1 000 110 100 110 110
; C 1238   1                            8D36      106466         1 000 110 100 110 110
; C 1239   1                            72C9      071311         0 111 001 011 001 001
; C 1240   1                            72C9      071311         0 111 001 011 001 001
; C 1241   1                            72C9      071311         0 111 001 011 001 001
; C 1242   1                            72C9      071311         0 111 001 011 001 001
; C 1243   1                            72C9      071311         0 111 001 011 001 001
; C 1244   1                            72C9      071311         0 111 001 011 001 001
```

```
; C 1245  1       Pattern 19     (LBN)   (LBN)            (LBN)
; C 1246  1                      B999    134631     1 011 100 110 011 001
; C 1247  1                      B999    134631     1 011 100 110 011 001
; C 1248  1                      4666    043146     0 100 011 001 100 110
; C 1249  1                      4666    043146     0 100 011 001 100 110
; C 1250  1                      4666    043146     0 100 011 001 100 110
; C 1251  1                      B999    134631     1 011 100 110 011 001
; C 1252  1                      B999    134631     1 011 100 110 011 001
; C 1253  1                      B999    134631     1 011 100 110 011 001
; C 1254  1                      B999    134631     1 011 100 110 011 001
; C 1255  1                      4666    043146     0 100 011 001 100 110
; C 1256  1                      4666    043146     0 100 011 001 100 110
; C 1257  1                      4666    043146     0 100 011 001 100 110
; C 1258  1                      4666    043146     0 100 011 001 100 110
; C 1259  1                      4666    043146     0 100 011 001 100 110
; C 1260  1                      B999    134631     1 011 100 110 011 001
; C 1261  1                      B999    134631     1 011 100 110 011 001
; C 1262  1                      B999    134631     1 011 100 110 011 001
; C 1263  1                      B999    134631     1 011 100 110 011 001
; C 1264  1                      B999    134631     1 011 100 110 011 001
; C 1265  1                      B999    134631     1 011 100 110 011 001
; C 1266  1
; C 1267  1       Pattern 20     B999    134631     1 011 100 110 011 001
; C 1268  1                      (LBN)   (LBN)            (LBN)
; C 1269  1                      4666    043146     0 100 011 001 100 110
; C 1270  1                      B999    134631     1 011 100 110 011 001
; C 1271  1                      B999    134631     1 011 100 110 011 001
; C 1272  1                      B999    134631     1 011 100 110 011 001
; C 1273  1                      4666    043146     0 100 011 001 100 110
; C 1274  1                      4666    043146     0 100 011 001 100 110
; C 1275  1                      4666    043146     0 100 011 001 100 110
; C 1276  1                      4666    043146     0 100 011 001 100 110
; C 1277  1                      B999    134631     1 011 100 110 011 001
; C 1278  1                      B999    134631     1 011 100 110 011 001
; C 1279  1                      B999    134631     1 011 100 110 011 001
; C 1280  1                      B999    134631     1 011 100 110 011 001
; C 1281  1                      B999    134631     1 011 100 110 011 001
; C 1282  1                      4666    043146     0 100 011 001 100 110
; C 1283  1                      4666    043146     0 100 011 001 100 110
; C 1284  1                      4666    043146     0 100 011 001 100 110
; C 1285  1                      4666    043146     0 100 011 001 100 110
; C 1286  1                      4666    043146     0 100 011 001 100 110
; C 1287  1                      4666    043146     0 100 011 001 100 110
; C 1288  1
; C 1289  1       Pattern 21     (LBN)   (LBN)            (LBN)
; C 1290  1
;   1291  1    )%
```

```
;    1292  1     %sbttl 'PROGRAM HEADER'
;    1293  1
;    1294  1     library 'ZRQAHO.L16';                          ! RDRX EXERCISER GLOBAL LIBRARY
;    1295  1
;    1296  1     !MMMrequire 'BLSMAC.REQ';                      ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    1297  1     require 'HSAXAO.BLB';                          ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    3038  1
;    3039  1     literal
;    3040  1         DS$NBR_OF_TESTS = 1;                       ! NUMBER OF TESTS IN THIS DIAGNOSTIC
;    3041  1
;    3042  1     EQUALS;
;    3043  1
;    3044  1     POINTER (ALL);
;    3045  1
;    3046  1     !+
;    3047  1     ! THE PROGRAM HEADER IS THE INTERFACE BETWEEN
;    3048  1     ! THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
;    3049  1     !-
;    3050  1
;    3051  1     !ZZZ HEADER (%ascii'ZRQA', %ascii'G', %ascii'0', 32000, 1, PRIO0);        !ZZZ NEED POSITIVE NUMBER
;    3052  1     HEADER (%ascii'ZRQA', %ascii'H', %ascii'0', 32000, 1, PRIO0,1);         !ZZZ FINAL 1 = NO TESTING ON TRAPS (SAVE TI
ME)
;    3053  1     !MMM HEADER (%ascii'ZRQA', %ascii'X', %ascii'8', 32000, 1, PRIO0,1);        !MMM FINAL 1 = NO TESTING ON TRAPS (SA
VE TIME)
;    3054  1
```

```
%sbttl 'DISPATCH TABLE'

!+
! THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
! IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
!-

DISPATCH (DS$NBR_OF_TESTS);
```

```
%sbttl 'GLOBAL DATA SECTION'

!+
! THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
! IN MORE THAN ONE TEST.
!-

psect
    global = $FFF$ (read, write, noexecute, global, concatenate);

global
    CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
                        ! RUN-TIME CONTROLLER STATUS TABLES
    CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
                        ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
    DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
                        ! DRIVER CONTROLLER TABLES
    DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
                        ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
    RDRX_ADDR : ref rdrx field (RC_REG),
                        ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
    IRDRX_ADDR : ref rdrx field (RC_REG),
                        ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER

    BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                          !ZZZ
                        !CONTAINS LO+ HI LBN FIELDS FOR SEQUENTIAL  !ZZZ
                        !I/O TRANSFER FOR EACH UNIT.                !ZZZ
    TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
                        ! STATISTICS TABLES
    T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
                        ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT

    !MMM DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),  !BUFFER FOR DUP  ZZZ
!MMM                    !INFO FROM RECEIVE + SEND CMDS                 ZZZ
    TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED] INITIAL (BYTE (REP     !ZZZ
            MAX_UNITS OF (1))),     !CURRENT TRACK DIRECTION           ZZZ
    RDM_CNT : WORD INITIAL (RDM_LEN),   !NO OF RANDOM NOS    \\KEEP    ZZZ
    RANDOM : VECTOR [RDM_LEN, WORD],    !RANDOM NO. TABLE    //TOGETHER ZZZ

    C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
                        ! STATISTICS TABLE FOR CONTROLLER ERRORS
    MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
                        ! MSCP PACKET POOL
    IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
                        ! ADDRESS OF AN MSCP PACKET (INTERUPT PROCESSING)
    PKT_USE : vector [PKT_CNT, byte, signed],
                        ! MSCP PACKET POOL ALLOCATION TABLE
    RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
                        ! RETURN PACKET POOL
    RP_USE : vector [RP_CNT, byte, signed],
                        ! RETURN PACKET POOL ALLOCATION TABLE
    RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
    RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
```

```
                                      ! CURRENT RETURN PACKET ADDRESS
    ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
                                      ! ERROR-LOG PACKET SAVE AREA
    BUFF_ADDR : vector [MAX_BUF_CNT],                   ! TABLE OF I/O BUFFER DESCRIPTORS
    BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],      ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
    IODQ : vector [IODQ_LEN, byte],                     ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
    IODQ_IN : word,                                     ! I/O DONE QUEUE IN POINTER
    IODQ_OUT : word,                                    ! I/O DONE QUEUE OUT POINTER
    ENTRY_REASON : byte,                                ! CURRENT OPERATOR COMMAND
    EOP_FLAG : byte,                                    ! END-OF-PASS FLAG
    DUP_FLAGS : WORD,                                   !DUP FLAGS      ZZZ
    CCTLR : word,                                       ! NUMBER OF "CURRENT" CONTROLLER
    CDISK : word,                                       ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
    CUOFF : word,                                       ! CURRENT UNIT CST OFFSET
    CTLR_CNT : word,                                    ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
    DUR : vector [MAX_UNITS, byte],                     ! DROP UNIT REASON
    QIO : vector [MAX_CTLR, byte],                      ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
    FREE_MEM_ADDR,                                      ! START OF FREE MEMORY
    BYTS_PER_QIO : word,                                ! SIZE (BYTES) OF AN I/O BUFFER
    ST_CODE : word,                                     ! CURRENT STATUS CODE
    SB_CODE : word,                                     ! CURRENT SUB-CODE
    STEP : word,                                        ! CURRENT STEP IN HARD_INIT
    OF_RC : signed word,                                ! OFFSET (0 OR 2) TO READ IP OR SA
    SA_REG : word,                                      ! STORAGE FOR SA REGISTER READS AND WRITES
    CMD_TIME : word,                                    ! COMMAND TIMEOUT VALUE (IN SECONDS)
    NEX : word,                                         ! NON-EXISTENT MEMORY TRAP INDICATOR
    CRN_LOW : word,                                     ! COMMAND REF NUMBER OF LAST COMMAND SENT
    CRN_HIGH : word,                                    ! COMMAND REF NUMBER (HI ORDER)
    TEMP1 : WORD,                                       !TEMPORARY STORAGE WD USED IN BGNCLN        !ZZZ
    TEMP2 : WORD,                                       !TEMPORARY STORAGE WD USED IN BGNCLN        !ZZZ
    CREDIT_BAL : word,                                  ! CREDIT BALANCE
    NEXT_PKT_USE : byte,                                ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
    HOURS : byte,                                       ! TIME OF DAY (HOURS)
    MINUTES : byte,                                     ! TIME OF DAY (MINUTES)
    CLK_TICKS : word,                                   ! TIME OF DAY (LINE-CLOCK TICKS)
    FER0_LBN : word,                                    !LO LBN ADR OF THE "FORCED ERROR" BLOCK           ZZZ
    FER1_LBN : word,                                    !HI LBN ADR OF THE "FORCED ERROR" BLOCK           ZZZ
    CLK_PRESENT : byte,                                 ! FLAG INDICATES IF LINE-CLOCK PRESENT
    HOE_FLAG : byte,                                    ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET

    TYPER : VECTOR [MAX_UNITS, WORD],    !READ I/O COUNTER                     ZZZ MMM
    TYPEW : VECTOR [MAX_UNITS, WORD],    !WRITE I/O COUNTER                    ZZZ MMM
    BAL_IN_PROGRESS : VECTOR [MAX_UNITS, WORD],   !FLAG SET TO BALANCE I/O TYPES      ZZZ MMM
    FORCE_WR : VECTOR [MAX_UNITS, WORD],         ! MMM
    CSR_MEM : WORD,                              ! MMM
    CSR_ADD : WORD INITIAL (%o'172100'),         ! MMM
!MMM    S_PATTERN : WORD,                   !PATTERN FOR DUP WRITES            ZZZ
!MMM    S_DUPPKT : WORD,                    !DBN BYTE COUNTER                  ZZZ
    P_INDEX : SIGNED WORD,               !CURRENT MESSAGE PACKET INDEX         ZZZ
    RD_COUNT : WORD INITIAL (0),         ! NUMBER OF WINCHESTER UNITS          ZZZ
    BRLEVEL : WORD,                      !BUS REQUEST LEVEL FROM OPERATOR      ZZZ
    D_FAIL : BYTE,                       !SIGNIFIES DUP TYPE ERROR             ZZZ
    FORCED_ERROR : byte,                                ! "FORCED ERROR" DETECTED IN LAST READ
```

```
;    3169  1          FER_LBN : word,                                   ! LBN OF THE "FORCED ERROR" BLOCK
;    3170  1          FER_BC : word,                                    ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    3171  1          INIT_OCCURED : byte initial (byte (FALSE)),       ! EXERCISER INITIALIZATION COMPLETE
;    3172  1          ADDR_VECT_OK : byte initial (byte (FALSE));       ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    3173  1
;    3174  1      ERRTBL;
```

```
%sbttl 'GLOBAL TEXT SECTION'

!+
! THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
! MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
! MORE THAN ONE TEST.
!-

global bind
!
! HARDWARE DIALOG
!
    PTCH1 = uplit (%asciz'                                        '),  !AREA TO PATCH IF NEEDED     ZZZ
    PTCH2 = uplit (%asciz'                                        '),  !                            ZZZ
    PTCH3 = uplit (%asciz'                                        '),  !                            ZZZ
    PTCH4 = uplit (%asciz'                                        '),  !                            ZZZ

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

```
    SWM1   = uplit (%asciz'The remaining questions only apply to unprotected disk units'),       !ZZZ
    SWQ27  = uplit (%asciz'Manufacturing Test'),                                                 !MMM
    SWQ28  = uplit (%asciz'Enable Host Memory (MSV11-P,L,J) Parity'),                            !MMM


    NULL   = uplit (%asciz''),

!++
!  THE FOLLOWING DBMs ARE DEBUG MESSAGES, AND SHOULD BE REMOVED BEFORE
!  RELEASING THE PROGRAM.  THEY INCLUDE THE NAMES OF EACH ROUTINE, PLUS
!  FORMAT STATEMENTS FOR PRINTING OUT OTHER INFORMATION.
!--

    DBM5    = uplit (%asciz'%N%A** Drop unit %D2'),
    DBM12   = uplit (%asciz'%N%A** PROC_RETPKT: Conn ID %O6%A received'),
    DBM15   = uplit (%asciz'%N%A** Multi-drive test'),
    DBM18   = uplit (%asciz'%N%A** FATAL_ERROR: RETPKT not available'),
    DBM19   = uplit (%asciz'%N%A** FSET_UPAR: Can''t find disk %D3%A in CST %D1'),
    DBM20   = uplit (%asciz'%N%A** Bad conn ID %O6%A received from %O6'),
    DBM21   = uplit (%asciz'%N%A** Message type %O2%A received in MSCP packet'),
    DBM22   = uplit (%asciz'%N%A** SEQUEN: RETPKT not available'),
    DBM23   = uplit (%asciz'%N%A** Error in SET_CTLR_CHAR'),
    DBM25   = uplit (%asciz'%N%A** Ctlr timeout = %D3%A. seconds'),
    DBM26   = uplit (%asciz'%N%A** Error in UNIT_INIT'),
    DBM27   = uplit (%asciz'%N%A** UNIT_INIT: RETPKT has bad ENDCODE'),
    DBM28A  = uplit (%asciz'%N%A** Unit size (Lo) = %D5%A.'),
    DBM28B  = uplit (%asciz'%N%A** Unit size (Hi) = %D5%A.'),
    DBM29   = uplit (%asciz'%N%A** ACCESS: RETPKT has bad ENDCODE'),
    DBM32   = uplit (%asciz'%N%A** QIO_UNIT: CST %D1%A no unit selected'),
    DBM101  = uplit (%asciz'%N%A** Unit # is: %O6'),
    DBM104  = uplit (%asciz'%N%A** Removable disk is selected'),
    DBM105  = uplit (%asciz'%N%A** Fixed disk is selected'),
    DBM107  = uplit (%asciz'%N%A** Illegal function: %O6'),
    DBM108  = uplit (%asciz'%N%A** Command ref # %O6%A/%O6%A (Oct) not sent by Host'),
    DBM109  = uplit (%asciz'%N%A** Unknown Error Log format %O3%A received'),
!   DBM110  = uplit (%asciz'%N%A** Error-Log save area full'),
    DBM111  = uplit (%asciz'%N%A** Op-code %O3%A, End-code %O3%A for ref # %O6%A/%O6%A (8)'),
    DBM112  = uplit (%asciz'%N%A** Cmd-bc %O6%A/%O6%A Rsp-bc %O6%A/%O6%A for %O6%A/%O6%A (8)'),
    DBM120  = uplit (%asciz'%N%A** Response already received for cmd %O6%A/%O6%A (8)'),
    DBM121  = uplit (%asciz'%N%A** Failure to send command after # %O6%A/%O6%A (8)'),
    DBM123  = uplit (%asciz'%N%A** Response has Error Log(s):'),                    !MMM
    DBM125  = uplit (%asciz'%N%A** HOST MEMORY ERROR, CSR =  %O6%A'),                !MMM
    DBM126  = uplit (%asciz'%N%A** HOST MEMORY ERROR, EXTENTED CSR =  %O6%A'),       !MMM
    DBM127  = uplit (%asciz'%N%A** BAD BLOCK REPLACEMENT REQUEST FLAG SET'),         !MMM
    DBM128  = uplit (%asciz'%N%A** ERROR DURING REPLACEMENT (BAD BLOCK) FLAG SET'),  !MMM


!
! DROP UNIT MESSAGES
!
    DU_MSG = uplit (%asciz'%N%AUNIT%D2%A DROPPED - '),
    DU_RSN = uplit (
             uplit (%asciz'%AUSER COMMAND%N'),
```

```
                    uplit (%asciz'%ACONFIGURATION ERROR%N'),
                    uplit (%asciz'%AINIT ERROR%N'),
                    uplit (%asciz'%ATRANSFER LIMIT REACHED%N'),
                    uplit (%asciz'%AERROR LIMIT REACHED%N'),
                    uplit (%asciz'%AUNRECOVERABLE DRIVE ERROR%N'),
                    uplit (%asciz'%AUNRECOVERABLE CONTROLLER ERROR%N'),
                    uplit (%asciz'%AFAILED TO COME ONLINE%N'),
                    uplit (%asciz'%AFAILED TO ACCESS EITHER FIRST OR LAST TRACK DURING INIT%N'),
                    uplit (%asciz'%ADISK WRITE PROTECTED%N'),
                    uplit (%asciz'%ACOMMAND TIME OUT%N')) : vector [11],
!
! SYSTEM MESSAGES (PRINTF)
!
    MSG_01 = uplit (%asciz'%N%APOWER DELAY - WAITING'),
    MSG_02 = uplit (%asciz'%N%AFUNCTIONAL TEST STARTED'),
    MSG_03 = uplit (%asciz'%N%N%AEXERCISER STARTED%N'),
!
! REPORT MESSAGES (PRINTS)
!
    RPT1  = uplit (%asciz'%N%N%AUNT DSK%S8%A# OF    # BYTES   # OF     # BYTES'),
    RPT2  = uplit (%asciz'%A  --HARD ERRORS-- --SOFT ERRORS--'),
    RPT3  = uplit (%asciz'%N%A #    #  TYPE  READS      READ   WRITES   WRITTEN'),
    RPT4  = uplit (%asciz'%A  SEK DAT DRV HST SEK DAT DRV HST'),
    RPT5  = uplit (%asciz'%N%A--- --- ----  -----  --------- ------  ---------'),
    RPT6  = uplit (%asciz'%A --- --- --- --- --- --- --- ---'),
    RPT7  = uplit (%asciz'%N%D2%D4%S2%T'),
    RPT8  = uplit (%asciz'%D4%Z3%D3%A,%Z3%A,%Z3'),
    RPT9  = uplit (%asciz'%D4%D4%D4%D4%D4%D4%D4%D4'),
    RPT10 = uplit (%asciz'%N%A .    .   CNTR           .  .........   .  .........'),
    RPT11 = uplit (%asciz'%A     .    .%D4%A    .    .    .%D4%A    .'),
    RPT12 = uplit (%asciz'%A     .    .%D4%A    .    .    .%D4%A    .'),
    RPT13 = UPLIT(%ASCIZ'%N%N%AUNIT  DISK              # OF    # BLKS       # OF    # BLKS '),
    RPT14 = UPLIT(%ASCIZ'%N%A  #      #     TYPE   PEADS     READ      WRITES  WRITTEN '),
    RPT15 = UPLIT(%ASCIZ'%N%A----  ----  -------  ------   ------     ------   ------  '),
    RPT16 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBN I/O  %D6%S3%D6%S5%D6%S3%D6'),
!ZZZ    RPT17  = uplit (%asciz'%N%D2%D4%A  RD52'),
!ZZZ    RPT18 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBNRD52  %D6%S3%D6%S5%D6%S3%D6'),
!ZZZ    RPT19  = uplit (%asciz'%N%D2%D4%A  ????'),

!
! GENERAL ERROR MESSAGES
!
!       SYSTEM FATAL (ERRSF)
!
    EGS_01 = uplit (%asciz'TOO MANY UNITS'),
    EGS_02 = uplit (%asciz'NOT ENOUGH FREE MEMORY FOR ALLOCATING READ/WRITE BUFFERS'),
!
!       DRIVE FATAL (ERRDF)
!
    EGD_10 = uplit (%asciz'REGISTER EXISTENCE TEST FAILED'),
    EGD_11 = uplit (%asciz'VECTOR TEST FAILED'),
    EGD_12 = uplit (%asciz'BR LEVEL TEST FAILED'),
    EGD_13 = uplit (%asciz'INIT SEQUENCE FAILED'),
```

```
;     1            EGD_14 = uplit (%asciz'FATAL CONTROLLER ERROR'),
;     1            EGD_15 = uplit (%asciz'ONLINE FAILED'),
;     1            EGD_16 = uplit (%asciz'WRITE-PROTECT CONFLICT'),
;     1            EGD_17 = uplit (%asciz'ACCESS FAILED'),
;     1            EGD_18 = uplit (%asciz'FATAL I/O ERROR'),
;     1        !   EGD_19 = uplit (%asciz'CONTROLLER TIMEOUT'),
;     1            EGD_19 = uplit (%asciz'DISK TYPE UNKNOWN TO EXERCISER'),
;     1            EGD_20 = uplit (%asciz'FAILED TO SEND SET-CONTROLLER-CHARACTERISTICS COMMAND'),
;     1            EGD_21 = uplit (%asciz'SET-CONTROLLER-CHARACTERISTICS RESPONSE HAS BAD ENDCODE OR FLAGS IN ERROR'),
;     1            EGD_22 = uplit (%asciz'FAILED TO SEND ON-LINE COMMAND'),
;     1            EGD_23 = uplit (%asciz'ON-LINE RESPONSE HAS BAD ENDCODE'),
;     1            EGD_24 = uplit (%asciz'ON-LINE RESPONSE HAS UNKNOWN DEVICE'),
;     1        !
;     1        !       HARD or SOFT (ERRHRD or ERRSOFT)
;     1        !
;     1            EGH_30 = uplit (%asciz'I/O REQUEST FAILED'),
;     1        !
;     1        ! BASIC ERROR MESSAGES (PRINTB)
;     1        !
;     1        !       SYSTEM FATAL (ERRSF)
;     1        !
;     1            EBS_01 = uplit (%asciz'%AMORE THAN %D2%A UNITS SPECIFIED'),
;     1        !
;     1        !       DRIVE FATAL (ERRDF)
;     1        !
;     1            EBD_10 = uplit (%asciz'%A* NO RESPONSE AT ADDRESS %O6'),
;     1            EBD_12 = uplit (%asciz'%A* INCORRECT BR LEVEL FOR DRIVE %O6'),
;     1            EBD_13 = uplit (%asciz'%A* STEP %D1%A READ ERROR'),
;     1            EBD_14 = uplit (%asciz'%A* BAD SA CODE FROM DRIVE %O6'),
;     1            EBD_18 = uplit (%asciz'%A* DISK%D2%A WENT OFFLINE'),
;     1            EBD_19 = uplit (%asciz'%A* DRIVE %O6%A NOT PROCESSING COMMAND PACKETS'),
;     1            EBD_24 = uplit (%asciz'%A* DISK%D2%A WENT TO THE "AVAILABLE" STATE'),
;     1        !
;     1        !
;     1        !       HARD or SOFT (ERRHRD or ERRSOFT)
;     1        !
;     1       EH_0 = UPLIT (%ASCIZ' - UNRECOGNIZED MESSAGE TYPE'),              !ZZZ
;     1       EH_1 = UPLIT (%ASCIZ' - UNRECOGNIZED CONNECTION ID'),             !ZZZ
;     1       EH_2 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN MESSAGE'),            !ZZZ
;     1       EH_3 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN PACKET'),             !ZZZ
;     1       EH_4 = UPLIT (%ASCIZ' - UNRECOGNIZED CRN'),                       !ZZZ
;     1       EH_5 = UPLIT (%ASCIZ' - UNRECOGNIZED OPCODE'),                    !ZZZ
;     1       EH_6 = UPLIT (%ASCIZ' - MSCP STATUS CODE ERR'),                   !ZZZ
;     1       EH_7 = UPLIT (%ASCIZ' - DUP STATUS CODE ERR'),                    !ZZZ
;     1       EH_8 = UPLIT (%ASCIZ' - UNRECOGNIZED STATUS CODE'),               !ZZZ
;     1       EH_9 = UPLIT (%ASCIZ' - LBN HOST COMPARE ERR'),                   !ZZZ
;     1       EH_10 = UPLIT (%ASCIZ' - DBN HOST COMPARE ERR'),                  !ZZZ
;     1       EH_12 = UPLIT (%ASCIZ' - UNABLE TO LOAD DUP MEDIA'),              !ZZZ
;     1       EH_13 = UPLIT (%ASCIZ' - ERR IN DUP PKT WHEN USING CTLR LC PROG'), !ZZZ
;     1       !
;     1           ERR_00  = uplit (%asciz'%A* DISK%D2'),
;     1           ERR_COD = uplit (
;     1                     uplit (%asciz'%AINVALID COMMAND'),
```

```
                        uplit (%asciz'%ACOMMAND ABORTED'),
                        uplit (%asciz'%AUNIT OFFLINE'),
                        uplit (%asciz'%ATRANSITION TO AVAILABLE STATE'),
                        uplit (%asciz'%AMEDIA FORMAT ERROR'),
                        uplit (%asciz'%AWRITE-PROTECTED'),
                        uplit (%asciz'%ADEVICE COMPARE ERROR'),
                        uplit (%asciz'%ADATA ERROR'),
                        uplit (%asciz'%AHOST BUFFER ACCESS ERROR'),
                        uplit (%asciz'%ACONTROLLER ERROR'),
                        uplit (%asciz'%ADRIVE ERROR'),
                        uplit (%asciz'%AMESSAGE FROM INTERNAL DIAGNOSTICS'),
                        uplit (%asciz'%AHOST COMPARE ERROR'),
                        uplit (%asciz'%ACOMMAND TIMEOUT'),
                        uplit (%asciz'%ABAD BLOCK REPLACEMENT COMPLETION')) : vector [15],        !MMM
        !
        !       ERROR LOG MESSAGE (ERRSOFT)
        !
            ELG_00  = uplit (%asciz'%AERROR LOG MESSAGE RECEIVED:%N'),
            ELG_FMT = uplit (
                        uplit (%asciz'%A* CONTROLLER ERROR%N'),
                        uplit (%asciz'%A* HOST MEMORY ACCESS ERROR%N'),
                        uplit (%asciz'%A* DISK%D2%A - DISK TRANSFER ERROR%N'),
                        uplit (%asciz'%A* DISK%D2%A - "STANDARD DISK INTERCONNECT" ERROR%N'),
                        uplit (%asciz'%A* DISK%D2%A - "SMALL DISK" ERROR%N'),
                        uplit (%asciz'%A* DISK%D2%A - "BAD BLOCK REPLACEMENT ATTEMPT"%N')) : vector [6],     !MMM
        !
        ! EXTENDED ERROR MESSAGES (PRINTX)
        !
            EX_SA   = uplit (%asciz'%N%A* SA: %O6'),
            EX_SC   = uplit (%asciz'%N%A* STATUS CODE: %O2'),
            EX_SBO  = uplit (%asciz'%O4'),
            EX_SB   = uplit (%asciz'%N%A* SUB_CODE: '),
            EX_CMD  = uplit (%asciz'%N%A* COMMAND: '),
            EX_RD   = uplit (%asciz'%AREAD'),
            EX_WRT  = uplit (%asciz'%AWRITE'),
            EX_CMP  = uplit (%asciz'%A-COMPARE'),
            EX_ONL  = uplit (%asciz'%AONLINE'),
            EX_ACC  = uplit (%asciz'%AACCESS'),
            EX_OP   = uplit (%asciz'%O3'),
        !ZZZ    EX_BB   = uplit (%asciz'%N%A* BAD BLOCK (Host replaceable): %D5%A. (OCT %O6%A)'),
        !ZZZ    EX_BB1  = uplit (%asciz'%N%A* 1st BAD BLOCK (Host replaceable): %D5%A. (OCT %O6%A)'),
        !ZZZ    EX_BBU  = uplit (%asciz'%N%A* BAD BLOCK REPORTED (Replaced): %D%A. (OCT %O6%A)'),
        !ZZZ    EX_LBN  = uplit (%asciz'%N%A* LBN: %D5%A. (OCT %O6%A)'),
        !ZZZ    EX_PBN  = uplit (%asciz'%N%A* PBN: %D5%A. (OCT %O6%A)'),
        !ZZZ    EX_LBR  = uplit (%asciz'%N%A* LBN: (READ) %D5%A. (OCT %O6%A)'),
        !ZZZ    EX_LBW  = uplit (%asciz'%N%A* LBN: (WRITE) %D5%A. (OCT %O6%A)'),
        !ZZZ    EX_RBN  = uplit (%asciz'%N%A* REPLACEMENT BLOCK NO. %D5%A. (OCT %O6%A)'),
            EX_BB2  = uplit (%asciz'%N%A* BAD BLOCK: %O6%A %O6%A  (OCTAL)'),                  !ZZZ
            EX_BB12 = uplit (%asciz'%N%A* 1ST BAD BLOCK: %O6%A %O6%A  (OCTAL)'),              !ZZZ
            EX_BBU2 = uplit (%asciz'%N%A* BAD BLOCK REPLACED: %O6%A %O6%A  (OCTAL)'),         !ZZZ
            EX_LBN2 = uplit (%asciz'%N%A* LBN: %O6%A %O6%A  (OCTAL)'),                        !ZZZ
            EX_PBN2 = uplit (%asciz'%N%A* PBN: %O6%A %O6%A  (OCTAL)'),                        !ZZZ
            EX_LBR2 = uplit (%asciz'%N%A* LBN READ: %O6%A %O6%A  (OCTAL)'),                   !ZZZ
```

```
EX_LBW2 = uplit (%asciz'%N%A* LBN WRITTEN: %O6%A %O6%A  (OCTAL)'),          !ZZZ
EX_RBN2 = uplit (%asciz'%N%A* RBN: %O6%A %O6%A  (OCTAL)'),                   !ZZZ
EX_CBC = uplit (%asciz'%N%A* BYTE COUNT IN COMMAND: %D5%A.'),
EX_CBR = uplit (%asciz'%N%A* BYTE COUNT IN READ COMMAND: %D5%A.'),
EX_CBW = uplit (%asciz'%N%A* BYTE COUNT IN WRITE COMMAND: %D5%A.'),
EX_BC  = uplit (%asciz'%N%A* ACTUAL # OF BYTES TRANSFERRED: %D5%A.'),
EX_BD  = uplit (%asciz'%N%A* I/O BUFFER ADDRESS (32 bits): %O6%A %O6'),
EX_BDR = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR READ (32 bits): %O6%A %O6'),
EX_BDW = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR WRITE (32 bits): %O6%A %O6'),
EX_RP  = uplit (%asciz'%N%ACONTENTS OF COMMAND/RESPONSE PACKET SAVE AREA:%N'),
EX_WRD = uplit (%asciz'%A %O6'),
EX_TIM = uplit (%asciz'%N%ATIME: %Z2%A:%Z2%A HOURS%N'),                      !MMM
!  EX_DUP = uplit (%asciz'%N%A* DUP FATAL TYPE MESSAGE, ERROR CODE: %O4%A (OCTAL)'),  !MMM
EX_LB = uplit (%asciz'%N%A* BAD LBN: %O6%A %O6%A  (OCTAL)'),                 !MMM


XX13 = UPLIT (%ASCIZ'%N%A * DISK : %D2'),                                   !ZZZ
XX23 = UPLIT (%ASCIZ'%N%ADBN: %D5%A. (OCT %O6%A)'),                        !ZZZ
XX32 = UPLIT (%ASCIZ'%N%ABYTE NUMBER: %D3'),                                !ZZZ
XX33 = UPLIT (%ASCIZ'%N%ARANDOM WRITTEN WORD :%B16'),                       !ZZZ
XX34 = UPLIT (%ASCIZ'%N%ARANDOM READ WORD bin:%B16%A oct:%O6'),            !ZZZ

!
! CONFIGURATION ERROR MESSAGES (PRINTF)
!
CER_01 = uplit (%asciz'%N%ADUPLICATE UNIT:%D2%A AT IP: %O6'),
CER_02 = uplit (%asciz'%N%AMORE THAN %D1%A DIFFERENT IP ADDRESSES'),
!
! ERROR/EVENT SUB CODES (PRINTX)
!
SC_SDI = uplit (%asciz'%ASPIN-DOWN IGNORED'),
SC_CON = uplit (%asciz'%ASTILL CONNECTED'),
SC_DUP = uplit (%asciz'%ADUPLICATE UNIT NUMBER'),
SC_ONL = uplit (%asciz'%AALREADY ONLINE'),
SC_SON = uplit (%asciz'%ASTILL ONLINE'),
SC_INR = uplit (%asciz'%AINCOMPLETE REPLACEMENT'),                          !MMM
SC_INV = uplit (%asciz'%AINVALID RCT'),                                     !MMM
SC_UNK = uplit (%asciz'%AUNIT UNKNOWN OR ONLINE TO ANOTHER CONTROLLER'),
SC_VOL = uplit (%asciz'%ANO VOLUME MOUNTED OR DRIVE DISABLED BY SWITCH'),
SC_IOP = uplit (%asciz'%AUNIT INOPERATIVE (RD51/52 write fault)'),
SC_DIS = uplit (%asciz'%AUNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTICS'),
SC_FER = uplit (%asciz'%A"FORCED ERROR" DETECTED WHILE ACCESSING FCT OR RCT'),
SC_FE2 = uplit (%asciz'%ASECTOR HAD BEEN WRITTEN WITH "FORCED ERROR" MODIFIER'),
SC_ISH = uplit (%asciz'%AFCT OR RCT UNREADABLE - INVALID SECTOR HEADER'),
SC_IS2 = uplit (%asciz'%AHEADER COMPARE ERROR (Valid header not found)'),
SC_DST = uplit (%asciz'%AFCT OR RCT UNREADABLE - DATA SYNC TIMEOUT'),
SC_DS2 = uplit (%asciz'%ADATA SYNC NOT FOUND (Data sync timeout)'),
SC_ECC = uplit (%asciz'%AFCT OR RCT UNREADABLE - UNCORRECTABLE ECC ERROR'),
SC_ECD = uplit (%asciz'%AUNCORRECTABLE ECC ERROR'),
SC_RCT = uplit (%asciz'%ARCT CORRUPTED'),
SC_FUL = uplit (%asciz'%ANO REPLACEMENT BLOCK AVAILABLE (RCT full)'),
SC_576 = uplit (%asciz'%ADISK NOT FORMATTED WITH 512 BYTE SECTORS'),
SC_FCT = uplit (%asciz'%ADISK NOT FORMATTED OR FCT CORRUPTED'),
```

```
	SC_EC1 = uplit (%asciz'%AONE SYMBOL ECC ERROR'),
	SC_EC2 = uplit (%asciz'%ATWO SYMBOL ECC ERROR'),
	SC_EC3 = uplit (%asciz'%ATHREE SYMBOL ECC ERROR'),
	SC_EC4 = uplit (%asciz'%AFOUR SYMBOL ECC ERROR'),
	SC_EC5 = uplit (%asciz'%AFIVE SYMBOL ECC ERROR'),
	SC_EC6 = uplit (%asciz'%ASIX SYMBOL ECC ERROR'),
	SC_EC7 = uplit (%asciz'%ASEVEN SYMBOL ECC ERROR'),
	SC_EC8 = uplit (%asciz'%AEIGHT SYMBOL ECC ERROR'),
	SC_EC9 = uplit (%asciz'%ACORRECTABLE ERROR IN ECC FIELD'),
	SC_SWP = uplit (%asciz'%AUNIT SOFTWARE WRITE PROTECTED'),
	SC_HWP = uplit (%asciz'%AUNIT HARDWARE WRITE PROTECTED'),
	SC_SAF = uplit (%asciz'%AUNIT DATA SAFETY WRITE PROTECTED'),	!MMM
	SC_ODA = uplit (%asciz'%AODD TRANSFER ADDRESS'),
	SC_ODB = uplit (%asciz'%AODD BYTE COUNT'),
	SC_NXM = uplit (%asciz'%ANON-EXISTENT HOST MEMORY'),
	SC_PAR = uplit (%asciz'%AHOST MEMORY PARITY ERROR'),
	SC_CTO = uplit (%asciz'%ACOMMAND TIMOUT OR RETRY LIMIT EXCEEDED'),
	SC_SDS = uplit (%asciz'%ASERIALIZER/DESERIALIZER OVERRUN OR UNDERRUN'),
	SC_EDC = uplit (%asciz'%A"ERROR DETECTION CODE" ERROR'),
	SC_IDS = uplit (%asciz'%AINCONSISTENT INTERNAL DATA STRUCTURE'),
	SC_SRT = uplit (%asciz'%ADRIVE COMMAND TIMEOUT (No response or seek incomplete)'),
	SC_SRI = uplit (%asciz'%ACONTROLLER DETECTED TRANSMISSION OR PROTOCOL ERROR'),
	SC_POE = uplit (%asciz'%APOSITION ERROR (Mis-seek)'),
	SC_RDY = uplit (%asciz'%ALOST READ/WRITE READY DURING/BETWEEN TRANSFERS'),
	SC_CLK = uplit (%asciz'%ADRIVE CLOCK DROPOUT'),
	SC_RSP = uplit (%asciz'%ALOST RECEIVER READY BETWEEN SECTORS'),
	SC_SUR = uplit (%asciz'%ADRIVE DETECTED ERROR'),
	SC_PSP = uplit (%asciz'%ACONTROLLER DETECTED PULSE OR STATE PARITY ERROR'),
	SC_REP = uplit (%asciz'%ABAD BLOCK SUCCESSFULLY REPLACED'),		!MMM
	SC_NBB = uplit (%asciz'%ABLOCK VERIFIED OK -- NOT A BAD BLOCK'),	!MMM
	SC_REF = uplit (%asciz'%AREPLACEMENT FAILURE'),			!MMM
!
! CONTROLLER GENERIC ERROR CODES
!
	CNTR_ERR = uplit (
		uplit (%asciz'%ACONTROLLER TIMEOUT'),
		uplit (%asciz'%AENVELOPE/PACKET READ ERROR (Parity or timeout)'),
		uplit (%asciz'%AENVELOPE/PACKET WRITE ERROR (Parity or timeout)'),
		uplit (%asciz'%ACONTROLLER ROM AND RAM PARITY ERROR'),
		uplit (%asciz'%ACONTROLLER RAM PARITY ERROR'),
		uplit (%asciz'%ACONTROLLER ROM PARITY ERROR'),
		uplit (%asciz'%ARING READ ERROR (Parity or timeout)'),
		uplit (%asciz'%ARING WRITE ERROR (Parity or timeout)'),
		uplit (%asciz'INTERRUPT MASTER FAILURE'),
		uplit (%asciz'%AHOST ACCESS TIMEOUT (Higher level protocol dependent)'),
		uplit (%asciz'%ACREDIT LIMIT EXCEEDED'),
		uplit (%asciz'%AQ-BUS MASTER ERROR'),
		uplit (%asciz'%ACONTROLLER FATAL ERROR'),
		uplit (%asciz'%AINSTRUCTION LOOP TIMEOUT'),
		uplit (%asciz'%AILLEGAL VIRTUAL CIRCUIT ID'),
		uplit (%asciz'%AINTERRUPT VECTOR ILLEGAL'),
		uplit (%asciz'%AMAINTENANCE READ/WRITE INVALID REGION IDENTIFIER'),
		uplit (%asciz'%AMAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER'),
```

```
;  1                      uplit (%asciz'%ACONTROLLER RAM ERROR (Non-parity)'),
;  1                      uplit (%asciz'%AINIT SEQUENCE ERROR'),
;  1                      uplit (%asciz'%AHIGHER LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
;  1                      uplit (%asciz'%APURGE/POLL HARDWARE FAILURE'),
;  1                      uplit (%asciz'%AMAPPING REGISTER READ FAILURE (Parity or timeout)')) : vector [23],
;  1        !
;  1        ! RD/RX CONTROLLER DEPENDENT ERRORS CODES
;  1        !
;  1           RDRX_ERR = uplit (
;  1                      uplit (%asciz'%AT11 CPU FAILURE'),
;  1                      uplit (%asciz'%ANON-PARITY RAM ERROR'),
;  1                      uplit (%asciz'%ASTATE MACHINE FAILURE - T11 ADDRESS REGISTER'),
;  1                      uplit (%asciz'%ASTATE MACHINE FAILURE - Q-BUS ADDRESS REGISTER'),
;  1                      uplit (%asciz'%ASTATE MACHINE FAILURE - CRC REGISTER'),
;  1                      uplit (%asciz'%ASTATE MACHINE FAILURE - SERIALIZER/DESERIALIZER REGISTER'),
;  1                      uplit (%asciz'%ASTATE MACHINE FAILURE - WRONG HARDWARE VERSION')) : vector [7],
;  1        !
;  1        ! PRINTOUTS THAT FAKE THE DRS ERROR MESSAGES
;  1        !
;  1           DF_MSG  = uplit (%asciz'%N%AZRQA DEV FTL  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %O6'),
;  1           HRD_MSG = uplit (%asciz'%N%AZRQA HRD ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %O6'),
;  1           SFT_MSG = uplit (%asciz'%N%AZRQA SFT ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %O6%N'),
;  1           HRD_SUB = uplit (%asciz'%N%AI/O REQUEST FAILED%N'),
;  1
;  1
;  1        !
;  1        ! MISCELLANEOUS
;  1        !
;  1           SPACE4   = uplit (%asciz'%S4'),
;  1           CRLF     = uplit (%asciz'%N'),
;  1           DASH     = uplit (%asciz'%A - '),
;  1           ASTERISK = uplit (%asciz'%A* ');
```

```
%sbttl 'DEFAULT HARDWARE P-TABLE'

!+
! THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
! THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
! IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES.
! AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
!-

BGNHW (DFPTBL);

global
    HWPT_IP_ADDR : word initial (INIT_IP_ADDR),         ! IP ADDRESS
    HWPT_VECTOR : word initial (INIT_INTR_VECT),        ! VECTOR ADDRESS
    HWPT_BR_LEVEL : word initial (INIT_BR_LEVEL),       ! BR LEVEL
    HWPT_DISK : WORD INITIAL (%o'000200'),              !PROTECT, WHOLE DISK, NO DUP   ZZZ
                                                        ! DK 0        ZZZ
    HWPTS0_LBN : word initial (0),                      ! STARTING TRACK LO ZZZ
    HWPTS1_LBN : word initial (0),                      ! STARTING TRACK HI ZZZ
    HWPTE0_LBN : word initial (%o'177777'),             ! ENDING TRACK LO   ZZZ
    HWPTE1_LBN : word initial (0),                      ! ENDING TRACK HI   ZZZ
    NAME_LO    : WORD INITIAL (%o'020040'),             !DISK TYPE          ZZZ
    NAME_HI    : WORD INITIAL (%o'020040');             !DISK TYPE          ZZZ

ENDHW;
```

```
%sbttl 'SOFTWARE P-TABLE'

!+
! THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
! PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
! SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
! AT RUN TIME.
!-

BGNSW (SFPTBL);

global
    SWP_ERROR : word initial (32),                                             ! HARD ERROR LIMIT FOR DROPPING UNIT
    SWP_XFER : WORD INITIAL (0),                                               !XFER LIMIT. DEFAULT = QUICK PASS       !ZZZ
    SWP_FLAGS : word initial (SWF_RDM or SWF_CRC or SWF_HWC or SWF_FER         ! FLAGS (SEE DOCUMENTATION)            !ZZZ
                              or SWF_HRD or SWF_BLK),                          !                                      !ZZZ

    SWP_DPAT : word initial (0),                                               ! DATA PATTERN NUMBER
    SWP_RAT : word initial (99),                                               ! RD51/52 OPERATION RATIO
    SWP_TIME : word initial (0),                                               ! START TIME (HHMM)
!MMM    DUPROUND : WORD INITIAL (11),                 !NO OF I/Os PER DBN TEST ZZZ

!       THE NEXT TWO LOCATIONS SHOULD BE TOGETHER

    SWP_UCNT : word initial (MAX_UDP_CNT),                                     ! USER DATA PATTERN COUNT
    SWP_UDPAT : vector [MAX_UDP_CNT, word],                                    ! USER DATA PATTERN

    DUPROUND : WORD INITIAL (11),                 !NO OF I/Os PER DBN TEST ZZZ
    MAN_TST : word initial (0),                                   ! Reduce Seek Duty cycle for Manufacturing MMM
    TST_PAR : word initial (1);                                   ! Enable Host Memory Parity MMM

ENDSW;
```

```
;     3634  1      %sbttl 'PROTECTION TABLE'
;     3635  1
;     3636  1      !+
;     3637  1      ! THIS TABLE IS USED BY THE RUNTIME SERVICES
;     3638  1      ! TO PROTECT THE LOAD MEDIA.
;     3639  1      !-
;     3640  1
;     3641  1      BGNPROT (0, -1, 6);
;     3642  1
;     3643  1      !1ST ARG =       OFFSET INTO P-TABLE FOR CSR ADDRESS
;     3644  1      !2ND ARG =       OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;     3645  1      !3RD ARG =       OFFSET INTO P-TABLE FOR DRIVE NUMBER
;     3646  1
;     3647  1      ENDPROT;
;     3648  1      end
;     3649  1
;     3650  0      eludom

                                                  .TITLE  ZRQAM1 RD/RX EXERCISER
                                                  .IDENT  /V02.3/
                                                  .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     110                                 .ASCII  /H/
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$SOFT
000022  023764'                        L$HPTP::.WORD   L$HW
000024  024014'                        L$SPTP::.WORD   L$SW
000026  000000G                        L$LADP::.WORD   L$LAST
000030  000000                         L$STA:: .WORD   0
000032  000000                         L$CO::  .WORD   0
000034  000001                         L$DTYP::.WORD   1
000036  000000                         L$APT:: .WORD   0
000040  000124'                        L$DTP:: .WORD   L$DISPATCH
000042  000000                         L$PRIO::.WORD   0
000044  000000                         L$ENVI::.WORD   0
000046  000000                         L$EXP1::.WORD   0
000050                                 L$MREV::
000050     004                                 .BYTE   4
000051     000                                 .BYTE   0
000052  000000                         L$EF::  .WORD   0
```

```
000054  000000                          .WORD   0
000056  000000              L$SPC::     .WORD   0
000060  000000G             L$DEVP::.WORD   L$DVTYP
000062  000000G             L$REPP::.WORD   L$RPT
000064  000000              L$EXP4::.WORD   0
000066  000000              L$EXP5::.WORD   0
000070  000000G             L$AUT::   .WORD  L$AU
000072  000000G             L$DUT::   .WORD  L$DU
000074  000000              L$LUN::   .WORD  0
000076  000000G             L$DESP::.WORD   L$DESC
000100  104035              L$LOAD::.WORD   -73743
000102  000126'             L$ETP::   .WORD  L$ERRTBL
000104  000000G             L$ICP::   .WORD  L$INIT
000106  000000G             L$CCP::   .WORD  L$CLEAN
000110  000000G             L$ACP::   .WORD  L$AUTO
000112  024102'             L$PRT::   .WORD  L$PROT
000114  000001              L$TEST::.WORD   1
000116  000000              L$DLY::   .WORD  0
000120  000000              L$HIME::.WORD   0
000122  000001              D$PCNT::.WORD   1
000124  000000G             L$DISPATCH::
                                        .WORD   T1
000126                      ERRTYP::.BLKW   1
000130                      ERRNBR::.BLKW   1
000132                      ERRMSG::.BLKW   1
000134                      ERRBLK::.BLKW   1
000136    040   040   040   P.AAA:  .ASCII  /   /
000141    040   040   040           .ASCII  /   /
000144    040   040   040           .ASCII  /   /
000147    040   040   040           .ASCII  /   /
000152    040   040   040           .ASCII  /   /
000155    040   040   040           .ASCII  /   /
000160    040   040   040           .ASCII  /   /
000163    040   040   040           .ASCII  /   /
000166    040   040   040           .ASCII  /   /
000171    040   040   040           .ASCII  /   /
000174    040   040   040           .ASCII  /   /
000177    040   040   040           .ASCII  /   /
000202    040   040   040           .ASCII  /   /
000205    040   040   000           .ASCII  /  /<00>
000210    040   040   040   P.AAB:  .ASCII  /   /
000213    040   040   040           .ASCII  /   /
000216    040   040   040           .ASCII  /   /
000221    040   040   040           .ASCII  /   /
000224    040   040   040           .ASCII  /   /
000227    040   040   040           .ASCII  /   /
000232    040   040   040           .ASCII  /   /
000235    040   040   040           .ASCII  /   /
000240    040   040   040           .ASCII  /   /
000243    040   040   040           .ASCII  /   /
000246    040   040   040           .ASCII  /   /
000251    040   040   040           .ASCII  /   /
000254    040   040   040           .ASCII  /   /
```

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

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

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

```
001177   123   124   117              .ASCII  /STO/
001202   115   105   122              .ASCII  /MER/
001205   040   104   101              .ASCII  / DA/
001210   124   101   040              .ASCII  /TA /
001213   101   122   105              .ASCII  /ARE/
001216   101   040   115              .ASCII  /A M/
001221   101   131   040              .ASCII  /AY /
001224   102   105   040              .ASCII  /BE /
001227   117   126   105              .ASCII  /OVE/
001232   122   127   122              .ASCII  /RWR/
001235   111   124   124              .ASCII  /ITT/
001240   105   116   041              .ASCII  /EN!/
001243   040   056   056              .ASCII  / ../
001246   056   040   103              .ASCII  /. C/
001251   117   116   106              .ASCII  /ONF/
001254   111   122   115              .ASCII  /IRM/
001257   000                          .ASCII  <00>
001260   110   141   162     P.AAQ:   .ASCII  /Har/
001263   144   040   145              .ASCII  /d e/
001266   162   162   157              .ASCII  /rro/
001271   162   040   154              .ASCII  /r l/
001274   151   155   151              .ASCII  /imi/
001277   164   000   000              .ASCII  /t/<00><00>
001302   124   162   141     P.AAR:   .ASCII  /Tra/
001305   156   163   146              .ASCII  /nsf/
001310   145   162   040              .ASCII  /er /
001313   154   151   155              .ASCII  /lim/
001316   151   164   040              .ASCII  /it /
001321   151   156   040              .ASCII  /in /
001324   155   145   147              .ASCII  /meg/
001327   141   142   171              .ASCII  /aby/
001332   164   145   163              .ASCII  /tes/
001335   040   050   060              .ASCII  / (0/
001340   040   146   157              .ASCII  / fo/
001343   162   040   161              .ASCII  /r q/
001346   165   151   143              .ASCII  /uic/
001351   153   040   160              .ASCII  /k p/
001354   141   163   163              .ASCII  /ass/
001357   051   000   000              .ASCII  /)/<00><00>
001362   122   141   156     P.AAS:   .ASCII  /Ran/
001365   144   157   155              .ASCII  /dom/
001370   040   163   145              .ASCII  / se/
001373   145   153   040              .ASCII  /ek /
001376   155   157   144              .ASCII  /mod/
001401   145   000   000              .ASCII  /e/<00><00>
001404   122   145   141     P.AAT:   .ASCII  /Rea/
001407   144   055   143              .ASCII  /d-c/
001412   157   155   160              .ASCII  /omp/
001415   141   162   145              .ASCII  /are/
001420   163   040   160              .ASCII  /s p/
001423   145   162   146              .ASCII  /erf/
001426   157   162   155              .ASCII  /orm/
001431   145   144   040              .ASCII  /ed /
```

```
001434  141  164  040          .ASCII  /at /
001437  164  150  145          .ASCII  /the/
001442  040  143  157          .ASCII  / co/
001445  156  164  162          .ASCII  /ntr/
001450  157  154  154          .ASCII  /oll/
001453  145  162  000          .ASCII  /er/<00>
001456  127  162  151  P.AAU:  .ASCII  /Wri/
001461  164  145  055          .ASCII  /te-/
001464  143  157  155          .ASCII  /com/
001467  160  141  162          .ASCII  /par/
001472  145  163  040          .ASCII  /es /
001475  160  145  162          .ASCII  /per/
001500  146  157  162          .ASCII  /for/
001503  155  145  144          .ASCII  /med/
001506  040  141  164          .ASCII  / at/
001511  040  164  150          .ASCII  / th/
001514  145  040  143          .ASCII  /e c/
001517  157  156  164          .ASCII  /ont/
001522  162  157  154          .ASCII  /rol/
001525  154  145  162          .ASCII  /ler/
001530  000  000               .ASCII  <00><00>
001532  103  150  145  P.AAV:  .ASCII  /Che/
001535  143  153  040          .ASCII  /ck /
001540  141  154  154          .ASCII  /all/
001543  040  167  162          .ASCII  / wr/
001546  151  164  145          .ASCII  /ite/
001551  163  040  141          .ASCII  /s a/
001554  164  040  150          .ASCII  /t h/
001557  157  163  164          .ASCII  /ost/
001562  040  142  171          .ASCII  / by/
001565  040  162  145          .ASCII  / re/
001570  141  144  151          .ASCII  /adi/
001573  156  147  000          .ASCII  /ng/<00>
001576  125  163  145  P.AAW:  .ASCII  /Use/
001601  162  055  144          .ASCII  /r-d/
001604  145  146  151          .ASCII  /efi/
001607  156  145  144          .ASCII  /ned/
001612  040  144  141          .ASCII  / da/
001615  164  141  040          .ASCII  /ta /
001620  160  141  164          .ASCII  /pat/
001623  164  145  162          .ASCII  /ter/
001626  156  000               .ASCII  /n/<00>
001630  123  145  154  P.AAX:  .ASCII  /Sel/
001633  145  143  164          .ASCII  /ect/
001636  040  160  162          .ASCII  / pr/
001641  145  055  144          .ASCII  /e-d/
001644  145  146  151          .ASCII  /efi/
001647  156  145  144          .ASCII  /ned/
001652  040  144  141          .ASCII  / da/
001655  164  141  040          .ASCII  /ta /
001660  160  141  164          .ASCII  /pat/
001663  164  145  162          .ASCII  /ter/
001666  156  040  050          .ASCII  /n (/
```

```
001671  060  040  146            .ASCII  /0 f/
001674  157  162  040            .ASCII  /or /
001677  163  145  161            .ASCII  /seq/
001702  165  145  156            .ASCII  /uen/
001705  164  151  141            .ASCII  /tia/
001710  154  040  163            .ASCII  /l s/
001713  145  154  145            .ASCII  /ele/
001716  143  164  151            .ASCII  /cti/
001721  157  156  051            .ASCII  /on)/
001724  000  000                 .ASCII  <00><00>
001726  116  165  155   P.AAY:   .ASCII  /Num/
001731  142  145  162            .ASCII  /ber/
001734  040  157  146            .ASCII  / of/
001737  040  167  157            .ASCII  / wo/
001742  162  144  163            .ASCII  /rds/
001745  040  151  156            .ASCII  / in/
001750  040  144  141            .ASCII  / da/
001753  164  141  040            .ASCII  /ta /
001756  160  141  164            .ASCII  /pat/
001761  164  145  162            .ASCII  /ter/
001764  156  040  050            .ASCII  /n (/
001767  061  066  040            .ASCII  /16 /
001772  155  141  170            .ASCII  /max/
001775  151  155  165            .ASCII  /imu/
002000  155  051  000            .ASCII  /m)/<00>
002003  000                      .ASCII  <00>
002004  120  141  164   P.AAZ:   .ASCII  /Pat/
002007  164  145  162            .ASCII  /ter/
002012  156  040  166            .ASCII  /n v/
002015  141  154  165            .ASCII  /alu/
002020  145  040  050            .ASCII  /e (/
002023  156  157  040            .ASCII  /no /
002026  154  145  141            .ASCII  /lea/
002031  144  151  156            .ASCII  /din/
002034  147  040  172            .ASCII  /g z/
002037  145  162  157            .ASCII  /ero/
002042  163  040  141            .ASCII  /s a/
002045  154  154  157            .ASCII  /llo/
002050  167  145  144            .ASCII  /wed/
002053  051  000  000            .ASCII  /)/<00><00>
002056  103  154  145   P.ABA:   .ASCII  /Cle/
002061  141  162  040            .ASCII  /ar /
002064  163  164  141            .ASCII  /sta/
002067  164  151  163            .ASCII  /tis/
002072  164  151  143            .ASCII  /tic/
002075  141  154  040            .ASCII  /al /
002100  164  141  142            .ASCII  /tab/
002103  154  145  163            .ASCII  /les/
002106  040  141  146            .ASCII  / af/
002111  164  145  162            .ASCII  /ter/
002114  040  160  162            .ASCII  / pr/
002117  151  156  164            .ASCII  /int/
002122  151  156  147            .ASCII  /ing/
```

```
002125  000                              .ASCII  <00>
002126  120    145    162      P.ABB:   .ASCII  /Per/
002131  143    145    156               .ASCII  /cen/
002134  164    141    147               .ASCII  /tag/
002137  145    040    157               .ASCII  /e o/
002142  146    040    042               .ASCII  /f "/
002145  106    151    170               .ASCII  /Fix/
002150  145    144    040               .ASCII  /ed /
002153  104    151    163               .ASCII  /Dis/
002156  153    042    040               .ASCII  /k" /
002161  157    160    145               .ASCII  /ope/
002164  162    141    164               .ASCII  /rat/
002167  151    157    156               .ASCII  /ion/
002172  163    040    157               .ASCII  /s o/
002175  165    164    040               .ASCII  /ut /
002200  157    146    040               .ASCII  /of /
002203  164    157    164               .ASCII  /tot/
002206  141    154    040               .ASCII  /al /
002211  157    160    145               .ASCII  /ope/
002214  162    141    164               .ASCII  /rat/
002217  151    157    156               .ASCII  /ion/
002222  163    000                      .ASCII  /s/<00>
002224  125    156    151      P.ABC:   .ASCII  /Uni/
002227  164    163    040               .ASCII  /ts /
002232  164    157    040               .ASCII  /to /
002235  142    145    040               .ASCII  /be /
002240  163    145    154               .ASCII  /sel/
002243  145    143    164               .ASCII  /ect/
002246  145    144    040               .ASCII  /ed /
002251  141    164    040               .ASCII  /at /
002254  162    141    156               .ASCII  /ran/
002257  144    157    155               .ASCII  /dom/
002262  040    050    116               .ASCII  / (N/
002265  157    054    040               .ASCII  /o, /
002270  151    155    160               .ASCII  /imp/
002273  154    151    145               .ASCII  /lie/
002276  163    040    163               .ASCII  /s s/
002301  145    161    165               .ASCII  /equ/
002304  145    156    164               .ASCII  /ent/
002307  151    141    154               .ASCII  /ial/
002312  051    000                      .ASCII  /)/<00>
002314  122    145    167      P.ABD:   .ASCII  /Rew/
002317  162    151    164               .ASCII  /rit/
002322  145    040    142               .ASCII  /e b/
002325  154    157    143               .ASCII  /loc/
002330  153    163    040               .ASCII  /ks /
002333  167    150    145               .ASCII  /whe/
002336  156    040    042               .ASCII  /n "/
002341  106    157    162               .ASCII  /For/
002344  143    145    144               .ASCII  /ced/
002347  040    105    162               .ASCII  / Er/
002352  162    157    162               .ASCII  /ror/
002355  042    040    144               .ASCII  /" d/
```

```
002360  145  164  145            .ASCII  /ete/
002363  143  164  145            .ASCII  /cta/
002366  144  040  157            .ASCII  /d o/
002371  156  040  162            .ASCII  /n r/
002374  145  141  144            .ASCII  /ead/
002377  163  000  000            .ASCII  /s/<00><00>
002402  110  141  154  P.ABE:    .ASCII  /Hal/
002405  164  040  157            .ASCII  /t o/
002410  156  040  157            .ASCII  /n o/
002413  164  150  145            .ASCII  /the/
002416  162  040  150            .ASCII  /r h/
002421  141  162  144            .ASCII  /ard/
002424  040  145  162            .ASCII  / er/
002427  162  157  162            .ASCII  /ror/
002432  163  040  050            .ASCII  /s (/
002435  043  163  040            .ASCII  /#s /
002440  063  061  055            .ASCII  /31-/
002443  063  064  054            .ASCII  /34,/
002446  040  063  066            .ASCII  / 36/
002451  055  063  067            .ASCII  /-37/
002454  054  040  063            .ASCII  /, 3/
002457  071  055  064            .ASCII  /9-4/
002462  065  051  000            .ASCII  /5)/<00>
002465  000                      .ASCII  <00>
002466  110  141  154  P.ABF:    .ASCII  /Hal/
002471  164  040  157            .ASCII  /t o/
002474  156  040  163            .ASCII  /n s/
002477  157  146  164            .ASCII  /oft/
002502  040  145  162            .ASCII  / er/
002505  162  157  162            .ASCII  /ror/
002510  163  040  050            .ASCII  /s (/
002513  043  163  040            .ASCII  /#s /
002516  065  060  055            .ASCII  /50-/
002521  065  064  051            .ASCII  /54)/
002524  000  000                 .ASCII  <00><00>
002526  110  141  154  P.ABG:    .ASCII  /Hal/
002531  164  040  157            .ASCII  /t o/
002534  156  040  142            .ASCII  /n b/
002537  141  144  055            .ASCII  /ad-/
002542  142  154  157            .ASCII  /blo/
002545  143  153  040            .ASCII  /ck /
002550  150  141  162            .ASCII  /har/
002553  144  040  145            .ASCII  /d e/
002556  162  162  157            .ASCII  /rro/
002561  162  163  040            .ASCII  /rs /
002564  050  043  163            .ASCII  /(#s/
002567  040  063  065            .ASCII  / 35/
002572  054  040  063            .ASCII  /, 3/
002575  070  051  000            .ASCII  /8)/<00>
002600  105  156  164  P.ABH:    .ASCII  /Ent/
002603  145  162  040            .ASCII  /er /
002606  164  151  155            .ASCII  /tim/
002611  145  040  141            .ASCII  /e a/
```

```
002614   163   040   110               .ASCII   /s H/
002617   110   115   115               .ASCII   /HMM/
002622   040   050   145               .ASCII   / (e/
002625   170   141   155               .ASCII   /xam/
002630   160   154   145               .ASCII   /ple/
002633   072   040   061               .ASCII   /: 1/
002636   063   060   065               .ASCII   /305/
002641   051   000   000               .ASCII   /)/<00><00>
002644   122   165   156      P.ABI:   .ASCII   /Run/
002647   156   151   156               .ASCII   /nin/
002652   147   040   165               .ASCII   /g u/
002655   156   144   145               .ASCII   /nde/
002660   162   040   164               .ASCII   /r t/
002663   150   145   040               .ASCII   /he /
002666   101   056   120               .ASCII   /A.P/
002671   056   124   056               .ASCII   /.T./
002674   040   115   157               .ASCII   / Mo/
002677   156   151   164               .ASCII   /nit/
002702   157   162   000               .ASCII   /or/<00>
002705   000                           .ASCII   <00>
002706   124   150   145      P.ABJ:   .ASCII   /The/
002711   040   162   145               .ASCII   / re/
002714   155   141   151               .ASCII   /mai/
002717   156   151   156               .ASCII   /nin/
002722   147   040   161               .ASCII   /g q/
002725   165   145   163               .ASCII   /ues/
002730   164   151   157               .ASCII   /tio/
002733   156   163   040               .ASCII   /ns /
002736   157   156   154               .ASCII   /onl/
002741   171   040   141               .ASCII   /y a/
002744   160   160   154               .ASCII   /ppl/
002747   171   040   164               .ASCII   /y t/
002752   157   040   165               .ASCII   /o u/
002755   156   160   162               .ASCII   /npr/
002760   157   164   145               .ASCII   /ote/
002763   143   164   145               .ASCII   /cte/
002766   144   040   144               .ASCII   /d d/
002771   151   163   153               .ASCII   /isk/
002774   040   165   156               .ASCII   / un/
002777   151   164   163               .ASCII   /its/
003002   000   000                     .ASCII   <00><00>
003004   115   141   156      P.ABK:   .ASCII   /Man/
003007   165   146   141               .ASCII   /ufa/
003012   143   164   165               .ASCII   /ctu/
003015   162   151   156               .ASCII   /rin/
003020   147   040   124               .ASCII   /g T/
003023   145   163   164               .ASCII   /est/
003026   000   000                     .ASCII   <00><00>
003030   105   156   141      P.ABL:   .ASCII   /Ena/
003033   142   154   145               .ASCII   /ble/
003036   040   110   157               .ASCII   / Ho/
003041   163   164   040               .ASCII   /st /
003044   115   145   155               .ASCII   /Mem/
```

```
003047  157  162  171             .ASCII  /ory/
003052  040  050  115             .ASCII  / (M/
003055  123  126  061             .ASCII  /SV1/
003060  061  055  120             .ASCII  /1-P/
003063  054  114  054             .ASCII  /,L,/
003066  112  051  040             .ASCII  /J) /
003071  120  141  162             .ASCII  /Par/
003074  151  164  171             .ASCII  /ity/
003077  000                       .ASCII  <00>
003100  000  000         P.ABM:   .ASCII  <00><00>
003102  045  116  045    P.ABN:   .ASCII  /%N%/
003105  101  052  052             .ASCII  /A**/
003110  040  104  162             .ASCII  / Dr/
003113  157  160  040             .ASCII  /op /
003116  165  156  151             .ASCII  /uni/
003121  164  040  045             .ASCII  /t %/
003124  104  062  000             .ASCII  /D2/<00>
003127  000                       .ASCII  <00>
003130  045  116  045    P.ABO:   .ASCII  /%N%/
003133  101  052  052             .ASCII  /A**/
003136  040  120  122             .ASCII  / PR/
003141  117  103  137             .ASCII  /OC_/
003144  122  105  124             .ASCII  /RET/
003147  120  113  124             .ASCII  /PKT/
003152  072  040  103             .ASCII  /: C/
003155  157  156  156             .ASCII  /onn/
003160  040  111  104             .ASCII  / ID/
003163  040  045  117             .ASCII  / %O/
003166  066  045  101             .ASCII  /6%A/
003171  040  162  145             .ASCII  / re/
003174  143  145  151             .ASCII  /cei/
003177  166  145  144             .ASCII  /ved/
003202  000  000                  .ASCII  <00><00>
003204  045  116  045    P.ABP:   .ASCII  /%N%/
003207  101  052  052             .ASCII  /A**/
003212  040  115  165             .ASCII  / Mu/
003215  154  164  151             .ASCII  /lti/
003220  055  144  162             .ASCII  /-dr/
003223  151  166  145             .ASCII  /ive/
003226  040  164  145             .ASCII  / te/
003231  163  164  000             .ASCII  /st/<00>
003234  045  116  045    P.ABQ:   .ASCII  /%N%/
003237  101  052  052             .ASCII  /A**/
003242  040  106  101             .ASCII  / FA/
003245  124  101  114             .ASCII  /TAL/
003250  137  105  122             .ASCII  /_ER/
003253  122  117  122             .ASCII  /ROR/
003256  072  040  122             .ASCII  /: R/
003261  105  124  120             .ASCII  /ETP/
003264  113  124  040             .ASCII  /KT /
003267  156  157  164             .ASCII  /not/
003272  040  141  166             .ASCII  / av/
003275  141  151  154             .ASCII  /ail/
```

```
003300  141  142  154            .ASCII  /abl/
003303  145  000  000            .ASCII  /e/<00><00>
003306  045  116  045   P.ABR:   .ASCII  /%N%/
003311  101  052  052            .ASCII  /A**/
003314  040  106  123            .ASCII  / FS/
003317  105  124  137            .ASCII  /ET_/
003322  125  120  101            .ASCII  /UPA/
003325  122  072  040            .ASCII  /R: /
003330  103  141  156            .ASCII  /Can/
003333  047  164  040            .ASCII  /'t /
003336  146  151  156            .ASCII  /fin/
003341  144  040  144            .ASCII  /d d/
003344  151  163  153            .ASCII  /isk/
003347  040  045  104            .ASCII  / %D/
003352  063  045  101            .ASCII  /3%A/
003355  040  151  156            .ASCII  / in/
003360  040  103  123            .ASCII  / CS/
003363  124  040  045            .ASCII  /T %/
003366  104  061  000            .ASCII  /D1/<00>
003371  000                      .ASCII  <00>
003372  045  116  045   P.ABS:   .ASCII  /%N%/
003375  101  052  052            .ASCII  /A**/
003400  040  102  141            .ASCII  / Ba/
003403  144  040  143            .ASCII  /d c/
003406  157  156  156            .ASCII  /onn/
003411  040  111  104            .ASCII  / ID/
003414  040  045  117            .ASCII  / %O/
003417  066  045  101            .ASCII  /6%A/
003422  040  162  145            .ASCII  / re/
003425  143  145  151            .ASCII  /cei/
003430  166  145  144            .ASCII  /ved/
003433  040  146  162            .ASCII  / fr/
003436  157  155  040            .ASCII  /om /
003441  045  117  066            .ASCII  /%O6/
003444  000  000                 .ASCII  <00><00>
003446  045  116  045   P.ABT:   .ASCII  /%N%/
003451  101  052  052            .ASCII  /A**/
003454  040  115  145            .ASCII  / Me/
003457  163  163  141            .ASCII  /ssa/
003462  147  145  040            .ASCII  /ge /
003465  164  171  160            .ASCII  /typ/
003470  145  040  045            .ASCII  /e %/
003473  117  062  045            .ASCII  /O2%/
003476  101  040  162            .ASCII  /A r/
003501  145  143  145            .ASCII  /ece/
003504  151  166  145            .ASCII  /ive/
003507  144  040  151            .ASCII  /d i/
003512  156  040  115            .ASCII  /n M/
003515  123  103  120            .ASCII  /SCP/
003520  040  160  141            .ASCII  / pa/
003523  143  153  145            .ASCII  /cke/
003526  164  000                 .ASCII  /t/<00>
003530  045  116  045   P.ABU:   .ASCII  /%N%/
```

```
003533   101   052   052               .ASCII  /A**/
003536   040   123   105               .ASCII  / SE/
003541   121   125   105               .ASCII  /QUE/
003544   116   072   040               .ASCII  /N: /
003547   122   105   124               .ASCII  /RET/
003552   120   113   124               .ASCII  /PKT/
003555   040   156   157               .ASCII  / no/
003560   164   040   141               .ASCII  /t a/
003563   166   141   151               .ASCII  /vai/
003566   154   141   142               .ASCII  /lab/
003571   154   145   000               .ASCII  /le/<00>
003574   045   116   045      P.ABV:   .ASCII  /%N%/
003577   101   052   052               .ASCII  /A**/
003602   040   105   162               .ASCII  / Er/
003605   162   157   162               .ASCII  /ror/
003610   040   151   156               .ASCII  / in/
003613   040   123   105               .ASCII  / SE/
003616   124   137   103               .ASCII  /T_C/
003621   124   114   122               .ASCII  /TLR/
003624   137   103   110               .ASCII  /_CH/
003627   101   122   000               .ASCII  /AR/<00>
003632   045   116   045      P.ABW:   .ASCII  /%N%/
003635   101   052   052               .ASCII  /A**/
003640   040   103   164               .ASCII  / Ct/
003643   154   162   040               .ASCII  /lr /
003646   164   151   155               .ASCII  /tim/
003651   145   157   165               .ASCII  /eou/
003654   164   040   075               .ASCII  /t =/
003657   040   045   104               .ASCII  / %D/
003662   063   045   101               .ASCII  /3%A/
003665   056   040   163               .ASCII  /. s/
003670   145   143   157               .ASCII  /eco/
003673   156   144   163               .ASCII  /nds/
003676   000   000                     .ASCII  <00><00>
003700   045   116   045      P.ABX:   .ASCII  /%N%/
003703   101   052   052               .ASCII  /A**/
003706   040   105   162               .ASCII  / Er/
003711   162   157   162               .ASCII  /ror/
003714   040   151   156               .ASCII  / in/
003717   040   125   116               .ASCII  / UN/
003722   111   124   137               .ASCII  /IT_/
003725   111   116   111               .ASCII  /INI/
003730   124   000                     .ASCII  /T/<00>
003732   045   116   045      P.ABY:   .ASCII  /%N%/
003735   101   052   052               .ASCII  /A**/
003740   040   125   116               .ASCII  / UN/
003743   111   124   137               .ASCII  /IT_/
003746   111   116   111               .ASCII  /INI/
003751   124   072   040               .ASCII  /T: /
003754   122   105   124               .ASCII  /RET/
003757   120   113   124               .ASCII  /PKT/
003762   040   150   141               .ASCII  / ha/
003765   163   040   142               .ASCII  /s b/
```

```
003770   141   144   040              .ASCII  /ad /
003773   105   116   104              .ASCII  /END/
003776   103   117   104              .ASCII  /COD/
004001   105   000   000              .ASCII  /E/<00><00>
004004   045   116   045     P.ABZ:   .ASCII  /%N%/
004007   101   052   052              .ASCII  /A**/
004012   040   125   156              .ASCII  / Un/
004015   151   164   040              .ASCII  /it /
004020   163   151   172              .ASCII  /siz/
004023   145   040   050              .ASCII  /e (/
004026   114   157   051              .ASCII  /Lo)/
004031   040   075   040              .ASCII  / = /
004034   045   104   065              .ASCII  /%D5/
004037   045   101   056              .ASCII  /%A./
004042   000   000                    .ASCII  <00><00>
004044   045   116   045     P.ACA:   .ASCII  /%N%/
004047   101   052   052              .ASCII  /A**/
004052   040   125   156              .ASCII  / Un/
004055   151   164   040              .ASCII  /it /
004060   163   151   172              .ASCII  /siz/
004063   145   040   050              .ASCII  /e (/
004066   110   151   051              .ASCII  /Hi)/
004071   040   075   040              .ASCII  / = /
004074   045   104   065              .ASCII  /%D5/
004077   045   101   056              .ASCII  /%A./
004102   000   000                    .ASCII  <00><00>
004104   045   116   045     P.ACB:   .ASCII  /%N%/
004107   101   052   052              .ASCII  /A**/
004112   040   101   103              .ASCII  / AC/
004115   103   105   123              .ASCII  /CES/
004120   123   072   040              .ASCII  /S: /
004123   122   105   124              .ASCII  /RET/
004126   120   113   124              .ASCII  /PKT/
004131   040   150   141              .ASCII  / ha/
004134   153   040   142              .ASCII  /s b/
004137   141   144   040              .ASCII  /ad /
004142   105   116   104              .ASCII  /END/
004145   103   117   104              .ASCII  /COD/
004150   105   000                    .ASCII  /E/<00>
004152   045   116   045     P.ACC:   .ASCII  /%N%/
004155   101   052   052              .ASCII  /A**/
004160   040   121   111              .ASCII  / QI/
004163   117   137   125              .ASCII  /O_U/
004166   116   111   124              .ASCII  /NIT/
004171   072   040   103              .ASCII  /: C/
004174   123   124   040              .ASCII  /ST /
004177   045   104   061              .ASCII  /%D1/
004202   045   101   040              .ASCII  /%A /
004205   156   157   040              .ASCII  /no /
004210   165   156   151              .ASCII  /uni/
004213   164   040   163              .ASCII  /t s/
004216   145   154   145              .ASCII  /ele/
004221   143   164   145              .ASCII  /cte/
```

```
004224   144   000                     .ASCII  /d/<00>
004226   045   116   045     P.ACD:    .ASCII  /%N%/
004231   101   052   052               .ASCII  /A**/
004234   040   125   156               .ASCII  / Un/
004237   151   164   040               .ASCII  /it /
004242   043   040   151               .ASCII  /# i/
004245   163   072   040               .ASCII  /s: /
004250   045   117   066               .ASCII  /%O6/
004253   000                           .ASCII  <00>
004254   045   116   045     P.ACE:    .ASCII  /%N%/
004257   101   052   052               .ASCII  /A**/
004262   040   122   145               .ASCII  / Re/
004265   155   157   166               .ASCII  /mov/
004270   141   142   154               .ASCII  /abl/
004273   145   040   144               .ASCII  /e d/
004276   151   163   153               .ASCII  /isk/
004301   040   151   163               .ASCII  / is/
004304   040   163   145               .ASCII  / se/
004307   154   145   143               .ASCII  /lec/
004312   164   145   144               .ASCII  /ted/
004315   000                           .ASCII  <00>
004316   045   116   045     P.ACF:    .ASCII  /%N%/
004321   101   052   052               .ASCII  /A**/
004324   040   106   151               .ASCII  / Fi/
004327   170   145   144               .ASCII  /xed/
004332   040   144   151               .ASCII  / di/
004335   163   153   040               .ASCII  /sk /
004340   151   163   040               .ASCII  /is /
004343   163   145   154               .ASCII  /sel/
004346   145   143   164               .ASCII  /ect/
004351   145   144   000               .ASCII  /ed/<00>
004354   045   116   045     P.ACG:    .ASCII  /%N%/
004357   101   052   052               .ASCII  /A**/
004362   040   111   154               .ASCII  / Il/
004365   154   145   147               .ASCII  /leg/
004370   141   154   040               .ASCII  /al /
004373   146   165   156               .ASCII  /fun/
004376   143   164   151               .ASCII  /cti/
004401   157   156   072               .ASCII  /on:/
004404   040   045   117               .ASCII  / %O/
004407   066   000   000               .ASCII  /6/<00><00>
004412   045   116   045     P.ACH:    .ASCII  /%N%/
004415   101   052   052               .ASCII  /A**/
004420   040   103   157               .ASCII  / Co/
004423   155   155   141               .ASCII  /mma/
004426   156   144   040               .ASCII  /nd /
004431   162   145   146               .ASCII  /ref/
004434   040   043   040               .ASCII  / # /
004437   045   117   066               .ASCII  /%O6/
004442   045   101   057               .ASCII  /%A/<57>
004445   045   117   066               .ASCII  /%O6/
004450   045   101   040               .ASCII  /%A /
004453   050   117   143               .ASCII  /(Oc/
```

```
004456   164   051   040              .ASCII  /t) /
004461   156   157   164              .ASCII  /not/
004464   040   163   145              .ASCII  / se/
004467   156   164   040              .ASCII  /nt /
004472   142   171   040              .ASCII  /by /
004475   110   157   163              .ASCII  /Hos/
004500   164   000                    .ASCII  /t/<00>
004502   045   116   045     P.ACI:   .ASCII  /%N%/
004505   101   052   052              .ASCII  /A**/
004510   040   125   156              .ASCII  / Un/
004513   153   156   157              .ASCII  /kno/
004516   167   156   040              .ASCII  /wn /
004521   105   162   162              .ASCII  /Err/
004524   157   162   040              .ASCII  /or /
004527   114   157   147              .ASCII  /Log/
004532   040   146   157              .ASCII  / fo/
004535   162   155   141              .ASCII  /rma/
004540   164   040   045              .ASCII  /t %/
004543   117   063   045              .ASCII  /O3%/
004546   101   040   162              .ASCII  /A r/
004551   145   143   145              .ASCII  /ece/
004554   151   166   145              .ASCII  /ive/
004557   144   000   000              .ASCII  /d/<00><00>
004562   045   116   045     P.ACJ:   .ASCII  /%N%/
004565   101   052   052              .ASCII  /A**/
004570   040   117   160              .ASCII  / Op/
004573   055   143   157              .ASCII  /-co/
004576   144   145   040              .ASCII  /de /
004601   045   117   063              .ASCII  /%O3/
004604   045   101   054              .ASCII  /%A,/
004607   040   105   156              .ASCII  / En/
004612   144   055   143              .ASCII  /d-c/
004615   157   144   145              .ASCII  /ode/
004620   040   045   117              .ASCII  / %O/
004623   063   045   101              .ASCII  /3%A/
004626   040   146   157              .ASCII  / fo/
004631   162   040   162              .ASCII  /r r/
004634   145   146   040              .ASCII  /ef /
004637   043   040   045              .ASCII  /# %/
004642   117   066   045              .ASCII  /O6%/
004645   101   057   045              .ASCII  /A/<57>/%/
004650   117   066   045              .ASCII  /O6%/
004653   101   040   050              .ASCII  /A (/
004656   070   051   000              .ASCII  /8)/<00>
004661   000                          .ASCII  <00>
004662   045   116   045     P.ACK:   .ASCII  /%N%/
004665   101   052   052              .ASCII  /A**/
004670   040   103   155              .ASCII  / Cm/
004673   144   055   142              .ASCII  /d-b/
004676   143   040   045              .ASCII  /c %/
004701   117   066   045              .ASCII  /O6%/
004704   101   057   045              .ASCII  /A/<57>/%/
004707   117   066   045              .ASCII  /O6%/
```

```
004712  101  040  122           .ASCII  /A R/
004715  163  160  055           .ASCII  /sp-/
004720  142  143  040           .ASCII  /bc /
004723  045  117  066           .ASCII  /%O6/
004726  045  101  057           .ASCII  /%A/<57>
004731  045  117  066           .ASCII  /%O6/
004734  045  101  040           .ASCII  /%A /
004737  146  157  162           .ASCII  /for/
004742  040  045  117           .ASCII  / %O/
004745  066  045  101           .ASCII  /6%A/
004750  057  045  117           .ASCII  <57>/%O/
004753  066  045  101           .ASCII  /6%A/
004756  040  050  070           .ASCII  / (8/
004761  051  000  000           .ASCII  /)/<00><00>
004764  045  116  045  P.ACL:   .ASCII  /%N%/
004767  101  052  052           .ASCII  /A**/
004772  040  122  145           .ASCII  / Re/
004775  163  160  157           .ASCII  /spo/
005000  156  163  145           .ASCII  /nse/
005003  040  141  154           .ASCII  / al/
005006  162  145  141           .ASCII  /rea/
005011  144  171  040           .ASCII  /dy /
005014  162  145  143           .ASCII  /rec/
005017  145  151  166           .ASCII  /eiv/
005022  145  144  040           .ASCII  /ed /
005025  146  157  162           .ASCII  /for/
005030  040  143  155           .ASCII  / cm/
005033  144  040  045           .ASCII  /d %/
005036  117  066  045           .ASCII  /O6%/
005041  101  057  045           .ASCII  /A/<57>/%/
005044  117  066  045           .ASCII  /O6%/
005047  101  040  050           .ASCII  /A (/
005052  070  051  000           .ASCII  /8)/<00>
005055  000                     .ASCII  <00>
005056  045  116  045  P.ACM:   .ASCII  /%N%/
005061  101  052  052           .ASCII  /A**/
005064  040  106  141           .ASCII  / Fa/
005067  151  154  165           .ASCII  /ilu/
005072  162  145  040           .ASCII  /re /
005075  164  157  040           .ASCII  /to /
005100  163  145  156           .ASCII  /sen/
005103  144  040  143           .ASCII  /d c/
005106  157  155  155           .ASCII  /omm/
005111  141  156  144           .ASCII  /and/
005114  040  141  146           .ASCII  / af/
005117  164  145  162           .ASCII  /ter/
005122  040  043  040           .ASCII  / # /
005125  045  117  066           .ASCII  /%O6/
005130  045  101  057           .ASCII  /%A/<57>
005133  045  117  066           .ASCII  /%O6/
005136  045  101  040           .ASCII  /%A /
005141  050  070  051           .ASCII  /(8)/
005144  000  000                .ASCII  <00><00>
```

```
005146  045  116  045   P.ACN:  .ASCII  /%N%/
005151  101  052  052           .ASCII  /A**/
005154  040  122  145           .ASCII  / Re/
005157  163  160  157           .ASCII  /spo/
005162  156  163  145           .ASCII  /nse/
005165  040  150  141           .ASCII  / ha/
005170  163  040  105           .ASCII  /s E/
005173  162  162  157           .ASCII  /rro/
005176  162  040  114           .ASCII  /r L/
005201  157  147  050           .ASCII  /og(/
005204  163  051  072           .ASCII  /s):/
005207  000                     .ASCII  <00>
005210  045  116  045   P.ACO:  .ASCII  /%N%/
005213  101  052  052           .ASCII  /A**/
005216  040  110  117           .ASCII  / HO/
005221  123  124  040           .ASCII  /ST /
005224  115  105  115           .ASCII  /MEM/
005227  117  122  131           .ASCII  /ORY/
005232  040  105  122           .ASCII  / ER/
005235  122  117  122           .ASCII  /ROR/
005240  054  040  103           .ASCII  /, C/
005243  123  122  040           .ASCII  /SR /
005246  075  040  040           .ASCII  /=  /
005251  045  117  066           .ASCII  /%O6/
005254  045  101  000           .ASCII  /%A/<00>
005257  000                     .ASCII  <00>
005260  045  116  045   P.ACP:  .ASCII  /%N%/
005263  101  052  052           .ASCII  /A**/
005266  040  110  117           .ASCII  / HO/
005271  123  124  040           .ASCII  /ST /
005274  115  105  115           .ASCII  /MEM/
005277  117  122  131           .ASCII  /ORY/
005302  040  105  122           .ASCII  / ER/
005305  122  117  122           .ASCII  /ROR/
005310  054  040  105           .ASCII  /, E/
005313  130  124  105           .ASCII  /XTE/
005316  116  124  105           .ASCII  /NTE/
005321  104  040  103           .ASCII  /D C/
005324  123  122  040           .ASCII  /SR /
005327  075  040  040           .ASCII  /=  /
005332  045  117  066           .ASCII  /%O6/
005335  045  101  000           .ASCII  /%A/<00>
005340  045  116  045   P.ACQ:  .ASCII  /%N%/
005343  101  052  052           .ASCII  /A**/
005346  040  102  101           .ASCII  / BA/
005351  104  040  102           .ASCII  /D B/
005354  114  117  103           .ASCII  /LOC/
005357  113  040  122           .ASCII  /K R/
005362  105  120  114           .ASCII  /EPL/
005365  101  103  105           .ASCII  /ACE/
005370  115  105  116           .ASCII  /MEN/
005373  124  040  122           .ASCII  /T R/
005376  105  121  125           .ASCII  /EQU/
```

```
005401  105  123  124           .ASCII  /EST/
005404  040  106  114           .ASCII  / FL/
005407  101  107  040           .ASCII  /AG /
005412  123  105  124           .ASCII  /SET/
005415  000                     .ASCII  <00>
005416  045  116  045   P.ACR:  .ASCII  /%N%/
005421  101  052  052           .ASCII  /A**/
005424  040  105  122           .ASCII  / ER/
005427  122  117  122           .ASCII  /ROR/
005432  040  104  125           .ASCII  / DU/
005435  122  111  116           .ASCII  /RIN/
005440  107  040  122           .ASCII  /G R/
005443  105  120  114           .ASCII  /EPL/
005446  101  103  105           .ASCII  /ACE/
005451  115  105  116           .ASCII  /MEN/
005454  124  040  050           .ASCII  /T (/
005457  102  101  104           .ASCII  /BAD/
005462  040  102  114           .ASCII  / BL/
005465  117  103  113           .ASCII  /OCK/
005470  051  040  106           .ASCII  /) F/
005473  114  101  107           .ASCII  /LAG/
005476  040  123  105           .ASCII  / SE/
005501  124  000  000           .ASCII  /T/<00><00>
005504  045  116  045   P.ACS:  .ASCII  /%N%/
005507  101  125  116           .ASCII  /AUN/
005512  111  124  045           .ASCII  /IT%/
005515  104  062  045           .ASCII  /D2%/
005520  101  040  104           .ASCII  /A D/
005523  122  117  120           .ASCII  /ROP/
005526  120  105  104           .ASCII  /PED/
005531  040  055  040           .ASCII  / - /
005534  000  000                .ASCII  <00><00>
005536  045  101  125   P.ACU:  .ASCII  /%AU/
005541  123  105  122           .ASCII  /SER/
005544  040  103  117           .ASCII  / CO/
005547  115  115  101           .ASCII  /MMA/
005552  116  104  045           .ASCII  /ND%/
005555  116  000  000           .ASCII  /N/<00><00>
005560  045  101  103   P.ACV:  .ASCII  /%AC/
005563  117  116  106           .ASCII  /ONF/
005566  111  107  125           .ASCII  /IGU/
005571  122  101  124           .ASCII  /RAT/
005574  111  117  116           .ASCII  /ION/
005577  040  105  122           .ASCII  / ER/
005602  122  117  122           .ASCII  /ROR/
005605  045  116  000           .ASCII  /%N/<00>
005610  045  101  111   P.ACW:  .ASCII  /%AI/
005613  116  111  124           .ASCII  /NIT/
005616  040  105  122           .ASCII  / ER/
005621  122  117  122           .ASCII  /ROR/
005624  045  116  000           .ASCII  /%N/<00>
005627  000                     .ASCII  <00>
005630  045  101  124   P.ACX:  .ASCII  /%AT/
```

```
005633  122  101  116           .ASCII  /RAN/
005636  123  106  105           .ASCII  /SFE/
005641  122  040  114           .ASCII  /R L/
005644  111  115  111           .ASCII  /IMI/
005647  124  040  122           .ASCII  /T R/
005652  105  101  103           .ASCII  /EAC/
005655  110  105  104           .ASCII  /HED/
005660  045  116  000           .ASCII  /%N/<00>
005663  000                     .ASCII  <00>
005664  045  101  105  P.ACY:   .ASCII  /%AE/
005667  122  122  117           .ASCII  /RRO/
005672  122  040  114           .ASCII  /R L/
005675  111  115  111           .ASCII  /IMI/
005700  124  040  122           .ASCII  /T R/
005703  105  101  103           .ASCII  /EAC/
005706  110  105  104           .ASCII  /HED/
005711  045  116  000           .ASCII  /%N/<00>
005714  045  101  125  P.ACZ:   .ASCII  /%AU/
005717  116  122  105           .ASCII  /NRE/
005722  103  117  126           .ASCII  /COV/
005725  105  122  101           .ASCII  /ERA/
005730  102  114  105           .ASCII  /BLE/
005733  040  104  122           .ASCII  / DR/
005736  111  126  105           .ASCII  /IVE/
005741  040  105  122           .ASCII  / ER/
005744  122  117  122           .ASCII  /ROR/
005747  045  116  000           .ASCII  /%N/<00>
005752  045  101  125  P.ADA:   .ASCII  /%AU/
005755  116  122  105           .ASCII  /NRE/
005760  103  117  126           .ASCII  /COV/
005763  105  122  101           .ASCII  /ERA/
005766  102  114  105           .ASCII  /BLE/
005771  040  103  117           .ASCII  / CO/
005774  116  124  122           .ASCII  /NTR/
005777  117  114  114           .ASCII  /OLL/
006002  105  122  040           .ASCII  /ER /
006005  105  122  122           .ASCII  /ERR/
006010  117  122  045           .ASCII  /OR%/
006013  116  000  000           .ASCII  /N/<00><00>
006016  045  101  106  P.ADB:   .ASCII  /%AF/
006021  101  111  114           .ASCII  /AIL/
006024  105  104  040           .ASCII  /ED /
006027  124  117  040           .ASCII  /TO /
006032  103  117  115           .ASCII  /COM/
006035  105  040  117           .ASCII  /E O/
006040  116  114  111           .ASCII  /NLI/
006043  116  105  045           .ASCII  /NE%/
006046  116  000                .ASCII  /N/<00>
006050  045  101  106  P.ADC:   .ASCII  /%AF/
006053  101  111  114           .ASCII  /AIL/
006056  105  104  040           .ASCII  /ED /
006061  124  117  040           .ASCII  /TO /
006064  101  103  103           .ASCII  /ACC/
```

```
006067    105     123     123                 .ASCII  /ESS/
006072    040     105     111                 .ASCII  / EI/
006075    124     110     105                 .ASCII  /THE/
006100    122     040     106                 .ASCII  /R F/
006103    111     122     123                 .ASCII  /IRS/
006106    124     040     117                 .ASCII  /T O/
006111    122     040     114                 .ASCII  /R L/
006114    101     123     124                 .ASCII  /AST/
006117    040     124     122                 .ASCII  / TR/
006122    101     103     113                 .ASCII  /ACK/
006125    040     104     125                 .ASCII  / DU/
006130    122     111     116                 .ASCII  /RIN/
006133    107     040     111                 .ASCII  /G I/
006136    116     111     124                 .ASCII  /NIT/
006141    045     116     000                 .ASCII  /%N/<00>
006144    045     101     104     P.ADD:      .ASCII  /%AD/
006147    111     123     113                 .ASCII  /ISK/
006152    040     127     122                 .ASCII  / WR/
006155    111     124     105                 .ASCII  /ITE/
006160    040     120     122                 .ASCII  / PR/
006163    117     124     105                 .ASCII  /OTE/
006166    103     124     105                 .ASCII  /CTE/
006171    104     045     116                 .ASCII  /D%N/
006174    000     000                         .ASCII  <00><00>
006176    045     101     103     P.ADE:      .ASCII  /%AC/
006201    117     115     115                 .ASCII  /OMM/
006204    101     116     104                 .ASCII  /AND/
006207    040     124     111                 .ASCII  / TI/
006212    115     105     040                 .ASCII  /ME /
006215    117     125     124                 .ASCII  /OUT/
006220    045     116     000                 .ASCII  /%N/<00>
006223    000                                 .ASCII  <00>
006224 005536'                        P.ACT:      .WORD   P.ACU
006226 005560'                                    .WORD   P.ACV
006230 005610'                                    .WORD   P.ACW
006232 005630'                                    .WORD   P.ACX
006234 005664'                                    .WORD   P.ACY
006236 005714'                                    .WORD   P.ACZ
006240 005752'                                    .WORD   P.ADA
006242 006016'                                    .WORD   P.ADB
006244 006050'                                    .WORD   P.ADC
006246 006144'                                    .WORD   P.ADD
006250 006176'                                    .WORD   P.ADE
006252    045     116     045     P.ADF:      .ASCII  /%N%/
006255    101     120     117                 .ASCII  /APO/
006260    127     105     122                 .ASCII  /WER/
006263    040     104     105                 .ASCII  / DE/
006266    114     101     131                 .ASCII  /LAY/
006271    040     055     040                 .ASCII  / - /
006274    127     101     111                 .ASCII  /WAI/
006277    124     111     116                 .ASCII  /TIN/
006302    107     000                         .ASCII  /G/<00>
006304    045     116     045     P.ADG:      .ASCII  /%N%/
```

```
006307  101  106  125           .ASCII  /AFU/
006312  116  103  124           .ASCII  /NCT/
006315  111  117  116           .ASCII  /ION/
006320  101  114  040           .ASCII  /AL /
006323  124  105  123           .ASCII  /TES/
006326  124  040  123           .ASCII  /T S/
006331  124  101  122           .ASCII  /TAR/
006334  124  105  104           .ASCII  /TED/
006337  000                     .ASCII  <00>
006340  045  116  045   P.ADH:  .ASCII  /%N%/
006343  116  045  101           .ASCII  /N%A/
006346  105  130  105           .ASCII  /EXE/
006351  122  103  111           .ASCII  /RCI/
006354  123  105  122           .ASCII  /SER/
006357  040  123  124           .ASCII  / ST/
006362  101  122  124           .ASCII  /ART/
006365  105  104  045           .ASCII  /ED%/
006370  116  000                .ASCII  /N/<00>
006372  045  116  045   P.ADI:  .ASCII  /%N%/
006375  116  045  101           .ASCII  /N%A/
006400  125  116  124           .ASCII  /UNT/
006403  040  104  123           .ASCII  / DS/
006406  113  045  123           .ASCII  /K%S/
006411  070  045  101           .ASCII  /8%A/
006414  043  040  117           .ASCII  /# O/
006417  106  040  040           .ASCII  /F  /
006422  040  043  040           .ASCII  / # /
006425  102  131  124           .ASCII  /BYT/
006430  105  123  040           .ASCII  /ES /
006433  040  040  043           .ASCII  /  #/
006436  040  117  106           .ASCII  / OF/
006441  040  040  040           .ASCII  /   /
006444  040  043  040           .ASCII  / # /
006447  102  131  124           .ASCII  /BYT/
006452  105  123  000           .ASCII  /ES/<00>
006455  000                     .ASCII  <00>
006456  045  101  040   P.ADJ:  .ASCII  /%A /
006461  040  055  055           .ASCII  / --/
006464  110  101  122           .ASCII  /HAR/
006467  104  040  105           .ASCII  /D E/
006472  122  122  117           .ASCII  /RRO/
006475  122  123  055           .ASCII  /RS-/
006500  055  040  055           .ASCII  /- -/
006503  055  123  117           .ASCII  /-SO/
006506  106  124  040           .ASCII  /FT /
006511  105  122  122           .ASCII  /ERR/
006514  117  122  123           .ASCII  /ORS/
006517  055  055  000           .ASCII  /--/<00>
006522  045  116  045   P.ADK:  .ASCII  /%N%/
006525  101  040  043           .ASCII  /A #/
006530  040  040  040           .ASCII  /   /
006533  043  040  040           .ASCII  /#  /
006536  124  131  120           .ASCII  /TYP/
```

```
006541  105  040  040          .ASCII  /E  /
006544  122  105  101          .ASCII  /REA/
006547  104  123  040          .ASCII  /DS /
006552  040  040  040          .ASCII  /   /
006555  040  122  105          .ASCII  / RE/
006560  101  104  040          .ASCII  /AD /
006563  040  040  127          .ASCII  /  W/
006566  122  111  124          .ASCII  /RIT/
006571  105  123  040          .ASCII  /ES /
006574  040  040  127          .ASCII  /  W/
006577  122  111  124          .ASCII  /RIT/
006602  124  105  116          .ASCII  /TEN/
006605  000                    .ASCII  <00>
006606  045  101  040  P.ADL:  .ASCII  /%A /
006611  040  123  105          .ASCII  / SE/
006614  113  040  104          .ASCII  /K D/
006617  101  124  040          .ASCII  /AT /
006622  104  122  126          .ASCII  /DRV/
006625  040  110  123          .ASCII  / HS/
006630  124  040  123          .ASCII  /T S/
006633  105  113  040          .ASCII  /EK /
006636  104  101  124          .ASCII  /DAT/
006641  040  104  122          .ASCII  / DR/
006644  126  040  110          .ASCII  /V H/
006647  123  124  000          .ASCII  /ST/<00>
006652  045  116  045  P.ADM:  .ASCII  /%N%/
006655  101  055  055          .ASCII  /A--/
006660  055  040  055          .ASCII  /- -/
006663  055  055  040          .ASCII  /-- /
006666  055  055  055          .ASCII  /---/
006671  055  040  040          .ASCII  /-  /
006674  055  055  055          .ASCII  /---/
006677  055  055  040          .ASCII  /-- /
006702  040  055  055          .ASCII  / --/
006705  055  055  055          .ASCII  /---/
006710  055  055  055          .ASCII  /---/
006713  055  040  055          .ASCII  /- -/
006716  055  055  055          .ASCII  /---/
006721  055  055  040          .ASCII  /-- /
006724  040  055  055          .ASCII  / --/
006727  055  055  055          .ASCII  /---/
006732  055  055  055          .ASCII  /---/
006735  055  000  000          .ASCII  /-/<00><00>
006740  045  101  040  P.ADN:  .ASCII  /%A /
006743  055  055  055          .ASCII  /---/
006746  040  055  055          .ASCII  / --/
006751  055  040  055          .ASCII  /- -/
006754  055  055  040          .ASCII  /-- /
006757  055  055  055          .ASCII  /---/
006762  040  055  055          .ASCII  / --/
006765  055  040  055          .ASCII  /- -/
006770  055  055  040          .ASCII  /-- /
006773  055  055  055          .ASCII  /---/
```

```
006776  040     055     055                     .ASCII  / --/
007001  055     000     000                     .ASCII  /-/<00><00>
007004  045     116     045             P.ADO:  .ASCII  /%N%/
007007  104     062     045                     .ASCII  /D2%/
007012  104     064     045                     .ASCII  /D4%/
007015  123     062     045                     .ASCII  /S2%/
007020  124     000                             .ASCII  /T/<00>
007022  045     104     064             P.ADP:  .ASCII  /%D4/
007025  045     132     063                     .ASCII  /%Z3/
007030  045     104     063                     .ASCII  /%D3/
007033  045     101     054                     .ASCII  /%A,/
007036  045     132     063                     .ASCII  /%Z3/
007041  045     101     054                     .ASCII  /%A,/
007044  045     132     063                     .ASCII  /%Z3/
007047  000                                     .ASCII  <00>
007050  045     104     064             P.ADQ:  .ASCII  /%D4/
007053  045     104     064                     .ASCII  /%D4/
007056  045     104     064                     .ASCII  /%D4/
007061  045     104     064                     .ASCII  /%D4/
007064  045     104     064                     .ASCII  /%D4/
007067  045     104     064                     .ASCII  /%D4/
007072  045     104     064                     .ASCII  /%D4/
007075  045     104     064                     .ASCII  /%D4/
007100  000     000                             .ASCII  <00><00>
007102  045     116     045             P.ADR:  .ASCII  /%N%/
007105  101     040     056                     .ASCII  /A ./
007110  040     040     040                     .ASCII  /   /
007113  056     040     040                     .ASCII  /.  /
007116  103     116     124                     .ASCII  /CNT/
007121  122     040     040                     .ASCII  /R  /
007124  040     040     040                     .ASCII  /   /
007127  040     056     040                     .ASCII  / . /
007132  040     056     056                     .ASCII  / ../
007135  056     056     056                     .ASCII  /.../
007140  056     056     056                     .ASCII  /.../
007143  056     040     040                     .ASCII  /.  /
007146  040     040     040                     .ASCII  /   /
007151  040     056     040                     .ASCII  / . /
007154  040     056     056                     .ASCII  / ../
007157  056     056     056                     .ASCII  /.../
007162  056     056     056                     .ASCII  /.../
007165  056     000     000                     .ASCII  /./<00><00>
007170  045     101     040             P.ADS:  .ASCII  /%A /
007173  040     040     056                     .ASCII  /  ./
007176  040     040     040                     .ASCII  /   /
007201  056     045     104                     .ASCII  /.%D/
007204  064     045     101                     .ASCII  /4%A/
007207  040     040     040                     .ASCII  /   /
007212  056     040     040                     .ASCII  /.  /
007215  040     056     040                     .ASCII  / . /
007220  040     040     056                     .ASCII  /  ./
007223  045     104     064                     .ASCII  /%D4/
007226  045     101     040                     .ASCII  /%A /
```

```
007231    040    040    056                   .ASCII  /  ./
007234    000    000                          .ASCII  <00><00>
007236    045    101    040        P.ADT:     .ASCII  /%A /
007241    040    040    056                   .ASCII  /  ./
007244    040    040    040                   .ASCII  /   /
007247    056    045    104                   .ASCII  /.%D/
007252    064    045    101                   .ASCII  /4%A/
007255    040    040    040                   .ASCII  /   /
007260    056    040    040                   .ASCII  /.  /
007263    040    056    040                   .ASCII  / . /
007266    040    040    056                   .ASCII  /  ./
007271    045    104    064                   .ASCII  /%D4/
007274    045    101    040                   .ASCII  /%A /
007277    040    040    056                   .ASCII  /  ./
007302    000    000                          .ASCII  <00><00>
007304    045    116    045        P.ADU:     .ASCII  /%N%/
007307    116    045    101                   .ASCII  /N%A/
007312    125    116    111                   .ASCII  /UNI/
007315    124    040    040                   .ASCII  /T  /
007320    104    111    123                   .ASCII  /DIS/
007323    113    040    040                   .ASCII  /K  /
007326    040    040    040                   .ASCII  /   /
007331    040    040    040                   .ASCII  /   /
007334    040    040    040                   .ASCII  /   /
007337    040    043    040                   .ASCII  / # /
007342    117    106    040                   .ASCII  /OF /
007345    040    040    043                   .ASCII  /  #/
007350    040    102    114                   .ASCII  / BL/
007353    113    123    040                   .ASCII  /KS /
007356    040    040    040                   .ASCII  /   /
007361    040    040    040                   .ASCII  /   /
007364    043    040    117                   .ASCII  /# O/
007367    106    040    040                   .ASCII  /F  /
007372    040    040    043                   .ASCII  /  #/
007375    040    102    114                   .ASCII  / BL/
007400    113    123    040                   .ASCII  /KS /
007403    000                                 .ASCII  <00>
007404    045    116    045        P.ADV:     .ASCII  /%N%/
007407    101    040    040                   .ASCII  /A  /
007412    043    040    040                   .ASCII  /#  /
007415    040    040    040                   .ASCII  /   /
007420    043    040    040                   .ASCII  /#  /
007423    040    040    040                   .ASCII  /   /
007426    124    131    120                   .ASCII  /TYP/
007431    105    040    040                   .ASCII  /E  /
007434    040    122    105                   .ASCII  / RE/
007437    101    104    123                   .ASCII  /ADS/
007442    040    040    040                   .ASCII  /   /
007445    040    040    122                   .ASCII  /  R/
007450    105    101    104                   .ASCII  /EAD/
007453    040    040    040                   .ASCII  /   /
007456    040    040    040                   .ASCII  /   /
007461    127    122    111                   .ASCII  /WRI/
```

```
007464  124  105  123            .ASCII  /TES/
007467  040  040  127            .ASCII  /  W/
007472  122  111  124            .ASCII  /RIT/
007475  124  105  116            .ASCII  /TEN/
007500  040  000                 .ASCII  / /<00>
007502  045  116  045    P.ADW:  .ASCII  /%N%/
007505  101  055  055            .ASCII  /A--/
007510  055  055  040            .ASCII  /-- /
007513  040  055  055            .ASCII  / --/
007516  055  055  040            .ASCII  /-- /
007521  040  055  055            .ASCII  / --/
007524  055  055  055            .ASCII  /---/
007527  055  055  040            .ASCII  /-- /
007532  040  055  055            .ASCII  / --/
007535  055  055  055            .ASCII  /---/
007540  055  040  040            .ASCII  /-  /
007543  040  055  055            .ASCII  / --/
007546  055  055  055            .ASCII  /---/
007551  055  040  040            .ASCII  /-  /
007554  040  040  040            .ASCII  /   /
007557  055  055  055            .ASCII  /---/
007562  055  055  055            .ASCII  /---/
007565  040  040  040            .ASCII  /   /
007570  055  055  055            .ASCII  /---/
007573  055  055  055            .ASCII  /---/
007576  040  040  000            .ASCII  /  /<00>
007601  000                      .ASCII  <00>
007602  045  116  045    P.ADX:  .ASCII  /%N%/
007605  123  061  045            .ASCII  /S1%/
007610  104  062  045            .ASCII  /D2%/
007613  123  064  045            .ASCII  /S4%/
007616  104  062  045            .ASCII  /D2%/
007621  101  040  040            .ASCII  /A  /
007624  040  104  102            .ASCII  / DB/
007627  116  040  111            .ASCII  /N I/
007632  057  117  040            .ASCII  <57>/O /
007635  040  045  104            .ASCII  / %D/
007640  066  045  123            .ASCII  /6%S/
007643  063  045  104            .ASCII  /3%D/
007646  066  045  123            .ASCII  /6%S/
007651  065  045  104            .ASCII  /5%D/
007654  066  045  123            .ASCII  /6%S/
007657  063  045  104            .ASCII  /3%D/
007662  066  000                 .ASCII  /6/<00>
007664  124  117  117    P.ADY:  .ASCII  /TOO/
007667  040  115  101            .ASCII  / MA/
007672  116  131  040            .ASCII  /NY /
007675  125  116  111            .ASCII  /UNI/
007700  124  123  000            .ASCII  /TS/<00>
007703  000                      .ASCII  <00>
007704  116  117  124    P.ADZ:  .ASCII  /NOT/
007707  040  105  116            .ASCII  / EN/
007712  117  125  107            .ASCII  /OUG/
```

```
007715   110   040   106            .ASCII  /H F/
007720   122   105   105            .ASCII  /REE/
007723   040   115   105            .ASCII  / ME/
007726   115   117   122            .ASCII  /MOR/
007731   131   040   106            .ASCII  /Y F/
007734   117   122   040            .ASCII  /OR /
007737   101   114   114            .ASCII  /ALL/
007742   117   103   101            .ASCII  /OCA/
007745   124   111   116            .ASCII  /TIN/
007750   107   040   122            .ASCII  /G R/
007753   105   101   104            .ASCII  /EAD/
007756   057   127   122            .ASCII  <57>/WR/
007761   111   124   105            .ASCII  /ITE/
007764   040   102   125            .ASCII  / BU/
007767   106   106   105            .ASCII  /FFE/
007772   122   123   000            .ASCII  /RS/<00>
007775   000                        .ASCII  <00>
007776   122   105   107   P.AEA:   .ASCII  /REG/
010001   111   123   124            .ASCII  /IST/
010004   105   122   040            .ASCII  /ER /
010007   105   130   111            .ASCII  /EXI/
010012   123   124   105            .ASCII  /STE/
010015   116   103   105            .ASCII  /NCE/
010020   040   124   105            .ASCII  / TE/
010023   123   124   040            .ASCII  /ST /
010026   106   101   111            .ASCII  /FAI/
010031   114   105   104            .ASCII  /LED/
010034   000   000                  .ASCII  <00><00>
010036   126   105   103   P.AEB:   .ASCII  /VEC/
010041   124   117   122            .ASCII  /TOR/
010044   040   124   105            .ASCII  / TE/
010047   123   124   040            .ASCII  /ST /
010052   106   101   111            .ASCII  /FAI/
010055   114   105   104            .ASCII  /LED/
010060   000   000                  .ASCII  <00><00>
010062   102   122   040   P.AEC:   .ASCII  /BR /
010065   114   105   126            .ASCII  /LEV/
010070   105   114   040            .ASCII  /EL /
010073   124   105   123            .ASCII  /TES/
010076   124   040   106            .ASCII  /T F/
010101   101   111   114            .ASCII  /AIL/
010104   105   104   000            .ASCII  /ED/<00>
010107   000                        .ASCII  <00>
010110   111   116   111   P.AED:   .ASCII  /INI/
010113   124   040   123            .ASCII  /T S/
010116   105   121   125            .ASCII  /EQU/
010121   105   116   103            .ASCII  /ENC/
010124   105   040   106            .ASCII  /E F/
010127   101   111   114            .ASCII  /AIL/
010132   105   104   000            .ASCII  /ED/<00>
010135   000                        .ASCII  <00>
010136   106   101   124   P.AEE:   .ASCII  /FAT/
010141   101   114   040            .ASCII  /AL /
```

```
010144  103  117  116           .ASCII  /CON/
010147  124  122  117           .ASCII  /TRO/
010152  114  114  105           .ASCII  /LLE/
010155  122  040  105           .ASCII  /R E/
010160  122  122  117           .ASCII  /RRO/
010163  122  000  000           .ASCII  /R/<00><00>
010166  117  116  114  P.AEF:   .ASCII  /ONL/
010171  111  116  105           .ASCII  /INE/
010174  040  106  101           .ASCII  / FA/
010177  111  114  105           .ASCII  /ILE/
010202  104  000                .ASCII  /D/<00>
010204  127  122  111  P.AEG:   .ASCII  /WRI/
010207  124  105  055           .ASCII  /TE-/
010212  120  122  117           .ASCII  /PRO/
010215  124  105  103           .ASCII  /TEC/
010220  124  040  103           .ASCII  /T C/
010223  117  116  106           .ASCII  /ONF/
010226  114  111  103           .ASCII  /LIC/
010231  124  000  000           .ASCII  /T/<00><00>
010234  101  103  103  P.AEH:   .ASCII  /ACC/
010237  105  123  123           .ASCII  /ESS/
010242  040  106  101           .ASCII  / FA/
010245  111  114  105           .ASCII  /ILE/
010250  104  000                .ASCII  /D/<00>
010252  106  101  124  P.AEI:   .ASCII  /FAT/
010255  101  114  040           .ASCII  /AL /
010260  111  057  117           .ASCII  /I/<57>/O/
010263  040  105  122           .ASCII  / ER/
010266  122  117  122           .ASCII  /ROR/
010271  000                     .ASCII  <00>
010272  104  111  123  P.AEJ:   .ASCII  /DIS/
010275  113  040  124           .ASCII  /K T/
010300  131  120  105           .ASCII  /YPE/
010303  040  125  116           .ASCII  / UN/
010306  113  116  117           .ASCII  /KNO/
010311  127  116  040           .ASCII  /WN /
010314  124  117  040           .ASCII  /TO /
010317  105  130  105           .ASCII  /EXE/
010322  122  103  111           .ASCII  /RCI/
010325  123  105  122           .ASCII  /SER/
010330  000  000                .ASCII  <00><00>
010332  106  101  111  P.AEK:   .ASCII  /FAI/
010335  114  105  104           .ASCII  /LED/
010340  040  124  117           .ASCII  / TO/
010343  040  123  105           .ASCII  / SE/
010346  116  104  040           .ASCII  /ND /
010351  123  105  124           .ASCII  /SET/
010354  055  103  117           .ASCII  /-CO/
010357  116  124  122           .ASCII  /NTR/
010362  117  114  114           .ASCII  /OLL/
010365  105  122  055           .ASCII  /ER-/
010370  103  110  101           .ASCII  /CHA/
010373  122  101  103           .ASCII  /RAC/
```

```
010376  124  105  122          .ASCII  /TER/
010401  111  123  124          .ASCII  /IST/
010404  111  103  123          .ASCII  /ICS/
010407  040  103  117          .ASCII  / CO/
010412  115  115  101          .ASCII  /MMA/
010415  116  104  000          .ASCII  /ND/<00>
010420  123  105  124  P.AEL:  .ASCII  /SET/
010423  055  103  117          .ASCII  /-CO/
010426  116  124  122          .ASCII  /NTR/
010431  117  114  114          .ASCII  /OLL/
010434  105  122  055          .ASCII  /ER-/
010437  103  110  101          .ASCII  /CHA/
010442  122  101  103          .ASCII  /RAC/
010445  124  105  122          .ASCII  /TER/
010450  111  123  124          .ASCII  /IST/
010453  111  103  123          .ASCII  /ICS/
010456  040  122  105          .ASCII  / RE/
010461  123  120  117          .ASCII  /SPO/
010464  116  123  105          .ASCII  /NSE/
010467  040  110  101          .ASCII  / HA/
010472  123  040  102          .ASCII  /S B/
010475  101  104  040          .ASCII  /AD /
010500  105  116  104          .ASCII  /END/
010503  103  117  104          .ASCII  /COD/
010506  105  040  117          .ASCII  /E O/
010511  122  040  106          .ASCII  /R F/
010514  114  101  107          .ASCII  /LAG/
010517  123  040  111          .ASCII  /S I/
010522  116  040  105          .ASCII  /N E/
010525  122  122  117          .ASCII  /RRO/
010530  122  000               .ASCII  /R/<00>
010532  106  101  111  P.AEM:  .ASCII  /FAI/
010535  114  105  104          .ASCII  /LED/
010540  040  124  117          .ASCII  / TO/
010543  040  123  105          .ASCII  / SE/
010546  116  104  040          .ASCII  /ND /
010551  117  116  055          .ASCII  /ON-/
010554  114  111  116          .ASCII  /LIN/
010557  105  040  103          .ASCII  /E C/
010562  117  115  115          .ASCII  /OMM/
010565  101  116  104          .ASCII  /AND/
010570  000  000               .ASCII  <00><00>
010572  117  116  055  P.AEN:  .ASCII  /ON-/
010575  114  111  116          .ASCII  /LIN/
010600  105  040  122          .ASCII  /E R/
010603  105  123  120          .ASCII  /ESP/
010606  117  116  123          .ASCII  /ONS/
010611  105  040  110          .ASCII  /E H/
010614  101  123  040          .ASCII  /AS /
010617  102  101  104          .ASCII  /BAD/
010622  040  105  116          .ASCII  / EN/
010625  104  103  117          .ASCII  /DCO/
010630  104  105  000          .ASCII  /DE/<00>
```

```
010633  000                          .ASCII  <00>
010634  117    116    055    P.AEO:  .ASCII  /ON-/
010637  114    111    116            .ASCII  /LIN/
010642  105    040    122            .ASCII  /E R/
010645  105    123    120            .ASCII  /ESP/
010650  117    116    123            .ASCII  /ONS/
010653  105    040    110            .ASCII  /E H/
010656  101    123    040            .ASCII  /AS /
010661  125    116    113            .ASCII  /UNK/
010664  116    117    127            .ASCII  /NOW/
010667  116    040    104            .ASCII  /N D/
010672  105    126    111            .ASCII  /EVI/
010675  103    105    000            .ASCII  /CE/<00>
010700  111    057    117    P.AEP:  .ASCII  /I/<57>/O/
010703  040    122    105            .ASCII  / RE/
010706  121    125    105            .ASCII  /QUE/
010711  123    124    040            .ASCII  /ST /
010714  106    101    111            .ASCII  /FAI/
010717  114    105    104            .ASCII  /LED/
010722  000    000                   .ASCII  <00><00>
010724  045    101    115    P.AEQ:  .ASCII  /%AM/
010727  117    122    105            .ASCII  /ORE/
010732  040    124    110            .ASCII  / TH/
010735  101    116    040            .ASCII  /AN /
010740  045    104    062            .ASCII  /%D2/
010743  045    101    040            .ASCII  /%A /
010746  125    116    111            .ASCII  /UNI/
010751  124    123    040            .ASCII  /TS /
010754  123    120    105            .ASCII  /SPE/
010757  103    111    106            .ASCII  /CIF/
010762  111    105    104            .ASCII  /IED/
010765  000                          .ASCII  <00>
010766  045    101    052    P.AER:  .ASCII  /%A*/
010771  040    116    117            .ASCII  / NO/
010774  040    122    105            .ASCII  / RE/
010777  123    120    117            .ASCII  /SPO/
011002  116    123    105            .ASCII  /NSE/
011005  040    101    124            .ASCII  / AT/
011010  040    101    104            .ASCII  / AD/
011013  104    122    105            .ASCII  /DRE/
011016  123    123    040            .ASCII  /SS /
011021  045    117    066            .ASCII  /%O6/
011024  000    000                   .ASCII  <00><00>
011026  045    101    052    P.AES:  .ASCII  /%A*/
011031  040    111    116            .ASCII  / IN/
011034  103    117    122            .ASCII  /COR/
011037  122    105    103            .ASCII  /REC/
011042  124    040    102            .ASCII  /T B/
011045  122    040    114            .ASCII  /R L/
011050  105    126    105            .ASCII  /EVE/
011053  114    040    106            .ASCII  /L F/
011056  117    122    040            .ASCII  /OR /
011061  104    122    111            .ASCII  /DRI/
```

```
011064  126  105  040            .ASCII  /VE /
011067  045  117  066            .ASCII  /%06/
011072  000  000                 .ASCII  <00><00>
011074  045  101  052  P.AET:    .ASCII  /%A*/
011077  040  123  124            .ASCII  / ST/
011102  105  120  040            .ASCII  /EP /
011105  045  104  061            .ASCII  /%D1/
011110  045  101  040            .ASCII  /%A /
011113  122  105  101            .ASCII  /REA/
011116  104  040  105            .ASCII  /D E/
011121  122  122  117            .ASCII  /RRO/
011124  122  000                 .ASCII  /P/<00>
011126  045  101  052  P.AEU:    .ASCII  /%A*/
011131  040  102  101            .ASCII  / BA/
011134  104  040  123            .ASCII  /D S/
011137  101  040  103            .ASCII  /A C/
011142  117  104  105            .ASCII  /ODE/
011145  040  106  122            .ASCII  / FR/
011150  117  115  040            .ASCII  /OM /
011153  104  122  111            .ASCII  /DRI/
011156  126  105  040            .ASCII  /VE /
011161  045  117  066            .ASCII  /%06/
011164  000  000                 .ASCII  <00><00>
011166  045  101  052  P.AEV:    .ASCII  /%A*/
011171  040  104  111            .ASCII  / DI/
011174  123  113  045            .ASCII  /SK%/
011177  104  062  045            .ASCII  /D2%/
011202  101  040  127            .ASCII  /A W/
011205  105  116  124            .ASCII  /ENT/
011210  040  117  106            .ASCII  / OF/
011213  106  114  111            .ASCII  /FLI/
011216  116  105  000            .ASCII  /NE/<00>
011221  000                      .ASCII  <00>
011222  045  101  052  P.AEW:    .ASCII  /%A*/
011225  040  104  122            .ASCII  / DR/
011230  111  126  105            .ASCII  /IVE/
011233  040  045  117            .ASCII  / %O/
011236  066  045  101            .ASCII  /6%A/
011241  040  116  117            .ASCII  / NO/
011244  124  040  120            .ASCII  /T P/
011247  122  117  103            .ASCII  /ROC/
011252  105  123  123            .ASCII  /ESS/
011255  111  116  107            .ASCII  /ING/
011260  040  103  117            .ASCII  / CO/
011263  115  115  101            .ASCII  /MMA/
011266  116  104  040            .ASCII  /ND /
011271  120  101  103            .ASCII  /PAC/
011274  113  105  124            .ASCII  /KET/
011277  123  000  000            .ASCII  /S/<00><00>
011302  045  101  052  P.AEX:    .ASCII  /%A*/
011305  040  104  111            .ASCII  / DI/
011310  123  113  045            .ASCII  /SK%/
011313  104  062  045            .ASCII  /D2%/
```

```
011316  101  040  127            .ASCII  /A W/
011321  105  116  124            .ASCII  /ENT/
011324  040  124  117            .ASCII  / TO/
011327  040  124  110            .ASCII  / TH/
011332  105  040  042            .ASCII  /E "/
011335  101  126  101            .ASCII  /AVA/
011340  111  114  101            .ASCII  /ILA/
011343  102  114  105            .ASCII  /BLE/
011346  042  040  123            .ASCII  /" S/
011351  124  101  124            .ASCII  /TAT/
011354  105  000                 .ASCII  /E/<00>
011356  040  055  040  P.AEY:    .ASCII  / - /
011361  125  116  122            .ASCII  /UNR/
011364  105  103  117            .ASCII  /ECO/
011367  107  116  111            .ASCII  /GNI/
011372  132  105  104            .ASCII  /ZED/
011375  040  115  105            .ASCII  / ME/
011400  123  123  101            .ASCII  /SSA/
011403  107  105  040            .ASCII  /GE /
011406  124  131  120            .ASCII  /TYP/
011411  105  000  000            .ASCII  /E/<00><00>
011414  040  055  040  P.AEZ:    .ASCII  / - /
011417  125  116  122            .ASCII  /UNR/
011422  105  103  117            .ASCII  /ECO/
011425  107  116  111            .ASCII  /GNI/
011430  132  105  104            .ASCII  /ZED/
011433  040  103  117            .ASCII  / CO/
011436  116  116  105            .ASCII  /NNE/
011441  103  124  111            .ASCII  /CTI/
011444  117  116  040            .ASCII  /ON /
011447  111  104  000            .ASCII  /ID/<00>
011452  040  055  040  P.AFA:    .ASCII  / - /
011455  125  116  122            .ASCII  /UNR/
011460  105  103  117            .ASCII  /ECO/
011463  107  116  111            .ASCII  /GNI/
011466  132  105  104            .ASCII  /ZED/
011471  040  122  105            .ASCII  / RE/
011474  124  125  122            .ASCII  /TUR/
011477  116  040  115            .ASCII  /N M/
011502  105  123  123            .ASCII  /ESS/
011505  101  107  105            .ASCII  /AGE/
011510  000  000                 .ASCII  <00><00>
011512  040  055  040  P.AFB:    .ASCII  / - /
011515  125  116  122            .ASCII  /UNR/
011520  105  103  117            .ASCII  /ECO/
011523  107  116  111            .ASCII  /GNI/
011526  132  105  104            .ASCII  /ZED/
011531  040  122  105            .ASCII  / RE/
011534  124  125  122            .ASCII  /TUR/
011537  116  040  120            .ASCII  /N P/
011542  101  103  113            .ASCII  /ACK/
011545  105  124  000            .ASCII  /ET/<00>
011550  040  055  040  P.AFC:    .ASCII  / - /
```

```
011553  125  116  122             .ASCII  /UNR/
011556  105  103  117             .ASCII  /ECO/
011561  107  116  111             .ASCII  /GNI/
011564  132  105  104             .ASCII  /ZED/
011567  040  103  122             .ASCII  / CR/
011572  116  000                  .ASCII  /N/<00>
011574  040  055  040    P.AFD:   .ASCII  / - /
011577  125  116  122             .ASCII  /UNR/
011602  105  103  117             .ASCII  /ECO/
011605  107  116  111             .ASCII  /GNI/
011610  132  105  104             .ASCII  /ZED/
011613  040  117  120             .ASCII  / OP/
011616  103  117  104             .ASCII  /COD/
011621  105  000  000             .ASCII  /E/<00><00>
011624  040  055  040    P.AFE:   .ASCII  / - /
011627  115  123  103             .ASCII  /MSC/
011632  120  040  123             .ASCII  /P S/
011635  124  101  124             .ASCII  /TAT/
011640  125  123  040             .ASCII  /US /
011643  103  117  104             .ASCII  /COD/
011646  105  040  105             .ASCII  /E E/
011651  122  122  000             .ASCII  /RR/<00>
011654  040  055  040    P.AFF:   .ASCII  / - /
011657  104  125  120             .ASCII  /DUP/
011662  040  123  124             .ASCII  / ST/
011665  101  124  125             .ASCII  /ATU/
011670  123  040  103             .ASCII  /S C/
011673  117  104  105             .ASCII  /ODE/
011676  040  105  122             .ASCII  / ER/
011701  122  000  000             .ASCII  /R/<00><00>
011704  040  055  040    P.AFG:   .ASCII  / - /
011707  125  116  122             .ASCII  /UNR/
011712  105  103  117             .ASCII  /ECO/
011715  107  116  111             .ASCII  /GNI/
011720  132  105  104             .ASCII  /ZED/
011723  040  123  124             .ASCII  / ST/
011726  101  124  125             .ASCII  /ATU/
011731  123  040  103             .ASCII  /S C/
011734  117  104  105             .ASCII  /ODE/
011737  000                       .ASCII  <00>
011740  040  055  040    P.AFH:   .ASCII  / - /
011743  114  102  116             .ASCII  /LBN/
011746  040  110  117             .ASCII  / HO/
011751  123  124  040             .ASCII  /ST /
011754  103  117  115             .ASCII  /COM/
011757  120  101  122             .ASCII  /PAR/
011762  105  040  105             .ASCII  /E E/
011765  122  122  000             .ASCII  /RR/<00>
011770  040  055  040    P.AFI:   .ASCII  / - /
011773  104  102  116             .ASCII  /DBN/
011776  040  110  117             .ASCII  / HO/
012001  123  124  040             .ASCII  /ST /
012004  103  117  115             .ASCII  /COM/
```

```
012007   120   101   122             .ASCII  /PAR/
012012   105   040   105             .ASCII  /E E/
012015   122   122   000             .ASCII  /RR/<00>
012020   040   055   040   P.AFJ:    .ASCII  / - /
012023   125   116   101             .ASCII  /UNA/
012026   102   114   105             .ASCII  /BLE/
012031   040   124   117             .ASCII  / TO/
012034   040   114   117             .ASCII  / LO/
012037   101   104   040             .ASCII  /AD /
012042   104   125   120             .ASCII  /DUP/
012045   040   115   105             .ASCII  / ME/
012050   104   111   101             .ASCII  /DIA/
012053   000                         .ASCII  <00>
012054   040   055   040   P.AFK:    .ASCII  / - /
012057   105   122   122             .ASCII  /ERR/
012062   040   111   116             .ASCII  / IN/
012065   040   104   125             .ASCII  / DU/
012070   120   040   120             .ASCII  /P P/
012073   113   124   040             .ASCII  /KT /
012076   127   110   105             .ASCII  /WHE/
012101   116   040   125             .ASCII  /N U/
012104   123   111   116             .ASCII  /SIN/
012107   107   040   103             .ASCII  /G C/
012112   124   114   122             .ASCII  /TLR/
012115   040   114   103             .ASCII  / LC/
012120   040   120   122             .ASCII  / PR/
012123   117   107   000             .ASCII  /OG/<00>
012126   045   101   052   P.AFL:    .ASCII  /%A*/
012131   040   104   111             .ASCII  / DI/
012134   123   113   045             .ASCII  /SK%/
012137   104   062   000             .ASCII  /D2/<00>
012142   045   101   111   P.AFN:    .ASCII  /%AI/
012145   116   126   101             .ASCII  /NVA/
012150   114   111   104             .ASCII  /LID/
012153   040   103   117             .ASCII  / CO/
012156   115   115   101             .ASCII  /MMA/
012161   116   104   000             .ASCII  /ND/<00>
012164   045   101   103   P.AFO:    .ASCII  /%AC/
012167   117   115   115             .ASCII  /OMM/
012172   101   116   104             .ASCII  /AND/
012175   040   101   102             .ASCII  / AB/
012200   117   122   124             .ASCII  /ORT/
012203   105   104   000             .ASCII  /ED/<00>
012206   045   101   125   P.AFP:    .ASCII  /%AU/
012211   116   111   124             .ASCII  /NIT/
012214   040   117   106             .ASCII  / OF/
012217   106   114   111             .ASCII  /FLI/
012222   116   105   000             .ASCII  /NE/<00>
012225   000                         .ASCII  <00>
012226   045   101   124   P.AFQ:    .ASCII  /%AT/
012231   122   101   116             .ASCII  /RAN/
012234   123   111   124             .ASCII  /SIT/
012237   111   117   116             .ASCII  /ION/
```

```
012242  040  124  117            .ASCII  / TO/
012245  040  101  126            .ASCII  / AV/
012250  101  111  114            .ASCII  /AIL/
012253  101  102  114            .ASCII  /ABL/
012256  105  040  123            .ASCII  /E S/
012261  124  101  124            .ASCII  /TAT/
012264  105  000                 .ASCII  /E/<00>
012266  045  101  115    P.AFR:  .ASCII  /%AM/
012271  105  104  111            .ASCII  /EDI/
012274  101  040  106            .ASCII  /A F/
012277  117  122  115            .ASCII  /ORM/
012302  101  124  040            .ASCII  /AT /
012305  105  122  122            .ASCII  /ERR/
012310  117  122  000            .ASCII  /OR/<00>
012313  000                      .ASCII  <00>
012314  045  101  127    P.AFS:  .ASCII  /%AW/
012317  122  111  124            .ASCII  /RIT/
012322  105  055  120            .ASCII  /E-P/
012325  122  117  124            .ASCII  /ROT/
012330  105  103  124            .ASCII  /ECT/
012333  105  104  000            .ASCII  /ED/<00>
012336  045  101  104    P.AFT:  .ASCII  /%AD/
012341  105  126  111            .ASCII  /EVI/
012344  103  105  040            .ASCII  /CE /
012347  103  117  115            .ASCII  /COM/
012352  120  101  122            .ASCII  /PAR/
012355  105  040  105            .ASCII  /E E/
012360  122  122  117            .ASCII  /RRO/
012363  122  000  000            .ASCII  /R/<00><00>
012366  045  101  104    P.AFU:  .ASCII  /%AD/
012371  101  124  101            .ASCII  /ATA/
012374  040  105  122            .ASCII  / ER/
012377  122  117  122            .ASCII  /ROR/
012402  000  000                 .ASCII  <00><00>
012404  045  101  110    P.AFV:  .ASCII  /%AH/
012407  117  123  124            .ASCII  /OST/
012412  040  102  125            .ASCII  / BU/
012415  106  106  105            .ASCII  /FFE/
012420  122  040  101            .ASCII  /R A/
012423  103  103  105            .ASCII  /CCE/
012426  123  123  040            .ASCII  /SS /
012431  105  122  122            .ASCII  /ERR/
012434  117  122  000            .ASCII  /OR/<00>
012437  000                      .ASCII  <00>
012440  045  101  103    P.AFW:  .ASCII  /%AC/
012443  117  116  124            .ASCII  /ONT/
012446  122  117  114            .ASCII  /ROL/
012451  114  105  122            .ASCII  /LER/
012454  040  105  122            .ASCII  / ER/
012457  122  117  122            .ASCII  /ROR/
012462  000  000                 .ASCII  <00><00>
012464  045  101  104    P.AFX:  .ASCII  /%AD/
012467  122  111  126            .ASCII  /RIV/
```

```
012472      105     040     105                 .ASCII  /E E/
012475      122     122     117                 .ASCII  /RRO/
012500      122     000                         .ASCII  /R/<00>
012502      045     101     115     P.AFY:      .ASCII  /%AM/
012505      105     123     123                 .ASCII  /ESS/
012510      101     107     105                 .ASCII  /AGE/
012513      040     106     122                 .ASCII  / FR/
012516      117     115     040                 .ASCII  /OM /
012521      111     116     124                 .ASCII  /INT/
012524      105     122     116                 .ASCII  /ERN/
012527      101     114     040                 .ASCII  /AL /
012532      104     111     101                 .ASCII  /DIA/
012535      107     116     117                 .ASCII  /GNO/
012540      123     124     111                 .ASCII  /STI/
012543      103     123     000                 .ASCII  /CS/<00>
012546      045     101     110     P.AFZ:      .ASCII  /%AH/
012551      117     123     124                 .ASCII  /OST/
012554      040     103     117                 .ASCII  / CO/
012557      115     120     101                 .ASCII  /MPA/
012562      122     105     040                 .ASCII  /RE /
012565      105     122     122                 .ASCII  /ERR/
012570      117     122     000                 .ASCII  /OR/<00>
012573      000                                 .ASCII  <00>
012574      045     101     103     P.AGA:      .ASCII  /%AC/
012577      117     115     115                 .ASCII  /OMM/
012602      101     116     104                 .ASCII  /AND/
012605      040     124     111                 .ASCII  / TI/
012610      115     105     117                 .ASCII  /MEO/
012613      125     124     000                 .ASCII  /UT/<00>
012616      045     101     102     P.AGB:      .ASCII  /%AB/
012621      101     104     040                 .ASCII  /AD /
012624      102     114     117                 .ASCII  /BLO/
012627      103     113     040                 .ASCII  /CK /
012632      122     105     120                 .ASCII  /REP/
012635      114     101     103                 .ASCII  /LAC/
012640      105     115     105                 .ASCII  /EME/
012643      116     124     040                 .ASCII  /NT /
012646      103     117     115                 .ASCII  /COM/
012651      120     114     105                 .ASCII  /PLE/
012654      124     111     117                 .ASCII  /TIO/
012657      116     000     000                 .ASCII  /N/<00><00>
012662   012142'                    P.AFM:      .WORD   P.AFN
012664   012164'                                .WORD   P.AFO
012666   012206'                                .WORD   P.AFP
012670   012226'                                .WORD   P.AFQ
012672   012266'                                .WORD   P.AFR
012674   012314'                                .WORD   P.AFS
012676   012336'                                .WORD   P.AFT
012700   012366'                                .WORD   P.AFU
012702   012404'                                .WORD   P.AFV
012704   012440'                                .WORD   P.AFW
012706   012464'                                .WORD   P.AFX
012710   012502'                                .WORD   P.AFY
```

```
012712  012546'                          .WORD   P.AFZ
012714  012574'                          .WORD   P.AGA
012716  012616'                          .WORD   P.AGB
012720     045     101     105   P.AGC:  .ASCII  /%AE/
012723     122     122     117           .ASCII  /RRO/
012726     122     040     114           .ASCII  /R L/
012731     117     107     040           .ASCII  /OG /
012734     115     105     123           .ASCII  /MES/
012737     123     101     107           .ASCII  /SAG/
012742     105     040     122           .ASCII  /E R/
012745     105     103     105           .ASCII  /ECE/
012750     111     126     105           .ASCII  /IVE/
012753     104     072     045           .ASCII  /D:%/
012756     116     000                   .ASCII  /N/<00>
012760     045     101     052   P.AGE:  .ASCII  /%A*/
012763     040     103     117           .ASCII  / CO/
012766     116     124     122           .ASCII  /NTR/
012771     117     114     114           .ASCII  /OLL/
012774     105     122     040           .ASCII  /ER /
012777     105     122     122           .ASCII  /ERR/
013002     117     122     045           .ASCII  /OR%/
013005     116     000     000           .ASCII  /N/<00><00>
013010     045     101     052   P.AGF:  .ASCII  /%A*/
013013     040     110     117           .ASCII  / HO/
013016     123     124     040           .ASCII  /ST /
013021     115     105     115           .ASCII  /MEM/
013024     117     122     131           .ASCII  /ORY/
013027     040     101     103           .ASCII  / AC/
013032     103     105     123           .ASCII  /CES/
013035     123     040     105           .ASCII  /S E/
013040     122     122     117           .ASCII  /RRO/
013043     122     045     116           .ASCII  /R%N/
013046     000     000                   .ASCII  <00><00>
013050     045     101     052   P.AGG:  .ASCII  /%A*/
013053     040     104     111           .ASCII  / DI/
013056     123     113     045           .ASCII  /SK%/
013061     104     062     045           .ASCII  /D2%/
013064     101     040     055           .ASCII  /A -/
013067     040     104     111           .ASCII  / DI/
013072     123     113     040           .ASCII  /SK /
013075     124     122     101           .ASCII  /TRA/
013100     116     123     106           .ASCII  /NSF/
013103     105     122     040           .ASCII  /ER /
013106     105     122     122           .ASCII  /ERR/
013111     117     122     045           .ASCII  /OR%/
013114     116     000                   .ASCII  /N/<00>
013116     045     101     052   P.AGH:  .ASCII  /%A*/
013121     040     104     111           .ASCII  / DI/
013124     123     113     045           .ASCII  /SK%/
013127     104     062     045           .ASCII  /D2%/
013132     101     040     055           .ASCII  /A -/
013135     040     042     123           .ASCII  / "S/
013140     124     101     116           .ASCII  /TAN/
```

```
013143    104    101    122                   .ASCII  /DAR/
013146    104    040    104                   .ASCII  /D D/
013151    111    123    113                   .ASCII  /ISK/
013154    040    111    116                   .ASCII  / IN/
013157    124    105    122                   .ASCII  /TER/
013162    103    117    116                   .ASCII  /CON/
013165    116    105    103                   .ASCII  /NEC/
013170    124    042    040                   .ASCII  /T" /
013173    105    122    122                   .ASCII  /ERR/
013176    117    122    045                   .ASCII  /OR%/
013201    116    000    000                   .ASCII  /N/<00><00>
013204    045    101    052         P.AGI:    .ASCII  /%A*/
013207    040    104    111                   .ASCII  / DI/
013212    123    113    045                   .ASCII  /SK%/
013215    104    062    045                   .ASCII  /D2%/
013220    101    040    055                   .ASCII  /A -/
013223    040    042    123                   .ASCII  / "S/
013226    115    101    114                   .ASCII  /MAL/
013231    114    040    104                   .ASCII  /L D/
013234    111    123    113                   .ASCII  /ISK/
013237    042    040    105                   .ASCII  /" E/
013242    122    122    117                   .ASCII  /RRO/
013245    122    045    116                   .ASCII  /R%N/
013250    000    000                          .ASCII  <00><00>
013252    045    101    052         P.AGJ:    .ASCII  /%A*/
013255    040    104    111                   .ASCII  / DI/
013260    123    113    045                   .ASCII  /SK%/
013263    104    062    045                   .ASCII  /D2%/
013266    101    040    055                   .ASCII  /A -/
013271    040    042    102                   .ASCII  / "B/
013274    101    104    040                   .ASCII  /AD /
013277    102    114    117                   .ASCII  /BLO/
013302    103    113    040                   .ASCII  /CK /
013305    122    105    120                   .ASCII  /REP/
013310    114    101    103                   .ASCII  /LAC/
013313    105    115    105                   .ASCII  /EME/
013316    116    124    040                   .ASCII  /NT /
013321    101    124    124                   .ASCII  /ATT/
013324    105    115    120                   .ASCII  /EMP/
013327    124    042    045                   .ASCII  /T"%/
013332    116    000                          .ASCII  /N/<00>
013334  012760'                     P.AGD:    .WORD   P.AGE
013336  013010'                               .WORD   P.AGF
013340  013050'                               .WORD   P.AGG
013342  013116'                               .WORD   P.AGH
013344  013204'                               .WORD   P.AGI
013346  013252'                               .WORD   P.AGJ
013350    045    116    045         P.AGK:    .ASCII  /%N%/
013353    101    052    040                   .ASCII  /A* /
013356    123    101    072                   .ASCII  /SA:/
013361    040    045    117                   .ASCII  / %O/
013364    066    000                          .ASCII  /6/<00>
013366    045    116    045         P.AGL:    .ASCII  /%N%/
```

```
013371  101  052  040             .ASCII  /A* /
013374  123  124  101             .ASCII  /STA/
013377  124  125  123             .ASCII  /TUS/
013402  040  103  117             .ASCII  / CO/
013405  104  105  072             .ASCII  /DE:/
013410  040  045  117             .ASCII  / %O/
013413  062  000  000             .ASCII  /2/<00><00>
013416  045  117  064    P.AGM:   .ASCII  /%O4/
013421  000                       .ASCII  <00>
013422  045  116  045    P.AGN:   .ASCII  /%N%/
013425  101  052  040             .ASCII  /A* /
013430  123  125  102             .ASCII  /SUB/
013433  137  103  117             .ASCII  /_CO/
013436  104  105  072             .ASCII  /DE:/
013441  040  000  000             .ASCII  / /<00><00>
013444  045  116  045    P.AGO:   .ASCII  /%N%/
013447  101  052  040             .ASCII  /A* /
013452  103  117  115             .ASCII  /COM/
013455  115  101  116             .ASCII  /MAN/
013460  104  072  040             .ASCII  /D: /
013463  000                       .ASCII  <00>
013464  045  101  122    P.AGP:   .ASCII  /%AR/
013467  105  101  104             .ASCII  /EAD/
013472  000  000                  .ASCII  <00><00>
013474  045  101  127    P.AGQ:   .ASCII  /%AW/
013477  122  111  124             .ASCII  /RIT/
013502  105  000                  .ASCII  /E/<00>
013504  045  101  055    P.AGR:   .ASCII  /%A-/
013507  103  117  115             .ASCII  /COM/
013512  120  101  122             .ASCII  /PAR/
013515  105  000  000             .ASCII  /E/<00><00>
013520  045  101  117    P.AGS:   .ASCII  /%AO/
013523  116  114  111             .ASCII  /NLI/
013526  116  105  000             .ASCII  /NE/<00>
013531  000                       .ASCII  <00>
013532  045  101  101    P.AGT:   .ASCII  /%AA/
013535  103  103  105             .ASCII  /CCE/
013540  123  123  000             .ASCII  /SS/<00>
013543  000                       .ASCII  <00>
013544  045  117  063    P.AGU:   .ASCII  /%O3/
013547  000                       .ASCII  <00>
013550  045  116  045    P.AGV:   .ASCII  /%N%/
013553  101  052  040             .ASCII  /A* /
013556  102  101  104             .ASCII  /BAD/
013561  040  102  114             .ASCII  / BL/
013564  117  103  113             .ASCII  /OCK/
013567  072  040  045             .ASCII  /: %/
013572  117  066  045             .ASCII  /O6%/
013575  101  040  045             .ASCII  /A %/
013600  117  066  045             .ASCII  /O6%/
013603  101  040  040             .ASCII  /A  /
013606  050  117  103             .ASCII  /(OC/
013611  124  101  114             .ASCII  /TAL/
```

```
013614   051   000                   .ASCII  /)/<00>
013616   045   116   045   P.AGW:    .ASCII  /%N%/
013621   101   052   040             .ASCII  /A* /
013624   061   123   124             .ASCII  /1ST/
013627   040   102   101             .ASCII  / BA/
013632   104   040   102             .ASCII  /D B/
013635   114   117   103             .ASCII  /LOC/
013640   113   072   040             .ASCII  /K: /
013643   045   117   066             .ASCII  /%O6/
013646   045   101   040             .ASCII  /%A /
013651   045   117   066             .ASCII  /%O6/
013654   045   101   040             .ASCII  /%A /
013657   040   050   117             .ASCII  / (O/
013662   103   124   101             .ASCII  /CTA/
013665   114   051   000             .ASCII  /L)/<00>
013670   045   116   045   P.AGX:    .ASCII  /%N%/
013673   101   052   040             .ASCII  /A* /
013676   102   101   104             .ASCII  /BAD/
013701   040   102   114             .ASCII  / BL/
013704   117   103   113             .ASCII  /OCK/
013707   040   122   105             .ASCII  / RE/
013712   120   114   101             .ASCII  /PLA/
013715   103   105   104             .ASCII  /CED/
013720   072   040   045             .ASCII  /: %/
013723   117   066   045             .ASCII  /O6%/
013726   101   040   045             .ASCII  /A %/
013731   117   066   045             .ASCII  /O6%/
013734   101   040   040             .ASCII  /A  /
013737   050   117   103             .ASCII  /(OC/
013742   124   101   114             .ASCII  /TAL/
013745   051   000   000             .ASCII  /)/<00><00>
013750   045   116   045   P.AGY:    .ASCII  /%N%/
013753   101   052   040             .ASCII  /A* /
013756   114   102   116             .ASCII  /LBN/
013761   072   040   045             .ASCII  /: %/
013764   117   066   045             .ASCII  /O6%/
013767   101   040   045             .ASCII  /A %/
013772   117   066   045             .ASCII  /O6%/
013775   101   040   040             .ASCII  /A  /
014000   050   117   103             .ASCII  /(OC/
014003   124   101   114             .ASCII  /TAL/
014006   051   000                   .ASCII  /)/<00>
014010   045   116   045   P.AGZ:    .ASCII  /%N%/
014013   101   052   040             .ASCII  /A* /
014016   120   102   116             .ASCII  /PBN/
014021   072   040   045             .ASCII  /: %/
014024   117   066   045             .ASCII  /O6%/
014027   101   040   045             .ASCII  /A %/
014032   117   066   045             .ASCII  /O6%/
014035   101   040   040             .ASCII  /A  /
014040   050   117   103             .ASCII  /(OC/
014043   124   101   114             .ASCII  /TAL/
014046   051   000                   .ASCII  /)/<00>
```

```
014050   045   116   045        P.AHA:  .ASCII  /%N%/
014053   101   052   040                .ASCII  /A* /
014056   114   102   116                .ASCII  /LBN/
014061   040   122   105                .ASCII  / RE/
014064   101   104   072                .ASCII  /AD:/
014067   040   045   117                .ASCII  / %O/
014072   066   045   101                .ASCII  /6%A/
014075   040   045   117                .ASCII  / %O/
014100   066   045   101                .ASCII  /6%A/
014103   040   040   050                .ASCII  /  (/
014106   117   103   124                .ASCII  /OCT/
014111   101   114   051                .ASCII  /AL)/
014114   000   000                      .ASCII  <00><00>
014116   045   116   045        P.AHB:  .ASCII  /%N%/
014121   101   052   040                .ASCII  /A* /
014124   114   102   116                .ASCII  /LBN/
014127   040   127   122                .ASCII  / WR/
014132   111   124   124                .ASCII  /ITT/
014135   105   116   072                .ASCII  /EN:/
014140   040   045   117                .ASCII  / %O/
014143   066   045   101                .ASCII  /6%A/
014146   040   045   117                .ASCII  / %O/
014151   066   045   101                .ASCII  /6%A/
014154   040   040   050                .ASCII  /  (/
014157   117   103   124                .ASCII  /OCT/
014162   101   114   051                .ASCII  /AL)/
014165   000                            .ASCII  <00>
014166   045   116   045        P.AHC:  .ASCII  /%N%/
014171   101   052   040                .ASCII  /A* /
014174   122   102   116                .ASCII  /RBN/
014177   072   040   045                .ASCII  /: %/
014202   117   066   045                .ASCII  /O6%/
014205   101   040   045                .ASCII  /A %/
014210   117   066   045                .ASCII  /O6%/
014213   101   040   040                .ASCII  /A  /
014216   050   117   103                .ASCII  /(OC/
014221   124   101   114                .ASCII  /TAL/
014224   051   000                      .ASCII  /)/<00>
014226   045   116   045        P.AHD:  .ASCII  /%N%/
014231   101   052   040                .ASCII  /A* /
014234   102   131   124                .ASCII  /BYT/
014237   105   040   103                .ASCII  /E C/
014242   117   125   116                .ASCII  /OUN/
014245   124   040   111                .ASCII  /T I/
014250   116   040   103                .ASCII  /N C/
014253   117   115   115                .ASCII  /OMM/
014256   101   116   104                .ASCII  /AND/
014261   072   040   045                .ASCII  /: %/
014264   104   065   045                .ASCII  /D5%/
014267   101   056   000                .ASCII  /A./<00>
014272   045   116   045        P.AHE:  .ASCII  /%N%/
014275   101   052   040                .ASCII  /A* /
014300   102   131   124                .ASCII  /BYT/
```

```
014303  105  040  103           .ASCII  /E C/
014306  117  125  116           .ASCII  /OUN/
014311  124  040  111           .ASCII  /T I/
014314  116  040  122           .ASCII  /N R/
014317  105  101  104           .ASCII  /EAD/
014322  040  103  117           .ASCII  / CO/
014325  115  115  101           .ASCII  /MMA/
014330  116  104  072           .ASCII  /ND:/
014333  040  045  104           .ASCII  / %D/
014336  065  045  101           .ASCII  /5%A/
014341  056  000  000           .ASCII  /./<00><00>
014344  045  116  045   P.AHF:  .ASCII  /%N%/
014347  101  052  040           .ASCII  /A* /
014352  102  131  124           .ASCII  /BYT/
014355  105  040  103           .ASCII  /E C/
014360  117  125  116           .ASCII  /OUN/
014363  124  040  111           .ASCII  /T I/
014366  116  040  127           .ASCII  /N W/
014371  122  111  124           .ASCII  /RIT/
014374  105  040  103           .ASCII  /E C/
014377  117  115  115           .ASCII  /OMM/
014402  101  116  104           .ASCII  /AND/
014405  072  040  045           .ASCII  /: %/
014410  104  065  045           .ASCII  /D5%/
014413  101  056  000           .ASCII  /A./<00>
014416  045  116  045   P.AHG:  .ASCII  /%N%/
014421  101  052  040           .ASCII  /A* /
014424  101  103  124           .ASCII  /ACT/
014427  125  101  114           .ASCII  /UAL/
014432  040  043  040           .ASCII  / # /
014435  117  106  040           .ASCII  /OF /
014440  102  131  124           .ASCII  /BYT/
014443  105  123  040           .ASCII  /ES /
014446  124  122  101           .ASCII  /TRA/
014451  116  123  106           .ASCII  /NSF/
014454  105  122  122           .ASCII  /ERR/
014457  105  104  072           .ASCII  /ED:/
014462  040  045  104           .ASCII  / %D/
014465  065  045  101           .ASCII  /5%A/
014470  056  000                .ASCII  /./<00>
014472  045  116  045   P.AHH:  .ASCII  /%N%/
014475  101  052  040           .ASCII  /A* /
014500  111  057  117           .ASCII  /I/<57>/O/
014503  040  102  125           .ASCII  / BU/
014506  106  106  105           .ASCII  /FFE/
014511  122  040  101           .ASCII  /R A/
014514  104  104  122           .ASCII  /DDR/
014517  105  123  123           .ASCII  /ESS/
014522  040  050  063           .ASCII  / (3/
014525  062  040  142           .ASCII  /2 b/
014530  151  164  163           .ASCII  /its/
014533  051  072  040           .ASCII  /): /
014536  045  117  066           .ASCII  /%O6/
```

```
014541   045   101   040                  .ASCII  /$A /
014544   045   117   066                  .ASCII  /$06/
014547   000                              .ASCII  <00>
014550   045   116   045          P.AHI:  .ASCII  /$N$/
014553   101   052   040                  .ASCII  /A* /
014556   111   057   117                  .ASCII  /I/<57>/O/
014561   040   102   125                  .ASCII  / BU/
014564   106   106   105                  .ASCII  /FFE/
014567   122   040   101                  .ASCII  /R A/
014572   104   104   122                  .ASCII  /DDR/
014575   105   123   123                  .ASCII  /ESS/
014600   040   106   117                  .ASCII  / FO/
014603   122   040   122                  .ASCII  /R R/
014606   105   101   104                  .ASCII  /EAD/
014611   040   050   063                  .ASCII  / (3/
014614   062   040   142                  .ASCII  /2 b/
014617   151   164   163                  .ASCII  /its/
014622   051   072   040                  .ASCII  /): /
014625   045   117   066                  .ASCII  /$06/
014630   045   101   040                  .ASCII  /$A /
014633   045   117   066                  .ASCII  /$06/
014636   000   000                        .ASCII  <00><00>
014640   045   116   045          P.AHJ:  .ASCII  /$N$/
014643   101   052   040                  .ASCII  /A* /
014646   111   057   117                  .ASCII  /I/<57>/O/
014651   040   102   125                  .ASCII  / BU/
014654   106   106   105                  .ASCII  /FFE/
014657   122   040   101                  .ASCII  /R A/
014662   104   104   122                  .ASCII  /DDR/
014665   105   123   123                  .ASCII  /ESS/
014670   040   106   117                  .ASCII  / FO/
014673   122   040   127                  .ASCII  /R W/
014676   122   111   124                  .ASCII  /RIT/
014701   105   040   050                  .ASCII  /E (/
014704   063   062   040                  .ASCII  /32 /
014707   142   151   164                  .ASCII  /bit/
014712   163   051   072                  .ASCII  /s):/
014715   040   045   117                  .ASCII  / $O/
014720   066   045   101                  .ASCII  /6$A/
014723   040   045   117                  .ASCII  / $O/
014726   066   000                        .ASCII  /6/<00>
014730   045   116   045          P.AHK:  .ASCII  /$N$/
014733   101   103   117                  .ASCII  /ACO/
014736   116   124   105                  .ASCII  /NTE/
014741   116   124   123                  .ASCII  /NTS/
014744   040   117   106                  .ASCII  / OF/
014747   040   103   117                  .ASCII  / CO/
014752   115   115   101                  .ASCII  /MMA/
014755   116   104   057                  .ASCII  /ND/<57>/
014760   122   105   123                  .ASCII  /RES/
014763   120   117   116                  .ASCII  /PON/
014766   123   105   040                  .ASCII  /SE /
014771   120   101   103                  .ASCII  /PAC/
```

```
014774  113  105  124              .ASCII  /KET/
014777  040  123  101              .ASCII  / SA/
015002  126  105  040              .ASCII  /VE /
015005  101  122  105              .ASCII  /ARE/
015010  101  072  045              .ASCII  /A:%/
015013  116  000  000              .ASCII  /N/<00><00>
015016  045  101  040     P.AHL:   .ASCII  /%A /
015021  045  117  066              .ASCII  /%O6/
015024  000  000                   .ASCII  <00><00>
015026  045  116  045     P.AHM:   .ASCII  /%N%/
015031  101  124  111              .ASCII  /ATI/
015034  115  105  072              .ASCII  /ME:/
015037  040  045  132              .ASCII  / %Z/
015042  062  045  101              .ASCII  /2%A/
015045  072  045  132              .ASCII  /:%Z/
015050  062  045  101              .ASCII  /2%A/
015053  040  110  117              .ASCII  / HO/
015056  125  122  123              .ASCII  /URS/
015061  045  116  000              .ASCII  /%N/<00>
015064  045  116  045     P.AHN:   .ASCII  /%N%/
015067  101  052  040              .ASCII  /A* /
015072  102  101  104              .ASCII  /BAD/
015075  040  114  102              .ASCII  / LB/
015100  116  072  040              .ASCII  /N: /
015103  045  117  066              .ASCII  /%O6/
015106  045  101  040              .ASCII  /%A /
015111  045  117  066              .ASCII  /%O6/
015114  045  101  040              .ASCII  /%A /
015117  040  050  117              .ASCII  / (O/
015122  103  124  101              .ASCII  /CTA/
015125  114  051  000              .ASCII  /L)/<00>
015130  045  116  045     P.AHO:   .ASCII  /%N%/
015133  101  040  052              .ASCII  /A */
015136  040  104  111              .ASCII  / DI/
015141  123  113  040              .ASCII  /SK /
015144  072  040  045              .ASCII  /: %/
015147  104  062  000              .ASCII  /D2/<00>
015152  045  116  045     P.AHP:   .ASCII  /%N%/
015155  101  104  102              .ASCII  /ADB/
015160  116  072  040              .ASCII  /N: /
015163  045  104  065              .ASCII  /%D5/
015166  045  101  056              .ASCII  /%A./
015171  040  050  117              .ASCII  / (O/
015174  103  124  040              .ASCII  /CT /
015177  045  117  066              .ASCII  /%O6/
015202  045  101  051              .ASCII  /%A)/
015205  000                        .ASCII  <00>
015206  045  116  045     P.AHQ:   .ASCII  /%N%/
015211  101  102  131              .ASCII  /ABY/
015214  124  105  040              .ASCII  /TE /
015217  116  125  115              .ASCII  /NUM/
015222  102  105  122              .ASCII  /BER/
015225  072  040  045              .ASCII  /: %/
```

```
015230   104   063   000                .ASCII  /D3/<00>
015233   000                            .ASCII  <00>
015234   045   116   045       P.AHR:   .ASCII  /%N%/
015237   101   122   101                .ASCII  /ARA/
015242   116   104   117                .ASCII  /NDO/
015245   115   040   127                .ASCII  /M W/
015250   122   111   124                .ASCII  /RIT/
015253   124   105   116                .ASCII  /TEN/
015256   040   127   117                .ASCII  / WO/
015261   122   104   040                .ASCII  /RD /
015264   072   045   102                .ASCII  /:%B/
015267   061   066   000                .ASCII  /16/<00>
015272   045   116   045       P.AHS:   .ASCII  /%N%/
015275   101   122   101                .ASCII  /ARA/
015300   116   104   117                .ASCII  /NDO/
015303   115   040   122                .ASCII  /M R/
015306   105   101   104                .ASCII  /EAD/
015311   040   127   117                .ASCII  / WO/
015314   122   104   040                .ASCII  /RD /
015317   142   151   156                .ASCII  /bin/
015322   072   045   102                .ASCII  /:%B/
015325   061   066   045                .ASCII  /16%/
015330   101   040   157                .ASCII  /A o/
015333   143   164   072                .ASCII  /ct:/
015336   045   117   066                .ASCII  /%O6/
015341   000                            .ASCII  <00>
015342   045   116   045       P.AHT:   .ASCII  /%N%/
015345   101   104   125                .ASCII  /ADU/
015350   120   114   111                .ASCII  /PLI/
015353   103   101   124                .ASCII  /CAT/
015356   105   040   125                .ASCII  /E U/
015361   116   111   124                .ASCII  /NIT/
015364   072   045   104                .ASCII  /:%D/
015367   062   045   101                .ASCII  /2%A/
015372   040   101   124                .ASCII  / AT/
015375   040   111   120                .ASCII  / IP/
015400   072   040   045                .ASCII  /: %/
015403   117   066   000                .ASCII  /O6/<00>
015406   045   116   045       P.AHU:   .ASCII  /%N%/
015411   101   115   117                .ASCII  /AMO/
015414   122   105   040                .ASCII  /RE /
015417   124   110   101                .ASCII  /THA/
015422   116   040   045                .ASCII  /N %/
015425   104   061   045                .ASCII  /D1%/
015430   101   040   104                .ASCII  /A D/
015433   111   106   106                .ASCII  /IFF/
015436   105   122   105                .ASCII  /ERE/
015441   116   124   040                .ASCII  /NT /
015444   111   120   040                .ASCII  /IP /
015447   101   104   104                .ASCII  /ADD/
015452   122   105   123                .ASCII  /RES/
015455   123   105   123                .ASCII  /SES/
015460   000   000                      .ASCII  <00><00>
```

```
015462  045  101  123      P.AHV:  .ASCII  /%AS/
015465  120  111  116              .ASCII  /PIN/
015470  055  104  117              .ASCII  /-DO/
015473  127  116  040              .ASCII  /WN /
015476  111  107  116              .ASCII  /IGN/
015501  117  122  105              .ASCII  /ORE/
015504  104  000                   .ASCII  /D/<00>
015506  045  101  123      P.AHW:  .ASCII  /%AS/
015511  124  111  114              .ASCII  /TIL/
015514  114  040  103              .ASCII  /L C/
015517  117  116  116              .ASCII  /ONN/
015522  105  103  124              .ASCII  /ECT/
015525  105  104  000              .ASCII  /ED/<00>
015530  045  101  104      P.AHX:  .ASCII  /%AD/
015533  125  120  114              .ASCII  /UPL/
015536  111  103  101              .ASCII  /ICA/
015541  124  105  040              .ASCII  /TE /
015544  125  116  111              .ASCII  /UNI/
015547  124  040  116              .ASCII  /T N/
015552  125  115  102              .ASCII  /UMB/
015555  105  122  000              .ASCII  /ER/<00>
015560  045  101  101      P.AHY:  .ASCII  /%AA/
015563  114  122  105              .ASCII  /LRE/
015566  101  104  131              .ASCII  /ADY/
015571  040  117  116              .ASCII  / ON/
015574  114  111  116              .ASCII  /LIN/
015577  105  000  000              .ASCII  /E/<00><00>
015602  045  101  123      P.AHZ:  .ASCII  /%AS/
015605  124  111  114              .ASCII  /TIL/
015610  114  040  117              .ASCII  /L O/
015613  116  114  111              .ASCII  /NLI/
015616  116  105  000              .ASCII  /NE/<00>
015621  000                        .ASCII  <00>
015622  045  101  111      P.AIA:  .ASCII  /%AI/
015625  116  103  117              .ASCII  /NCO/
015630  115  120  114              .ASCII  /MPL/
015633  105  124  105              .ASCII  /ETE/
015636  040  122  105              .ASCII  / RE/
015641  120  114  101              .ASCII  /PLA/
015644  103  105  115              .ASCII  /CEM/
015647  105  116  124              .ASCII  /ENT/
015652  000  000                   .ASCII  <00><00>
015654  045  101  111      P.AIB:  .ASCII  /%AI/
015657  116  126  101              .ASCII  /NVA/
015662  114  111  104              .ASCII  /LID/
015665  040  122  103              .ASCII  / RC/
015670  124  000                   .ASCII  /T/<00>
015672  045  101  125      P.AIC:  .ASCII  /%AU/
015675  116  111  124              .ASCII  /NIT/
015700  040  125  116              .ASCII  / UN/
015703  113  116  117              .ASCII  /KNO/
015706  127  116  040              .ASCII  /WN /
015711  117  122  040              .ASCII  /OR /
```

```
015714  117  116  114           .ASCII  /ONL/
015717  111  116  105           .ASCII  /INE/
015722  040  124  117           .ASCII  / TO/
015725  040  101  116           .ASCII  / AN/
015730  117  124  110           .ASCII  /OTH/
015733  105  122  040           .ASCII  /ER /
015736  103  117  116           .ASCII  /CON/
015741  124  122  117           .ASCII  /TRO/
015744  114  114  105           .ASCII  /LLE/
015747  122  000  000           .ASCII  /R/<00><00>
015752  045  101  116   P.AID:  .ASCII  /%AN/
015755  117  040  126           .ASCII  /O V/
015760  117  114  125           .ASCII  /OLU/
015763  115  105  040           .ASCII  /ME /
015766  115  117  125           .ASCII  /MOU/
015771  116  124  105           .ASCII  /NTE/
015774  104  040  117           .ASCII  /D O/
015777  122  040  104           .ASCII  /R D/
016002  122  111  126           .ASCII  /RIV/
016005  105  040  104           .ASCII  /E D/
016010  111  123  101           .ASCII  /ISA/
016013  102  114  105           .ASCII  /BLE/
016016  104  040  102           .ASCII  /D B/
016021  131  040  123           .ASCII  /Y S/
016024  127  111  124           .ASCII  /WIT/
016027  103  110  000           .ASCII  /CH/<00>
016032  045  101  125   P.AIE:  .ASCII  /%AU/
016035  116  111  124           .ASCII  /NIT/
016040  040  111  116           .ASCII  / IN/
016043  117  120  105           .ASCII  /OPE/
016046  122  101  124           .ASCII  /RAT/
016051  111  126  105           .ASCII  /IVE/
016054  040  050  122           .ASCII  / (R/
016057  104  065  061           .ASCII  /D51/
016062  057  065  062           .ASCII  <57>/52/
016065  040  167  162           .ASCII  / wr/
016070  151  164  145           .ASCII  /ite/
016073  040  146  141           .ASCII  / fa/
016076  165  154  164           .ASCII  /ult/
016101  051  000  000           .ASCII  /)/<00><00>
016104  045  101  125   P.AIF:  .ASCII  /%AU/
016107  116  111  124           .ASCII  /NIT/
016112  040  104  111           .ASCII  / DI/
016115  123  101  102           .ASCII  /SAB/
016120  114  105  104           .ASCII  /LED/
016123  040  102  131           .ASCII  / BY/
016126  040  106  111           .ASCII  / FI/
016131  105  114  104           .ASCII  /ELD/
016134  040  123  105           .ASCII  / SE/
016137  122  126  111           .ASCII  /RVI/
016142  103  105  040           .ASCII  /CE /
016145  117  122  040           .ASCII  /OR /
016150  111  116  124           .ASCII  /INT/
```

```
016153   105   122   116           .ASCII  /ERN/
016156   101   114   040           .ASCII  /AL /
016161   104   111   101           .ASCII  /DIA/
016164   107   116   117           .ASCII  /GNO/
016167   123   124   111           .ASCII  /STI/
016172   103   123   000           .ASCII  /CS/<00>
016175   000                       .ASCII  <00>
016176   045   101   042   P.AIG:  .ASCII  /%A"/
016201   106   117   122           .ASCII  /FOR/
016204   103   105   104           .ASCII  /CED/
016207   040   105   122           .ASCII  / ER/
016212   122   117   122           .ASCII  /ROR/
016215   042   040   104           .ASCII  /" D/
016220   105   124   105           .ASCII  /ETE/
016223   103   124   105           .ASCII  /CTE/
016226   104   040   127           .ASCII  /D W/
016231   110   111   114           .ASCII  /HIL/
016234   105   040   101           .ASCII  /E A/
016237   103   103   105           .ASCII  /CCE/
016242   123   123   111           .ASCII  /SSI/
016245   116   107   040           .ASCII  /NG /
016250   106   103   124           .ASCII  /FCT/
016253   040   117   122           .ASCII  / OR/
016256   040   122   103           .ASCII  / RC/
016261   124   000   000           .ASCII  /T/<00><00>
016264   045   101   123   P.AIH:  .ASCII  /%AS/
016267   105   103   124           .ASCII  /ECT/
016272   117   122   040           .ASCII  /OR /
016275   110   101   104           .ASCII  /HAD/
016300   040   102   105           .ASCII  / BE/
016303   105   116   040           .ASCII  /EN /
016306   127   122   111           .ASCII  /WRI/
016311   124   124   105           .ASCII  /TTE/
016314   116   040   127           .ASCII  /N W/
016317   111   124   110           .ASCII  /ITH/
016322   040   042   106           .ASCII  / "F/
016325   117   122   103           .ASCII  /ORC/
016330   105   104   040           .ASCII  /ED /
016333   105   122   122           .ASCII  /ERR/
016336   117   122   042           .ASCII  /OR"/
016341   040   115   117           .ASCII  / MO/
016344   104   111   106           .ASCII  /DIF/
016347   111   105   122           .ASCII  /IER/
016352   000   000                 .ASCII  <00><00>
016354   045   101   106   P.AII:  .ASCII  /%AF/
016357   103   124   040           .ASCII  /CT /
016362   117   122   040           .ASCII  /OR /
016365   122   103   124           .ASCII  /RCT/
016370   040   125   116           .ASCII  / UN/
016373   122   105   101           .ASCII  /REA/
016376   104   101   102           .ASCII  /DAB/
016401   114   105   040           .ASCII  /LE /
016404   055   040   111           .ASCII  /- I/
```

```
016407   116   126   101            .ASCII  /NVA/
016412   114   111   104            .ASCII  /LID/
016415   040   123   105            .ASCII  / SE/
016420   103   124   117            .ASCII  /CTO/
016423   122   040   110            .ASCII  /R H/
016426   105   101   104            .ASCII  /EAD/
016431   105   122   000            .ASCII  /ER/<00>
016434   045   101   110   P.AIJ:   .ASCII  /%AH/
016437   105   101   104            .ASCII  /EAD/
016442   105   122   040            .ASCII  /ER /
016445   103   117   115            .ASCII  /COM/
016450   120   101   122            .ASCII  /PAR/
016453   105   040   105            .ASCII  /E E/
016456   122   122   117            .ASCII  /RRO/
016461   122   040   050            .ASCII  /R (/
016464   126   141   154            .ASCII  /Val/
016467   151   144   040            .ASCII  /id /
016472   150   145   141            .ASCII  /hea/
016475   144   145   162            .ASCII  /der/
016500   040   156   157            .ASCII  / no/
016503   164   040   146            .ASCII  /t f/
016506   157   165   156            .ASCII  /oun/
016511   144   051   000            .ASCII  /d)/<00>
016514   045   101   106   P.AIK:   .ASCII  /%AF/
016517   103   124   040            .ASCII  /CT /
016522   117   122   040            .ASCII  /OR /
016525   122   103   124            .ASCII  /RCT/
016530   040   125   116            .ASCII  / UN/
016533   122   105   101            .ASCII  /REA/
016536   104   101   102            .ASCII  /DAB/
016541   114   105   040            .ASCII  /LE /
016544   055   040   104            .ASCII  /- D/
016547   101   124   101            .ASCII  /ATA/
016552   040   123   131            .ASCII  / SY/
016555   116   103   040            .ASCII  /NC /
016560   124   111   115            .ASCII  /TIM/
016563   105   117   125            .ASCII  /EOU/
016566   124   000                  .ASCII  /T/<00>
016570   045   101   104   P.AIL:   .ASCII  /%AD/
016573   101   124   101            .ASCII  /ATA/
016576   040   123   131            .ASCII  / SY/
016601   116   103   040            .ASCII  /NC /
016604   116   117   124            .ASCII  /NOT/
016607   040   106   117            .ASCII  / FO/
016612   125   116   104            .ASCII  /UND/
016615   040   050   104            .ASCII  / (D/
016620   141   164   141            .ASCII  /ata/
016623   040   163   171            .ASCII  / sy/
016626   156   143   040            .ASCII  /nc /
016631   164   151   155            .ASCII  /tim/
016634   145   157   165            .ASCII  /eou/
016637   164   051   000            .ASCII  /t)/<00>
016642   045   101   106   P.AIM:   .ASCII  /%AF/
```

```
016645   103   124   040              .ASCII  /CT /
016650   117   122   040              .ASCII  /OR /
016653   122   103   124              .ASCII  /RCT/
016656   040   125   116              .ASCII  / UN/
016661   122   105   101              .ASCII  /REA/
016664   104   101   102              .ASCII  /DAB/
016667   114   105   040              .ASCII  /LE /
016672   055   040   125              .ASCII  /- U/
016675   116   103   117              .ASCII  /NCO/
016700   122   122   105              .ASCII  /RRE/
016703   103   124   101              .ASCII  /CTA/
016706   102   114   105              .ASCII  /BLE/
016711   040   105   103              .ASCII  / EC/
016714   103   040   105              .ASCII  /C E/
016717   122   122   117              .ASCII  /RRO/
016722   122   000                    .ASCII  /R/<00>
016724   045   101   125    P.AIN:    .ASCII  /%AU/
016727   116   103   117              .ASCII  /NCO/
016732   122   122   105              .ASCII  /RRE/
016735   103   124   101              .ASCII  /CTA/
016740   102   114   105              .ASCII  /BLE/
016743   040   105   103              .ASCII  / EC/
016746   103   040   105              .ASCII  /C E/
016751   122   122   117              .ASCII  /RRO/
016754   122   000                    .ASCII  /R/<00>
016756   045   101   122    P.AIO:    .ASCII  /%AR/
016761   103   124   040              .ASCII  /CT /
016764   103   117   122              .ASCII  /COR/
016767   122   125   120              .ASCII  /RUP/
016772   124   105   104              .ASCII  /TED/
016775   000                          .ASCII  <00>
016776   045   101   116    P.AIP:    .ASCII  /%AN/
017001   117   040   122              .ASCII  /O R/
017004   105   120   114              .ASCII  /EPL/
017007   101   103   105              .ASCII  /ACE/
017012   115   105   116              .ASCII  /MEN/
017015   124   040   102              .ASCII  /T B/
017020   114   117   103              .ASCII  /LOC/
017023   113   040   101              .ASCII  /K A/
017026   126   101   111              .ASCII  /VAI/
017031   114   101   102              .ASCII  /LAB/
017034   114   105   040              .ASCII  /LE /
017037   050   122   103              .ASCII  /(RC/
017042   124   040   146              .ASCII  /T f/
017045   165   154   154              .ASCII  /ull/
017050   051   000                    .ASCII  /)/<00>
017052   045   101   104    P.AIQ:    .ASCII  /%AD/
017055   111   123   113              .ASCII  /ISK/
017060   040   116   117              .ASCII  / NO/
017063   124   040   106              .ASCII  /T F/
017066   117   122   115              .ASCII  /ORM/
017071   101   124   124              .ASCII  /ATT/
017074   105   104   040              .ASCII  /ED /
```

```
017077   127   111   124              .ASCII  /WIT/
017102   110   040   065              .ASCII  /H 5/
017105   061   062   040              .ASCII  /12 /
017110   102   131   124              .ASCII  /BYT/
017113   105   040   123              .ASCII  /E S/
017116   105   103   124              .ASCII  /ECT/
017121   117   122   123              .ASCII  /ORS/
017124   000   000                    .ASCII  <00><00>
017126   045   101   104     P.AIR:   .ASCII  /%AD/
017131   111   123   113              .ASCII  /ISK/
017134   040   116   117              .ASCII  / NO/
017137   124   040   106              .ASCII  /T F/
017142   117   122   115              .ASCII  /ORM/
017145   101   124   124              .ASCII  /ATT/
017150   105   104   040              .ASCII  /ED /
017153   117   122   040              .ASCII  /OR /
017156   106   103   124              .ASCII  /FCT/
017161   040   103   117              .ASCII  / CO/
017164   122   122   125              .ASCII  /RRU/
017167   120   124   105              .ASCII  /PTE/
017172   104   000                    .ASCII  /D/<00>
017174   045   101   117     P.AIS:   .ASCII  /%AO/
017177   116   105   040              .ASCII  /NE /
017202   123   131   115              .ASCII  /SYM/
017205   102   117   114              .ASCII  /BOL/
017210   040   105   103              .ASCII  / EC/
017213   103   040   105              .ASCII  /C E/
017216   122   122   117              .ASCII  /RRO/
017221   122   000   000              .ASCII  /R/<00><00>
017224   045   101   124     P.AIT:   .ASCII  /%AT/
017227   127   117   040              .ASCII  /WO /
017232   123   131   115              .ASCII  /SYM/
017235   102   117   114              .ASCII  /BOL/
017240   040   105   103              .ASCII  / EC/
017243   103   040   105              .ASCII  /C E/
017246   122   122   117              .ASCII  /RRO/
017251   122   000   000              .ASCII  /R/<00><00>
017254   045   101   124     P.AIU:   .ASCII  /%AT/
017257   110   122   105              .ASCII  /HRE/
017262   105   040   123              .ASCII  /E S/
017265   131   115   102              .ASCII  /YMB/
017270   117   114   040              .ASCII  /OL /
017273   105   103   103              .ASCII  /ECC/
017276   040   105   122              .ASCII  / ER/
017301   122   117   122              .ASCII  /ROR/
017304   000   000                    .ASCII  <00><00>
017306   045   101   106     P.AIV:   .ASCII  /%AF/
017311   117   125   122              .ASCII  /OUR/
017314   040   123   131              .ASCII  / SY/
017317   115   102   117              .ASCII  /MBO/
017322   114   040   105              .ASCII  /L E/
017325   103   103   040              .ASCII  /CC /
017330   105   122   122              .ASCII  /ERR/
```

```
017333   117   122   000                .ASCII  /OR/<00>
017336   045   101   106     P.AIW:     .ASCII  /%AF/
017341   111   126   105                .ASCII  /IVE/
017344   040   123   131                .ASCII  / SY/
017347   115   102   117                .ASCII  /MBO/
017352   114   040   105                .ASCII  /L E/
017355   103   103   040                .ASCII  /CC /
017360   105   122   122                .ASCII  /ERR/
017363   117   122   000                .ASCII  /OR/<00>
017366   045   101   123     P.AIX:     .ASCII  /%AS/
017371   111   130   040                .ASCII  /IX /
017374   123   131   115                .ASCII  /SYM/
017377   102   117   114                .ASCII  /BOL/
017402   040   105   103                .ASCII  / EC/
017405   103   040   105                .ASCII  /C E/
017410   122   122   117                .ASCII  /RRO/
017413   122   000   000                .ASCII  /R/<00><00>
017416   045   101   123     P.AIY:     .ASCII  /%AS/
017421   105   126   105                .ASCII  /EVE/
017424   116   040   123                .ASCII  /N S/
017427   131   115   102                .ASCII  /YMB/
017432   117   114   040                .ASCII  /OL /
017435   105   103   103                .ASCII  /ECC/
017440   040   105   122                .ASCII  / ER/
017443   122   117   122                .ASCII  /ROR/
017446   000   000                      .ASCII  <00><00>
017450   045   101   105     P.AIZ:     .ASCII  /%AE/
017453   111   107   110                .ASCII  /IGH/
017456   124   040   123                .ASCII  /T S/
017461   131   115   102                .ASCII  /YMB/
017464   117   114   040                .ASCII  /OL /
017467   105   103   103                .ASCII  /ECC/
017472   040   105   122                .ASCII  / ER/
017475   122   117   122                .ASCII  /ROR/
017500   000   000                      .ASCII  <00><00>
017502   045   101   103     P.AJA:     .ASCII  /%AC/
017505   117   122   122                .ASCII  /ORR/
017510   105   103   124                .ASCII  /ECT/
017513   101   102   114                .ASCII  /ABL/
017516   105   040   105                .ASCII  /E E/
017521   122   122   117                .ASCII  /RRO/
017524   122   040   111                .ASCII  /R I/
017527   116   040   105                .ASCII  /N E/
017532   103   103   040                .ASCII  /CC /
017535   106   111   105                .ASCII  /FIE/
017540   114   104   000                .ASCII  /LD/<00>
017543   000                            .ASCII  <00>
017544   045   101   125     P.AJB:     .ASCII  /%AU/
017547   116   111   124                .ASCII  /NIT/
017552   040   123   117                .ASCII  / SO/
017555   106   124   127                .ASCII  /FTW/
017560   101   122   105                .ASCII  /ARE/
017563   040   127   122                .ASCII  / WR/
```

```
017566   111   124   105              .ASCII  /ITE/
017571   040   120   122              .ASCII  / PR/
017574   117   124   105              .ASCII  /OTE/
017577   103   124   105              .ASCII  /CTE/
017602   104   000                    .ASCII  /D/<00>
017604   045   101   125    P.AJC:    .ASCII  /%AU/
017607   116   111   124              .ASCII  /NIT/
017612   040   110   101              .ASCII  / HA/
017615   122   104   127              .ASCII  /RDW/
017620   101   122   105              .ASCII  /ARE/
017623   040   127   122              .ASCII  / WR/
017626   111   124   105              .ASCII  /ITE/
017631   040   120   122              .ASCII  / PR/
017634   117   124   105              .ASCII  /OTE/
017637   103   124   105              .ASCII  /CTE/
017642   104   000                    .ASCII  /D/<00>
017644   045   101   125    P.AJD:    .ASCII  /%AU/
017647   116   111   124              .ASCII  /NIT/
017652   040   104   101              .ASCII  / DA/
017655   124   101   040              .ASCII  /TA /
017660   123   101   106              .ASCII  /SAF/
017663   105   124   131              .ASCII  /ETY/
017666   040   127   122              .ASCII  / WR/
017671   111   124   105              .ASCII  /ITE/
017674   040   120   122              .ASCII  / PR/
017677   117   124   105              .ASCII  /OTE/
017702   103   124   105              .ASCII  /CTE/
017705   104   000   000              .ASCII  /D/<00><00>
017710   045   101   117    P.AJE:    .ASCII  /%AO/
017713   104   104   040              .ASCII  /DD /
017716   124   122   101              .ASCII  /TRA/
017721   116   123   106              .ASCII  /NSF/
017724   105   122   040              .ASCII  /ER /
017727   101   104   104              .ASCII  /ADD/
017732   122   105   123              .ASCII  /RES/
017735   123   000   000              .ASCII  /S/<00><00>
017740   045   101   117    P.AJF:    .ASCII  /%AO/
017743   104   104   040              .ASCII  /DD /
017746   102   131   124              .ASCII  /BYT/
017751   105   040   103              .ASCII  /E C/
017754   117   125   116              .ASCII  /OUN/
017757   124   000   000              .ASCII  /T/<00><00>
017762   045   101   116    P.AJG:    .ASCII  /%AN/
017765   117   116   055              .ASCII  /ON-/
017770   105   130   111              .ASCII  /EXI/
017773   123   124   105              .ASCII  /STE/
017776   116   124   040              .ASCII  /NT /
020001   110   117   123              .ASCII  /HOS/
020004   124   040   115              .ASCII  /T M/
020007   105   115   117              .ASCII  /EMO/
020012   122   131   000              .ASCII  /RY/<00>
020015   000                          .ASCII  <00>
020016   045   101   110    P.AJH:    .ASCII  /%AH/
```

```
020021  117  123  124               .ASCII  /OST/
020024  040  115  105               .ASCII  / ME/
020027  115  117  122               .ASCII  /MOR/
020032  131  040  120               .ASCII  /Y P/
020035  101  122  111               .ASCII  /ARI/
020040  124  131  040               .ASCII  /TY /
020043  105  122  122               .ASCII  /ERR/
020046  117  122  000               .ASCII  /OR/<00>
020051  000                         .ASCII  <00>
020052  045  101  103     P.AJI:    .ASCII  /%AC/
020055  117  115  115               .ASCII  /OMM/
020060  101  116  104               .ASCII  /AND/
020063  040  124  111               .ASCII  / TI/
020066  115  117  125               .ASCII  /MOU/
020071  124  040  117               .ASCII  /T O/
020074  122  040  122               .ASCII  /R R/
020077  105  124  122               .ASCII  /ETR/
020102  131  040  114               .ASCII  /Y L/
020105  111  115  111               .ASCII  /IMI/
020110  124  040  105               .ASCII  /T E/
020113  130  103  105               .ASCII  /XCE/
020116  105  104  105               .ASCII  /EDE/
020121  104  000  000               .ASCII  /D/<00><00>
020124  045  101  123     P.AJJ:    .ASCII  /%AS/
020127  105  122  111               .ASCII  /ERI/
020132  101  114  111               .ASCII  /ALI/
020135  132  105  122               .ASCII  /ZER/
020140  057  104  105               .ASCII  <57>/DE/
020143  123  105  122               .ASCII  /SER/
020146  111  101  114               .ASCII  /IAL/
020151  111  132  105               .ASCII  /IZE/
020154  122  040  117               .ASCII  /R O/
020157  126  105  122               .ASCII  /VER/
020162  122  125  116               .ASCII  /RUN/
020165  040  117  122               .ASCII  / OR/
020170  040  125  116               .ASCII  / UN/
020173  104  105  122               .ASCII  /DER/
020176  122  125  116               .ASCII  /RUN/
020201  000                         .ASCII  <00>
020202  045  101  042     P.AJK:    .ASCII  /%A"/
020205  105  122  122               .ASCII  /ERR/
020210  117  122  040               .ASCII  /OR /
020213  104  105  124               .ASCII  /DET/
020216  105  103  124               .ASCII  /ECT/
020221  111  117  116               .ASCII  /ION/
020224  040  103  117               .ASCII  / CO/
020227  104  105  042               .ASCII  /DE"/
020232  040  105  122               .ASCII  / ER/
020235  122  117  122               .ASCII  /ROR/
020240  000  000                    .ASCII  <00><00>
020242  045  101  111     P.AJL:    .ASCII  /%AI/
020245  116  103  117               .ASCII  /NCO/
020250  116  123  111               .ASCII  /NSI/
```

```
020253  123  124  105           .ASCII  /STE/
020256  116  124  040           .ASCII  /NT /
020261  111  116  124           .ASCII  /INT/
020264  105  122  116           .ASCII  /ERN/
020267  101  114  040           .ASCII  /AL /
020272  104  101  124           .ASCII  /DAT/
020275  101  040  123           .ASCII  /A S/
020300  124  122  125           .ASCII  /TRU/
020303  103  124  125           .ASCII  /CTU/
020306  122  105  000           .ASCII  /RE/<00>
020311  000                     .ASCII  <00>
020312  045  101  104   P.AJM:  .ASCII  /%AD/
020315  122  111  126           .ASCII  /RIV/
020320  105  040  103           .ASCII  /E C/
020323  117  115  115           .ASCII  /OMM/
020326  101  116  104           .ASCII  /AND/
020331  040  124  111           .ASCII  / TI/
020334  115  105  117           .ASCII  /MEO/
020337  125  124  040           .ASCII  /UT /
020342  050  116  157           .ASCII  /(No/
020345  040  162  145           .ASCII  / re/
020350  163  160  157           .ASCII  /spo/
020353  156  163  145           .ASCII  /nse/
020356  040  157  162           .ASCII  / or/
020361  040  163  145           .ASCII  / se/
020364  145  153  040           .ASCII  /ek /
020367  151  156  143           .ASCII  /inc/
020372  157  155  160           .ASCII  /omp/
020375  154  145  164           .ASCII  /let/
020400  145  051  000           .ASCII  /e)/<00>
020403  000                     .ASCII  <00>
020404  045  101  103   P.AJN:  .ASCII  /%AC/
020407  117  116  124           .ASCII  /ONT/
020412  122  117  114           .ASCII  /ROL/
020415  114  105  122           .ASCII  /LER/
020420  040  104  105           .ASCII  / DE/
020423  124  105  103           .ASCII  /TEC/
020426  124  105  104           .ASCII  /TED/
020431  040  124  122           .ASCII  / TR/
020434  101  116  123           .ASCII  /ANS/
020437  115  111  123           .ASCII  /MIS/
020442  123  111  117           .ASCII  /SIO/
020445  116  040  117           .ASCII  /N O/
020450  122  040  120           .ASCII  /R P/
020453  122  117  124           .ASCII  /ROT/
020456  117  103  117           .ASCII  /OCO/
020461  114  040  105           .ASCII  /L E/
020464  122  122  117           .ASCII  /RRO/
020467  122  000  000           .ASCII  /R/<00><00>
020472  045  101  120   P.AJO:  .ASCII  /%AP/
020475  117  123  111           .ASCII  /OSI/
020500  124  111  117           .ASCII  /TIO/
020503  116  040  105           .ASCII  /N E/
```

```
020506  122  122  117             .ASCII  /RRO/
020511  122  040  050             .ASCII  /R (/
020514  115  151  163             .ASCII  /Mis/
020517  055  163  145             .ASCII  /-se/
020522  145  153  051             .ASCII  /ek)/
020525  000                       .ASCII  <00>
020526  045  101  114    P.AJP:   .ASCII  /%AL/
020531  117  123  124             .ASCII  /OST/
020534  040  122  105             .ASCII  / RE/
020537  101  104  057             .ASCII  /AD/<57>
020542  127  122  111             .ASCII  /WRI/
020545  124  105  040             .ASCII  /TE /
020550  122  105  101             .ASCII  /REA/
020553  104  131  040             .ASCII  /DY /
020556  104  125  122             .ASCII  /DUR/
020561  111  116  107             .ASCII  /ING/
020564  057  102  105             .ASCII  <57>/BE/
020567  124  127  105             .ASCII  /TWE/
020572  105  116  040             .ASCII  /EN /
020575  124  122  101             .ASCII  /TRA/
020600  116  123  106             .ASCII  /NSF/
020603  105  122  123             .ASCII  /ERS/
020606  000  000                  .ASCII  <00><00>
020610  045  101  104    P.AJQ:   .ASCII  /%AD/
020613  122  111  126             .ASCII  /RIV/
020616  105  040  103             .ASCII  /E C/
020621  114  117  103             .ASCII  /LOC/
020624  113  040  104             .ASCII  /K D/
020627  122  117  120             .ASCII  /ROP/
020632  117  125  124             .ASCII  /OUT/
020635  000                       .ASCII  <00>
020636  045  101  114    P.AJR:   .ASCII  /%AL/
020641  117  123  124             .ASCII  /OST/
020644  040  122  105             .ASCII  / RE/
020647  103  105  111             .ASCII  /CEI/
020652  126  105  122             .ASCII  /VER/
020655  040  122  105             .ASCII  / RE/
020660  101  104  131             .ASCII  /ADY/
020663  040  102  105             .ASCII  / BE/
020666  124  127  105             .ASCII  /TWE/
020671  105  116  040             .ASCII  /EN /
020674  123  105  103             .ASCII  /SEC/
020677  124  117  122             .ASCII  /TOR/
020702  123  000                  .ASCII  /S/<00>
020704  045  101  104    P.AJS:   .ASCII  /%AD/
020707  122  111  126             .ASCII  /RIV/
020712  105  040  104             .ASCII  /E D/
020715  105  124  105             .ASCII  /ETE/
020720  103  124  105             .ASCII  /CTE/
020723  104  040  105             .ASCII  /D E/
020726  122  122  117             .ASCII  /RRO/
020731  122  000  000             .ASCII  /R/<00><00>
020734  045  101  103    P.AJT:   .ASCII  /%AC/
```

```
020737   117   116   124             .ASCII  /ONT/
020742   122   117   114             .ASCII  /ROL/
020745   114   105   122             .ASCII  /LER/
020750   040   104   105             .ASCII  / DE/
020753   124   105   103             .ASCII  /TEC/
020756   124   105   104             .ASCII  /TED/
020761   040   120   125             .ASCII  / PU/
020764   114   123   105             .ASCII  /LSE/
020767   040   117   122             .ASCII  / OR/
020772   040   123   124             .ASCII  / ST/
020775   101   124   105             .ASCII  /ATE/
021000   040   120   101             .ASCII  / PA/
021003   122   111   124             .ASCII  /RIT/
021006   131   040   105             .ASCII  /Y E/
021011   122   122   117             .ASCII  /RRO/
021014   122   000                   .ASCII  /R/<00>
021016   045   101   102    P.AJU:   .ASCII  /%AB/
021021   101   104   040             .ASCII  /AD /
021024   102   114   117             .ASCII  /BLO/
021027   103   113   040             .ASCII  /CK /
021032   123   125   103             .ASCII  /SUC/
021035   103   105   123             .ASCII  /CES/
021040   123   106   125             .ASCII  /SFU/
021043   114   114   131             .ASCII  /LLY/
021046   040   122   105             .ASCII  / RE/
021051   120   114   101             .ASCII  /PLA/
021054   103   105   104             .ASCII  /CED/
021057   000                         .ASCII  <00>
021060   045   101   102    P.AJV:   .ASCII  /%AB/
021063   114   117   103             .ASCII  /LOC/
021066   113   040   126             .ASCII  /K V/
021071   105   122   111             .ASCII  /ERI/
021074   106   111   105             .ASCII  /FIE/
021077   104   040   117             .ASCII  /D O/
021102   113   040   055             .ASCII  /K -/
021105   055   040   116             .ASCII  /- N/
021110   117   124   040             .ASCII  /OT /
021113   101   040   102             .ASCII  /A B/
021116   101   104   040             .ASCII  /AD /
021121   102   114   117             .ASCII  /BLO/
021124   103   113   000             .ASCII  /CK/<00>
021127   000                         .ASCII  <00>
021130   045   101   122    P.AJW:   .ASCII  /%AR/
021133   105   120   114             .ASCII  /EPL/
021136   101   103   105             .ASCII  /ACE/
021141   115   105   116             .ASCII  /MEN/
021144   124   040   106             .ASCII  /T F/
021147   101   111   114             .ASCII  /AIL/
021152   125   122   105             .ASCII  /URE/
021155   000                         .ASCII  <00>
021156   045   101   103    P.AJY:   .ASCII  /%AC/
021161   117   116   124             .ASCII  /ONT/
021164   122   117   114             .ASCII  /ROL/
```

```
021167  114  105  122            .ASCII  /LER/
021172  040  124  111            .ASCII  / TI/
021175  115  105  117            .ASCII  /MEO/
021200  125  124  000            .ASCII  /UT/<00>
021203  000                      .ASCII  <00>
021204  045  101  105  P.AJZ:    .ASCII  /%AE/
021207  116  126  105            .ASCII  /NVE/
021212  114  117  120            .ASCII  /LOP/
021215  105  057  120            .ASCII  /E/<57>/P/
021220  101  103  113            .ASCII  /ACK/
021223  105  124  040            .ASCII  /ET /
021226  122  105  101            .ASCII  /REA/
021231  104  040  105            .ASCII  /D E/
021234  122  122  117            .ASCII  /RRO/
021237  122  040  050            .ASCII  /R (/
021242  120  141  162            .ASCII  /Par/
021245  151  164  171            .ASCII  /ity/
021250  040  157  162            .ASCII  / or/
021253  040  164  151            .ASCII  / ti/
021256  155  145  157            .ASCII  /meo/
021261  165  164  051            .ASCII  /ut)/
021264  000  000                 .ASCII  <00><00>
021266  045  101  105  P.AKA:    .ASCII  /%AE/
021271  116  126  105            .ASCII  /NVE/
021274  114  117  120            .ASCII  /LOP/
021277  105  057  120            .ASCII  /E/<57>/P/
021302  101  103  113            .ASCII  /ACK/
021305  105  124  040            .ASCII  /ET /
021310  127  122  111            .ASCII  /WRI/
021313  124  105  040            .ASCII  /TE /
021316  105  122  122            .ASCII  /ERR/
021321  117  122  040            .ASCII  /OR /
021324  050  120  141            .ASCII  /(Pa/
021327  162  151  164            .ASCII  /rit/
021332  171  040  157            .ASCII  /y o/
021335  162  040  164            .ASCII  /r t/
021340  151  155  145            .ASCII  /ime/
021343  157  165  164            .ASCII  /out/
021346  051  000                 .ASCII  /)/<00>
021350  045  101  103  P.AKB:    .ASCII  /%AC/
021353  117  116  124            .ASCII  /ONT/
021356  122  117  114            .ASCII  /ROL/
021361  114  105  122            .ASCII  /LER/
021364  040  122  117            .ASCII  / RO/
021367  115  040  101            .ASCII  /M A/
021372  116  104  040            .ASCII  /ND /
021375  122  101  115            .ASCII  /RAM/
021400  040  120  101            .ASCII  / PA/
021403  122  111  124            .ASCII  /RIT/
021406  131  040  105            .ASCII  /Y E/
021411  122  122  117            .ASCII  /RRO/
021414  122  000                 .ASCII  /R/<00>
021416  045  101  103  P.AKC:    .ASCII  /%AC/
```

```
021421  117  116  124           .ASCII  /ONT/
021424  122  117  114           .ASCII  /ROL/
021427  114  105  122           .ASCII  /LER/
021432  040  122  101           .ASCII  / RA/
021435  115  040  120           .ASCII  /M P/
021440  101  122  111           .ASCII  /ARI/
021443  124  131  040           .ASCII  /TY /
021446  105  122  122           .ASCII  /ERR/
021451  117  122  000           .ASCII  /OR/<00>
021454  045  101  103  P.AKD:   .ASCII  /%AC/
021457  117  116  124           .ASCII  /ONT/
021462  122  117  114           .ASCII  /ROL/
021465  114  105  122           .ASCII  /LER/
021470  040  122  117           .ASCII  / RO/
021473  115  040  120           .ASCII  /M P/
021476  101  122  111           .ASCII  /ARI/
021501  124  131  040           .ASCII  /TY /
021504  105  122  122           .ASCII  /ERR/
021507  117  122  000           .ASCII  /OR/<00>
021512  045  101  122  P.AKE:   .ASCII  /%AR/
021515  111  116  107           .ASCII  /ING/
021520  040  122  105           .ASCII  / RE/
021523  101  104  040           .ASCII  /AD /
021526  105  122  122           .ASCII  /ERR/
021531  117  122  040           .ASCII  /OR /
021534  050  120  141           .ASCII  /(Pa/
021537  162  151  164           .ASCII  /rit/
021542  171  040  157           .ASCII  /y o/
021545  162  040  164           .ASCII  /r t/
021550  151  155  145           .ASCII  /ime/
021553  157  165  164           .ASCII  /out/
021556  051  000                .ASCII  /)/<00>
021560  045  101  122  P.AKF:   .ASCII  /%AR/
021563  111  116  107           .ASCII  /ING/
021566  040  127  122           .ASCII  / WR/
021571  111  124  105           .ASCII  /ITE/
021574  040  105  122           .ASCII  / ER/
021577  122  117  122           .ASCII  /ROR/
021602  040  050  120           .ASCII  / (P/
021605  141  162  151           .ASCII  /ari/
021610  164  171  040           .ASCII  /ty /
021613  157  162  040           .ASCII  /or /
021616  164  151  155           .ASCII  /tim/
021621  145  157  165           .ASCII  /eou/
021624  164  051  000           .ASCII  /t)/<00>
021627  000                     .ASCII  <00>
021630  111  116  124  P.AKG:   .ASCII  /INT/
021633  105  122  122           .ASCII  /ERR/
021636  125  120  124           .ASCII  /UPT/
021641  040  115  101           .ASCII  / MA/
021644  123  124  105           .ASCII  /STE/
021647  122  040  106           .ASCII  /R F/
021652  101  111  114           .ASCII  /AIL/
```

```
021655  125  122  105            .ASCII  /URE/
021660  000  000                 .ASCII  <00><00>
021662  045  101  110   P.AKH:   .ASCII  /%AH/
021665  117  123  124            .ASCII  /OST/
021670  040  101  103            .ASCII  / AC/
021673  103  105  123            .ASCII  /CES/
021676  123  040  124            .ASCII  /S T/
021701  111  115  105            .ASCII  /IME/
021704  117  125  124            .ASCII  /OUT/
021707  040  050  110            .ASCII  / (H/
021712  151  147  150            .ASCII  /igh/
021715  145  162  040            .ASCII  /er /
021720  154  145  166            .ASCII  /lev/
021723  145  154  040            .ASCII  /el /
021726  160  162  157            .ASCII  /pro/
021731  164  157  143            .ASCII  /toc/
021734  157  154  040            .ASCII  /ol /
021737  144  145  160            .ASCII  /dep/
021742  145  156  144            .ASCII  /end/
021745  145  156  164            .ASCII  /ent/
021750  051  000                 .ASCII  /)/<00>
021752  045  101  103   P.AKI:   .ASCII  /%AC/
021755  122  105  104            .ASCII  /RED/
021760  111  124  040            .ASCII  /IT /
021763  114  111  115            .ASCII  /LIM/
021766  111  124  040            .ASCII  /IT /
021771  105  130  103            .ASCII  /EXC/
021774  105  105  104            .ASCII  /EED/
021777  105  104  000            .ASCII  /ED/<00>
022002  045  101  121   P.AKJ:   .ASCII  /%AQ/
022005  055  102  125            .ASCII  /-BU/
022010  123  040  115            .ASCII  /S M/
022013  101  123  124            .ASCII  /AST/
022016  105  122  040            .ASCII  /ER /
022021  105  122  122            .ASCII  /ERR/
022024  117  122  000            .ASCII  /OR/<00>
022027  000                      .ASCII  <00>
022030  045  101  103   P.AKK:   .ASCII  /%AC/
022033  117  116  124            .ASCII  /ONT/
022036  122  117  114            .ASCII  /ROL/
022041  114  105  122            .ASCII  /LER/
022044  040  106  101            .ASCII  / FA/
022047  124  101  114            .ASCII  /TAL/
022052  040  105  122            .ASCII  / ER/
022055  122  117  122            .ASCII  /ROR/
022060  000  000                 .ASCII  <00><00>
022062  045  101  111   P.AKL:   .ASCII  /%AI/
022065  116  123  124            .ASCII  /NST/
022070  122  125  103            .ASCII  /RUC/
022073  124  111  117            .ASCII  /TIO/
022076  116  040  114            .ASCII  /N L/
022101  117  117  120            .ASCII  /OOP/
022104  040  124  111            .ASCII  / TI/
```

```
022107  115  105  117           .ASCII  /MEO/
022112  125  124  000           .ASCII  /UT/<00>
022115  000                     .ASCII  <00>
022116  045  101  111  P.AKM:   .ASCII  /%AI/
022121  114  114  105           .ASCII  /LLE/
022124  107  101  114           .ASCII  /GAL/
022127  040  126  111           .ASCII  / VI/
022132  122  124  125           .ASCII  /RTU/
022135  101  114  040           .ASCII  /AL /
022140  103  111  122           .ASCII  /CIR/
022143  103  125  111           .ASCII  /CUI/
022146  124  040  111           .ASCII  /T I/
022151  104  000  000           .ASCII  /D/<00><00>
022154  045  101  111  P.AKN:   .ASCII  /%AI/
022157  116  124  105           .ASCII  /NTE/
022162  122  122  125           .ASCII  /RRU/
022165  120  124  040           .ASCII  /PT /
022170  126  105  103           .ASCII  /VEC/
022173  124  117  122           .ASCII  /TOR/
022176  040  111  114           .ASCII  / IL/
022201  114  105  107           .ASCII  /LEG/
022204  101  114  000           .ASCII  /AL/<00>
022207  000                     .ASCII  <00>
022210  045  101  115  P.AKO:   .ASCII  /%AM/
022213  101  111  116           .ASCII  /AIN/
022216  124  105  116           .ASCII  /TEN/
022221  101  116  103           .ASCII  /ANC/
022224  105  040  122           .ASCII  /E R/
022227  105  101  104           .ASCII  /EAD/
022232  057  127  122           .ASCII  <57>/WR/
022235  111  124  105           .ASCII  /ITE/
022240  040  111  116           .ASCII  / IN/
022243  126  101  114           .ASCII  /VAL/
022246  111  104  040           .ASCII  /ID /
022251  122  105  107           .ASCII  /REG/
022254  111  117  116           .ASCII  /ION/
022257  040  111  104           .ASCII  / ID/
022262  105  116  124           .ASCII  /ENT/
022265  111  106  111           .ASCII  /IFI/
022270  105  122  000           .ASCII  /ER/<00>
022273  000                     .ASCII  <00>
022274  045  101  115  P.AKP:   .ASCII  /%AM/
022277  101  111  116           .ASCII  /AIN/
022302  124  105  116           .ASCII  /TEN/
022305  101  116  103           .ASCII  /ANC/
022310  105  040  127           .ASCII  /E W/
022313  122  111  124           .ASCII  /RIT/
022316  105  040  114           .ASCII  /E L/
022321  117  101  104           .ASCII  /OAD/
022324  040  124  117           .ASCII  / TO/
022327  040  116  117           .ASCII  / NO/
022332  116  055  114           .ASCII  /N-L/
022335  117  101  104           .ASCII  /OAD/
```

```
022340  101  102  114          .ASCII  /ABL/
022343  105  040  103          .ASCII  /E C/
022346  117  116  124          .ASCII  /ONT/
022351  122  117  114          .ASCII  /ROL/
022354  114  105  122          .ASCII  /LER/
022357  000                    .ASCII  <00>
022360  045  101  103  P.AKQ:  .ASCII  /%AC/
022363  117  116  124          .ASCII  /ONT/
022366  122  117  114          .ASCII  /ROL/
022371  114  105  122          .ASCII  /LER/
022374  040  122  101          .ASCII  / RA/
022377  115  040  105          .ASCII  /M E/
022402  122  122  117          .ASCII  /RRO/
022405  122  040  050          .ASCII  /R (/
022410  116  157  156          .ASCII  /Non/
022413  055  160  141          .ASCII  /-pa/
022416  162  151  164          .ASCII  /rit/
022421  171  051  000          .ASCII  /y)/<00>
022424  045  101  111  P.AKR:  .ASCII  /%AI/
022427  116  111  124          .ASCII  /NIT/
022432  040  123  105          .ASCII  / SE/
022435  121  125  105          .ASCII  /QUE/
022440  116  103  105          .ASCII  /NCE/
022443  040  105  122          .ASCII  / ER/
022446  122  117  122          .ASCII  /ROR/
022451  000                    .ASCII  <00>
022452  045  101  110  P.AKS:  .ASCII  /%AH/
022455  111  107  110          .ASCII  /IGH/
022460  105  122  040          .ASCII  /ER /
022463  114  105  126          .ASCII  /LEV/
022466  105  114  040          .ASCII  /EL /
022471  120  122  117          .ASCII  /PRO/
022474  124  117  103          .ASCII  /TOC/
022477  117  114  040          .ASCII  /OL /
022502  111  116  103          .ASCII  /INC/
022505  117  115  120          .ASCII  /OMP/
022510  101  124  111          .ASCII  /ATI/
022513  102  111  114          .ASCII  /BIL/
022516  111  124  131          .ASCII  /ITY/
022521  040  105  122          .ASCII  / ER/
022524  122  117  122          .ASCII  /ROR/
022527  000                    .ASCII  <00>
022530  045  101  120  P.AKT:  .ASCII  /%AP/
022533  125  122  107          .ASCII  /URG/
022536  105  057  120          .ASCII  /E/<57>/P/
022541  117  114  114          .ASCII  /OLL/
022544  040  110  101          .ASCII  / HA/
022547  122  104  127          .ASCII  /RDW/
022552  101  122  105          .ASCII  /ARE/
022555  040  106  101          .ASCII  / FA/
022560  111  114  125          .ASCII  /ILU/
022563  122  105  000          .ASCII  /RE/<00>
022566  045  101  115  P.AKU:  .ASCII  /%AM/
```

```
022571   101    120    120            .ASCII  /APP/
022574   111    116    107            .ASCII  /ING/
022577   040    122    105            .ASCII  / RE/
022602   107    111    123            .ASCII  /GIS/
022605   124    105    122            .ASCII  /TER/
022610   040    122    105            .ASCII  / RE/
022613   101    104    040            .ASCII  /AD /
022616   106    101    111            .ASCII  /FAI/
022621   114    125    122            .ASCII  /LUR/
022624   105    040    050            .ASCII  /E (/
022627   120    141    162            .ASCII  /Par/
022632   151    164    171            .ASCII  /ity/
022635   040    157    162            .ASCII  / or/
022640   040    164    151            .ASCII  / ti/
022643   155    145    157            .ASCII  /meo/
022646   165    164    051            .ASCII  /ut)/
022651   000                          .ASCII  <00>
022652  021156'              P.AJX:   .WORD   P.AJY
022654  021204'                       .WORD   P.AJZ
022656  021266'                       .WORD   P.AKA
022660  021350'                       .WORD   P.AKB
022662  021416'                       .WORD   P.AKC
022664  021454'                       .WORD   P.AKD
022666  021512'                       .WORD   P.AKE
022670  021560'                       .WORD   P.AKF
022672  021630'                       .WORD   P.AKG
022674  021662'                       .WORD   P.AKH
022676  021752'                       .WORD   P.AKI
022700  022002'                       .WORD   P.AKJ
022702  022030'                       .WORD   P.AKK
022704  022062'                       .WORD   P.AKL
022706  022116'                       .WORD   P.AKM
022710  022154'                       .WORD   P.AKN
022712  022210'                       .WORD   P.AKO
022714  022274'                       .WORD   P.AKP
022716  022360'                       .WORD   P.AKQ
022720  022424'                       .WORD   P.AKR
022722  022452'                       .WORD   P.AKS
022724  022530'                       .WORD   P.AKT
022726  022566'                       .WORD   P.AKU
022730   045    101    124   P.AKW:   .ASCII  /%AT/
022733   061    061    040            .ASCII  /11 /
022736   103    120    125            .ASCII  /CPU/
022741   040    106    101            .ASCII  / FA/
022744   111    114    125            .ASCII  /ILU/
022747   122    105    000            .ASCII  /RE/<00>
022752   045    101    116   P.AKX:   .ASCII  /%AN/
022755   117    116    055            .ASCII  /ON-/
022760   120    101    122            .ASCII  /PAR/
022763   111    124    131            .ASCII  /ITY/
022766   040    122    101            .ASCII  / RA/
022771   115    040    105            .ASCII  /M E/
022774   122    122    117            .ASCII  /RRO/
```

```
022777   122   000   000               .ASCII  /R/<00><00>
023002   045   101   123      P.AKY:   .ASCII  /%AS/
023005   124   101   124               .ASCII  /TAT/
023010   105   040   115               .ASCII  /E M/
023013   101   103   110               .ASCII  /ACH/
023016   111   116   105               .ASCII  /INE/
023021   040   106   101               .ASCII  / FA/
023024   111   114   125               .ASCII  /ILU/
023027   122   105   040               .ASCII  /RE /
023032   055   040   124               .ASCII  /- T/
023035   061   061   040               .ASCII  /11 /
023040   101   104   104               .ASCII  /ADD/
023043   122   105   123               .ASCII  /RES/
023046   123   040   122               .ASCII  /S R/
023051   105   107   111               .ASCII  /EGI/
023054   123   124   105               .ASCII  /STE/
023057   122   000   000               .ASCII  /R/<00><00>
023062   045   101   123      P.AKZ:   .ASCII  /%AS/
023065   124   101   124               .ASCII  /TAT/
023070   105   040   115               .ASCII  /E M/
023073   101   103   110               .ASCII  /ACH/
023076   111   116   105               .ASCII  /INE/
023101   040   106   101               .ASCII  / FA/
023104   111   114   125               .ASCII  /ILU/
023107   122   105   040               .ASCII  /RE /
023112   055   040   121               .ASCII  /- Q/
023115   055   102   125               .ASCII  /-BU/
023120   123   040   101               .ASCII  /S A/
023123   104   104   122               .ASCII  /DDR/
023126   105   123   123               .ASCII  /ESS/
023131   040   122   105               .ASCII  / RE/
023134   107   111   123               .ASCII  /GIS/
023137   124   105   122               .ASCII  /TER/
023142   000   000                     .ASCII  <00><00>
023144   045   101   123      P.ALA:   .ASCII  /%AS/
023147   124   101   124               .ASCII  /TAT/
023152   105   040   115               .ASCII  /E M/
023155   101   103   110               .ASCII  /ACH/
023160   111   116   105               .ASCII  /INE/
023163   040   106   101               .ASCII  / FA/
023166   111   114   125               .ASCII  /ILU/
023171   122   105   040               .ASCII  /RE /
023174   055   040   103               .ASCII  /- C/
023177   122   103   040               .ASCII  /RC /
023202   122   105   107               .ASCII  /REG/
023205   111   123   124               .ASCII  /IST/
023210   105   122   000               .ASCII  /ER/<00>
023213   000                           .ASCII  <00>
023214   045   101   123      P.ALB:   .ASCII  /%AS/
023217   124   101   124               .ASCII  /TAT/
023222   105   040   115               .ASCII  /E M/
023225   101   103   110               .ASCII  /ACH/
023230   111   116   105               .ASCII  /INE/
```

```
023233     040    106    101            .ASCII  / FA/
023236     111    114    125            .ASCII  /ILU/
023241     122    105    040            .ASCII  /RE /
023244     055    040    123            .ASCII  /- S/
023247     105    122    111            .ASCII  /ERI/
023252     101    114    111            .ASCII  /ALI/
023255     132    105    122            .ASCII  /ZER/
023260     057    104    105            .ASCII  <57>/DE/
023263     123    105    122            .ASCII  /SER/
023266     111    101    114            .ASCII  /IAL/
023271     111    132    105            .ASCII  /IZE/
023274     122    040    122            .ASCII  /R R/
023277     105    107    111            .ASCII  /EGI/
023302     123    124    105            .ASCII  /STE/
023305     122    000    000            .ASCII  /R/<00><00>
023310     045    101    123   P.ALC:   .ASCII  /%AS/
023313     124    101    124            .ASCII  /TAT/
023316     105    040    115            .ASCII  /E M/
023321     101    103    110            .ASCII  /ACH/
023324     111    116    105            .ASCII  /INE/
023327     040    106    101            .ASCII  / FA/
023332     111    114    125            .ASCII  /ILU/
023335     122    105    040            .ASCII  /RE /
023340     055    040    127            .ASCII  /- W/
023343     122    117    116            .ASCII  /RON/
023346     107    040    110            .ASCII  /G H/
023351     101    122    104            .ASCII  /ARD/
023354     127    101    122            .ASCII  /WAR/
023357     105    040    126            .ASCII  /E V/
023362     105    122    123            .ASCII  /ERS/
023365     111    117    116            .ASCII  /ION/
023370     000    000                   .ASCII  <00><00>
023372  022730'                  P.AKV:   .WORD   P.AKW
023374  022752'                           .WORD   P.AKX
023376  023002'                           .WORD   P.AKY
023400  023062'                           .WORD   P.AKZ
023402  023144'                           .WORD   P.ALA
023404  023214'                           .WORD   P.ALB
023406  023310'                           .WORD   P.ALC
023410     045    116    045   P.ALD:   .ASCII  /%N%/
023413     101    132    122            .ASCII  /AZR/
023416     121    101    040            .ASCII  /QA /
023421     104    105    126            .ASCII  /DEV/
023424     040    106    124            .ASCII  / FT/
023427     114    040    040            .ASCII  /L  /
023432     045    132    065            .ASCII  /%Z5/
023435     045    101    040            .ASCII  /%A /
023440     117    116    040            .ASCII  /ON /
023443     125    116    111            .ASCII  /UNI/
023446     124    040    045            .ASCII  /T %/
023451     132    062    045            .ASCII  /Z2%/
023454     101    040    124            .ASCII  /A T/
023457     123    124    040            .ASCII  /ST /
```

```
023462  060  060  061           .ASCII  /001/
023465  040  123  125           .ASCII  / SU/
023470  102  040  060           .ASCII  /B 0/
023473  060  060  040           .ASCII  /00 /
023476  120  103  072           .ASCII  /PC:/
023501  040  045  117           .ASCII  / %O/
023504  066  000                .ASCII  /6/<00>
023506  045  116  045   P.ALE:  .ASCII  /%N%/
023511  101  132  122           .ASCII  /AZR/
023514  121  101  040           .ASCII  /QA /
023517  110  122  104           .ASCII  /HRD/
023522  040  105  122           .ASCII  / ER/
023525  122  040  040           .ASCII  /R  /
023530  045  132  065           .ASCII  /%Z5/
023533  045  101  040           .ASCII  /%A /
023536  117  116  040           .ASCII  /ON /
023541  125  116  111           .ASCII  /UNI/
023544  124  040  045           .ASCII  /T %/
023547  132  062  045           .ASCII  /Z2%/
023552  101  040  124           .ASCII  /A T/
023555  123  124  040           .ASCII  /ST /
023560  060  060  061           .ASCII  /001/
023563  040  123  125           .ASCII  / SU/
023566  102  040  060           .ASCII  /B 0/
023571  060  060  040           .ASCII  /00 /
023574  120  103  072           .ASCII  /PC:/
023577  040  045  117           .ASCII  / %O/
023602  066  000                .ASCII  /6/<00>
023604  045  116  045   P.ALF:  .ASCII  /%N%/
023607  101  132  122           .ASCII  /AZR/
023612  121  101  040           .ASCII  /QA /
023615  123  106  124           .ASCII  /SFT/
023620  040  105  122           .ASCII  / ER/
023623  122  040  040           .ASCII  /R  /
023626  045  132  065           .ASCII  /%Z5/
023631  045  101  040           .ASCII  /%A /
023634  117  116  040           .ASCII  /ON /
023637  125  116  111           .ASCII  /UNI/
023642  124  040  045           .ASCII  /T %/
023645  132  062  045           .ASCII  /Z2%/
023650  101  040  124           .ASCII  /A T/
023653  123  124  040           .ASCII  /ST /
023656  060  060  061           .ASCII  /001/
023661  040  123  125           .ASCII  / SU/
023664  102  040  060           .ASCII  /B 0/
023667  060  060  040           .ASCII  /00 /
023672  120  103  072           .ASCII  /PC:/
023675  040  045  117           .ASCII  / %O/
023700  066  045  116           .ASCII  /6%N/
023703  000                     .ASCII  <00>
023704  045  116  045   P.ALG:  .ASCII  /%N%/
023707  101  111  057           .ASCII  /AI/<57>
023712  117  040  122           .ASCII  /O R/
```

```
023715  105  121  125              .ASCII  /EQU/
023720  105  123  124              .ASCII  /EST/
023723  040  106  101              .ASCII  / FA/
023726  111  114  105              .ASCII  /ILE/
023731  104  045  116              .ASCII  /D%N/
023734  000  000                   .ASCII  <00><00>
023736  045  123  064     P.ALH:   .ASCII  /%S4/
023741  000                        .ASCII  <00>
023742  045  116  000     P.ALI:   .ASCII  /%N/<00>
023745  000                        .ASCII  <00>
023746  045  101  040     P.ALJ:   .ASCII  /%A /
023751  055  040  000              .ASCII  /- /<00>
023754  045  101  052     P.ALK:   .ASCII  /%A*/
023757  040  000  000              .ASCII  / /<00><00>
023762  00000C                L$HWLEN::
                                   .WORD   <<L$NDHW-L$HWLEN>/2>
023764  172150                HWPT.IP.ADDR::
                                   .WORD   -5630
023766  000154                HWPT.VECTOR::
                                   .WORD   154
023770  000004                HWPT.BR.LEVEL::
                                   .WORD   4
023772  000200                HWPT.DISK::
                                   .WORD   200
023774  000000                HWPTS0.LBN::
                                   .WORD   0
023776  000000                HWPTS1.LBN::
                                   .WORD   0
024000  177777                HWPTE0.LBN::
                                   .WORD   -1
024002  000000                HWPTE1.LBN::
                                   .WORD   0
024004  020040                NAME.LO::
                                   .WORD   20040
024006  020040                NAME.HI::
                                   .WORD   20040
024010                        L$NDHW:: .BLKW  1
024012  00000C                L$SWLEN::
                                   .WORD   <<L$NDSW-L$SWLEN>/2>
024014  000040                SWP.ERROR::
                                   .WORD   40
024016  000000                SWP.XFER::
                                   .WORD   0
024020  054046                SWP.FLAGS::
                                   .WORD   54046
024022  000000                SWP.DPAT::
                                   .WORD   0
024024  000143                SWP.RAT::
                                   .WORD   143
024026  000000                SWP.TIME::
                                   .WORD   0
024030  000020                SWP.UCNT::
                                   .WORD   20
```

```
024032                  SWP.UDPAT::
                                .BLKW   20
024072  000013          DUPROUND::
                                .WORD   13
024074  000000          MAN.TST::
                                .WORD   0
024076  000001          TST.PAR::
                                .WORD   1
024100                  L$NDSW::.BLKW   1
024102  000000          L$PROT::.WORD   0
024104  177777                  .WORD   -1
024106  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                  TRK.SGN::
000532     001                  .BYTE   1
000533     001                  .BYTE   1
000534     001                  .BYTE   1
000535     001                  .BYTE   1
000536  000020          RDM.CNT::
                                .WORD   20
000540                  RANDOM::.BLKW   20
000600                  C.ERR.TBL::
                                .BLKW   1
000602                  MSCP.PKT::
                                .BLKW   644
002312                  IPKT.ADDR::
                                .BLKW   1
002314                  PKT.USE::
                                .BLKW   6
002330                  RETPKT::.BLKW   260
003070                  RP.USE::.BLKW   4
003100                  RP.INDX::
                                .BLKW   1
003102                  RP.ADDR::
                                .BLKW   1
003104                  ELOG.PKT::
                                .BLKW   655
```

```
004636          BUFF.ADDR::
                        .BLKW   10
004656          BUFF.OWN::
                        .BLKW   4
004666          IODQ::  .BLKW   4
004676          IODQ.IN::
                        .BLKW   1
004700          IODQ.OUT::
                        .BLKW   1
004702          ENTRY.REASON::
                        .BLKB   1
004703          EOP.FLAG::
                        .BLKB   1
004704          DUP.FLAGS::
                        .BLKW   1
004706          CCTLR:: .BLKW   1
004710          CDISK:: .BLKW   1
004712          CUOFF:: .BLKW   1
004714          CTLR.CNT::
                        .BLKW   1
004716          DUR::   .BLKW   2
004722          QIO::   .BLKB   1
                        .EVEN
004724          FREE.MEM.ADDR::
                        .BLKW   1
004726          BYTS.PER.QIO::
                        .BLKW   1
004730          ST.CODE::
                        .BLKW   1
004732          SB.CODE::
                        .BLKW   1
004734          STEP::  .BLKW   1
004736          OF.RC:: .BLKW   1
004740          SA.REG::.BLKW   1
004742          CMD.TIME::
                        .BLKW   1
004744          NEX::   .BLKW   1
004746          CRN.LOW::
                        .BLKW   1
004750          CRN.HIGH::
                        .BLKW   1
004752          TEMP1:: .BLKW   1
004754          TEMP2:: .BLKW   1
004756          CREDIT.BAL::
                        .BLKW   1
004760          NEXT.PKT.USE::
                        .BLKB   1
004761          HOURS:: .BLKB   1
004762          MINUTES::
                        .BLKB   1
                        .EVEN
004764          CLK.TICKS::
                        .BLKW   1
```

```
004766                          FER0.LBN::
                                        .BLKW   1
004770                          FER1.LBN::
                                        .BLKW   1
004772                          CLK.PRESENT::
                                        .BLKB   1
004773                          HOE.FLAG::
                                        .BLKB   1
004774                          TYPER:: .BLKW   4
005004                          TYPEW:: .BLKW   4
005014                          BAL.IN.PROGRESS::
                                        .BLKW   4
005024                          FORCE.WR::
                                        .BLKW   4
005034                          CSR.MEM::
                                        .BLKW   1
005036  172100                  CSR.ADD::
                                        .WORD    5700
005040                          P.INDEX::
                                        .BLKW   1
005042  000000                  RD.COUNT::
                                        .WORD   0
005044                          BRLEVEL::
                                        .BLKW   1
005046                          D.FAIL::.BLKB   1
005047                          FORCED.ERROR::
                                        .BLKB   1
005050                          FER.LBN::
                                        .BLKW   1
005052                          FER.BC::.BLKW   1
005054                          INIT.OCCURED::
005054     000                          .BYTE   0
005055                          ADDR.VECT.OK::
005055     000                          .BYTE   0

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

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

```
000040     BIT05==          40
000020     BIT04==          20
000010     BIT03==          10
000004     BIT02==          4
000002     BIT01==          2
000001     BIT00==          1
001000     BIT9==           1000
000400     BIT8==           400
000200     BIT7==           200
000100     BIT6==           100
000040     BIT5==           40
000020     BIT4==           20
000010     BIT3==           10
000004     BIT2==           4
000002     BIT1==           2
000001     BIT0==           1
000035     EF.NEW==         35
000034     EF.PWR==         34
000040     EF.START==       40
000037     EF.RESTART==     37
000036     EF.CONTINUE==    36
000340     PRI07==          340
000300     PRI06==          300
000240     PRI05==          240
000200     PRI04==          200
000140     PRI03==          140
000100     PRI02==          100
000040     PRI01==          40
000000     PRI00==          0
000004     EVL==            4
000010     LOT==            10
000020     ADR==            20
000040     IDU==            40
000100     ISR==            100
000200     UAM==            200
000400     BOE==            400
001000     PNT==            1000
002000     PRI==            2000
004000     IXE==            4000
010000     IBE==            10000
020000     IER==            20000
040000     LOE==            40000
100000     HOE==            -100000
000126'    L$ERRTBL==       ERRTYP
024014'    L$SW==           L$SWLEN+2
023764'    L$HW==           L$HWLEN+2
000011'    L$DEPO==         L$REV+1
000136'    PTCH1==          P.AAA
000210'    PTCH2==          P.AAB
000262'    PTCH3==          P.AAC
000334'    PTCH4==          P.AAD
000406'    PTCH5==          P.AAE
000460'    HWQ1==           P.AAF
```

```
000474'     HWQ2==       P.AAG
000504'     HWQ3==       P.AAH
000546'     HWQ4==       P.AAI
000564'     HWQ5==       P.AAJ
000634'     HWQ6A==      P.AAK
000706'     HWQ6B==      P.AAL
000762'     HWQ7A==      P.AAM
001032'     HWQ7B==      P.AAN
001102'     HWQ8==       P.AAO
001160'     HWQ9==       P.AAP
001260'     SW[illegible]==     P.AAQ
001302'     SWQ2==       P.AAR
001362'     SWQ4==       P.AAS
001404'     SWQ7==       P.AAT
001456'     SWQ9==       P.AAU
001532'     SWQ10==      P.AAV
001576'     SWQ11==      P.AAW
001630'     SWQ12==      P.AAX
001726'     SWQ13==      P.AAY
002004'     SWQ14==      P.AAZ
002056'     SWQ15==      P.ABA
002126'     SWQ17==      P.ABB
002224'     SWQ19==      P.ABC
002314'     SWQ20==      P.ABD
002402'     SWQ21==      P.ABE
002466'     SWQ22==      P.ABF
002526'     SWQ23==      P.ABG
002600'     SWQ24==      P.ABH
002644'     SWQ26==      P.ABI
002706'     SWM1==       P.ABJ
003004'     SWQ27==      P.ABK
003030'     SWQ28==      P.ABL
003100'     NULL==       P.ABM
003102'     DBM5==       P.ABN
003130'     DBM12==      P.ABO
003204'     DBM15==      P.ABP
003234'     DBM18==      P.ABQ
003306'     DBM19==      P.ABR
003372'     DBM20==      P.ABS
003446'     DBM21==      P.ABT
003530'     DBM22==      P.ABU
003574'     DBM23==      P.ABV
003632'     DBM25==      P.ABW
003700'     DBM26==      P.ABX
003732'     DBM27==      P.ABY
004004'     DBM28A==     P.ABZ
004044'     DBM28B==     P.ACA
004104'     DBM29==      P.ACB
004152'     DBM32==      P.ACC
004226'     DBM101==     P.ACD
004254'     DBM104==     P.ACE
004316'     DBM105==     P.ACF
004354'     DBM107==     P.ACG
```

004412'	DBM108==	P.ACH
004502'	DBM109==	P.ACI
004562'	DBM111==	P.ACJ
004662'	DBM112==	P.ACK
004764'	DBM120==	P.ACL
005056'	DBM121==	P.ACM
005146'	DBM123==	P.ACN
005210'	DBM125==	P.ACO
005260'	DBM126==	P.ACP
005340'	DBM127==	P.ACQ
005416'	DBM128==	P.ACR
005504'	DU.MSG==	P.ACS
006224'	DU.RSN==	P.ACT
006252'	MSG.01==	P.ADF
006304'	MSG.02==	P.ADG
006340'	MSG.03==	P.ADH
006372'	RPT1==	P.ADI
006456'	RPT2==	P.ADJ
006522'	RPT3==	P.ADK
006606'	RPT4==	P.ADL
006652'	RPT5==	P.ADM
006740'	RPT6==	P.ADN
007004'	RPT7==	P.ADO
007022'	RPT8==	P.ADP
007050'	RPT9==	P.ADQ
007102'	RPT10==	P.ADR
007170'	RPT11==	P.ADS
007236'	RPT12==	P.ADT
007304'	RPT13==	P.ADU
007404'	RPT14==	P.ADV
007502'	RPT15==	P.ADW
007602'	RPT16==	P.ADX
007664'	EGS.01==	P.ADY
007704'	EGS.02==	P.ADZ
007776'	EGD.10==	P.AEA
010036'	EGD.11==	P.AEB
010062'	EGD.12==	P.AEC
010110'	EGD.13==	P.AED
010136'	EGD.14==	P.AEE
010166'	EGD.15==	P.AEF
010204'	EGD.16==	P.AEG
010234'	EGD.17==	P.AEH
010252'	EGD.18==	P.AEI
010272'	EGD.19==	P.AEJ
010332'	EGD.20==	P.AEK
010420'	EGD.21==	P.AEL
010532'	EGD.22==	P.AEM
010572'	EGD.23==	P.AEN
010634'	EGD.24==	P.AEO
010700'	EGH.30==	P.AEP
010724'	EBS.01==	P.AEQ
010766'	EBD.10==	P.AER
011026'	EBD.12==	P.AES

```
011074'     EBD.13==     P.AET
011126'     EBD.14==     P.AEU
011166'     EBD.18==     P.AEV
011222'     EBD.19==     P.AEW
011302'     EBD.24==     P.AEX
011356'     EH.0==       P.AEY
011414'     EH.1==       P.AEZ
011452'     EH.2==       P.AFA
011512'     EH.3==       P.AFB
011550'     EH.4==       P.AFC
011574'     EH.5==       P.AFD
011624'     EH.6==       P.AFE
011654'     EH.7==       P.AFF
011704'     EH.8==       P.AFG
011740'     EH.9==       P.AFH
011770'     EH.10==      P.AFI
012020'     EH.12==      P.AFJ
012054'     EH.13==      P.AFK
012126'     ERR.00==     P.AFL
012662'     ERR.COD==    P.AFM
012720'     ELG.00==     P.AGC
013334'     ELG.FMT==    P.AGD
013350'     EX.SA==      P.AGK
013366'     EX.SC==      P.AGL
013416'     EX.SBO==     P.AGM
013422'     EX.SB==      P.AGN
013444'     EX.CMD==     P.AGO
013464'     EX.RD==      P.AGP
013474'     EX.WRT==     P.AGQ
013504'     EX.CMP==     P.AGR
013520'     EX.ONL==     P.AGS
013532'     EX.ACC==     P.AGT
013544'     EX.OP==      P.AGU
013550'     EX.BB2==     P.AGV
013616'     EX.BB12==    P.AGW
013670'     EX.BBU2==    P.AGX
013750'     EX.LBN2==    P.AGY
014010'     EX.PBN2==    P.AGZ
014050'     EX.LBR2==    P.AHA
014116'     EX.LBW2==    P.AHB
014166'     EX.RBN2==    P.AHC
014226'     EX.CBC==     P.AHD
014272'     EX.CBR==     P.AHE
014344'     EX.CBW==     P.AHF
014416'     EX.BC==      P.AHG
014472'     EX.BD==      P.AHH
014550'     EX.BDR==     P.AHI
014640'     EX.BDW==     P.AHJ
014730'     EX.RP==      P.AHK
015016'     EX.WRD==     P.AHL
015026'     EX.TIM==     P.AHM
015064'     EX.LB==      P.AHN
015130'     XX13==       P.AHO
```

015152'	XX23==	P.AHP
015206'	XX32==	P.AHQ
015234'	XX33==	P.AHR
015272'	XX34==	P.AHS
015342'	CER.01==	P.AHT
015406'	CER.02==	P.AHU
015462'	SC.SDI==	P.AHV
015506'	SC.CON==	P.AHW
015530'	SC.DUP==	P.AHX
015560'	SC.ONL==	P.AHY
015602'	SC.SON==	P.AHZ
015622'	SC.INR==	P.AIA
015654'	SC.INV==	P.AIB
015672'	SC.UNK==	P.AIC
015752'	SC.VOL==	P.AID
016032'	SC.IOP==	P.AIE
016104'	SC.DIS==	P.AIF
016176'	SC.FER==	P.AIG
016264'	SC.FE2==	P.AIH
016354'	SC.ISH==	P.AII
016434'	SC.IS2==	P.AIJ
016514'	SC.DST==	P.AIK
016570'	SC.DS2==	P.AIL
016642'	SC.ECC==	P.AIM
016724'	SC.ECD==	P.AIN
016756'	SC.RCT==	P.AIO
016776'	SC.FUL==	P.AIP
017052'	SC.576==	P.AIQ
017126'	SC.FCT==	P.AIR
017174'	SC.EC1==	P.AIS
017224'	SC.EC2==	P.AIT
017254'	SC.EC3==	P.AIU
017306'	SC.EC4==	P.AIV
017336'	SC.EC5==	P.AIW
017366'	SC.EC6==	P.AIX
017416'	SC.EC7==	P.AIY
017450'	SC.EC8==	P.AIZ
017502'	SC.EC9==	P.AJA
017544'	SC.SWP==	P.AJB
017604'	SC.HWP==	P.AJC
017644'	SC.SAF==	P.AJD
017710'	SC.ODA==	P.AJE
017740'	SC.ODB==	P.AJF
017762'	SC.NXM==	P.AJG
020016'	SC.PAR==	P.AJH
020052'	SC.CTO==	P.AJI
020124'	SC.SDS==	P.AJJ
020202'	SC.EDC==	P.AJK
020242'	SC.IDS==	P.AJL
020312'	SC.SRT==	P.AJM
020404'	SC.SRI==	P.AJN
020472'	SC.POE==	P.AJO
020526'	SC.RDY==	P.AJP

```
 020610'         SC.CLK==        P.AJQ
 020636'         SC.RSP==        P.AJR
 020704'         SC.SUR==        P.AJS
 020734'         SC.PSP==        P.AJT
 021016'         SC.REP==        P.AJU
 021060'         SC.NBB==        P.AJV
 021130'         SC.REF==        P.AJW
 022652'         CNTR.ERR==      P.AJX
 023372'         RDRX.ERR==      P.AKV
 023410'         DF.MSG==        P.ALD
 023506'         HRD.MSG==       P.ALE
 023604'         SFT.MSG==       P.ALF
 023704'         HRD.SUB==       P.ALG
 023736'         SPACE4==        P.ALH
 023742'         CRLF==          P.ALI
 023746'         DASH==          P.ALJ
 023754'         ASTERISK==      P.ALK
 023764'         DFPTBL==        L$HWLEN+2
 024014'         SFPTBL==        L$SWLEN+2


;                        PSECT SUMMARY
;
;     Psect Name         Words   Attributes
;      $CODE$             5156     RO ,  I  ,  LCL,  REL,  CON
;      $FFF$              1303     RW ,  D  ,  GBL,  REL,  CON


;                   Library Statistics
;
;                               -------- Symbols --------    Pages     Processing
;     File                      Total   Loaded   Percent     Mapped    Time
;
;  DISK$USER2:[MLYNAR.ZRQ]ZRQAHO.L16;10   411     178      43        21       00:00.1



;                        COMMAND QUALIFIERS

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

```
;   0   module ZRQAM2 (
;   0
;   0   %title 'RD/RX EXERCISER'
;   0               ident = 'V02.3',
;   0               addressing_mode (absolute),
;   0               environment (noeis)
;   0               ) =
;   0
;   1   begin
;   1
;   1   %sbttl 'DECLARATIONS'
;   1
;   1   library 'ZRQAHO.L16';                          ! RDRX EXERCISER GLOBAL LIBRARY
;   1
;   1   !MMM require 'BLSMAC.REQ';                     ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;   1   require 'HSAXAO.BLB';                          ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;   1
;   1   forward routine
;   1       NEX_TRAP : L$ISR novalue,
;   1       EMS_01 : novalue,
;   1       EMS_TIM : novalue,
;   1   !MMM    EMS_DBN : NOVALUE,                                          !ZZZ
;   1       EMS_BLK : NOVALUE,                                      !ZZZ
;   1       SET_CPAR : novalue,
;   1       SET_UPAR : novalue;
;   1
;   1   external
;   1       CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;   1                       ! RUN-TIME CONTROLLER STATUS TABLES
;   1       CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;   1                       ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;   1       DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;   1                       ! DRIVER CONTROLLER TABLES
;   1       DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;   1                       ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;   1       RDRX_ADDR : ref rdrx field (RC_REG),
;   1                       ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;   1       IRDRX_ADDR : ref rdrx field (RC_REG),
;   1                       ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;   1       BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                 !ZZZ
;   1                       !CONTAINS LBNS (HI + LO FIELDS) FOR SEQUENTIAL  !ZZZ
;   1                       !I/O TRANSFER FOR EACH UNIT.                    !ZZZ
;   1       TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;   1                       ! STATISTICS TABLES
;   1       T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;   1                       ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;   1   !MMM    DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),   !BUFFER FOR DUP  ZZZ
;   1   !MMM                    !INFO FROM RECEIVE AND SEND CMDS            ZZZ
;   1       TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TRACK DIRECTION ZZZ
;   1       RDM_CNT : WORD,             !NO OF RANDOM NOS    \\KEEP        ZZZ
;   1       RANDOM : VECTOR [RDM_LEN, WORD],    !RANDOM NO TABLE    //TOGETHER    ZZZ
;   1       C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;   1                       ! STATISTICS TABLE FOR CONTROLLER ERRORS
```

```
MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
                        ! MSCP PACKET POOL
IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
                        ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
PKT_USE : vector [PKT_CNT, byte, signed],
                        ! MSCP PACKET POOL ALLOCATION TABLE
RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
                        ! RETURN PACKET POOL
RP_USE : vector [RP_CNT, byte, signed],
                        ! RETURN PACKET POOL ALLOCATION TABLE
RP_INDX : word,         ! CURRENT RETURN PACKET INDEX
RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
                        ! CURRENT RETURN PACKET ADDRESS
ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
                        ! ERROR-LOG PACKET SAVE AREA
BUFF_ADDR : vector [MAX_BUF_CNT],                 ! TABLE OF I/O BUFFER DESCRIPTORS
BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],    ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
IODQ : vector [IODQ_LEN, byte],                   ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
IODQ_IN : word,                                   ! I/O DONE QUEUE IN POINTER
IODQ_OUT : word,                                  ! I/O DONE QUEUE OUT POINTER
ENTRY_REASON : byte,                              ! CURRENT OPERATOR COMMAND
EOP_FLAG : byte,                                  ! END-OF-PASS FLAG
DUP_FLAGS : WORD,                                 !DUP FLAGS        ZZZ
CCTLR : word,                                     ! NUMBER OF "CURRENT" CONTROLLER
CDISK : word,                                     ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
CUOFF : word,                                     ! CURRENT UNIT CST OFFSET
CTLR_CNT : word,                                  ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
DUR : vector [MAX_UNITS, byte],                   ! DROP UNIT REASON
QIO : vector [MAX_CTLR, byte],                    ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
FREE_MEM_ADDR,                                    ! START OF FREE MEMORY
BYTS_PER_QIO : word,                              ! SIZE (BYTES) OF AN I/O BUFFER
ST_CODE : word,                                   ! CURRENT STATUS CODE
SB_CODE : word,                                   ! CURRENT SUB-CODE
STEP : word,                                      ! CURRENT STEP IN HARD_INIT
OF_RC : signed word,                              ! OFFSET (0 OR 2) TO READ IP OR SA
SA_REG : word,                                    ! STORAGE FOR SA REGISTER READS AND WRITES
CMD_TIME : word,                                  ! COMMAND TIMEOUT VALUE (IN SECONDS)
NEX : word,                                       ! NON-EXISTENT MEMORY TRAP INDICATOR
CRN_LOW : word,                                   ! COMMAND REF NUMBER OF LAST COMMAND SENT
CRN_HIGH : word,                                  ! COMMAND REF NUMBER (HI ORDER)
TEMP1 : WORD,                                     !TEMPORARY STORAGE WD USED IN BGNCLN           !ZZZ
TEMP2 : WORD,                                     !TEMPORARY STORAGE WD USED IN BGNCLN           !ZZZ
CREDIT_BAL : word,                                ! CREDIT BALANCE
NEXT_PKT_USE : byte,                              ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
HOURS: byte,                                      ! TIME OF DAY (HOURS)
MINUTES : byte,                                   ! TIME OF DAY (MINUTES)
CLK_TICKS : word,                                 ! TIME OF DAY (LINE-CLOCK TICKS)
FER0_LBN : word,                                  !LO LBN ADR OF THE "FORCED ERROR" BLOCK              ZZZ
FER1_LBN : word,                                  !HI LBN ADR OF THE "FORCED ERROR" BLOCK              ZZZ
CLK_PRESENT : byte,                               ! FLAG INDICATES IF LINE-CLOCK PRESENT
HOE_FLAG : byte,                                  ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
FORCED_ERROR : byte,                              ! "FORCED ERROR" DETECTED IN LAST READ
FER_LBN : word,                                   ! LBN OF THE "FORCED ERROR" BLOCK
```

```
;  1      FER_BC : word,                                          ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;  1      INIT_OCCURED : byte,                                    ! EXERCISER INITIALIZATION COMPLETE
;  1      ADDR_VECT_OK : byte,                                    ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;  1      DBM5,
;  1      DBM125,                                                 !MMM
;  1      DBM126,                                                 !MMM
;  1
;  1      TYPER : VECTOR [MAX_UNITS, WORD],    !READ I/O COUNTER                  ZZZ MMM
;  1      TYPEW : VECTOR [MAX_UNITS, WORD],    !WRITE I/O COUNTER                 ZZZ MMM
;  1      BAL_IN_PROGRESS : VECTOR [MAX_UNITS, WORD],        !FLAG SET TO BALANCE I/O TYPES     ZZZ MMM
;  1      FORCE_WR : VECTOR [MAX_UNITS, WORD],           ! MMM
;  1      CSR_MEM : WORD,                                ! MMM
;  1      CSR_ADD : WORD,                                ! MMM
;  1      P_INDEX : SIGNED WORD,               !CURRENT MESSAGE PACKET INDEX      ZZZ
;  1  !MMM    S_PATTERN : WORD,                      !PATTERN FOR DUP WRITES            ZZZ
;  1  !MMM    S_DUPPKT : WORD,                       !DBN BYTE COUNTER                  ZZZ
;  1      RD_COUNT : WORD,                     ! NUMBER OF WINCHESTER UNITS       ZZZ
;  1      BRLEVEL : WORD,                      !BUS REQUEST PRIORITY LEVEL        ZZZ
;  1      D_FAIL : BYTE,                       !SIGNIFIES DUP TYPE ERROR          ZZZ
;  1      DBM107,
;  1      DU_MSG,
;  1      DU_RSN : vector [11],
;  1      RPT1,
;  1      RPT2,
;  1      RPT3,
;  1      RPT4,
;  1      RPT5,
;  1      RPT6,
;  1      RPT7,
;  1      RPT8,
;  1      RPT9,
;  1      RPT10,
;  1      RPT11,
;  1      RPT12,
;  1      RPT13,
;  1      RPT14,
;  1      RPT15,
;  1      RPT16,
;  1  !ZZZ    RPT17,
;  1  !ZZZ    RPT18,
;  1  !ZZZ    RPT19,
;  1
;  1      MSG_01,
;  1      EGS_01,
;  1      EBS_01,
;  1      EBD_10,
;  1      EBD_12,
;  1      EBD_13,
;  1      EBD_14,
;  1      EBD_18,
;  1      EBD_19,
;  1      EBD_24,
;  1      ERR_00,
```

```
;    1    ERR_COD : vector [14],
;    1    ELG_OO,
;    1    ELG_FMT : vector [5],
;    1    EX_TIM,
;    1    !EX_DUP,                                              !MMM
;    1    EX_LB,                                                !MMM
;    1    XX13,                                                 !ZZZ
;    1    XX23,                                                 !ZZZ
;    1    XX32,                                                 !ZZZ
;    1    XX33,                                                 !ZZZ
;    1    XX34,                                                 !ZZZ
;    1    EX_SA,
;    1    EX_SC,
;    1    EX_SBO,
;    1    EX_SB,
;    1    EX_RP,
;    1    EX_WRD,
;    1    EX_CMD,
;    1    EX_RD,
;    1    EX_WRT,
;    1    EX_CMP,
;    1    EX_ONL,
;    1    EX_ACC,
;    1    EX_OP,
;    1    EX_BB2,
;    1    EX_BB12,
;    1    EX_BBU2,
;    1    EX_LBN2,
;    1    EX_PBN2,
;    1    EX_LBR2,
;    1    EX_LBW2,
;    1    EX_RBN2,
;    1    EX_CBC,
;    1    EX_CBR,
;    1    EX_CBW,
;    1    EX_BC,
;    1    EX_BD,
;    1    EX_BDR,
;    1    EX_BDW,
;    1    SC_SDI,
;    1    SC_CON,
;    1    SC_DUP,
;    1    SC_ONL,
;    1    SC_SON,
;    1    SC_INR,                    !MMM
;    1    SC_INV,                    !MMM
;    1    SC_UNK,
;    1    SC_VOL,
;    1    SC_IOP,
;    1    SC_DIS,
;    1    SC_FER,
;    1    SC_FE2,
;    1    SC_ISH,
```

```
;  1        SC_IS2,
;  1        SC_DST,
;  1        SC_DS2,
;  1        SC_ECC,
;  1        SC_ECD,
;  1        SC_RCT,
;  1        SC_FUL,
;  1        SC_576,
;  1        SC_FCT,
;  1        SC_SWP,
;  1        SC_HWP,
;  1        SC_SAF,                                      !MMM
;  1        SC_EC1,
;  1        SC_EC2,
;  1        SC_EC3,
;  1        SC_EC4,
;  1        SC_EC5,
;  1        SC_EC6,
;           SC_EC7,
;  1        SC_EC8,
;  1        SC_EC9,
;  1        SC_ODA,
;  1        SC_ODB,
;  1        SC_NXM,
;  1        SC_PAR,
;  1        SC_CTO,
;  1        SC_SDS,
;  1        SC_EDC,
;  1        SC_IDS,
;  1        SC_SRT,
;  1        SC_SRI,
;  1        SC_POE,
;  1        SC_RDY,
;  1        SC_CLK,
;  1        SC_RSP,
;  1        SC_SUR,
;  1        SC_PSP,
;  1        SC_REP,                      !MMM
;  1        SC_NBB,                      !MMM
;  1        SC_REF,                      !MMM
;  1        CER_01,
;  1        CER_02,
;  1        CNTR_ERR : vector [23],
;  1        RDRX_ERR : vector [7],
;  1        SPACE4,
;  1        CRLF,
;  1        DASH,
;  1        ASTERISK,
;  1        HWQ1,
;  1        HWQ2,
;  1        HWQ3,
;  1        HWQ4,
;  1        HWQ5,
```

```
        HWQ6A,
        HWQ6B,
        HWQ7A,
        HWQ7B,
        HWQ8,
        HWQ9,
!MMMHWQ10,                                              !ZZZ
!MMM    HWQ11,                                                    !ZZZ
        SWQ1,
        SWQ2,
        SWQ4,
        SWQ7,
        SWQ9,
        SWQ10,
        SWQ11,
        SWQ12,
        SWQ13,
        SWQ14,
        SWQ15,
        SWQ17,
        SWQ19
        SWQ20,
        SWQ21,
        SWQ22,
        SWQ23,
        SWQ24,
        !MMMSWQ25,
        SWQ26,                                          !ZZZ
        SWQ27,                                          !MMM
        SWQ28,                                          !MMM
        EH_0,                                           !ZZZ
        EH_1,                                           !ZZZ
        EH_2,                                           !ZZZ
        EH_3,                                           !ZZZ
        EH_4,                                           !ZZZ
        EH_5,                                           !ZZZ
        EH_6,                                           !ZZZ
        EH_7,                                           !ZZZ
        EH_8,                                           !ZZZ
        EH_9,                                           !ZZZ
        EH_10,                                          !ZZZ
        EH_12,                                          !ZZZ
        EH_13,                                          !ZZZ
        SWM1,
        NULL,
        SWP_FLAGS : word,
        L$HIMEM,
        L$LUN,
        L$UNIT;
!       O_BRK;

own
!MMM    TBL_SUC : vector [17] initial (NULL, SC_SDI, SC CON, NULL, SC_DUP, NULL, NULL,
```

```
!MMM            NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SON),
    TBL_SUC : vector [19] initial (NULL, SC_SDI, SC_CON, NULL, SC_DUP, NULL, NULL,
              NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SON, SC_INR, SC_INV),      !MMM
    TBL_OFL : vector [9] initial (SC_UNK, SC_VOL, SC_IOP, NULL, SC_DUP, NULL, NULL,
              NULL, SC_DIS),
    TBL_MFE : vector [11] initial (SC_FER, NULL, SC_ISH, SC_DST, SC_EC9, SC_576,
              SC_FCT, SC_ECC, SC_RCT, SC_FUL, SC_EC1),
!MMM    TBL_WPT : vector [3] initial (NULL, SC_SWP, SC_HWP),
    TBL_WPT : vector [4] initial (NULL, SC_SWP, SC_HWP, SC_SAF),                                    !MMM
    TBL_DAT : vector [16] initial (SC_FE2, NULL, SC_IS2, SC_DS2, SC_EC9, NULL, NULL,
              SC_ECD, SC_EC1, SC_EC2, SC_EC3, SC_EC4, SC_EC5, SC_EC6, SC_EC7, SC_EC8),
    TBL_HST : vector [5] initial (NULL, SC_ODA, SC_ODB, SC_NXM, SC_PAR),
    TBL_CNT : vector [4] initial (SC_CTO, SC_SDS, SC_EDC, SC_IDS),
    TBL_DRV : vector [9] initial (NULL, SC_SRT, SC_SRI, SC POE, SC_RDY, SC CLK, SC RSP,
              SC_SUR, SC_PSP),
    TBL_BRC : vector [5] initial (SC_REP, SC NBB, SC REF, SC_REF, SC_REF);                          !MMM
```

```
%sbttl 'TYPE AND DESCRIPTION'

EQUALS;

DEVTYP (%asciz 'RQDX or RUX50');                    ! NAME OF DEVICE SUPPORTED BY PROGRAM
DESCPIPT (%asciz 'RD/RX EXERCISER ');       ! TEST DESCRIPTION
```

```
%sbttl 'HARDWARE PARAMETER CODING SECTION'

!+
! THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
! WITH THE OPERATOR.
!-

BGNHRD;

GPRMA (HWQ1, 0, 0, %c'160000', %o'177777', YES, 1);                      ! IP ADDRESS
GPRMA (HWQ2, 2, 0, %o'4', %o'774', YES, 1);                               ! VECTOR
GPRMD (HWQ3, 4, 0, %o'377', %o'0', %o'7', YES, 1);                        ! BR LEVEL
GPRMD (HWQ4, 6, 0, %o'17', %decimal'0', %decimal'15', YES, 1);            ! RDRX DRIVE NUMBER
!MMMGPRML (HWQ10, 6, %o'000040', YES, 1);                                        !ALSO RUN DUP EXERCISER    ZZZ
!MMMXFERF (NODU);                                                              !                        ZZZ
!MMMGPRML (HWQ11, 6, %o'000100', YES, 1);                                         !WRITE DIAG AREA          ZZZ
!MMM$L (NODU);                                                            !                           ZZZ
GPRML (HWQ5, 6, %o'000200', YES, 1);                                      !TEST ENTIRE CUSTOMER AREA? ZZZ
XFERT (TOQ8);                                                             !BR IF YES                  ZZZ
GPRMD (HWQ6A, 8, 0, %o'177777', %decimal'0', %o'177777', YES, 1);         ! STARTING LBN LO           ZZZ
GPRMD (HWQ6B, 10, 0, %o'177777', %decimal'0', %o'177777', YES, 1);        ! STARTING LBN HI           ZZZ
GPRMD (HWQ7A, 12, 0, %o'177777', GP$ATLO (8), %o'177777', YES, 1);        ! ENDING LBN LO             ZZZ
GPRMD (HWQ7B, 14, 0, %o'177777', %decimal'0', %o'177777', YES, 1);        ! ENDING LBN HI             ZZZ
$L (TOQ8);                                                                !                           ZZZ
GPRML (HWQ8, 6, %o'100000', NO, 0);                                       ! WRITE ON CUST DATA AREA
XFERF (HWDONE);                                                           ! NO - DONE
GPRML (HWQ9, 6, %o'100000', NO, 1);                                       ! ** WARNING / CONFIRM
$L (HWDONE);

ENDHRD;
```

```
%sbttl  SOFTWARE PARAMETER CODING SECTION'

!+
! THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
! WITH THE OPERATOR.
!-

BGNSFT;

!GPRML (SWQ16, 4, SWF_TRC, YES, 1);                          ! ENABLE DIAGNOSTIC TRACE
GPRMD (SWQ24, 10, D, %o'177777', 0, 2359, YES, 1);           ! START TIME
GPRMD (SWQ1, 0, D, %o'177777', 0, 65535, YES, 1);            ! ERROR LIMIT
GPRMD (SWQ2, 2, D, %o'177777', 0, 99, YES, 1);               ! TRANSFER LIMIT
GPRMD (SWQ17, 8, D, %o'177777', 0, 100, YES, 1);             ! PERCENT OF RD OPERATIONS
GPRML (SWQ15, 4, SWF_CST, YES, 1);                           ! CLEAR STATISTICAL TABLES ?
GPRML (SWQ20, 4, SWF_FER, YES, 1);                           ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT SET?
GPRML (SWQ23, 4, SWF_BLK, YES, 1);                           ! HALT ON BAD-BLOCK TYPE ERRORS WITH 'HOE' FLAG?
GPRML (SWQ21, 4, SWF_HRD, YES, 1);                           ! HALT ON HARD ERRORS WITH 'HOE' FLAG SET?
GPRML (SWQ22, 4, SWF_SFT, YES, 1);                           ! HALT ON SOFT ERRORS WITH 'HOE' FLAG SET?
!MMMGPRML (SWQ25, 4, SWF_TRY, YES, 1);                       ! COUNT EACH RETRY AS ANOTHER SOFT-ERROR?
GPRML (SWQ4, 4, SWF_RDM, YES, 1);                            ! RANDOM SEEK MODE ?
XFERF (SW1);                                                 ! IF NO, DO NEXT QUESTION
XFER (SW2);                                                  !
$L (SW1);                                                    !
GPRML (SWQ19, 4, SWF_SEQ, YES, 1);                           ! RANDOM OR SEQUENTIAL SELECTION OF DRIVES
$L (SW2);                                                    !
GPRML (SWQ7, 4, SWF_CRC, YES, 1);                            ! READ-COMPARES AT CONTROLLER ?
GPRML (SWQ26, 4, SWF_APT, YES, 1);                           !RUNNING UNDER A.P.T. MONITOR?          ZZZ
DISPLAY (SWM1);                                              ! REMAINING QUESTIONS ONLY APPLY ...
GPRML (SWQ9, 4, SWF_CWC, YES, 1);                            ! WRITE-COMPARES AT CONTROLLER ?
XFERF (SW3);                                                 ! IF NO, DO NEXT QUESTION
XFER (SW4);                                                  !
$L (SW3);                                                    !
GPRML (SWQ10, 4, SWF_HWC, YES, 1);                           ! CHECK WRITES AT HOST BY READING ?
$L (SW4);                                                    !
GPRML (SWQ11, 4, SWF_UDP, YES, 1);                           ! USER-DEFINED DATA PATTERN ?
XFERF (SW5);                                                 ! IF NO, DO NEXT QUESTION
XFER (SW6);                                                  !
$L (SW5);                                                    !
GPRMD (SWQ12, 6, D, %o'177777', 0, DP_CNT, YES, 1);          ! SELECT PRE-DEFINED DATA PATTERN
XFER (SW7);                                                  ! DONE
$L (SW6);                                                    !
GPRMD (SWQ13, 12, D, %o'177777', 1, MAX_UDP_CNT, YES, 1);    ! NO. OF WORDS IN USER DATA PATTERN
!MMMGPRMD (SWQ14, 14, O, %o'177777', 0, %o'177777', NO, 12); ! PATTERN VALUES
GPRMD (SWQ14, 14, O, %o'177777', 0, %o'177777', YES, 12);    ! PATTERN VALUES MMM
$L (SW7);
GPRML (SWQ27, 48, %o'000001', YES, 1);                               ! Manufacturing Test (Reduce Duty Cycle) MMM
GPRML (SWQ28, 50, %o'000001', YES, 1);                                ! Enable Host Memory Parity MMM
ENDSFT;
```

```
;     1
;     1
;     1    %sbttl 'REPORT CODING SECTION'
;     1
;     1
;     1    !+
;     1    ! THE REPORT CODING SECTION CONTAINS THE
;     1    ! "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;     1    !-
;     1
;     1
;     2    BGNRPT;
;     2
;     2    local
;     2        CUR_PRIORITY : word;
;     2
;     2    GETPRI (CUR_PRIORITY);
;     2    !ZZZ SETPRI (PRIO4);                              !ZZZ
;     2    SETPRI (.BRLEVEL);                        !ZZZ
;     2
;     2
;     2    PRINTS (RPT1);
;     2    PRINTS (RPT2);
;     2    PRINTS (RPT3);
;     2    PRINTS (RPT4);
;     2    PRINTS (RPT5);
;     2    PRINTS (RPT6);
;     2
;     2    incr CTLR from 0 to MAX_CTLR - 1 do
;     2
;     3        begin
;     3        SET_CPAR (.CTLR);
;     3
;     3        incr DISK from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;     3
;     4            begin
;     4            SET_UPAR (.DISK);
;     4
;     4
;     4            if .CST_ADDR [.DISK + OF_DATA, D_PRES] eql PRESENT
;     4            then
;     4
;     5                begin
; P   5                PRINTS (RPT7,
;     5                    .L$LUN, .CST_ADDR [.DISK + OF_DATA, D_DISK_NUM], CST [.CTLR, .DISK + OF_NAME_O, D_NAME_O]);
; P   5                PRINTS (RPT8,
; P   5                    .T_ADDR [TOT_READS_HI], .T_ADDR [TOT_READS_LO],
;     5                    .T_ADDR [MTOT_BYT_RED], .T_ADDR [TOT_BYT_RED_HI], .T_ADDR [TOT_BYT_RED_LO]);
; P   5                PRINTS (RPT8,
; P   5                    .T_ADDR [TOT_WRITES_HI], .T_ADDR [TOT_WRITES_LO],
;     5                    .T_ADDR [MTOT_BYT_WRT], .T_ADDR [TOT_BYT_WRT_HI], .T_ADDR [TOT_BYT_WRT_LO]);
; P   5                PRINTS (RPT9,
; P   5                    .T_ADDR [ERR_HRD_SEK], .T_ADDR [ERR_HRD_DAT], .T_ADDR [ERR_HRD_DRV], .T_ADDR [ERR_HRD_HST],
```

```
;  5              .T_ADDR [ERR_SFT_SEK], .T_ADDR [ERR_SFT_DAT], .T_ADDR [ERR_SFT_DRV], .T_ADDR [ERR_SFT_HST]);
;  4          end;
;  3      end;
;  3
;  3
;  3    if .CST [.CTLR, STATE] eql PRESENT
;  3    then
;  3
;  4        begin
;  4        PRINTS (RPT10);
;  4        PRINTS (RPT11, .C_ERR_TBL [.CTLR, C_ERR_HRD], .C_ERR_TBL [.CTLR, C_ERR_SFT]);
;  3        end;
;  3
;  3
;  2    end;
;  2
;  2  SETPRI (.CUR_PRIORITY);
;  2
;  2  !MMMIF .RD_COUNT NEQ 0                                                  !IF THERE IS A WINCHESTER     ZZZ
;  2  !MMMTHEN                                                                !THEN OUTPUT EXTRA LINES      ZZZ
;  2
;  2  !MMMbegin                                                                 ! PRINTS DUP DATA
;  2  !MMMprints(crlf);
;  2  !MMMPRINTS(RPT13);
;  2  !MMMPRINTS(PPT14);
;  2  !MMMPRINTS(RPT15);
;  2  !MMMINCR CTLR FROM 0 TO MAX_CTLR-1 DO
;  2  !MMM            BEGIN
;  2  !MMM    SET_CPAR(.CTLR);
;  2  !MMM    INCR DISK FROM (0+OF_UN) TO (3*UNIT_SIZE+OF_UN) BY UNIT_SIZE DO
;  2  !MMM            BEGIN
;  2  !MMM            SET_UPAR(.DISK);
;  2  !MMM            IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_51 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;  2  !MMM            THEN
;  2  !MMM        PRINTS (RPT16,
;  2  !MMM            .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;  2  !MMM            .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;  2
;  2  !ZZZ            IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_52 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;  2  !ZZZ            THEN
;  2  !ZZZ        PRINTS (RPT18,
;  2  !ZZZ            .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;  2  !ZZZ            .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;  2  !MMM            END;
;  2  !MMM    END;
;  2  !MMM           PRINTS (CRLF);
;  2  !MMMEND;
;  2
;  2  PRINTS (CRLF);                                  !MMM
;  2  EMS_TIM ();                                     !MMM
;  2
;  1  ENDRPT;
```

```
                                                  .TITLE  ZRQAM2 RD/RX EXERCISER
                                                  .IDENT  /V02.3/
                                                  .ENABL  AMA

000000                                            .PSECT  $CODE$,  RO
000000      122     121     104       L$DVTYP::
                                                  .ASCII  /RQD/
000003      130     040     157                   .ASCII  /X o/
000006      162     040     122                   .ASCII  /r R/
000011      125     130     065                   .ASCII  /UX5/
000014      060     000                           .ASCII  /0/<00>
000016                                            .BLKB   2
000020      122     104     057       L$DESC::.ASCII  /RD/<57>
000023      122     130     040                   .ASCII  /RX /
000026      105     130     105                   .ASCII  /EXE/
000031      122     103     111                   .ASCII  /RCI/
000034      123     105     122                   .ASCII  /SER/
000037      000                                   .ASCII  <00>
000040                                            .BLKB   2
000042   000000C                      L$HRDLN::
                                                  .WORD   <<<L$NDHRD-L$HRDLN>/2>-1>
000044   000031                       GP$1::      .WORD   31
000046   000000G                                  .WORD   HWQ1
000050   160000                                   .WORD   -20000
000052   177777                                   .WORD   -1
000054   001031                       GP$2::      .WORD   1031
000056   000000G                                  .WORD   HWQ2
000060   000004                                   .WORD   4
000062   000774                                   .WORD   774
000064   002032                       GP$3::      .WORD   2032
000066   000000G                                  .WORD   HWQ3
000070   000377                                   .WORD   377
000072   000000                                   .WORD   0
000074   000007                                   .WORD   7
000076   003052                       GP$4::      .WORD   3052
000100   000000G                                  .WORD   HWQ4
000102   000017                                   .WORD   17
000104   000000                                   .WORD   0
000106   000017                                   .WORD   17
000110   003130                       GP$5::      .WORD   3130
000112   000000G                                  .WORD   HWQ5
000114   000200                                   .WORD   200
000116   000000C                      $TOQ8:      .WORD   <<<<$LTOQ8-$TOQ8>*400>+4>+20>
000120   004032                       GP$6::      .WORD   4032
000122   000000G                                  .WORD   HWQ6A
000124   177777                                   .WORD   -1
000126   000000                                   .WORD   0
000130   177777                                   .WORD   -1
000132   005032                       GP$7::      .WORD   5032
000134   000000G                                  .WORD   HWQ6B
000136   177777                                   .WORD   -1
```

```
000140  000000                         .WORD   0
000142  177777                         .WORD   -1
000144  006432              GP$ ::     .WORD   6432
000146  000000G                        .WORD   HWQ7A
000150  177777                         .WORD   -1
000152  000004                         .WORD   4
000154  177777                         .WORD   -1
000156  000001                         .WORD   1
000160  007032              GP$9::     .WORD   7032
000162  000000G                        .WORD   HWQ7B
000164  177777                         .WORD   -1
000166  000000                         .WORD   0
000170  177777                         .WORD   -1
000172  001004              $LTOQ8:    .WORD   1004
000174  003120              GP$10::    .WORD   3120
000176  000000G                        .WORD   HWQ8
000200  100000                         .WORD   -100000
000202  000000C             $HWDONE::  .WORD   <<<<$LHWDONE-$HWDONE>*400>+4>+40>
000204  003120              GP$11::    .WORD   3120
000206  000000G                        .WORD   HWQ9
000210  100000                         .WORD   -100000
000212  001004              $LHWDONE:
                                       .WORD   1004
000214                      L$NDHRD::
                                       .BLKW   1
000216  000000C             L$SFTLN::
                                       .WORD   <<<L$NDSFT-L$SFTLN>/2>-1>
000220  005052              GP$12::    .WORD   5052
000222  000000G                        .WORD   SWQ24
000224  177777                         .WORD   -1
000226  000000                         .WORD   0
000230  004467                         .WORD   4467
000232  000052              GP$13::    .WORD   52
000234  000000G                        .WORD   SWQ1
000236  177777                         .WORD   -1
000240  000000                         .WORD   0
000242  177777                         .WORD   -1
000244  001052              GP$14::    .WORD   1052
000246  000000G                        .WORD   SWQ2
000250  177777                         .WORD   -1
000252  000000                         .WORD   0
000254  000143                         .WORD   143
000256  004052              GP$15::    .WORD   4052
000260  000000G                        .WORD   SWQ17
000262  177777                         .WORD   -1
000264  000000                         .WORD   0
000266  C00144                         .WORD   144
000270  002130              GP$16::    .WORD   2130
000272  000000G                        .WURD   SWQ15
000274  000200                         .WORD   200
000276  002130              GP$17::    .WORD   2130
000300  000000G                        .WORD   SWQ20
000302  004000                         .WORD   4000
```

```
000304  002130            GP$18:: .WORD   2130
000306  000000G                   .WORD   SWQ23
000310  040000                    .WORD   40000
000312  002130            GP$19:: .WORD   2130
000314  000000G                   .WORD   SWQ21
000316  010000                    .WORD   10000
000320  002130            GP$20:: .WORD   2130
000322  000000G                   .WORD   SWQ22
000324  020000                    .WORD   20000
000326  002130            GP$21:: .WORD   2130
000330  000000G                   .WORD   SWQ4
000332  000002                    .WORD   2
000334  000000C           $SW1:   .WORD   <<<<$LSW1-$SW1>*400>+4>+40>
000336  000000C           $SW2:   .WORD   <<<$LSW2-$SW2>*400>+4>
000340  001004            $LSW1:  .WORD   1004
000342  002130            GP$22:: .WORD   2130
000344  000000G                   .WORD   SWQ19
000346  001000                    .WORD   1000
000350  001004            $LSW2:  .WORD   1004
000352  002130            GP$23:: .WORD   2130
000354  000000G                   .WORD   SWQ7
000356  000004                    .WORD   4
000360  002130            GP$24:: .WORD   2130
000362  000000G                   .WORD   SWQ26
000364  000001                    .WORD   1
000366  000003            GP$DISP::
                                  .WORD   3
000370  000000G                   .WORD   SWM1
000372  002130            GP$25:: .WORD   2130
000374  000000G                   .WORD   SWQ9
000376  000020                    .WORD   20
000400  000000C           $SW3:   .WORD   <<<<$LSW3-$SW3>*400>+4>+40>
000402  000000C           $SW4:   .WORD   <<<$LSW4-$SW4>*400>+4>
000404  001004            $LSW3:  .WORD   1004
000406  002130            GP$26:: .WORD   2130
000410  000000G                   .WORD   SWQ10
000412  000040                    .WORD   40
000414  001004            $LSW4:  .WORD   1004
000416  002130            GP$27:: .WORD   2130
000420  000000G                   .WORD   SWQ11
000422  000100                    .WORD   100
000424  000000C           $SW5:   .WORD   <<<<$LSW5-$SW5>*400>+4>+40>
000426  000000C           $SW6:   .WORD   <<<$LSW6-$SW6>*400>+4>
000430  001004            $LSW5:  .WORD   1004
000432  003052            GP$28:: .WORD   3052
000434  000000G                   .WORD   SWQ12
000436  177777                    .WORD   -1
000440  000000                    .WORD   0
000442  000025                    .WORD   25
000444  000000C           $SW7:   .WORD   <<<$LSW7-$SW7>*400>+4>
000446  001004            $LSW6:  .WORD   1004
000450  006052            GP$29:: .WORD   6052
000452  000000G                   .WORD   SWQ13
```

```
000454  177777                          .WORD   -1
000456  000001                          .WORD   1
000460  000020                          .WORD   20
000462  007232                GP$30:: .WORD   7232
000464  000000G                         .WORD   SWQ14
000466  177777                          .WORD   -1
000470  000000                          .WORD   0
000472  177777                          .WORD   -1
000474  000006                          .WORD   6
000476  001004                $LSW7:  .WORD   1004
000500  030130                GP$31:: .WORD   30130
000502  000000G                         .WORD   SWQ27
000504  000001                          .WORD   1
000506  031130                GP$32:: .WORD   31130
000510  000000G                         .WORD   SWQ28
000512  000001                          .WORD   1
000514                        L$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                         .WORD   SC.INR
000044  000000G                         .WORD   SC.INV
000046  000000G               TBL.OFL:.WORD   SC.UNK
000050  000000G                         .WORD   SC.VOL
000052  000000G                         .WORD   SC.IOP
000054  000000G                         .WORD   NULL
000056  000000G                         .WORD   SC.DUP
000060  000000G                         .WORD   NULL
000062  000000G                         .WORD   NULL
000064  000000G                         .WORD   NULL
000066  000000G                         .WORD   SC.DIS
000070  000000G               TBL.MFE:.WORD   SC.FER
000072  000000G                         .WORD   NULL
000074  000000G                         .WORD   SC.ISH
```

```
000076  000000G                 .WORD   SC.DST
000100  000000G                 .WORD   SC.EC9
000102  000000G                 .WORD   SC.576
000104  000000G                 .WORD   SC.FCT
000106  000000G                 .WORD   SC.ECC
000110  000000G                 .WORD   SC.RCT
000112  000000G                 .WORD   SC.FUL
000114  000000G                 .WORD   SC.EC1
000116  000000G       TBL.WPT:: .WORD   NULL
000120  000000G                 .WORD   SC.SWP
000122  000000G                 .WORD   SC.HWP
000124  000000G                 .WORD   SC.SAF
000126  000000G       TBL.DAT:: .WORD   SC.FE2
000130  000000G                 .WORD   NULL
000132  000000G                 .WORD   SC.IS2
000134  000000G                 .WORD   SC.DS2
000136  000000G                 .WORD   SC.EC9
000140  000000G                 .WORD   NULL
000142  000000G                 .WORD   NULL
000144  000000G                 .WORD   SC.ECD
000146  000000G                 .WORD   SC.EC1
000150  000000G                 .WORD   SC.EC2
000152  000000G                 .WORD   SC.EC3
000154  000000G                 .WORD   SC.EC4
000156  000000G                 .WORD   SC.EC5
000160  000000G                 .WORD   SC.EC6
000162  000000G                 .WORD   SC.EC7
000164  000000G                 .WORD   SC.EC8
000166  000000G       TBL.HST:: .WORD   NULL
000170  000000G                 .WORD   SC.ODA
000172  000000G                 .WORD   SC.ODB
000174  000000G                 .WORD   SC.NXM
000176  000000G                 .WORD   SC.PAR
000200  000000G       TBL.CNT:: .WORD   SC.CTO
000202  000000G                 .WORD   SC.SDS
000204  000000G                 .WORD   SC.EDC
000206  000000G                 .WORD   SC.IDS
000210  000000G       TBL.DRV:: .WORD   NULL
000212  000000G                 .WORD   SC.SRT
000214  000000G                 .WORD   SC.SRI
000216  000000G                 .WORD   SC.POE
000220  000000G                 .WORD   SC.RDY
000222  000000G                 .WORD   SC.CLK
000224  000000G                 .WORD   SC.RSP
000226  000000G                 .WORD   SC.SUR
000230  000000G                 .WORD   SC.PSP
000232  000000G       TBL.BRC:: .WORD   SC.REP
000234  000000G                 .WORD   SC.NBB
000236  000000G                 .WORD   SC.REF
000240  000000G                 .WORD   SC.REF
000242  000000G                 .WORD   SC.REF
```

```
.GLOBL  CST, CST.ADDR, DCT, DCT.ADDR, RDRX.ADDR
.GLOBL  IRDRX.ADDR, BST, TALLY, T.ADDR
.GLOBL  TRK.SGN, RDM.CNT, RANDOM, C.ERR.TBL
.GLOBL  MSCP.PKT, IPKT.ADDR, PKT.USE, RETPKT
.GLOBL  RP.USE, RP.INDX, RP.ADDR, ELOG.PKT
.GLOBL  BUFF.ADDR, BUFF.OWN, IODQ, IODQ.IN
.GLOBL  IODQ.OUT, ENTRY.REASON, EOP.FLAG
.GLOBL  DUP.FLAGS, CCTLR, CDISK, CUOFF
.GLOBL  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, FER0.LBN
.GLOBL  FER1.LBN, CLK.PRESENT, HOE.FLAG
.GLOBL  FORCED.ERROR, FER.LBN, FER.BC
.GLOBL  INIT.OCCURED, ADDR.VECT.OK, DBM5
.GLOBL  DBM125, DBM126, TYPER, TYPEW, BAL.IN.PROGRESS
.GLOBL  FORCE.WR, CSR.MEM, CSR.ADD, P.INDEX
.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  EX.LB, XX13, XX23, XX32, XX33
.GLOBL  XX34, EX.SA, EX.SC, EX.SBO, EX.SB
.GLOBL  EX.RP, EX.WRD, EX.CMD, EX.RD, EX.WRT
.GLOBL  EX.CMP, EX.ONL, EX.ACC, EX.OP
.GLOBL  EX.BB2, EX.BB12, EX.BBU2, EX.LBN2
.GLOBL  EX.PBN2, EX.LBR2, EX.LBW2, EX.RBN2
.GLOBL  EX.CBC, EX.CBR, EX.CBW, EX.BC
.GLOBL  EX.BD, EX.BDR, EX.BDW, SC.SDI
.GLOBL  SC.CON, SC.DUP, SC.ONL, SC.SON
.GLOBL  SC.INR, SC.INV, SC.UNK, SC.VOL
.GLOBL  SC.IOP, SC.DIS, SC.FER, SC.FE2
.GLOBL  SC.ISH, SC.IS2, SC.DST, SC.DS2
.GLOBL  SC.ECC, SC.ECD, SC.RCT, SC.FUL
.GLOBL  SC.576, SC.FCT, SC.SWP, SC.HWP
.GLOBL  SC.SAF, SC.EC1, SC.EC2, SC.EC3
.GLOBL  SC.EC4, SC.EC5, SC.EC6, SC.EC7
.GLOBL  SC.EC8, SC.EC9, SC.ODA, SC.ODB
.GLOBL  SC.NXM, SC.PAR, SC.CTO, SC.SDS
.GLOBL  SC.EDC, SC.IDS, SC.SRT, SC.SRI
.GLOBL  SC.POE, SC.RDY, SC.CLK, SC.RSP
.GLOBL  SC.SUR, SC.PSP, SC.REP, SC.NBB
.GLOBL  SC.REF, CER.01, CER.02, CNTR.ERR
.GLOBL  RDRX.ERR, SPACE4, CRLF, DASH, ASTERISK
.GLOBL  HWQ1, HWQ2, HWQ3, HWQ4, HWQ5, HWQ6A
.GLOBL  HWQ6B, HWQ7A, HWQ7B, HWQ8, HWQ9
.GLOBL  SWQ1, SWQ2, SWQ4, SWQ7, SWQ9, SWQ10
```

```
                                 .GLOBL  SWQ11, SWQ12, SWQ13, SWQ14, SWQ15
                                 .GLOBL  SWQ17, SWQ19, SWQ20, SWQ21, SWQ22
                                 .GLOBL  SWQ23, SWQ24, SWQ26, SWQ27, SWQ28
                                 .GLOBL  EH.0, EH.1, EH.2, EH.3, EH.4, EH.5
                                 .GLOBL  EH.6, EH.7, EH.8, EH.9, EH.10
                                 .GLOBL  EH.12, EH.13, SWM1, NULL, SWP.FLAGS
                                 .GLOBL  L$HIMEM, L$LUN, L$UNIT

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

```
        000040                  IDU==           40
        000100                  ISR==           100
        000200                  UAM==           200
        000400                  BOE==           400
        001000                  PNT==           1000
        002000                  PRI==           2000
        004000                  IXE==           4000
        010000                  IBE==           10000
        020000                  IER==           20000
        040000                  LOE==           40000
        100000                  HOE==            100000
        000044'                 L$HARD==        L$HRDLN+2
        000220'                  $SOFT==        L$SFTLN+2

                                        .SBTTL  LRPT REPORT CODING SECTION
000516                                  .PSECT  $CODE$,  RO

000000  004137  000000G         LRPT:   JSR     R1,$SAVE4               ;                       5819
000004  104440                          TRAP    40                      ;                       5836
000006  010004                          MOV     R0,R4                   ; *.CUR.PRIORITY
000010  013700  000000G                 MOV     BRLEVEL,R0              ;                       5838
000014  104441                          TRAP    41
000016  012746  000000G                 MOV     #RPT1,-(SP)             ;                       5841
000022  012746  000001                  MOV     #1,-(SP)
000026  010600                          MOV     SP,R0                   ; SP,*
000030  104416                          TRAP    16
000032  012716  000000G                 MOV     #RPT2,(SP)              ;                       5842
000036  012746  000001                  MOV     #1,-(SP)
000042  010600                          MOV     SP,R0                   ; SP,*
000044  104416                          TRAP    16
000046  012716  000000G                 MOV     #RPT3,(SP)              ;                       5843
000052  012746  000001                  MOV     #1,-(SP)
000056  010600                          MOV     SP,R0                   ; SP,*
000060  104416                          TRAP    16
000062  012716  000000G                 MOV     #RPT4,(SP)              ;                       5844
000066  012746  000001                  MOV     #1,-(SP)
000072  010600                          MOV     SP,R0                   ; SP,*
000074  104416                          TRAP    16
000076  012716  000000G                 MOV     #RPT5,(SP)              ;                       5845
000102  012746  000001                  MOV     #1,-(SP)
000106  010600                          MOV     SP,R0                   ; SP,*
000110  104416                          TRAP    16
000112  012716  000000G                 MOV     #RPT6,(SP)              ;                       5846
000116  012746  000001                  MOV     #1,-(SP)
000122  010600                          MOV     SP,R0                   ; SP,*
000124  104416                          TRAP    16
000126  005002                          CLR     R2                      ; CTLR                  5848
000130  010216                  1$:     MOV     R2,(SP)                 ; CTLR,*                5851
000132  004737  000000V                 JSR     PC,SET.CPAR
000136  012703  000003                  MOV     #3,R3                   ; *,DISK                5853
000142  010316                  2$:     MOV     R3,(SP)                 ; DISK,*                5856
000144  004737  000000V                 JSR     PC,SET.UPAR
```

```
000150  010301                  MOV     R3,R1                   ; DISK,*        5859
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,*        5864
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               ;               5867
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               ;               5870
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               ;               5873
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                  ;                       5862
000460  062703  000012      3$:     ADD     #12,R3                  ; *,DISK                5853
000464  020327  000041              CMP     R3,#41                  ; DISK,*
000470  003624                      BLE     2$
000472  010216                      MOV     R2,(SP)                 ; CTLR,*                5878
000474  012746  000126              MOV     #1_6,-(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)             ;                       5882
000520  012746  000001              MOV     #1,-(SP)
000524  010600                      MOV     SP,R0                   ; SP,*
000526  104416                      TRAP    16
000530  010200                      MOV     R2,R0                   ; CTLR,*                5883
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                  ;                       5881
000570  005202              4$:     INC     R2                      ; CTLR                  5848
000572  000243                      .WORD CLV!CLC

000574  003002                      BGT     5$
000576  000137  000646'             JMP     1$
000602  010400              5$:     MOV     R4,R0                   ; CUR.PRIORITY,*        5889
000604  104441                      TRAP    41
000606  012716  000000G             MOV     #CRLF,(SP)              ;                       5921
000612  012746  000001              MOV     #1,-(SP)
000616  010600                      MOV     SP,R0                   ; SP,*
000620  104416                      TRAP    16
000622  004737  000000V             JSR     PC,EMS.TIM              ;                       5922
000626  062706  000020              ADD     #20,SP                  ;                       5819
000632  000207                      RTS     PC

; Routine Size:  206 words,     Routine Base:  $CODE$ + 0516
; Maximum stack depth per invocation:  40 words
```

```
                                        .SBTTL  L$RPT REPORT CODING SECTION
000000  004737  000516'         L$RPT:: JSR     PC,LRPT                         ;               5922
000004  104425                          TRAP    25
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 1352
; Maximum stack depth per invocation:  2 words

;   5925  1
```

```
;  1   %sbttl 'INITIALIZE SECTION'
;  1
;  2   BGNINIT;
;  2
;  2   local
;  2       DELAY_MULT : word,
;  2       FLAG : byte,
;  2       TEMP : word,
;  2       HWPT_REF : ref block [HWPT_LEN, word] field (HWP_FIELDS),
;  2       CLEAR_TABLES : byte,
;  2       SMALLEST_DRIVE : byte,
;  2       BLANKS : WORD INITIAL (%o'020040'),                    !ZZZ
;  2       HWPT_ADDRESS : vector [MAX_UNITS, word];
;  2
;  2   SETPRI (PRIO7);                                   ! NO INTERRUPTS ALLOWED DURING INIT
;  2
;  2   if READEF (EF_NEW)                                ! IS THIS A NEW PASS?
;  2   then
;  3       begin
;  3       ENTRY_REASON = NEW_PASS;
;  3
;  4       if not BIT_TST (SWP_FLAGS, SWF_CST)
;  3       then
;  3           CLEAR_TABLES = FALSE
;  3       else
;  3           CLEAR_TABLES = TRUE;
;  3
;  2       end;
;  2
;  2   if READEF (EF_START)                              ! IS THIS A START?
;  2   then
;  3       begin
;  3       BRESET;
;  3       ENTRY_REASON = START;
;  3       CLEAR_TABLES = TRUE;
;  3       ADDR_VECT_OK = FALSE;
;  3       INIT_OCCURED = FALSE;
;  2       end;
;  2
;  2   if READEF (EF_RESTART)                            ! IS THIS A RESTART?
;  2   then
;  3       begin
;  3       ENTRY_REASON = RESTART;
;  3       CLEAR_TABLES = TRUE;
;  2       end;
;  2
;  2   if READEF (EF_CONTINUE)                           ! IS THIS A CONTINUE?
;  2   then
;  3       begin
;  3       ENTRY_REASON = CONT;
;  3
;  4       if not BIT_TST (SWP_FLAGS, SWF_CST)
;  3       then
```

```
;   3              CLEAR_TABLES = FALSE
;   3          else
;   3              CLEAR_TABLES = TRUE;
;   3
;   2          end;
;   2
;   2      if READEF (EF_PWR)                                                       ! ARE WE HERE BECAUSE OF POWER FAIL
;   2      then
;   3          begin
;   3          ENTRY_REASON = PWR_FAIL;
;   3          ADDR_VECT_OK = FALSE;
;   3          INIT_OCCURED = FALSE;
;   3          CLEAR_TABLES = TRUE;
;   3          PRINTF (MSG_ 1);                                                     ! "POWER DELAY - WAITING"
;   3
;   3          incr COUNT from 0 to 60 do                                           ! WAIT APPROX. 60 SECONDS
;   4              begin
;   4              DELAY_MULT = 333;
;   4              DELAY (.DELAY_MULT);
;   4              BREAK;                                                           ! BREAK FOR ACT
;   3              end;
;   3
;   2          end;
;   2
;   2      !SETVEC (O_TVEC, O_BRK, PRIO7);                                          ! SET ODT TRAP VECTOR
;   2
;   2      !+
;   2      !       MAKE SURE THAT NOT MORE THAN MAX_UNITS HAVE BEEN SPECIFIED.
;   2      !       IF THERE ARE TOO MANY, NOTIFY USER AND RETURN TO SUPERVISOR.
;   2      !       (DIAGNOSTIC IS ABORTED).
;   2      !-
;   2
;   2      if .L$UNIT gtru MAX_UNITS
;   2      then
;   3          begin
;   3          ERRSF (1, EGS_01, EMS_01);
;   3          DOCLN;
;   2          end;
;   2
;   2      !+
;   2      !       THE FOLLOWING CODE IS EXECUTED FOR ALL ENTRY REASONS EXCEPT NEW_PASS.
;   2      !       ALL RUN-TIME CONTROLLER STATUS TABLES (CSTs) ARE CLEARED TO 0, THEN
;   2      !       LOADED WITH CONFIGURATION DATA FROM THE HARDWARE P-TABLES.
;   2      !-
;   2
;   2      if .ENTRY_REASON neq NEW_PASS
;   2      then
;   3          begin
;   3          SMALLEST_DRIVE = 255;                                                ! LARGEST DISK NO. ALLOWED BY MSCP
;   3
;   3          incr COUNT from 0 to ((MAX_CTLR * CST_LEN * 2) - 2) by 2 do
;   3              (CST + .COUNT) = 0;
;   3
```

```
3       incr UNIT from 0 to (.L$UNIT - 1) do                                        ! LOOP THROUGH ALL UNITS
3
3           if (HWPT_ADDRESS [.UNIT] = GPHARD (.UNIT, HWPT_REF)) neqa 0              ! IF HWP TABLE FOUND
3           then
3
3               if .HWPT_REF [HWP_DISK_NUM] lssu .SMALLEST_DRIVE                     ! FIND OUT THE SMALLEST DISK NUMBER
3               then
3                   SMALLEST_DRIVE = .HWPT_REF [HWP_DISK_NUM];
3
3       incr UNIT from 0 to (.L$UNIT - 1) do                                        ! LOOP THROUGH ALL UNITS
3
3           if .HWPT_ADDRESS [.UNIT] neqa 0                                          ! IF HWP TABLE FOUND
3           then
4               begin
4               FLAG = NOT_FOUND;
4               HWPT_REF = .HWPT_ADDRESS [.UNIT];
4
4               incr CTLR from 0 to (MAX_CTLR - 1) do                                ! LOOP THROUGH ALL CSTs
4
4                   if .CST [.CTLR, IP_ADDR] eqla .HWPT_REF [HWP_IP_ADDR]
4                   then
4
4                       if .CST [.CTLR, (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE
4                               + OF_UN + OF_DATA, D_PRES] eql NOT_PRESENT
4                       then
5                           begin                                                    ! IF EMPTY SLOT FOUND
5                           TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
5                           CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
5                                                                                    ! COPY DISK ADDR AND PROT BIT
5                           CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
5                           CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
5                           CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
5
5                           IF .HWPT_REF [HWP_ENTIRE] EQL TRUE               !ZZZ IF DEFAULT TEST RANGE,
5                           THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;          !ZZZ MAKE HI ADDR ALL ONES
5
5                           CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =
5                                   .HWPT_REF [HWP_BEG_TRK];                  !ZZZ
5                           CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =            !ZZZ
5                                   .HWPT_REF [HWP_BEG_TRK1];                 !ZZZ
5                           CST [.CTLR, .TEMP + OF_END, D_END0] =             !ZZZ
5                                   .HWPT_REF [HWP_END_TRK];                  !ZZZ
5                           CST [.CTLR, .TEMP + OF_END1, D_END1] =            !ZZZ
5                                   .HWPT_REF [HWP_END_TRK1];                 !ZZZ
5
5                           CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;         !ZZZ BLANK NAME
5                           CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;         !ZZZ BLANK NAME
5
5
5                           CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;      !ZZZ
5                           CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =    !ZZZ
5                                   NOT (.HWPT_REF [HWP_DISK_DUPEX]);         !ZZZ
5
```

```
;   5                               CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =        !ZZZ
;   5                                       (.HWPT_REF [HWP_DISK_DUPWT]);               !ZZZ
;   5                               CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;         !ZZZ
;   5                               FLAG = FOUND;
;   5                               exitloop;
;   5                               end
;   4                           else
;   5                               begin                                               ! DUPLICATE UNIT
;   5                               PRINTF (CER_01, .HWPT_REF [HWP_DISK_NUM], .HWPT_REF [HWP_IP_ADDR]);
;   5                                                                                   ! "DUPLICATE UNIT: X AT IP: XXXXXX"
;   5                               DUR [.UNIT] = DU_CONF;                              ! CONFIGURATION ERROR
;   5                               DODU (.UNIT);                                       ! DROP UNIT
;   5                               FLAG = FOUND;
;   5                               exitloop;
;   4                               end;
;   4
;   4                   if .FLAG eql NOT_FOUND                                          ! IF NO IP MATCH TO EXISTING CST
;   4                   then
;   5                       begin
;   5
;   5                       incr CTLR from 0 to (MAX_CTLR - 1) do                       ! LOOP THROUGH EACH CST
;   5
;   5                           if .CST [.CTLR, IP_ADDR] eqla 0                         ! IF EMPTY CST FOUND
;   5                           then
;   6                               begin
;   6                               CST [.CTLR, IP_ADDR] = .HWPT_REF [HWP_IP_ADDR];
;   6                               CST [.CTLR, VEC_ADDR] = .HWPT_REF [HWP_VECTOR];
;   6                               CST [.CTLR, BR_LEV] = .HWPT_REF [HWP_BR_LEVEL];
;   6                               TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;   6                               CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;   6                                                                                   ! COPY DISK ADDR AND PROT BIT
;   6                               CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;   6                               CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;   6                               CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;   6
;   6                               IF .HWPT_REF [HWP_ENTIRE] EQL TRUE              !ZZZ IF DEFAULT TEST RANGE,
;   6                               THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;        !ZZZ MAKE HI ADDR ALL ONES
;   6
;   6                               CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =           !ZZZ
;   6                                       .HWPT_REF [HWP_BEG_TRK];                !ZZZ
;   6                               CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =          !ZZZ
;   6                                       .HWPT_REF [HWP_BEG_TRK1];               !ZZZ
;   6                               CST [.CTLR, .TEMP + OF_END, D_END0] =           !ZZZ
;   6                                       .HWPT_REF [HWP_END_TRK];                !ZZZ
;   6                               CST [.CTLR, .TEMP + OF_END1, D_END1] =          !ZZZ
;   6                                       .HWPT_REF [HWP_END_TRK1];               !ZZZ
;   6
;   6                               CST [.CTLR, .TEMP + OF_NAME_C, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;   6                               CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;   6
;   6
;   6                               CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;    !ZZZ
;   6                               CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =  !ZZZ
```

```
; 6                                   NOT (.HWPT_REF [HWP_DISK_DUPEX]);              !ZZZ
; 6                            CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =     !ZZZ
; 6                                   (.HWPT_REF [HWP_DISK_DUPWT]);             !ZZZ
; 6                            CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;      !ZZZ
; 6                            FLAG = FOUND;
; 6                            exitloop;
; 5                            end;                                              ! IF EMPTY CST FOUND
; 5
; 5                    if .FLAG eql NOT_FOUND                                    ! IF NO EMPTY CST FOUND
; 5                    then
; 6                        begin
; 6                        PRINTF (CER_02, MAX_CTLR);                            ! "MORE THAN X IP ADDRESSES."
; 6                        DUR [.UNIT] = DU_CONF;                                ! CONFIGURATION ERROR
; 6                        DODU (.UNIT);                                         ! DROP UNIT
; 5                        end;
; 5
; 4                    end;                                                      ! IF NO IP ADDR MATCH IN CST
; 4
; 3                end;                                                          ! IF GPHARD RETURNS A HWP TABLE
; 3        !
; 3        ! CONFIGURATON CHECK FOR LEGAL RDRX UNIT MIX BECAUSE WE HAVE DIFFERENT
; 3        ! DRIVES :  THE RD51, RD52, AND RX50.
; 3        ! (NEEDED?)
; 3        !
; 2        end;                                                                  ! END OF "NON NEW_PASS" INIT
; 2
; 2    if .ENTRY_REASON eql NEW_PASS
; 2    then
; 3        begin
; 3
; 3        incr UNIT from 0 to (.L$UNIT - 1) do
; 3            GPHARD (.UNIT, HWPT_REF);                                         ! DUMMY GPHARDs FOR NEW PASS
; 3
; 3        incr CTLR from 0 to (MAX_CTLR - 1) do
; 4            begin
; 4            CST [.CTLR, U_CNT] = 0;                                           ! REINITIALIZE UNIT COUNT
; 4
; 4            incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
; 4                CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;             ! START EACH UNIT AS OFFLINE
; 4
; 3            end;
; 3
; 2        end;
; 2
; 2    if .ENTRY_REASON eql START
; 2    then
; 3        begin
; 3        CTLR_CNT = 0;                                                         ! NUMBER OF CONFIGURED CONTROLLERS
; 3
; 3        incr CTLR from 0 to (MAX_CTLR - 1) do
; 3
; 3            if .CST [.CTLR, IP_ADDR] neqa 0                                   ! IF CONTROLLER IS PRESENT
; 3            then
```

```
                    CTLR_CNT = .CTLR_CNT + 1;                   ! INCREMENT CONTROLLER COUNT

            MEMORY (FREE_MEM_ADDR);                             ! GET START OF FREE MEMORY

            end;                                                ! END OF "START" INITIALIZATION

        !+
        !       CLEAR STATISTICS TABLES
        !-

        incr UNITS from 0 to MAX_UNITS - 1 do                   ! CLEAR CURRENT STATISTICS
            incr COUNT from 0 to TALLY_CLEAR - 1 do
                TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;

        if .CLEAR_TABLES                                        ! IF CLEAR TABLES ON EVERY PASS
        then
            incr UNITS from 0 to MAX_UNITS - 1 do
                incr COUNT from TALLY_CLEAR to TALLY_LEN - 1 do   ! INITIALIZE TOTALS
                    TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;      !

        if .CLEAR_TABLES
        then
            incr CTLR from 0 to MAX_CTLR - 1 do
                begin
                C_ERR_TBL [.CTLR, C_ERR_HRD] = 0;                 ! INITALIZE CONTROLLE  ERRORS
                C_ERR_TBL [.CTLR, C_ERR_SFT] = 0;                 !
                end;

        !+
        !       MISCELLANEOUS INITIALIZATON
        !-

        incr CTLR from 0 to (MAX_CTLR - 1) do                   ! INIT NO. OF OUTSTANDING QIOs
            QIO [.CTLR] = 0;

        incr COUNT from 0 to (RP_CNT - 1) do                    ! INITIALIZE RETURN PACKET POOL
            RP_USE [.COUNT] = -1;

        if .CLK_PRESENT                                         ! STOP CLOCK IF PRESENT
        then
            LINE_CLOCK = 0;

        IODQ_IN = IODQ_OUT = 0;                                 ! INIT I/O DONE QUEUE POINTERS
        CRN_LOW = CRN_HIGH = 0;                                 ! INIT COMMAND REFERENCE NUMBER
        SETPRI (PRIO0);                                         ! SET PROGRAM PRIORITY TO 0


        ENDINIT;

                                .GLOBL  L$DLY
```

```
                                                .SBTTL  LINIT INITIALIZE SECTION
000000  004137  000000G             LINIT:  JSR     R1,$SAVE5                       ;                           5924
000004  162706  000030                      SUB     #30,SP
000010  012746  020040                      MOV     #20040,-(SP)                    ; *,BLANKS
000014  012700  000340                      MOV     #340,R0                         ;                           5940
000020  104441                              TRAP    41
000022  012700  000035                      MOV     #35,R0                          ;                           5942
000026  104447                              TRAP    47
000030  103014                              BHIS    2$
000032  112737  000005  000000G             MOVB    #5,ENTRY.REASON                 ;                           5945
000040  105737  000000G                     TSTB    SWP.FLAGS                       ;                           5947
000044  100403                              BMI     1$
000046  105066  000012                      CLRB    12(SP)                          ; CLEAR.TABLES              5949
000052  000403                              BR      2$                              ;                           5947
000054  112766  000001  000012      1$:     MOVB    #1,12(SP)                       ; *,CLEAR.TABLES            5951
000062  012700  000040              2$:     MOV     #40,R0                          ;                           5955
000066  104447                              TRAP    47
000070  103013                              BHIS    3$
000072  104433                              TRAP    33
000074  112737  000001  000000G             MOVB    #1,ENTRY.REASON                 ;                           5957
000102  112766  000001  000012              MOVB    #1,12(SP)                       ; *,CLEAR.TABLES            5959
000110  105037  000000G                     CLRB    ADDR.VECT.OK                    ;                           5960
000114  105037  000000G                     CLRB    INIT.OCCURED                    ;                           5961
000120  012700  000037              3$:     MOV     #37,R0                          ;                           5962
000124  104447                              TRAP    47                              ;                           5965
000126  103006                              BHIS    4$
000130  112737  000002  000000G             MOVB    #2,ENTRY.REASON                 ;                           5968
000136  112766  000001  000012              MOVB    #1,12(SP)                       ; *,CLEAR.TABLES            5969
000144  012700  000036              4$:     MOV     #36,R0                          ;                           5972
000150  104447                              TRAP    47
000152  103014                              BHIS    6$
000154  112737  000003  000000G             MOVB    #3,ENTRY.REASON                 ;                           5975
000162  105737  000000G                     TSTB    SWP.FLAGS                       ;                           5977
000166  100403                              BMI     5$
000170  105066  000012                      CLRB    12(SP)                          ; CLEAR.TABLES              5979
000174  000403                              BR      6$                              ;                           5977
000176  112766  000001  000012      5$:     MOVB    #1,12(SP)                       ; *,CLEAR.TABLES            5981
000204  012700  000034              6$:     MOV     #34,R0                          ;                           5985
000210  104447                              TRAP    47
000212  103043                              BHIS    12$
000214  112737  000004  000000G             MOVB    #4,ENTRY.REASON                 ;                           5988
000222  105037  000000G                     CLRB    ADDR.VECT.OK                    ;                           5989
000226  105037  000000G                     CLRB    INIT.OCCURED                    ;                           5990
000232  112766  000001  000012              MOVB    #1,12(SP)                       ; *,CLEAR.TABLES            5991
000240  012746  000000G                     MOV     #MSG.01,-(SP)                   ;                           5992
000244  012746  000001                      MOV     #1,-(SP)
000250  010600                              MOV     SP,R0                           ; SP,*
000252  104417                              TRAP    17
000254  012702  000075                      MOV     #75,R2                          ; *,COUNT                   5994
000260  012703  000515              7$:     MOV     #515,R3                         ; *,DELAY.MULT              5996
000264  010301                              MOV     R3,R1                           ; DELAY.MULT,$$TMP2         5997
000266  001411                      8$:     BEQ     11$
000270  013700  000000G                     MOV     L$DLY,R0                        ; *,$$TMP1
```

```
000274  001404                              BEQ     10$
000276  005066  000024              9$:     CLR     24(SP)                          ; $$TMP
000302  005300                              DEC     R0                              ; $$TMP1
000304  001374                              BNE     9$
000306  005301                      10$:    DEC     R1                              ; $$TMP2
000310  000766                              BR      8$
000312  104422                      11$:    TRAP    22
000314  005302                              DEC     R2                              ; COUNT                         5994
000316  001360                              BNE     7$
000320  022626                              CMP     (SP)+,(SP)+                     ;                               5987
000322  023727  000000G 000004      12$:    CMP     L$UNIT,#4                       ;                               6011
000330  101405                              BLOS    13$
000332  104454                              TRAP    54                              ;                               6014
000334  000001                              .WORD   1
000336  000000G                             .WORD   EGS.01
000340  000000V                             .WORD   EMS.01
000342  104444                              TRAP    44
000344  123727  000000G 000005      13$:    CMPB    ENTRY.REASON,#5                 ;                               6024
000352  001002                              BNE     14$
000354  000137  003462'                     JMP     43$
000360  112766  000377  000010      14$:    MOVB    #377,10(SP)                     ; *.SMALLEST.DRIVE              6027
000366  005000                              CLR     R0                              ; COUNT                         6029
000370  005060  000000G             15$:    CLR     CST(R0)                         ; *(COUNT)                      6030
000374  062700  000002                      ADD     #2,R0                           ; *.COUNT                       6029
000400  020027  000124                      CMP     R0,#124                         ; COUNT,*
000404  003771                              BLE     15$
000406  013704  000000G                     MOV     L$UNIT,R4                       ;                               6032
000412  005003                              CLR     R3                              ; UNIT
000414  000435                              BR      18$
000416  010302                      16$:    MOV     R3,R2                           ; UNIT,*                        6034
000420  006302                              ASL     R2
000422  012700  000022                      MOV     #22,R0
000426  060600                              ADD     SP,R0                           ; HWPT.ADDRESS,*
000430  060002                              ADD     R0,R2
000432  010300                              MOV     R3,R0                           ; UNIT,*
000434  104442                              TRAP    42
000436  010001                              MOV     R0,R1                           ; *,HWPT.REF
000440  010112                              MOV     R1,(R2)                         ; HWPT.REF,*
000442  001421                              BEQ     17$
000444  005002                              CLR     R2                              ;                               6037
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),*                 6039
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$:    INC     R3                              ; UNIT                          6032
000510  020304                      18$:    CMP     R3,R4                           ; UNIT,*
000512  002741                              BLT     16$
000514  013766  000000G 000016              MOV     L$UNIT,16(SP)                   ;                               6041
```

```
000522  005004                            CLR    R4                     ; UNIT
000524  000137  003440'                   JMP    41$
000530  010400                   19$:     MOV    R4,R0                  ; UNIT,*                 6043
000532  006300                            ASL    R0
000534  012703  000022                    MOV    #22,R3
000540  060603                            ADD    SP,R3                  ; HWPT.ADDRESS,*
000542  060300                            ADD    R3,R0
000544  005710                            TST    (R0)
000546  001002                            BNE    20$
000550  000137  003436'                   JMP    40$
000554  105066  000006           20$:     CLRB   6(SP)                  ; FLAG                   6046
000560  011001                            MOV    (R0),R1                ; *,HWPT.REF             6047
000562  005066  000002                    CLR    2(SP)                  ; CTLR                   6049
000566  016646  000002           21$:     MOV    2(SP),-(SP)            ; CTLR,*                 6051
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  002660'                   JMP    28$
000616  012766  000001  000014   22$:     MOV    #1,14(SP)              ;                        6088
000624  112766  000001  000006            MOVB   #1,6(SP)               ; *,FLAG
000632  012705  000006                    MOV    #6,R5                  ;                        6054
000636  060105                            ADD    R1,R5                  ; HWPT.REF,*
000640  111546                            MOVB   (R5),-(SP)
000642  042716  177760                    BIC    #177760,(SP)
000646  005000                            CLR    R0
000650  156600  000012                    BISB   12(SP),R0              ; SMALLEST.DRIVE,*
000654  160016                            SUB    R0,(SP)
000656  012746  000012                    MOV    #12,-(SP)
000662  004737  000000G                   JSR    PC,BL$MUL
000666  010066  000010                    MOV    R0,10(SP)
000672  005726                            TST    (SP)+
000674  016616  000004                    MOV    4(SP),(SP)             ; CTLR,*                 6055
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                 6058
000736  062702  000003                    ADD    #3,R2                  ; *,TEMP
000742  010300                            MOV    R3,R0                  ;                        6059
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,*                 6061
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)            ;                       6062
001000 052710 040000                   BIS     #40000,(R0)            ;                       6063
001004 105715                          TSTB    (R5)                   ;                       6065
001006 100003                          BPL     23$
001010 012761 177777 000016            MOV     #-1,16(R1)             ; *,*(HWPT.REF)         6066
001016 010300                   23$:   MOV     R3,R0                  ;                       6068
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                  ;                       6070
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                  ;                       6072
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                  ;                       6074
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                  ;                       6077
001100 060200                          ADD     R2,R0                  ; TEMP,*
001102 006300                          ASL     R0
001104 011660 000012G                  MOV     (SP),CST+12(R0)        ; BLANKS,*
001110 010300                          MOV     R3,R0                  ;                       6078
001112 060200                          ADD     R2,R0                  ; TEMP,*
001114 006300                          ASL     R0
001116 011660 000014G                  MOV     (SP),CST+14(R0)        ; BLANKS,*
001122 010300                          MOV     R3,R0                  ;                       6081
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)             ;                       6083
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$:   TST     (SP)+
001176 111516                          MOVB    (R5),(SP)              ;                       6085
001200 042710 010000                   BIC     #10000,(R0)
001204 032726 000100                   BIT     #100,(SP)+
001210 001402                          BEQ     26$
001212 052710 010000                   BIS     #10000,(R0)
001216 010300                   26$:   MOV     R3,R0                  ;                       6087
```

```
001220  060200                          ADD     R2,R0                         ; TEMP,*
001222  006300                          ASL     R0
001224  005060  000022G                 CLR     CST+22(R0)
001230  000430                          BR      29$                           ;                          6057
001232  011146                   27$:   MOV     (R1),-(SP)                    ; HWPT.REF,*               6093
001234  111546                          MOVB    (R5),-(SP)
001236  042716  177760                  BIC     #177760,(SP)
001242  012746  000000G                 MOV     #CER.01,-(SP)
001246  012746  000003                  MOV     #3,-(SP)
001252  010600                          MOV     SP,R0                         ; SP,*
001254  104417                          TRAP    17
001256  062706  000010                  ADD     #10,SP
001262  112764  000001  000000G         MOVB    #1,DUR(R4)                    ; *,*(UNIT)                6095
001270  010400                          MOV     R4,R0                         ; UNIT,*                   6096
001272  104451                          TRAP    51
001274  000406                          BR      29$                           ;                          6092
001276  005266  000002           28$:   INC     2(SP)                         ; CTLR                     6049
001302  000243                          .WORD   CLV!CLC

001304  003002                          BGT     29$
001306  000137  002150'                 JMP     21$
001312  105766  000006           29$:   TSTB    6(SP)                         ; FLAG                     6101
001316  001402                          BEQ     30$
001320  000137  003436'                 JMP     40$
001324  005066  000014           30$:   CLR     14(SP)                        ; CTLR                     6105
001330  016646  000014           31$:   MOV     14(SP),-(SP)                  ; CTLR,*                   6107
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  003356'                 JMP     37$
001360  011160  000000G          32$:   MOV     (R1),CST(R0)                  ; HWPT.REF,*               6110
001364  016103  000002                  MOV     2(R1),R3                      ; *(HWPT.REF),*            6111
001370  042703  177000                  BIC     #177000,R3
001374  042760  000777  000002G         BIC     #777,CST+2(R0)
001402  050360  000002G                 BIS     R3,CST+2(R0)
001406  116160  000004  000004G         MOVB    4(R1),CST+4(R0)               ; *(HWPT.REF),*            6112
001414  012705  000006                  MOV     #6,R5                         ;                          6113
001420  060105                          ADD     R1,R5                         ; HWPT.REF,*
001422  111546                          MOVB    (R5),-(SP)
001424  042716  177760                  BIC     #177760,(SP)
001430  005000                          CLR     R0
001432  156600  000012                  BISB    12(SP),R0                     ; SMALLEST.DRIVE,*
001436  160016                          SUB     R0,(SP)
001440  012746  000012                  MOV     #12,-(SP)
001444  004737  000000G                 JSR     PC,BL$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,*                   6114
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,*             6116
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)            ;                    6117
001534  052710  040000                   BIS   #40000,(R0)            ;                    6118
001540  105715                           TSTB  (R5)                   ;                    6120
001542  100003                           BPL   33$
001544  012761  177777  000016           MOV   #-1,16(R1)             ; *,*(HWPT.REF)      6121
001552  010300                   33$:    MOV   R3,R0                  ;                    6123
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                  ;                    6125
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                  ;                    6127
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                  ;                    6129
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                  ;                    6132
001634  060200                           ADD   R2,R0                  ; TEMP,*
001636  006300                           ASL   R0
001640  011660  000012G                  MOV   (SP),CST+12(R0)        ; BLANKS,*
001644  010300                           MOV   R3,R0                  ;                    6133
001646  060200                           ADD   R2,R0                  ; TEMP,*
001650  006300                           ASL   R0
001652  011660  000014G                  MOV   (SP),CST+14(R0)        ; BLANKS,*
001656  010300                           MOV   R3,R0                  ;                    6136
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)             ;                    6138
001674  005046                           CLR   -(SP)
001676  032766  000040  000002           BIT   #40,2(SP)
001704  001401                           BEQ   34$
001706  005216                           INC   (SP)
001710  005116                   34$:    COM   (SP)
001712  011646                           MOV   (SP),-(SP)
001714  042710  100000                   BIC   #100000,(R0)
```

```
001720  006026                          ROR     (SP)+
001722  103002                          BCC     35$
001724  052710  100000                  BIS     #100000,(R0)
001730  005726                  35$:    TST     (SP)+
001732  111516                          MOVB    (R5),(SP)                ;                      6139
001734  042710  010000                  BIC     #10000,(R0)
001740  032726  000100                  BIT     #100,(SP)+
001744  001402                          BEQ     36$
001746  052710  010000                  BIS     #10000,(R0)
001752  010300                  36$:    MOV     R3,R0                    ;                      6141
001754  060200                          ADD     R2,R0                    ; TEMP,*
001756  006300                          ASL     R0
001760  005060  000022G                 CLR     CST+22(R0)
001764  112766  000001  000006          MOVB    #1,6(SP)                 ; *,FLAG               6142
001772  000410                          BR      39$                      ;                      6109
001774  005266  000014          37$:    INC     14(SP)                   ; CTLR                 6105
002000  000243                          .WORD CLV!CLC

002002  003002                          BGT     38$
002004  000137  002712'                 JMP     31$
002010  105766  000006          38$:    TSTB    6(SP)                    ; FLAG                 6146
002014  001017                  39$:    BNE     40$
002016  012746  000001                  MOV     #1,-(SP)                 ;                      6149
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)            6150
002044  010400                          MOV     R4,R0                    ; UNIT,*               6151
002046  104451                          TRAP    51
002050  062706  000006                  ADD     #6,SP                    ;                      6148
002054  005204                  40$:    INC     R4                       ; UNIT                 6041
002056  020466  000016          41$:    CMP     R4,16(SP)                ; UNIT,*
002062  002002                          BGE     42$
002064  000137  002112'                 JMP     19$
002070  123727  000000G 000005  42$:    CMPB    ENTRY.REASON,#5          ;                      6164
002076  001051                          BNE     48$
002100  013703  000000G         43$:    MOV     L$UNIT,R3                ;                      6168
002104  005004                          CLR     R4                       ; UNIT
002106  000404                          BR      45$
002110  010400                  44$:    MOV     R4,R0                    ; UNIT,*               6169
002112  104442                          TRAP    42
002114  010001                          MOV     R0,R1                    ; *,HWPT.REF
002116  005204                          INC     R4                       ; UNIT                 6168
002120  020403                  45$:    CMP     R4,R3                    ; UNIT,*
002122  002772                          BLT     44$
002124  005003                          CLR     R3                       ; CTLR                 6171
002126  010346                  46$:    MOV     R3,-(SP)                 ; CTLR,*               6173
002130  012746  000126                  MOV     #126,-(SP)
002134  004737  000000G                 JSR     PC,BL$MUL
002140  105060  000005G                 CLRB    CST+5(R0)
002144  010316                          MOV     R3,(SP)                  ; CTLR,*               6176
002146  012746  000053                  MOV     #53,-(SP)
```

```
002152  004737  000000G                     JSR     PC,BL$MUL
002156  012701  000003                      MOV     #3,R1                   ; *,OFFSET                       6175
002162  010002                      47$:    MOV     R0,R2                   ;                                6176
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                       6175
002202  020127  000041                      CMP     R1,#41                  ; OFFSET,*
002206  003765                              BLE     47$
002210  062706  000006                      ADD     #6,SP                   ;                                6172
002214  005203                              INC     R3                      ; CTLR                           6171
002216  000243                              .WORD   CLV!CLC

002220  003742                              BLE     46$
002222  123727  000000G 000001      48$:    CMPB    ENTRY.REASON,#1         ;                                6182
002230  001017                              BNE     51$
002232  005037  000000G                     CLR     CTLR.CNT                ;                                6185
002236  005000                              CLR     R0                      ; CTLR                           6187
002240  005760  C00000G             49$:    TST     CST(R0)                 ; *(CTLR)                        6189
002244  001402                              BEQ     50$
002246  005237  000000G                     INC     CTLR.CNT                ;                                6191
002252  062700  000126              50$:    ADD     #126,R0                 ; *,CTLR                         6187
002256  000243                              .WORD   CLV!CLC

002260  003767                              BLE     49$
002262  104431                              TRAP    31                      ;                                6193
002264  010037  000000G                     MOV     R0,FREE.MEM.ADDR
002270  005001                      51$:    CLR     R1                      ; UNITS                          6201
002272  005003                      52$:    CLR     R3                      ; COUNT                          6202
002274  010300                      53$:    MOV     R3,R0                   ; COUNT,*                        6203
002276  060100                              ADD     R1,R0                   ; UNITS,*
002300  006300                              ASL     R0
002302  005060  000000G                     CLR     TALLY(R0)
002306  005203                              INC     R3                      ; COUNT                          6202
002310  020327  000006                      CMP     R3,#6                   ; COUNT,*
002314  003767                              BLE     53$
002316  062701  000033                      ADD     #33,R1                  ; *,UNITS                        6201
002322  020127  000121                      CMP     R1,#121                 ; UNITS,*
002326  003761                              BLE     52$
002330  032766  000001  000012              BIT     #1,12(SP)               ; *,CLEAR.TABLES                 6205
002336  001436                              BEQ     57$
002340  005001                              CLR     R1                      ; UNITS                          6207
002342  012703  000007              54$:    MOV     #7,R3                   ; *,COUNT                        6208
002346  010300                      55$:    MOV     R3,R0                   ; COUNT,*                        6209
002350  060100                              ADD     R1,R0                   ; UNITS,*
002352  006300                              ASL     R0
002354  005060  000000G                     CLR     TALLY(R0)
002360  005203                              INC     R3                      ; COUNT                          6208
002362  020327  000032                      CMP     R3,#32                  ; COUNT,*
002366  003767                              BLE     55$
002370  062701  000033                      ADD     #33,R1                  ; *,UNITS                        6207
002374  020127  000121                      CMP     R1,#121                 ; UNITS,*
002400  003760                              BLE     54$
```

```
002402  032766  000001  000012                  BIT     #1,12(SP)               ; *.CLEAR.TABLES        6211
002410  001411                                  BEQ     57$
002412  005000                                  CLR     R0                      ; CTLR                  6213
002414  105060  000000G                 56$:    CLRB    C.ERR.TBL(R0)           ; *(CTLR)               6215
002420  105060  000001G                         CLRB    C.ERR.TBL+1(R0)         ; *(CTLR)               6216
002424  062700  000002                          ADD     #2,R0                   ; *.CTLR                6213
002430  000243                                  .WORD CLV!CLC

002432  003770                                  BLE     56$
002434  005000                          57$:    CLR     R0                      ; CTLR                  6223
002436  105060  000000G                 58$:    CLRB    QIO(R0)                 ; *(CTLR)               6224
002442  005200                                  INC     R0                      ; CTLR                  6223
002444  000243                                  .WORD CLV!CLC

002446  003773                                  BLE     58$
002450  005000                                  CLR     R0                      ; COUNT                 6226
002452  112760  000377  000000G         59$:    MOVB    #377,RP.USE(R0)         ; *.*(COUNT)            6227
002460  005200                                  INC     R0                      ; COUNT                 6226
002462  020027  000007                          CMP     R0,#7                   ; COUNT,*
002466  003771                                  BLE     59$
002470  132737  000001  000000G                 BITB    #1,CLK.PRESENT          ;                       6229
002476  001402                                  BEQ     60$
002500  005037  177546                          CLR     @#177546                ;                       6231
002504  005037  000000G                 60$:    CLR     IODQ.OUT                ;                       6233
002510  005037  000000G                         CLR     IODQ.IN
002514  005037  000000G                         CLR     CRN.HIGH                ;                       6234
002520  005037  000000G                         CLR     CRN.LOW
002524  005000                                  CLR     R0                      ;                       6235
002526  104441                                  TRAP    41
002530  062706  000032                          ADD     #32,SP                  ;                       5924
002534  000207                                  RTS     PC

; Routine Size:  687 words,     Routine Base:  $CODE$ + 1362
; Maximum stack depth per invocation:  25 words



                                                .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  001362'                 L$INIT::JSR     PC,LINIT                ;                       6235
000004  104411                                  TRAP    11
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4120
; Maximum stack depth per invocation:  2 words
```

```
;  1      %sbttl 'AUTODROP SECTION'
;  1
;  1      !+
;  1      ! THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;  1      ! THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
;  1      ! SFE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
;  1      ! DROPPED FROM TESTING.
;  1      !-
;  1
;  2      BGNAUTO;
;  2
;  2      !if BIT_TST (SWP_FLAGS, SWF_TRC)
;  2      !then
;  2      !   PRINTF (DBM3);
;  2
;  2      return;
;  2
;  1      ENDAUTO;


                                         .SBTTL  LAUTO AUTODROP SECTION
000000  000207                   LAUTO:  RTS     PC                                  ;     6238

; Routine Size:  1 word,       Routine Base:  $CODE$ + 4130
; Maximum stack depth per invocation:  0 words



                                         .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  004130'          L$AUTO::JSR     PC,LAUTO                            ;     6254
000004  104461                           TRAP    61
000006  000207                           RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 4132
; Maximum stack depth per invocation:  2 words
```

```
%sbttl 'CLEANUP CODING SECTION'

!+
! THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
! AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
!-

BGNCLN;

LABEL                                                                          !ZZZ
    LZ1;                                                                       !ZZZ

DORPT;

!CLRVEC (O_TVEC);                                   ! RETURN ODT TRAP TO DIAGNOSTIC SUPERVISER

if .CLK_PRESENT
then
    begin
    LINE_CLOCK = 0;                                 ! STOP THE LINE-CLOCK
!   CLRVEC (%o'100');                               ! RETURN LINE-CLOCK'S VECTOR TO SUPERVISOR
    end;

incr CTLR from 0 to (MAX_CTLR - 1) do               ! FOR EACH CONTROLLER

    if (RDRX_ADDR = .CST [.CTLR, IP_ADDR]) neqa 0   ! IF CONTROLLER EXISTS
    then
        begin

        if .ADDR_VECT_OK
        then
LZ1:        begin                                                                  !ZZZ

            if .DCT [.CTLR, STAT] eql ONLINE        ! IF CONTROLLER ALIVE
            then

!MMM                    incr COUNT from 1 to 10000 do
                        incr COUNT from 1 to 200 do                                !MMM
                    begin
                    DELAY (1);
                    BREAK;                                                         !MMM
                    if .DCT [.CTLR, CRING_CNT] eql 0    ! WAIT TILL OUTSTANDING COMMANDS FINISHED
                    then

                        INCR Z  FROM 0 TO 3  DO                                    !ZZZ
                           BEGIN                                                   !ZZZ
                           TEMP1 =  (.DCT [.CTLR, RR_BEG]) + 4 * .Z;      !DESCRIPTOR ADDRESS      !ZZZ
                           TEMP2 = ..TEMP1;                               !PACKET ADDRESS          !ZZZ
                           IF ..TEMP2  EQL CRN_LOW                        !CRN                     !ZZZ
                           THEN                                           !IF THE LAST CRN IS BACK, !ZZZ
                              (WRT_RDRX (RCIP, RC_ALL, ALL_ONES); LEAVE LZ1); !THEN STOP WAITING   !ZZZ
                           END;                                                    !ZZZ
                    end;
```

```
;    6310  4
;    6311  4                 WRT_RDRX (RCIP, RC_ALL, ALL_ONES);        ! WRITE IP TO STOP DEVICE
;    6312  3                 end;
;    6313  3
;    6314  3             CLRVEC (.CST[.CTLR, VEC_ADDR]);               ! RETURN CONTROLLER'S TRAP VECTOR TO SUPERVISOR
;    6315  2             end;
;    6316  2
;    6317  1      ENDCLN;

                                                .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  004137  000000G             LCLEAN: JSR     R1,$SAVE5                       ;                       6256
000004  005746                              TST     -(SP)
000006  104424                              TRAP    24                              ;                       6267
000010  132737  000001  000000G             BITB    #1,CLK.PRESENT                  ;                       6273
000016  001402                              BEQ     1$
000020  005037  177546                      CLR     @#177546                        ;                       6276
000024  005005                      1$:     CLR     R5                              ; CTLR                  6280
000026  010546                      2$:     MOV     R5,-(SP)                        ; CTLR,*                6282
000030  012746  000126                      MOV     #126,-(SP)
000034  004737  000000G                     JSR     PC,BL$MUL
000040  010003                              MOV     R0,R3
000042  022626                              CMP     (SP)+,(SP)+
000044  016337  000000G 000000G             MOV     CST(R3),RDRX.ADDR
000052  001501                              BEQ     13$
000054  132737  000001  000000G             BITB    #1,ADDR.VECT.OK                 ;                       6286
000062  001470                              BEQ     12$
000064  010546                              MOV     R5,-(SP)                        ; CTLR,*                6290
000066  012746  000022                      MOV     #22,-(SP)
000072  004737  000000G                     JSR     PC,BL$MUL
000076  010001                              MOV     R0,R1
000100  022626                              CMP     (SP)+,(SP)+
000102  005761  000000G                     TST     DCT(R1)
000106  100052                              BPL     11$
000110  012704  000310                      MOV     #310,R4                         ; *,COUNT               6294
000114  012702  000001              3$:     MOV     #1,R2                           ; *,$$TMP2              6296
000120  001410                      4$:     BEQ     7$
000122  013700  000000G                     MOV     L$DLY,R0                        ; *,$$TMP1
000126  001403                              BEQ     6$
000130  005016                      5$:     CLR     (SP)                            ; $$TMP
000132  005300                              DEC     R0                              ; $$TMP1
000134  001375                              BNE     5$
000136  005302                      6$:     DEC     R2                              ; $$TMP2
000140  000767                              BR      4$
000142  104422                      7$:     TRAP    22
000144  105761  000000G                     TSTB    DCT(R1)                         ;                       6298
000150  001027                              BNE     10$
000152  005000                              CLR     R0                              ; Z                     6301
000154  016137  000004G 000000G     8$:     MOV     DCT+4(R1),TEMP1                 ;                       6303
000162  060037  000000G                     ADD     R0,TEMP1                        ; Z,*
000166  017737  000000G 000000G             MOV     @TEMP1,TEMP2                    ;                       6304
000174  027727  000000G 000000G             CMP     @TEMP2,#CRN.LOW                 ;                       6305
000202  001005                              BNE     9$
```

```
000204  012702  177777                MOV     #-1,R2               ; *,RC.REG          6307
000210  010277  000000G               MOV     R2,@RDRX.ADDR        ; RC.REG,*
000214  000413                        BR      12$
000216  062700  000004       9$:      ADD     #4,R0                ; *,Z               6301
000222  020027  000014                CMP     R0,#14               ; Z,*
000226  003752                        BLE     8$
000230  005304               10$:     DEC     R4                   ; COUNT             6294
000232  001330                        BNE     3$
000234  012700  177777       11$:     MOV     #-1,R0               ; *,RC.REG          6311
000240  010077  000000G               MOV     R0,@RDRX.ADDR        ; RC.REG,*
000244  016300  000002G      12$:     MOV     CST+2(R3),R0         ;                   6314
000250  042700  177000                BIC     #177000,R0
000254  104436                        TRAP    36
000256  005205               13$:     INC     R5                   ; CTLR              6280
000260  000243                        .WORD CLV!CLC

000262  003661                        BLE     2$
000264  005726                        TST     (SP)+                ;                   6256
000266  000207                        RTS     PC

; Routine Size:  92 words,      Routine Base:  $CODE$ + 4142
; Maximum stack depth per invocation:  10 words



                                      .SBTTL  L$CLEAN CLEANUP CODING SECTION
000000  004737  004142'      L$CLEAN::
                                      JSR     PC,LCLEAN            ;                   6315
000004  104412                        TRAP    12
000006  000207                        RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4432
; Maximum stack depth per invocation:  2 words
```

```
%sbttl 'DROP UNIT SECTION'

!+
! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
! TO NO LONGER BE TESTED.
!-

BGNDU;

local
    UNIT : word,                                                          ! UNIT NUMBER
    PRINT : byte initial (byte (FALSE));                                  ! NO PRINTING

label
    SEARCH;

begin

register
    INPUT = 0;                                                            ! R0 = UNIT NO.

UNIT = .INPUT;                                                            ! GET UNIT NUMBER
end;                                                                      ! UNDECLARE R0

!ZZZif BIT_TST (SWP_FLAGS, SWF_TRC)
!ZZZthen
!ZZZ    PRINTF (DBM5, .UNIT);

SEARCH :                                                                  ! SEARCH BLOCK
begin

incr CTLR from 0 to (MAX_CTLR - 1) do                                     ! FOR EACH CNTR

    incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do   ! FOR EACH UNIT

        if (.CST [.CTLR, .OFFSET + OF_DATA, D_UNIT] eql .UNIT) and        ! IF UNIT MATCHES
           (.CST [.CTLR, .OFFSET + OF_DATA, D_PRES] eql PRESENT)
        then
            begin

            if (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE) or    ! IF UNIT ALIVE
               (.DUR [.UNIT] eql DU_ONLINE) or
               (.DUR [.UNIT] eql DU_PROTECT)
            then
                begin
                PRINT = TRUE;                                             ! O.K. TO PRINT

                if (.CST [.CTLR, U_CNT] gtru 0) and
                   (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
                then
                    CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] - 1;         ! DECREMENT COUNT

                if (.CST [.CTLR, U_CNT] eql 0) and
```

```
;    6371  6                  (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;    6372  5               then
;    6373  5                   EOP_FLAG = TRUE;                                        ! ALL UNITS OFFLINE
;    6374  5
;    6375  5               CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;           ! MARK UNIT OFFLINE
;    6376  4               end;                                                        ! IF UNIT ALIVE
;    6377  4
;    6378  4           leave SEARCH;                                                   ! EXIT SEARCH BLOCK
;    6379  3           end;                                                            ! IF UNIT FOUND
;    6380  3
;    6381  2   end;
;    6382  2
;    6383  2   if .PRINT or                                                            ! IF OK TO PRINT
;    6384  2      (.DUR [.UNIT] eql DU_CONF) or
;    6385  2      (.DUR [.UNIT] eql DU_INIT) or
;    6386  2      (.DUR [.UNIT] eql DU_ONLINE) or
;    6387  3      (.DUR [.UNIT] eql DU_PROTECT)
;    6388  2   then
;    6389  3       begin
;    6390  3       PRINTF (DU_MSG, .UNIT);                                             ! "UNIT XX DROPPED"
;    6391  3       PRINTF (.DU_RSN [.DUR [.UNIT]]);                                    ! REASON
;    6392  2       end;
;    6393  2
;    6394  1   ENDDU;


                                              .SBTTL  LDU DROP UNIT SECTION
000000  004137  000000G           LDU:        JSR     R1,$SAVE5               ;                        6317
000004  024646                                CMP     -(SP),-(SP)
000006  105066  000002                        CLRB    2(SP)                   ; PRINT
000012  010001                                MOV     R0,R1                   ; INPUT,UNIT             6339
000014  005005                                CLR     R5                      ; CTLR                   6349
000016  010546                    1$:         MOV     R5,-(SP)                ; CTLR,*                 6353
000020  012746  000053                        MOV     #53,-(SP)
000024  004737  000000G                       JSR     PC,BL$MUL
000030  010066  000004                        MOV     R0,4(SP)
000034  012703  000003                        MOV     #3,R3                   ; *,OFFSET               6351
000040  010300                    2$:         MOV     R3,R0                   ; OFFSET,*               6353
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)             ;                        6354
000100  001452                                BEQ     8$
000102  005004                                CLR     R4                      ;                        6358
000104  032712  020000                        BIT     #20000,(R2)
000110  001402                                BEQ     3$
```

```
000112  005204                                  INC     R4
000114  000410                                  BR      4$
000116  126127  000000G 000007         3$:      CMPB    DUR(R1),#7                     ; *(UNIT),*               6359
000124  001404                                  BEQ     4$
000126  126127  000000G 000011                  CMPB    DUR(R1),#11                    ; *(UNIT),*               6360
000134  001032                                  BNE     7$
000136  112766  000001  000006         4$:      MOVB    #1,6(SP)                       ; *,PRINT                 6363
000144  010516                                  MOV     R5,(SP)                        ; CTLR,*                  6365
000146  012746  000126                          MOV     #126,-(SP)
000152  004737  000000G                         JSR     PC,BL$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                             ;                         6366
000174  105660  000001                          SBCB    1(R0)                          ;                         6368
000200  001006                                  BNE     6$                             ;                         6370
000202  032712  020000                 5$:      BIT     #20000,(R2)                    ;                         6371
000206  001403                                  BEQ     6$
000210  112737  000001  000000G                 MOVB    #1,EOP.FLAG                    ;                         6373
000216  042712  020000                 6$:      BIC     #20000,(R2)                    ;                         6375
000222  022626                         7$:      CMP     (SP)+,(SP)+                    ;                         6356
000224  000411                                  BR      9$
000226  062703  000012                 8$:      ADD     #12,R3                         ; *,OFFSET                6351
000232  020327  000041                          CMP     R3,#41                         ; OFFSET,*
000236  003700                                  BLE     2$
000240  022626                                  CMP     (SP)+,(SP)+
000242  005205                                  INC     R5                             ; CTLR                    6349
000244  000243                                  .WORD CLV!CLC

000246  003663                                  BLE     1$
000250  032766  000001  000002         9$:      BIT     #1,2(SP)                       ; *,PRINT                 6383
000256  001020                                  BNE     10$
000260  126127  000000G 000001                  CMPB    DUR(R1),#1                     ; *(UNIT),*               6384
000266  001414                                  BEQ     10$
000270  126127  000000G 000002                  CMPB    DUR(R1),#2                     ; *(UNIT),*               6385
000276  001410                                  BEQ     10$
000300  126127  000000G 000007                  CMPB    DUR(R1),#7                     ; *(UNIT),*               6386
000306  001404                                  BEQ     10$
000310  126127  000000G 000011                  CMPB    DUR(R1),#11                    ; *(UNIT),*               6387
000316  001024                                  BNE     11$
000320  010146                         10$:     MOV     R1,-(SP)                       ; UNIT,*                  6390
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),*               6391
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                                  ;        6389
000370  022626                    11$:    CMP     (SP)+,(SP)+                             ;        6317
000372  000207                            RTS     PC

; Routine Size:  126 words,     Routine Base:  $CODE$ + 4442
; Maximum stack depth per invocation:  14 words


                                          .SBTTL  L$DU DROP UNIT SECTION
000000  004737  004442'           L$DU::  JSR     PC,LDU                                  ;        6392
000004  104453                            TRAP    53
000006  000207                            RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5036
; Maximum stack depth per invocation:  2 words
```

```
;   6395  1     %sbttl 'ADD UNIT SECTION'
;   6396  1
;   6397  1     !+
;   6398  1     ! THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;   6399  1     !  TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;   6400  1     !  TO THE TEST CYCLE.
;   6401  1     !-
;   6402  1
;   6403  2     BGNAU;
;   6404  2
;   6405  2     local
;   6406  2         STINDX : word,
;   6407  2         ENDIDX : word;
;   6408  2
;   6409  2     register
;   6410  2         UNIT = 0;                                  ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
;   6411  2
;   6412  3     if BIT_TST (SWP_FLAGS, SWF_CST)
;   6413  2     then
;   6414  3         begin                                      ! IF CLEAR STAT. TABLES TRUE....
;   6415  3         STINDX = .UNIT * TALLY_LEN;                ! ZERO OUT
;   6416  3         ENDIDX = .STINDX + TALLY_LEN - 1;          !    ADDED
;   6417  3
;   6418  3         incr COUNT from .STINDX to .ENDIDX do      !    UNIT'S
;   6419  3             TALLY [.COUNT] = 0;                    !    STATISTICS
;   6420  3
;   6421  2         end;
;   6422  2
;   6423  1     ENDAU;


                                            .SBTTL  LAU ADD UNIT SECTION
000000  004137  000000G             LAU:    JSR     R1,$SAVE2                                     6394
000004  105737  000000G                     TSTB    SWP.FLAGS               ;                     6412
000010  100023                              BPL     3$                      ;
000012  010046                              MOV     R0,-(SP)                ; UNIT,*              6415
000014  012746  000033                      MOV     #33,-(SP)
000020  004737  000000G                     JSR     PC,BL$MUL
000024  010002                              MOV     R0,R2                   ; STINDX,ENDIDX       6416
000026  062702  000032                      ADD     #32,R2                  ; *,ENDIDX
000032  010001                              MOV     R0,R1                   ; STINDX,COUNT        6418
000034  005301                              DEC     R1                      ; COUNT
000036  000404                              BR      2$
000040  010100                      1$:     MOV     R1,R0                   ; COUNT,*             6419
000042  006300                              ASL     R0
000044  005060  000000G                     CLR     TALLY(R0)
000050  005201                      2$:     INC     R1                      ; COUNT               6418
000052  020102                              CMP     R1,R2                   ; COUNT,ENDIDX
000054  003771                              BLE     1$
000056  022626                              CMP     (SP)+,(SP)+             ;                     6414
000060  000207                      3$:     RTS     PC                      ;                     6394

; Routine Size:  25 words,      Routine Base:  $CODE$ + 5046
```

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


                                                  .SBTTL  L$AU ADD UNIT SECTION
000000  004737  005046'                  L$AU::   JSR     PC.LAU                      ;                        6421
000004  104452                                    TRAP    52
000006  000207                                    RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5130
; Maximum stack depth per invocation:  2 words
```

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


                                        .SBTTL  NEX.TRAP NON-EXISTENT MEMORY TRAP HANDLER
000000  012737  000001  000000G     NEX.TRAP::
                                        MOV     #1,NEX                          ;                       6435
000006  000002                          RTI                                     ;                       6433

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5140
; Maximum stack depth per invocation:  0 words
```

```
;     6438  1        %sbttl 'HOST MEMORY PARITY TRAP HANDLER'                         !MMM
;     6439  1
;     6440  1        !+
;     6441  1        !       THIS TRAP HANDLER IS VECTORED FROM LOCATION 114 FOR ALL HOST MEMORY PARITY
;     6442  1        !       ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO READ A MEMORY LOCATION WITH
;     6443  1        !       PARITY/UNCORRECTABLE ECC ERROR DETECTED BY THE MEMORY'S PARITY OR ECC LOGIC.
;     6444  1        !-
;     6445  1
;     6446  2        BGNSRV (PARITY);
;     6447  2
;     6448  2        builtin
;     6449  2             HALT;
;     6450  2
;     6451  2        PRINTF (DBM125, ..CSR_ADD);
;     6452  2
;     6453  2        .CSR_ADD = %o'40000';
;     6454  2
;     6455  2        PRINTF (DBM126, ..CSR_ADD);
;     6456  2
;     6457  2        HALT ();
;     6458  2
;     6459  1        ENDSRV;


                                                  .SBTTL  PARITY HOST MEMORY PARITY TRAP HANDLER
000000  010046                          PARITY::MOV     R0,-(SP)                        ;              6446
000002  017746  000000G                         MOV     @CSR.ADD,-(SP)                  ;              6451
000006  012746  000000G                         MOV     #DBM125,-(SP)
000012  012746  000002                          MOV     #2,-(SP)
000016  010600                                  MOV     SP,R0                           ; SP,*
000020  104417                                  TRAP    17
000022  012777  040000  000000G                 MOV     #40000,@CSR.ADD                 ;              6453
000030  012716  040000                          MOV     #40000,(SP)                     ;              6455
000034  012746  000000G                         MOV     #DBM126,-(SP)
000040  012746  000002                          MOV     #2,-(SP)
000044  010600                                  MOV     SP,R0                           ; SP,*
000046  104417                                  TRAP    17
000050  000000                                  HALT                                    ;              6457
000052  062706  000012                          ADD     #12,SP                          ;              6446
000056  012600                                  MOV     (SP)+,R0
000060  000002                                  RTI

; Routine Size:  25 words,      Routine Base:  $CODE$ + 5150
; Maximum stack depth per invocation:  8 words


;    6460  1
```

```
;   6461  1    %sbttl 'TIME OF DAY'
;   6462  1
;   6463  1    !+
;   6464  1    !       THIS INTERRUPT SERVICE ROUTINE KEEPS TRACK OF THE TIME-OF-DAY
;   6465  1    !-
;   6466  1
;   6467  2    BGNSRV (TIME);
;   6468  2
;   6469  2    CLK_TICKS = .CLK_TICKS + 1;                ! INCREMENT CLOCK-TICKS
;   6470  2
;   6471  2    if .CLK_TICKS gequ 3600
;   6472  2    then
;   6473  3        begin
;   6474  3        MINUTES = .MINUTES + 1;                ! UPDATE MINUTE COUNT
;   6475  3        CLK_TICKS = 0;
;   6476  2        end;
;   6477  2
;   6478  2    if .MINUTES gequ 60
;   6479  2    then
;   6480  3        begin
;   6481  3        HOURS = .HOURS + 1;                    ! UPDATE HOUR COUNT
;   6482  3        MINUTES = 0;
;   6483  2        end;
;   6484  2
;   6485  2    if .HOURS gequ 24
;   6486  2    then
;   6487  2        HOURS = 0;                             ! RATIONALIZE HOURS
;   6488  2
;   6489  1    ENDSRV;


                                                   .SBTTL  TIME TIME OF DAY
000000  005237  000000G               TIME::  INC     CLK.TICKS             ;   6469
000004  023727  000000G 007020                CMP     CLK.TICKS,#7020       ;   6471
000012  103404                                BLO     1$
000014  105237  000000G                       INCB    MINUTES               ;   6474
000020  005037  000000G                       CLR     CLK.TICKS             ;   6475
000024  123727  000000G 000074        1$:     CMPB    MINUTES,#74           ;   6478
000032  103404                                BLO     2$
000034  105237  000000G                       INCB    HOURS                 ;   6481
000040  105037  000000G                       CLRB    MINUTES               ;   6482
000044  123727  000000G 000030        2$:     CMPB    HOURS,#30             ;   6485
000052  103402                                BLO     3$
000054  105037  000000G                       CLRB    HOURS                 ;   6487
000060  000002                        3$:     RTI                           ;   6467

; Routine Size:  25 words,    Routine Base:  $CODE$ + 5232
; Maximum stack depth per invocation:  0 words
```

```
;    6490  1      %sbttl 'GLOBAL ROUTINES'
;    6491  1
;    6492  1      global routine SET_CPAR (CTLR) : novalue =
;    6493  1
;    6494  1      !+
;    6495  1      !       THIS ROUTINE SETS UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    6496  1      !       FOR THE GIVEN CONTROLLER NUMBER.
;    6497  1      !
;    6498  1      !       INPUTS:
;    6499  1      !               CTLR - CONTROLLER NUMBER
;    6500  1      !
;    6501  1      !       IMPLICIT OUTPUTS:
;    6502  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    6503  1      !               CST_ADDR - ADDRESS OF CONTROLLER'S STATUS TABLE
;    6504  1      !               DCT_ADDR - ADDRESS OF CONTROLLER'S DRIVER TABLE
;    6505  1      !               RDRX_ADDR - ADDRESS OF CONTROLLER'S IP REGISTER
;    6506  1      !-
;    6507  1
;    6508  2          begin
;    6509  2          CCTLR = .CTLR;                              ! SET CURRENT CONTROLLER NUMBER
;    6510  2          CST_ADDR = CST + (.CTLR * CST_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER'S CST
;    6511  2          DCT_ADDR = DCT + (.CTLR * DCT_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER'S DCT
;    6512  2          RDRX_ADDR = .CST_ADDR [IP_ADDR];            ! GET CONTROLLER'S DEVICE ADDRESS
;    6513  1          end;


                                                 .SBTTL  SET.CPAR GLOBAL ROUTINES
000000  010146                          SET.CPAR::
                                                 MOV     R1,-(SP)                 ;                    6492
000002  016601  000004                           MOV     4(SP),R1                 ; CTLR,*             6509
000006  010137  000000G                          MOV     R1,CCTLR
000012  010146                                   MOV     R1,-(SP)                 ;                    6510
000014  012746  000126                           MOV     #126,-(SP)
000020  004737  000000G                          JSR     PC,BL$MUL
000024  062700  000000G                          ADD     #CST,R0
000030  010037  000000G                          MOV     R0,CST.ADDR
000034  010116                                   MOV     R1,(SP)                  ;                    6511
000036  012746  000022                           MOV     #22,-(SP)
000042  004737  000000G                          JSR     PC,BL$MUL
000046  062700  000000G                          ADD     #DCT,R0
000052  010037  000000G                          MOV     R0,DCT.ADDR
000056  017737  000000G 000000G                  MOV     @CST.ADDR,RDRX.ADDR      ;                    6512
000064  062706  000006                           ADD     #6,SP                    ;                    6508
000070  012601                                   MOV     (SP)+,R1                 ;                    6492
000072  000207                                   RTS     PC

; Routine Size:  30 words,      Routine Base:  $CODE$ + 5314
; Maximum stack depth per invocation:  5 words
```

```
;    6514  1    global routine SET_UPAR (OFFSET) : novalue =
;    6515  1    !+
;    6516  1    !       THIS ROUTINE SETS UP THE COMMONLY-USED UNIT-RELATED DATA ITEMS FOR
;    6517  1    !       THE CURRENT CONTROLLER AND GIVEN CST OFFSET.
;    6518  1    !
;    6519  1    !       INPUTS:
;    6520  1    !               OFFSET - WORD OFFSET INTO CURRENT CONTROLLER'S CST WHICH
;    6521  1    !                               DESCRIBES A UNIT
;    6522  1    !
;    6523  1    !       IMPLICIT INPUTS:
;    6524  1    !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    6525  1    !
;    6526  1    !       IMPLICIT OUTPUTS:
;    6527  1    !               CUOFF - CURRENT UNIT'S CST OFFSET
;    6528  1    !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    6529  1    !               L$LUN - CURRENT UNIT NUMBER (DRS UNIT NUMBER)
;    6530  1    !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    6531  1    !-
;    6532  2        begin
;    6533  2        CUOFF = .OFFSET;
;    6534  2        CDISK = .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM];
;    6535  2        L$LUN = .CST_ADDR [.OFFSET + OF_DATA, D_UNIT];
;    6536  2        T_ADDR = TALLY + (.L$LUN * TALLY_LEN * 2);
;    6537  1        end;


                                                .SBTTL  SET.UPAR GLOBAL ROUTINES
000000  010146                          SET.UPAR::
                                                MOV     R1,-(SP)                ;                       6514
000002  016637  000004  000000G                 MOV     4(SP),CUOFF             ; OFFSET,*              6533
000010  016600  000004                          MOV     4(SP),R0                ; CUOFF,*               6534
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                 ;                       6535
000036  000301                                  SWAB    R1
000040  042701  177760                          BIC     #177760,R1
000044  010137  000000G                         MOV     R1,L$LUN
000050  010146                                  MOV     R1,-(SP)                ; L$LUN,*               6536
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)+             ;                       6532
000074  012601                                  MOV     (SP)+,R1                ;                       6514
000076  000207                                  RTS     PC

; Routine Size:  32 words,      Routine Base:  $CODE$ + 5410
; Maximum stack depth per invocation:  4 words
```

```
global routine GET_PKT (CTLR) =

!+
!       THIS ROUTINE SEARCHES THE MSCP PACKET POOL ALLOCATION TABLE (PKT_USE)
!       FOR A FREE MSCP PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
!       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED
!       TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
!
!       INPUTS:
!               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
!-

    begin

    local
        index : signed word initial (-1),
        RING_ADDR : word,
        PACKET_OWNED : byte,
        NEXT_PACKET : byte;


    NEXT_PACKET = .NEXT_PKT_USE;                        ! NEXT PACKET TO TRY

    incr COUNT from 0 to (PKT_CNT - 1) do               ! FOR EACH ENTRY IN ALLOCATION TABLE
        begin
        PACKET_OWNED = FALSE;

        if .PKT_USE [.NEXT_PACKET] lss 0                ! IF ENTRY INDICATES FREE PACKET

        then
            begin
            RING_ADDR = .DCT_ADDR [RR_BEG];             ! FIRST RESPONSE PACKET'S ADDRESS

            incr I from 1 to (RRING_LEN + CRING_LEN) do ! FOR EACH PACKET ADDRESS

                if (..RING_ADDR eqla .MSCP_PKT [.NEXT_PACKET, PKT_LO]) and
!MMM               (((.RING_ADDR + 2) and ED_OWN) eql ED_OWN)
                   (((.(.RING_ADDR + 2)) and ED_OWN) eql ED_OWN)                               !MMM

                then
                    begin                               ! CHECK ADDRESS AND OWNERSHIP
                    PACKET_OWNED = TRUE;                ! PACKET OWNED BY CONTROLLER
                    exitloop;
                    end
                else
                    RING_ADDR = .RING_ADDR + 4;         ! ADDRESS OF NEXT PACKET IN RING

            if not .PACKET_OWNED                        ! IF NOT ALREADY USED

            then
                begin
                PKT_USE [.NEXT_PACKET] = .CTLR;         ! ALLOCATE PACKET TO CONTROLLER
```

```
;  5                  index = .NEXT_PACKET;
;  5
;  5                  incr J from 2 to (PKT_LEN - 1) do        ! ZERO OUT PACKET
;  5                      MSCP_PKT [.NEXT_PACKET, .J, 0, 16, 0] = 0;
;  5
;  5                  exitloop;                                 ! DONE
;  5
;  4                  end;
;  4
;  3              end;
;  3
;  3          NEXT_PACKET = .NEXT_PACKET + 1;                   ! TRY NEXT PACKET IN RING
;  3
;  3          if .NEXT_PACKET gequ PKT_CNT
;  3
;  3          then
;  3              NEXT_PACKET = 0;                              ! IF BEYOND ALL PACKETS, START AT THE TOP
;  3
;  2          end;
;  2
;  2      if (.index geq 0) and                                 ! IF PACKET FOUND
;  3         (.PKT_USE [.index] geq 0)
;  2      then
;  2
;  3          begin
;  3          MSCP_PKT [.index, MSGLEN] = SZ_GEN;               ! PACKET SIZE - ONLY ONLINE AND SCC CHANGE IT
;  3          MSCP_PKT [.index, CREDITS] = 1;                   ! CREDIT SIZE
;  3          NEXT_PKT_USE = .NEXT_PACKET + 1;                  ! NEXT PACKET TO ALLOCATE
;  3
;  3          if.NEXT_PKT_USE gequ PKT_CNT
;  3          then
;  3              NEXT_PKT_USE = 0;                             ! CYCLE BACK TO BEGINNING IF AT END
;  3
;  2          end;
;  2
;  2      return .index;
;  2
;  1      end;


                                      .SBTTL  GET.PKT GLOBAL ROUTINES
000000  004137  000000G         GET.PKT::
                                      JSR     R1,$SAVE5                  ;                    6539
000004  162706  000006                SUB     #6,SP
000010  012704  177777                MOV     #-1,R4                     ; *.INDEX            6551
000014  113766  000000G 000004        MOVB    NEXT.PKT.USE,4(SP)         ; *.NEXT.PACKET      6560
000022  012766  000014  000002        MOV     #14,2(SP)                  ; *.COUNT            6562
000030  105016                  1$:   CLRB    (SP)                       ; PACKET.OWNED       6564
000032  005001                        CLR     R1                         ;                    6566
000034  156601  000004                BISB    4(SP),R1                   ; NEXT.PACKET.*
000040  105761  000000G               TSTB    PKT.USE(R1)
000044  002071                        BGE     7$
000046  013700  000000G               MOV     DCT.ADDR,R0                ;                    6570
```

```
000052 016005 000004                   MOV     4(R0),R5                ; *,RING.ADDR
000056 010146                          MOV     R1,-(SP)                ;                               6574
000060 012746 000106                   MOV     #106,-(SP)
000064 004737 000000G                  JSR     PC,BL$MUL
000070 012702 000010                   MOV     #10,R2                  ; *,I                           6572
000074 021560 000000G          2$:     CMP     (R5),MSCP.PKT(R0)       ; RING.ADDR,*                   6574
000100 001013                          BNE     3$
000102 016503 000002                   MOV     2(R5),R3                ; *(RING.ADDR),*                6576
000106 042703 077777                   BIC     #77777,R3
000112 020327 100000                   CMP     R3,#-100000
000116 001004                          BNE     3$
000120 112766 000001 000004            MOVB    #1,4(SP)                ; *,PACKET.OWNED                6580
000126 000404                          BR      4$                      ;                               6579
000130 062705 000004           3$:     ADD     #4,R5                   ; *,RING.ADDR                   6584
000134 005302                          DEC     R2                      ; I                             6572
000136 001356                          BNE     2$
000140 032766 000001 000004    4$:     BIT     #1,4(SP)                ; *,PACKET.OWNED                6586
000146 001027                          BNE     6$
000150 116661 000030 000000G           MOVB    30(SP),PKT.USE(R1)      ; CTLR,*                        6590
000156 010104                          MOV     R1,R4                   ; *,INDEX                       6591
000160 010116                          MOV     R1,(SP)                 ;                               6594
000162 012746 000043                   MOV     #43,-(SP)
000166 004737 000000G                  JSR     PC,BL$MUL
000172 005726                          TST     (SP)+
000174 012702 000002                   MOV     #2,R2                   ; *,J                           6593
000200 010003                  5$:     MOV     R0,R3                   ;                               6594
000202 060203                          ADD     R2,R3                   ; J,*
000204 006303                          ASL     R3
000206 005063 000000G                  CLR     MSCP.PKT(R3)
000212 005202                          INC     R2                      ; J                             6593
000214 020227 000042                   CMP     R2,#42                  ; J,*
000220 003767                          BLE     5$
000222 022626                          CMP     (SP)+,(SP)+             ;                               6589
000224 000414                          BR      9$
000226 022626                  6$:     CMP     (SP)+,(SP)+             ;                               6569
000230 105266 000004           7$:     INCB    4(SP)                   ; NEXT.PACKET                   6602
000234 126627 000004 000014            CMPB    4(SP),#14               ; NEXT.PACKET,*                 6604
000242 103402                          BLO     8$
000244 105066 000004                   CLRB    4(SP)                   ; NEXT.PACKET                   6607
000250 005366 000002           8$:     DEC     2(SP)                   ; COUNT                         6562
000254 001265                          BNE     1$
000256 005704                  9$:     TST     R4                      ; INDEX                         6611
000260 002435                          BLT     11$
000262 105764 000000G                  TSTB    PKT.USE(R4)             ; *(INDEX)                      6612
000266 002432                          BLT     11$
000270 010446                          MOV     R4,-(SP)                ; INDEX,*                       6616
000272 012746 000106                   MOV     #106,-(SP)
000276 004737 000000G                  JSR     PC,BL$MUL
000302 012760 000040 000006G           MOV     #40,MSCP.PKT+6(R0)
000310 142760 000017 000010G           BICB    #17,MSCP.PKT+10(R0)     ;                               6617
000316 152760 000001 000010G           BISB    #1,MSCP.PKT+10(R0)
000324 005000                          CLR     R0                      ;                               6618
000326 156600 000010                   BISB    10(SP),R0               ; NEXT.PACKET,*
```

```
000332  005200                              INC     R0
000334  110037  000000G                     MOVB    R0,NEXT.PKT.USE
000340  120027  000014                      CMPB    R0,#14                  ; NEXT.PKT.USE,*                6620
000344  103402                              BLO     10$
000346  105037  000000G                     CLRB    NEXT.PKT.USE
000352  022626                      10$:    CMP     (SP)+,(SP)+             ;                               6622
000354  010400                      11$:    MOV     R4,R0                   ;                               6615
000356  062706  000006                      ADD     #6,SP                   ; INDEX,*                       6551
000362  000207                              RTS     PC                      ;                               6539

; Routine Size:  122 words,     Routine Base:  $CODE$ + 5510
; Maximum stack depth per invocation:  13 words

;     6629  1
;     6630  1
```

```
global routine PUT_PKT (index) : novalue =

!+
!       THE MSCP PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
!       ROUTINE.
!-


    begin


    local
        RING_ADDR : word,
        OWNER : word;

    RING_ADDR = .DCT_ADDR [RR_BEG];                                 ! ADDRESS IN FIRST RESPONSE RING

    incr COUNT from 1 to (RRING_LEN + CRING_LEN) do                 ! FOR EACH ADDRESS IN THE RINGS
        begin

        if .MSCP_PKT [.index, PKT_LO] eqla ..RING_ADDR              ! IF ADDRESS MATCHES

        then
            begin
            OWNER = .RING_ADDR + 2;                                 ! ADDRESS OF OWNERSHIP WORD
            .OWNER = ..OWNER and (not (ED_OWN)) and (not (ED_FLAG)) ;  ! GIVE OWNERSHIP TO HOST
            end;


        RING_ADDR = .RING_ADDR + 4;                                 ! LOOK AT NEXT PACKET ADDRESS IN RING
        end;


    PKT_USE [.index] = -1;

    end;


                                        .SBTTL  PUT.PKT GLOBAL ROUTINES
000000  004137  000000G         PUT.PKT::
                                        JSR     R1,$SAVE4                       ;                       6633
000004  013700  000000G                 MOV     DCT.ADDR,R0                     ;                       6648
000010  016001  000004                  MOV     4(R0),R1                        ; *,RING.ADDR
000014  016602  000014                  MOV     14(SP),R2                       ; INDEX,*               6653
000020  010246                          MOV     R2,-(SP)
000022  012746  000106                  MOV     #106,-(SP)
000026  004737  000000G                 JSR     PC,BL$MUL
000032  012704  000010                  MOV     #10,R4                          ; *,COUNT               6650
000036  026011  000000G         1$:     CMP     MSCP.PKT(R0),(R1)               ; *,RING.ADDR           6653
000042  001005                          BNE     2$
000044  012703  000002                  MOV     #2,R3                           ; *,OWNER               6657
```

```
000050  060103                          ADD     R1,R3                 ; RING.ADDR,OWNER
000052  042713  140000                  BIC     #140000,(R3)          ; .OWNER                  6658
000056  062701  000004          2$:     ADD     #4,R1                 ; .RING.ADDR              6662
000062  005304                          DEC     R4                    ; COUNT                   6650
000064  001364                          BNE     1$
000066  112762  000377  000000G         MOVB    #377,PKT.USE(R2)      ;                         6666
000074  022626                          CMP     (SP)+,(SP)+           ;                         6641
000076  000207                          RTS     PC                    ;                         6633

; Routine Size:  32 words.      Routine Base:  $CODE$ + 6074
; Maximum stack depth per invocation:  8 words


;    6669  1
;    6670  1
```

```
;     6671  1     routine PUTA_PKT (CTLR) : novalue =
;     6672  1
;     6673  1     !+
;     6674  1     !       THIS ROUTINE DEALLOCATES ALL MSCP PACKETS WHICH HAVE BEEN ALLOCATED
;     6675  1     !       TO A PARTICULAR CONTROLLER.
;     6676  1     !
;     6677  1     !       INPUTS:
;     6678  1     !               CTLR - CONTROLLER NUMBER
;     6679  1     !-
;     6680  1
;     6681  1         incr COUNT from 0 to (PKT_CNT - 1) do        ! FOR EACH ENTRY IN ALLOCATION TABLE
;     6682  1
;     6683  1             if .PKT_USE [.COUNT] eql .CTLR           ! IF PACKET IS ALLOCATED TO GIVEN CONTROLLER
;     6684  1             then
;     6685  1                 PKT_USE [.COUNT] = 1;                ! DEALLOCATE IT


                                                 .SBTTL  PUTA.PKT GLOBAL ROUTINES
000000  010146                         PUTA.PKT:
                                                 MOV     R1,-(SP)                   ;                         6671
000002  005000                                   CLR     R0                         ; COUNT                   6681
000004  116001  000000G                1$:       MOVB    PKT.USE(R0),R1             ; *(COUNT),*              6683
000010  020166  000004                           CMP     R1,4(SP)                   ; *,CTLR
000014  001003                                   BNE     2$
000016  112760  000377  000000G                  MOVB    #377,PKT.USE(R0)           ; *,*(COUNT)              6685
000024  005200                         2$:       INC     R0                         ; COUNT                   6681
000026  020027  000013                           CMP     R0,#13                     ; COUNT,*
000032  003764                                   BLE     1$
000034  012601                                   MOV     (SP)+,R1                   ;                         6671
000036  000207                                   RTS     PC

; Routine Size:  16 words,      Routine Base:  $CODE$ + 6174
; Maximum stack depth per invocation:  2 words
```

```
;   6686  1      global routine GET_RETPKT (CTLR) =
;   6687  1
;   6688  1      !+
;   6689  1      !       THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
;   6690  1      !       FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;   6691  1      !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
;   6692  1      !       THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;   6693  1      !
;   6694  1      !       INPUTS:
;   6695  1      !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;   6696  1      !-
;   6697  1
;   6698  2          begin
;   6699  2
;   6700  2          local
;   6701  2              index : signed word initial (-1);       ! ASSUME NONE AVAILABLE
;   6702  2
;   6703  2          incr COUNT from 0 to (RP CNT - 1) do        ! FOR EACH ENTRY IN TABLE
;   6704  2
;   6705  2              if .RP_USE [.COUNT] lss 0               ! IF FREE RETPKT IS FOUND
;   6706  2              then
;   6707  3                  begin
;   6708  3                  RP_USE [.COUNT] = .CTLR;            ! ALLOCATE RETURN PACKET TO CONTROLLER
;   6709  3                  index = .COUNT;
;   6710  3
;   6711  3                  incr J from 0 to (RP_LEN - 1) do    ! ZERO OUT RETPKT
;   6712  3                      RETPKT [.COUNT, .J, 0, 16, 0] = 0;
;   6713  3
;   6714  3                  exitloop;                           ! DONE
;   6715  2                  end;
;   6716  2
;   6717  2          return .index;                              ! RETURN PACKET INDEX (OR  1) TO CALLER
;   6718  1          end;


                                                .SBTTL  GET.RETPKT GLOBAL ROUTINES
000000  004137  000000G                 GET.RETPKT::
                                                JSR     R1,$SAVE4                       ;                       6686
000004  012703  177777                          MOV     #-1,R3                          ; *,INDEX               6698
000010  005001                                  CLR     R1                              ; COUNT                 6703
000012  105761  000000G                 1$:     TSTB    RP.USE(R1)                      ; *(COUNT)              6705
000016  002025                                  BGE     3$
000020  116661  000014  000000G                 MOVB    14(SP),RP.USE(R1)               ; CTLR,*(COUNT)         6708
000026  010103                                  MOV     R1,R3                           ; COUNT,INDEX           6709
000030  010146                                  MOV     R1,-(SP)                        ; COUNT,*               6712
000032  012746  000026                          MOV     #26,-(SP)
000036  004737  000000G                         JSR     PC,BL$MUL
000042  022626                                  CMP     (SP)+,(SP)+
000044  005002                                  CLR     R2                              ; J                     6711
000046  010004                          2$:     MOV     R0,R4                           ;                       6712
000050  060204                                  ADD     R2,R4                           ; J,*
000052  006304                                  ASL     R4
000054  005064  000000G                         CLR     RETPKT(R4)
```

```
000060  005202                   INC    R2              ; J                    6711
000062  020227  000025           CMP    R2,#25          ; J,*
000066  003767                   BLE    2$
000070  000404                   BR     4$              ;                      6707
000072  005201           3$:     INC    R1              ; COUNT                6703
000074  020127  000007           CMP    R1,#7           ; COUNT,*
000100  003744                   BLE    1$
000102  010300           4$:     MOV    R3,R0           ; INDEX,*              6698
000104  000207                   RTS    PC              ;                      6686

; Routine Size:  35 words,      Routine Base:  $CODE$ + 6234
; Maximum stack depth per invocation:  8 words
```

```
;    6719  1      global routine PUT_RETPKT (index) : novalue =
;    6720  1
;    6721  1      !+
;    6722  1      !        THE RETURN PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;    6723  1      !        ROUTINE.
;    6724  1      !-
;    6725  1
;    6726  1          RP_USE [.index] = -1;


                                                .SBTTL  PUT.RETPKT GLOBAL ROUTINES
000000  016600  000002                  PUT.RETPKT::
                                                MOV     2(SP),R0                        ; INDEX,*                 6726
000004  112760  000377  000000G                 MOVB    #377,RP.USE(R0)
000012  000207                                  RTS     PC                              ;                         6719

; Routine Size:  6 words,       Routine Base:  $CODE$ + 6342
; Maximum stack depth per invocation:  0 words
```

```
;   1
;   1
;   1    global routine GET_IO_BUFF (ADDR) : novalue =
;   1
;   1
;   1    !+
;   1    !
;   1    !       THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
;   1    !       POOL.
;   1    !
;   1    !       INPUTS:
;   1    !               ADDR - ADDRESS TO STORE THE 2-WORD BUFFER DESCRIPTOR
;   1    !
;   1    !       IMPLICIT INPUTS:
;   1    !               CCTLR - CURRENT CONTROLLER NUMBER
;   1    !
;   1    !       OUTPUTS:
;   1    !               THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
;   1    !               WORDS AT "ADDR" AND "ADDR + 2". OTHERWISE, A ZERO IS RETURNED
;   1    !               AT "ADDR" IF NO BUFFERS ARE AVAILABLE.
;   1    !-
;   1
;   1
;   1
;   2        begin
;   2        .ADDR = 0;                                              ! ASSUME FAILURE
;   2
;   2        incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do   ! FOR EACH ENTRY IN BUFFER TABLE
;   2
;   2            if .BUFF_OWN [.COUNT] lss 0                         ! IF BUFFER IS FREE
;   2
;   2            then
;   2
;   3                begin
;   3                BUFF_OWN [.COUNT] = .CCTLR;                     ! ALLOCATE BUFFER TO CONTROLLER
;   3                .ADDR = .BUFF_ADDR [.COUNT];                    ! RETURN BUFFER DESCRIPTOR
;   3                exitloop;                                       ! DONE
;   2                end;
;   2
;   2
;   1        end;                                                    ! ROUTINE GET_IO_BUFF


                                                .SBTTL  GET.IO.BUFF GLOBAL ROUTINES
000000  010146                          GET.IO.BUFF::
                                                MOV     R1,-(SP)                        ;                       6729
000002  005076  000004                          CLR     @4(SP)                          ; ADDR                  6751
000006  005001                                  CLR     R1                              ; COUNT                 6753
000010  105761  000000G                 1$:     TSTB    BUFF.OWN(R1)                    ; *(COUNT)              6755
000014  002011                                  BGE     2$
000016  113761  000000G 000000G                 MOVB    CCTLR,BUFF.OWN(R1)              ; *,*(COUNT)            6760
000024  010100                                  MOV     R1,R0                           ; COUNT,*               6761
000026  006300                                  ASL     R0
000030  016076  000000G 000004                  MOV     BUFF.ADDR(R0),@4(SP)            ; *,ADDR
```

```
000036  000404                   BR      3$                          ;                     6759
000040  005201          2$:      INC     R1                          ; COUNT               6753
000042  020127  000007           CMP     R1,#7                       ; COUNT,*
000046  003760                   BLE     1$
000050  012601          3$:      MOV     (SP)+,R1                    ;                     6729
000052  000207                   RTS     PC

; Routine Size:  22 words,      Routine Base:  $CODE$ + 6356
; Maximum stack depth per invocation:  2 words


;    6767  1
;    6768  1
```

```
;   6769 1      global routine PUT_IO_BUFF (ADDR) : novalue =
;   6770 1
;   6771 1      !+
;   6772 1      !       THIS ROUTINE HANDLES THE DEALLOCATION OF AN I/O BUFFER, RETURNING IT
;   6773 1      !       TO THE BUFFER POOL.
;   6774 1      !
;   6775 1      !       INPUTS:
;   6776 1      !               ADDR - ADDRESS OF THE 2-WORD BUFFER DESCRIPTOR TO BE
;   6777 1      !                                  DEALLOCATED
;   6778 1      !-
;   6779 1
;   6780 1          incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;   6781 1
;   6782 1              if .BUFF_ADDR [.COUNT] eqla ..ADDR                      ! IF THIS IS THE BUFFER'S ENTRY
;   6783 1              then
;   6784 2                  begin
;   6785 2                  BUFF_OWN [.COUNT] = -1;                             ! DEALLOCATE BUFFER
;   6786 2                  exitloop;                                           ! DONE
;   6787 1                  end;


                                                 .SBTTL  PUT.IO.BUFF GLOBAL ROUTINES
000000  010146                           PUT.IO.BUFF::
                                                 MOV     R1,-(SP)                        ;                       6769
000002  005001                                   CLR     R1                              ; COUNT                 6780
000004  010100                           1$:     MOV     R1,R0                           ; COUNT,*               6782
000006  006300                                   ASL     R0
000010  026076  000000G 000004                   CMP     BUFF.ADDR(R0),@4(SP)            ; *,ADDR
000016  001004                                   BNE     2$
000020  112761  000377  000000G                  MOVB    #377,BUFF.OWN(R1)               ; *,*(COUNT)            6785
000026  000404                                   BR      3$                              ;                       6784
000030  005201                           2$:     INC     R1                              ; COUNT                 6780
000032  020127  000007                           CMP     R1,#7                           ; COUNT,*
000036  003762                                   BLE     1$
000040  012601                           3$:     MOV     (SP)+,R1                        ;                       6769
000042  000207                                   RTS     PC

; Routine Size:  18 words,      Routine Base:  $CODE$ + 6432
; Maximum stack depth per invocation:  2 words
```

```
;    6788  1      global routine PUTA_BUFF : novalue =
;    6789  1
;    6790  1      !+
;    6791  1      !         THIS ROUTINE DEALLOCATES ALL I/O BUFFERS WHICH HAVE BEEN ALLOCATED TO
;    6792  1      !         THE CURRENT CONTROLLER (CCTLR).
;    6793  1      !-
;    6794  1
;    6795  1          incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do      ! FOR EACH ENTRY IN BUFFER TABLE
;    6796  1
;    6797  1              if .BUFF_OWN [.COUNT] eql .CCTLR                       ! IF THIS BUFFER ALLOCATED TO CURRENT CONTROLLER
;    6798  1              then
;    6799  1                  BUFF_OWN [.COUNT] = -1;                            ! DEALLOCATE IT


                                             .SBTTL  PUTA.BUFF GLOBAL ROUTINES
000000  010146                       PUTA.BUFF::
                                             MOV     R1,-(SP)                        ;                         6788
000002  005000                               CLR     R0                              ; COUNT                   6795
000004  116001  C00000G              1$:     MOVB    BUFF.OWN(R0),R1                 ; *(COUNT),*              6797
000010  020137  000000G                      CMP     R1,CCTLR
000014  001003                               BNE     2$
000016  112760  000377  000000G              MOVB    #377,BUFF.OWN(RO)               ; *,*(COUNT)              6799
000024  005200                       2$:     INC     R0                              ; COUNT                   6795
000026  020027  000007                       CMP     R0,#7                           ; COUNT,*
000032  003764                               BLE     1$
000034  012601                               MOV     (SP)+,R1                        ;                         6788
000036  000207                               RTS     PC

; Routine Size:  16 words,      Routine Base:  $CODE$ + 6476
; Maximum stack depth per invocation:  2 words
```

```
;   6800  1    global routine OUT_IODQ =
;   6801  1
;   6802  1    !+
;   6803  1    !       THIS ROUTINE RETURNS TO THE CALLER THE NEXT RETPKT INDEX TO BE
;   6804  1    !       PROCESSED FROM THE I/O DONE QUEUE (IODQ). THE "OUT" POINTER TO THE
;   6805  1    !       QUEUE IS ALSO UPDATED.
;   6806  1    !
;   6807  1    !       INPUTS:
;   6808  1    !               NONE
;   6809  1    !
;   6810  1    !       OUTPUTS:
;   6811  1    !               THE INDEX OF THE NEXT RETPKT TO BE PROCESSED.
;   6812  1    !-
;   6813  1
;   6814  2        begin
;   6815  2
;   6816  2        local
;   6817  2            index : word;
;   6818  2
;   6819  2        index = .IODQ [.IODQ_OUT];                      ! GET NEXT RETPKT INDEX
;   6820  2        IODQ_OUT = .IODQ_OUT + 1;                       ! ADVANCE "OUT" POINTER
;   6821  2
;   6822  2        if .IODQ_OUT gequ IODQ_LEN                      ! IF BEYOND END OF QUEUE
;   6823  2        then
;   6824  2            IODQ_OUT = 0;                               ! SET POINTER TO BEGINNING OF QUEUE
;   6825  2
;   6826  2        return .index;                                  ! RETURN INDEX TO CALLER
;   6827  1        end;


                                               .SBTTL  OUT.IODQ GLOBAL ROUTINES
000000  013700  000000G                 OUT.IODQ::
                                               MOV     IODQ.OUT,R0                 ;                  6819
000004  116000  000000G                        MOVB    IODQ(R0),R0                 ; *.INDEX
000010  042700  177400                         BIC     #177400,R0                  ; *.INDEX
000014  005237  000000G                        INC     IODQ.OUT                    ;                  6820
000020  023727  000000G 000010                 CMP     IODQ.OUT,#10                ;                  6822
000026  103402                                 BLO     1$
000030  005037  000000G                        CLR     IODQ.OUT                    ;                  6824
000034  000207                          1$:    RTS     PC                          ;                  6800

; Routine Size:  15 words,      Routine Base:  $CODE$ + 6536
; Maximum stack depth per invocation:  0 words
```

```
;    6828  1      global routine IN_IODQ (index) : novalue =
;    6829  1
;    6830  1      !+
;    6831  1      !       THIS ROUTINE INSERTS A RETURN PACKET INDEX INTO THE I/O DONE QUEUE, AND
;    6832  1      !       UPDATES THE IODQ_IN POINTER.
;    6833  1      !-
;    6834  1
;    6835  1          if ((.IODQ_IN + 1) eql .IODQ_OUT) or
;    6836  2             (.IODQ_IN - (IODQ_LEN - 1) eql .IODQ_OUT)
;    6837  1          then
;    6838  1              return
;    6839  1          else
;    6840  2              begin
;    6841  2              IODQ [.IODQ_IN] = .index;                ! LOAD INDEX INTO QUEUE
;    6842  2              IODQ_IN = .IODQ_IN + 1;                  ! ADVANCE "IN" POINTER
;    6843  2
;    6844  2              if .IODQ_IN gequ IODQ_LEN                ! IF BEYOND END OF QUEUE
;    6845  2              then
;    6846  2                  IODQ_IN = 0;                         ! CYCLE BACK TO BEGINNING OF QUEUE
;    6847  2
;    6848  1              end;                                     ! IF IODQ IS NOT FULL


                                                  .SBTTL  IN.IODQ GLOBAL ROUTINES
000000  010146                          IN.IODQ::
                                                  MOV     R1,-(SP)                                    6828
000002  013701  000000G                           MOV     IODQ.IN,R1                    ;             6835
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
000022  162700  000007                            SUB     #7,R0                         ;             6836
000026  020037  000000G                           CMP     R0,IODQ.OUT
000032  001413                                    BEQ     1$
000034  116661  000004  000000G                   MOVB    4(SP),IODQ(R1)                ; INDEX,*     6838
                                                                                                      6841
000042  005237  000000G                           INC     IODQ.IN                       ;             6842
000046  023727  000000G 000010                    CMP     IODQ.IN,#10                   ;             6844
000054  103402                                    BLO     1$
000056  005037  000000G                           CLR     IODQ.IN                       ;             6846
000062  012601                          1$:       MOV     (SP)+,R1                      ;             6828
000064  000207                                    RTS     PC

; Routine Size:  27 words,      Routine Base:  $CODE$ + 6574
; Maximum stack depth per invocation:  2 words
```

```
;    6849  1
;    6850  1
;    6851  1     global routine DROP_CTLR (CTLR, REASON) : novalue =
;    6852  1
;    6853  1
;    6854  1
;    6855  1     !+
;    6856  1     !       THIS ROUTINE DROPS ALL UNITS ASSOCIATED WITH THE CONTROLLER DESIGNATED
;    6857  1     !       BY "CTLR". THE REASON FOR DROPPING THE DEVICE IS LOADED INTO THE DUR
;    6858  1     !       VECTOR FOR EACH ATTACHED UNIT. THIS DATA IS THEN USED BY THE DROP UNIT
;    6859  1     !       SECTION.
;    6860  1     !-
;    6861  1
;    6862  1
;    6863  2         begin
;    6864  2
;    6865  2         local
;    6866  2             UNIT;
;    6867  2
;    6868  2         incr N from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do      ! FOR EACH UNIT
;    6869  2
;    6870  2             if .CST [.CTLR, .N + OF_DATA, D_PRES] eql PRESENT                                          ! IF CONFIGURED
;    6871  2             then
;    6872  3                 begin
;    6873  3                 UNIT = .CST [.CTLR, .N + OF_DATA, D_UNIT];                                             ! DRS UNIT NUMBER
;    6874  3                 DUR [.UNIT] = .REASON;                                                                 ! DROP REASON
;    6875  3                 DODU (.UNIT);                                                                          ! DROP UNIT
;    6876  2                 end;
;    6877  2
;    6878  1         end;


                                                    .SBTTL  DROP.CTLR GLOBAL ROUTINES
000000  004137  000000G                    DROP.CTLR::
                                                    JSR     R1,$SAVE3                       ;                       6851
000004  016646  000014                              MOV     14(SP),(SP)                     ; CTLR,*                6870
000010  012746  000053                              MOV     #53,-(SP)
000014  004737  000000G                             JSR     PC,BL$MUL
000020  010003                                      MOV     R0,R3
000022  012702  000003                              MOV     #3,R2                           ; *,N                   6868
000026  010300                             1$:      MOV     R3,R0                           ;                       6870
000030  060200                                      ADD     R2,R0                           ; N,*
000032  006300                                      ASL     R0
000034  032760  040000  000000G                     BIT     #40000,CST(R0)
000042  001412                                      BEQ     2$
000044  016001  000000G                             MOV     CST(R0),R1                      ; *,UNIT                6873
000050  000301                                      SWAB    R1                              ; UNIT
000052  042701  177760                              BIC     #177760,R1                      ; *,UNIT
000056  116661  000016  000000G                     MOVB    16(SP),DUR(R1)                  ; REASON,*(UNIT)        6874
000064  010100                                      MOV     R1,R0                           ; UNIT,*                6875
000066  104451                                      TRAP    51
000070  062702  000012                     2$:      ADD     #12,R2                          ; *,N                   6868
000074  020227  000041                              CMP     R2,#41                          ; N,*
```

```
000100  003752                    BLE     1$
000102  022626                    CMP     (SP)+,(SP)+
000104  000207                    RTS     PC                    ;           6863
                                                                ;           6851

; Routine Size:  35 words,      Routine Base:  $CODE$ + 6662
; Maximum stack depth per invocation:  8 words


;    6879  1
;    6880  1
```

```
;    6881  1     global routine DRV_CTLERR (CTLR) : novalue =
;    6882  1
;    6883  1     !+
;    6884  1     !       THIS ROUTINE IS CALLED BY DRV_TIMCHK AND FATAL_ERROR WHENEVER AN
;    6885  1     !       UNRECOVERABLE CONTROLLER ERROR HAS BEEN DETECTED. ITS PURPOSE IS TO
;    6886  1     !       CLEAN UP ALL CONTROLLER-RELATED DATA IN THE "DRIVER" PORTION OF THE
;    6887  1     !       PROGRAM. THIS INCLUDES MARKING THE CONTROLLER OFFLINE, CLEARING THE
;    6888  1     !       C-RING COUNT, AND DEALLOCATING MSCP PACKETS DESCRIBED IN THE RESPONSE
;    6889  1     !       RING.
;    6890  1     !
;    6891  1     !       INPUTS:
;    6892  1     !               CTLR - DYING CONTROLLER NUMBER
;    6893  1     !-
;    6894  1
;    6895  2         begin
;    6896  2
;    6897  2         local
;    6898  2             D_ADDR : ref block [DCT_LEN, word] field (DCT FIELDS);  ! CONTROLLER'S DCT ADDRESS
;    6899  2
;    6900  2         D_ADDR = DCT + (.CTLR * DCT_LEN * 2);                       ! GET CONTROLLER S DCT ADDR
;    6901  2         D_ADDR [WORD0] = OFFLINE;                                   ! MARK DCT OFFLINE AND CLEAR CRING_CNT
;    6902  2         PUTA_PKT (.CTLR);                                           ! RELEASE ALL PACKETS ALLOCATED TO CONTROLLER
;    6903  2         DROP_CTLR (.CTLR, DU_CFATAL);                               ! DROP ALL UNITS ON THE CONTROLLER
;    6904  1         end;                                                        ! ROUTINE DRV CTLERR


                                          .SBTTL  DRV.CTLERR GLOBAL ROUTINES
000000  010146                    DRV.CTLERR::
                                          MOV     R1, (SP)                 ;                 6881
000002  016601  000004                    MOV     4(SP),R1                 ; CTLR,*          6900
000006  010146                            MOV     R1,-(SP)
000010  012746  000022                    MOV     #22,-(SP)
000014  004737  000000G                   JSR     PC,BL$MUL
000020  062700  000000G                   ADD     #DCT,R0
000024  005010                            CLR     (R0)                     ; D.ADDR
000026  010116                            MOV     R1,(SP)                                    6901
000030  004737  006174'                   JSR     PC,PUTA.PKT              ;                 6902
000034  010116                            MOV     R1,(SP)
000036  012746  000006                    MOV     #6,-(SP)                 ;                 6903
000042  004737  006662'                   JSR     PC,DROP.CTLR
000046  062706  000006                    ADD     #6,SP
000052  012601                            MOV     (SP)+,R1                 ;                 6895
000054  000207                            RTS     PC                       ;                 6881

; Routine Size:  23 words,     Routine Base:  $CODE$ + 6770
; Maximum stack depth per invocation:  5 words
```

```
global routine SEND (index) =

!+
!       IF THE CURRENT RDRX IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
!       ROUTINE "SENDS" A COMMAND TO THE RDRX BY LOADING THE PACKET
!       DESCRIPTOR OF AN MSCP PACKET INTO THE COMMAND RING AND READING THE
!       DEVICE'S IP REGISTER.  IF THE
!       CURRENT RDRX IN NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
!       THE CALLER, AND NO ACTION IS TAKEN.
!
!       INPUTS:
!               INDEX - INDEX OF MSCP PACKET CONTAINING THE COMMAND TO
!                       BE SENT
!
!       IMPLICIT INPUTS:
!               CCTLR - CURRENT CONTROLLER NUMBER
!               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
!-

    begin

    local
        SLOT_ADDR,
        TEMP : word,
        CUR_PRIORITY : word;

    if (.DCT_ADDR [CRING_CNT] lssu CRING_LEN) and                        ! IF CRING IS NOT FULL AND
      ((.DCT_ADDR [STAT] eql ONLINE) or                                  !   IF DEVICE IS ONLINE OR
        (.MSCP_PKT [.index, OPCODE] eql OP_SCC))                         !   IT IS A SET-CTRL-CHAR COMMAND
    then

        if (not ((.MSCP_PKT [.index, OPCODE] eql OP_ACC) or (.MSCP_PKT [.index, OPCODE] eql OP_ONL) or
                (.MSCP_PKT [.index, OPCODE] eql OP_RD) or (.MSCP_PKT [.index, OPCODE] eql OP_SCC) or
        (.MSCP_PKT [.INDEX, OPCODE] EQL OP_SDD) OR                       !ZZZ
        (.MSCP_PKT [.INDEX, OPCODE] EQL OP_RCD) OR                       !ZZZ
        (.MSCP_PKT [.INDEX, OPCODE] EQL OP_GDS) OR                       !ZZZ
        (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ELP) OR                       !ZZZ
        (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ABT) OR                       !ZZZ
        (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ESP) OR                       !ZZZ
                (.MSCP_PKT [.index, OPCODE] eql OP_WRT)))
        then
            begin
            PRINTF (DBM107, .MSCP_PKT [.index, OPCODE]);
            return FAILURE;
            end
        else
            begin

            do
                BREAK                                                    ! LOOP TILL CREDIT BALANCE POSITIVE
            until ((.MSCP_PKT [.index, CMD_TYPE] eql IMM_CMD) and
                    (.CREDIT_BAL gequ 1)) or
                  (.CREDIT_BAL gtru 1);
```

```
;  3
;  3                MSCP_PKT [.index, CRN_LO] = (CRN_LOW = .CRN_LOW + 1);     ! ASSIGN CMD REF NUM
;  3
;  3                if .CRN_LOW eql 0
;  3                then
;  3                    CRN_HIGH = .CRN_HIGH + 1;                           ! CMD REF NUM (HIGH ORDER)
;  3
;  3                MSCP_PKT [.index, CRN_HI] = .CRN_HIGH;                  !
;  3                SLOT_ADDR = .DCT_ADDR [CR_NEXT];                        ! ADDR OF NEXT COMMAND SLOT
;  3
;  3                do                                                      ! WAIT TILL NEXT SLOT HOST OWNED
;  3                    BREAK                                               !
;  3                until ((.(.SLOT_ADDR + 2) and ED_OWN) eql 0);           !
;  3
;  3                GETPRI (CUR_PRIORITY);                                  ! NO INTERRUPTS WHILE POINTERS UPDATED
;  3    !ZZZ        SETPRI (PRIO4);                                         !
;  3                SETPRI (.BRLEVEL);                                      !          ZZZ
;  3
;  3                .SLOT_ADDR = .MSCP_PKT [.index, PKT_LO];                ! LOAD BUFF DESC (LO) INTO COMMAND SLOT
;  3                SLOT_ADDR = .SLOT_ADDR + 2;                             ! ADVANCE TO NEXT WORD
;  3                .SLOT_ADDR = .MSCP_PKT [.index, PKT_HI];                ! LOAD BUFF DESC (HI) INTO COMMAND SLOT
;  3                .SLOT_ADDR = ..SLOT_ADDR and (not (ED_FLAG));           ! CLEAR INTERRUPT FLAG IN CASE SET
;  3                .SLOT_ADDR = ..SLOT_ADDR or ED_OWN;                     ! GIVE OWNERSHIP TO CONTROLLER
;  3                SLOT_ADDR = .SLOT_ADDR + 2;                             ! ADVANCE TO NEXT COMMAND SLOT
;  3
;  3                if .SLOT_ADDR gtra .DCT_ADDR [CR_END]                   ! IF BEYOND END OF CRING
;  3                then
;  3                    SLOT_ADDR = .DCT_ADDR [CR_BEG];                     ! CYCLE BACK TO BEGINNING
;  3
;  3                DCT_ADDR [CR_NEXT] = .SLOT_ADDR;                        ! RESTORE CR_NEXT POINTER IN DCT
;  3                DCT_ADDR [CRING_CNT] = .DCT_ADDR [CRING_CNT] + 1;       ! INCR # OF COMMANDS IN CRING
;  4                IF (.MSCP_PKT [.INDEX, CONNID] EQL CID_MSCP)           !IF MSCP COMMAND              ZZZ
;  3                THEN (CREDIT_BAL = .CREDIT_BAL - 1);                    !DECR CREDIT BALANCE          ZZZ
;  3                TEMP = .RDRX_ADDR [RCIP, RC_ALL];                       ! READ IP TO FORCE PORT TO POLL
;  3                SETPRI (.CUR_PRIORITY);                                 ! LOWER PRIORITY
;  3                return SUCCESS;
;  3                end
;  3
;  2        else
;  2            return FAILURE;                                             ! IF DEVICE IS NOT ONLINE
;  2
;  1        end;                                                            ! ROUTINE SEND


                                            .SBTTL  SEND GLOBAL ROUTINES
000000  004137  000000G             SEND::  JSR     R1,$SAVE3               ;                               6905
000004  005746                              TST     -(SP)
000006  127727  000000G 000004              CMPB    @DCT.ADDR,#4            ;                               6931
000014  103100                              BHIS    2$
000016  005777  000000G                     TST     @DCT.ADDR               ;                               6932
000022  100413                              BMI     1$
000024  016646  000014                      MOV     14(SP),-(SP)            ; INDEX,*                       6933
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$:  MOV     14(SP),(SP)                 ; INDEX,*                6936
000056  012746  000106               MOV     #106,-(SP)
000062  004737  000000G              JSR     PC,BL$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                      ;                        6937
000120  001437                       BEQ     3$
000122  020027  000004               CMP     R0,#4
000126  001434                       BEQ     3$
000130  020027  000005               CMP     R0,#5                       ;                        6939
000134  001431                       BEQ     3$
000136  020027  000001               CMP     R0,#1                       ;                        6940
000142  001426                       BEQ     3$
000144  020027  000003               CMP     R0,#3                       ;                        6941
000150  001423                       BEQ     3$
000152  020027  000006               CMP     R0,#6                       ;                        6942
000156  001420                       BEQ     3$
000160  020027  000002               CMP     R0,#2                       ;                        6943
000164  001415                       BEQ     3$
000166  020027  000042               CMP     R0,#42                      ;                        6944
000172  001412                       BEQ     3$
000174  010046                       MOV     R0,-(SP)                    ;                        6947
000176  012746  000000G              MOV     #DBM107,-(SP)
000202  012746  000002               MOV     #2,-(SP)
000206  010600                       MOV     SP,R0                       ; SP,*
000210  104417                       TRAP    17
000212  062706  000006               ADD     #6,SP                       ;                        6946
000216  000504                  2$:  BR      10$                         ;                        6936
000220  104422                  3$:  TRAP    22                          ;                        6953
000222  105762  000004G              TSTB    MSCP.PKT+4(R2)              ;                        6955
000226  001003                       BNE     4$
000230  005737  000000G              TST     CREDIT.BAL                  ;                        6956
000234  001004                       BNE     5$
000236  023727  000000G 000001  4$:  CMP     CREDIT.BAL,#1               ;                        6957
000244  101765                       BLOS    3$
000246  013700  000000G         5$:  MOV     CRN.LOW,R0                  ;                        6959
000252  005200                       INC     R0
000254  010037  000000G              MOV     R0,CRN.LOW
000260  010062  000012G              MOV     R0,MSCP.PKT+12(R2)
000264  001002                       BNE     6$
000266  005237  000000G              INC     CRN.HIGH                    ;                        6961
000272  013762  000000G 000014G 6$:  MOV     CRN.HIGH,MSCP.PKT+14(R2)    ;                        6963
                                                                         ;                        6965
000300  013700  000000G              MOV     DCT.ADDR,R0                 ;                        6966
```

```
000304  016001  000020                 MOV    20(R0),R1                ; •.SLOT.ADDR
000310  104422                   7$:   TRAP   22                       ;                         6968
000312  032761  100000  000002         BIT    #-100000,2(R1)           ; •.•(SLOT.ADDR)          6970
000320  001373                         BNE    7$
000322  104440                         TRAP   40                       ;                         6972
000324  010003                         MOV    R0,R3                    ; •.CUR.PRIORITY
000326  013700  000000G                MOV    BRLEVEL,R0               ;                         6974
000332  104441                         TRAP   41
000334  016221  000000G                MOV    MSCP.PKT(R2),(R1)+       ; •.SLOT.ADDR             6976
000340  016211  000002G                MOV    MSCP.PKT+2(R2),(R1)      ; •.SLOT.ADDR             6978
000344  042711  040000                 BIC    #40000,(R1)              ; •.SLOT.ADDR             6979
000350  052721  100000                 BIS    #100000,(R1)+            ; •.SLOT.ADDR             6980
000354  013700  000000G                MOV    DCT.ADDR,R0              ;                         6983
000360  020160  000012                 CMP    R1,12(R0)                ; SLOT.ADDR,•
000364  101402                         BLOS   8$
000366  016001  000010                 MOV    10(R0),R1                ; •.SLOT.ADDR             6985
000372  010160  000020           8$:   MOV    R1,20(R0)                ; SLOT.ADDR,•             6987
000376  105210                         INCB   (R0)                     ;                         6988
000400  105762  000011G                TSTB   MSCP.PKT+11(R2)          ;                         6989
000404  001002                         BNE    9$
000406  005337  000000G                DEC    CREDIT.BAL               ;                         6990
000412  017716  000000G          9$:   MOV    @RDRX.ADDR,(SP)          ; •.RC.REG                6991
000416  010300                         MOV    R3,R0                    ; CUR.PRIORITY,•          6992
000420  104441                         TRAP   41
000422  012700  000001                 MOV    #1,R0                    ;                         6996
000426  000401                         BR     11$                      ;                         6997
000430  005000                   10$:  CLR    R0
000432  005726                   11$:  TST    (SP)+                    ;                         6905
000434  000207                         RTS    PC
```

; Routine Size: 143 words, Routine Base: $CODE$ + 7046
; Maximum stack depth per invocation: 10 words

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


                                                   .SBTTL  WAIT GLOBAL ROUTINES
000000  104422                           WAIT::
000000                                   1$:      TRAP    22                                       ;      7008
000002  023737  000000G 000000G                   CMP     IODQ.IN,IODQ.OUT                         ;      7010
000010  001773                                    BEQ     1$
000012  000207                                    RTS     PC                                       ;      7000

; Routine Size:  6 words,       Routine Base:  $CODE$ + 7504
; Maximum stack depth per invocation:  2 words


;   1
```

```
;   1
;   1   GLOBAL ROUTINE MODULAS (LO_LIMIT, HI LIMIT) =                              !ZZZ
;   1                                                                              !ZZZ
;   1   !+ THE PURPOSE OF THIS ROUTINE IS TO GET A RANDOM NUMBER BETWEEN           !ZZZ
;   1   !  THE LOW AND HIGH LIMITS. THIS SHOULD WORK FOR A 16 BIT WORD.            !ZZZ
;   1   !- THE "MOD" FUNC ONLY WORKS ON 15 BITS.                                   !ZZZ
;   1                                                                              !ZZZ
;   2   BEGIN                                                                      !ZZZ
;   2   OWN     X : WORD;               !VARIABLE FOR RANDOM WD TABLE              !ZZZ
;   2   LOCAL   ANSWER : UNSIGNED WORD, !FINAL ANSWER                              !ZZZ
;   2           SAVESZ : UNSIGNED WORD, !SAVES SIZE OF WINDOW                      !ZZZ
;   2           SIZE : UNSIGNED WORD;   !SIZE OF WINDOW                            !ZZZ
;   2                                                                              !ZZZ
;   2                                                                              !ZZZ
;   2   X = .X + 1;                                                                !ZZZ
;   2   IF .X GEQ RDM_LEN                                                          !ZZZ
;   2   THEN X = 0;                     !KEEP ROTATING RANDOM NUMBERS USED         !ZZZ
;   2
                        !ZZZ
;   2   SIZE = .HI_LIMIT - .LO_LIMIT;
;   2   SAVESZ = .HI_LIMIT - .LO_LIMIT;                                            !ZZZ
;   3   IF (.SIZE LEQU %O'077777')      !IF BIT 15 NOT SET                         !ZZZ
;   3   THEN ANSWER = ((.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1))              !ZZZ
;   3                                   !ONLY 15 BIT WD, SO TAKE RANDOM SAMPLE     !ZZZ
;   2   ELSE                            !16 BIT WD                                 !ZZZ
;   3           BEGIN                                                              !ZZZ
;   3           SIZE = .SIZE ^ -1;      !MAKES SIZE A 15 BIT LENGTH, OR DIV BY 2   !ZZZ
;   3           ANSWER = (.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1);            !ZZZ
;   3                                   !GIVES 15 BIT RANDOM NUMBER                !ZZZ
;   3           ANSWER = .ANSWER ^ 1;   !BUILD UP TO REGULAR SIZE                  !ZZZ
;   3           ANSWER = .ANSWER + (.RANDOM [.X + 1] AND 1);                       !ZZZ
;   3                                   !RANDOMLY FILL BIT 0                       !ZZZ
;   4           IF (.ANSWER GTRU SAVESZ) !ITS POSSIBLE TO BE 1 LARGER THAN SIZE    !ZZZ
;   3           THEN ANSWER = .SAVESZ;  !SO CHECK.                                 !ZZZ
;   2           END;                                                               !ZZZ
;   2   RETURN .ANSWER;
;   1   END;                            !END MODULAS ROUTINE                       !ZZZ


007520                                  X:          .BLKW   1


                                                    .SBTTL  MODULAS GLOBAL ROUTINES
000000  004137  000000G                 MODULAS::
                                                    JSR     R1,$SAVE2           ;                7013
000004  005746                                      TST     -(SP)
000006  005237  007520'                             INC     X                   ;                7026
000012  023727  007520'  000020                     CMP     X,#20               ;                7027
000020  002402                                      BLT     1$
000022  005037  007520'                             CLR     X                   ;                7028
000026  016600  000012                  1$:         MOV     12(SP),R0           ; HI.LIMIT,*     7030
000032  166600  000014                              SUB     14(SP),R0           ; LO.LIMIT,*
000036  010001                                      MOV     R0,R1               ; *,SIZE
000040  010016                                      MOV     R0,(SP)             ; *,SAVESZ       7031
```

```
000042  013700  007520'          MOV     X,R0                   ;                       7033
000046  006300                   ASL     R0
000050  020127  077777           CMP     R1,#77777              ; SIZE,*                7032
000054  101011                   BHI     2$
000056  016046  000000G          MOV     RANDOM(R0),(SP)        ;                       7033
000062  042716  100000           BIC     #100000,(SP)
000066  010146                   MOV     R1,-(SP)               ; SIZE,*
000070  005216                   INC     (SP)
000072  004737  000000G          JSR     PC,BL$MOD
000076  000431                   BR      3$
000100  006201           2$:     ASR     R1                     ;                       7032
000102  016046  000000G          MOV     RANDOM(R0),-(SP)       ; SIZE                  7037
000106  042716  100000           BIC     #100000,(SP)           ;                       7038
000112  010146                   MOV     R1,-(SP)               ; SIZE,*
000114  005216                   INC     (SP)
000116  004737  000000G          JSR     PC,BL$MOD
000122  006300                   ASL     R0                     ; ANSWER                7040
000124  013701  007520'          MOV     X,R1                   ;                       7041
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                  ;                       7043
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         7044
000162  062706  000006   3$:     ADD     #6,SP                  ;                       7013
000166  000207                   RTS     PC

; Routine Size:  60 words,       Routine Base:  $CODE$ + 7522
; Maximum stack depth per invocation:  7 words
```

```
;    7048  1     %sbttl 'ERROR MESSAGE SUBROUTINES'
;    7049  1
;    7050  1     routine EMS_SA : novalue =
;    7051  1
;    7052  1     !+
;    7053  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SA_REG" WHICH CONTAINS
;    7054  1     !       THE CONTENTS OF THE SA REGISTER.
;    7055  1     !-
;    7056  1
;    7057  2         begin
;    7058  2
;    7059  2         if .SA_REG eql %o'177777'                                      ! IF CONTROLLER TIME-OUT
;    7060  2         then
;    7061  3             begin
;    7062  3             PRINTX (CRLF);
;    7063  3             PRINTX (ASTERISK);
;    7064  3             PRINTX (.CNTR_ERR [0]);
;    7065  3             end
;    7066  2         else
;    7067  2
;    7068  2             if (.SA_REG and %o'003777') lequ 22                        ! IF GENERIC CONTROLLER ERROR
;    7069  2             then
;    7070  3                 begin
;    7071  3                 PRINTX (CRLF);
;    7072  3                 PRINTX (ASTERISK);
;    7073  3                 PRINTX (.CNTR_ERR [.SA_REG and %o'003777']);
;    7074  3                 end
;    7075  2             else
;    7076  2
;    7077  2                 if ((.SA_REG and %o'003777') - 400) lequ 6             ! IF RDRX SPECIFIC CONTROLLER ERROR
;    7078  2                 then
;    7079  3                     begin
;    7080  3                     PRINTX (CRLF);
;    7081  3                     PRINTX (ASTERISK);
;    7082  3                     PRINTX (.RDRX_ERR [(.SA_REG and %o'003777') - 400]);
;    7083  3                     end
;    7084  2                 else
;    7085  2                     PRINTX (EX_SA, .SA_REG);                           ! JUST PRINT CONTENTS OF SA
;    7086  2
;    7087  2         EMS_TIM ();                                                    ! TIME
;    7088  1         end;


                                        .SBTTL  EMS.SA ERROR MESSAGE SUBROUTINES
000000  010146                EMS.SA:   MOV     R1,-(SP)                  ;              7050
000002  013701  000000G                 MOV     SA.REG,R1                 ;              705
000006  020127  177777                  CMP     R1,#-1
000012  001023                          BNE     1$
000014  012746  000000G                 MOV     #CRLF,-(SP)               ;              7062
000020  012746  000001                  MOV     #1,-(SP)
000024  010600                          MOV     SP,R0                     ; SP,*
000026  104415                          TRAP    15
000030  012716  000000G                 MOV     #ASTERISK,(SP)            ;              7063
```

```
000034  012746  000001                MOV     #1,-(SP)
000040  010600                        MOV     SP,R0                   ; SP,*
000042  104415                        TRAP    15
000044  013716  000000G               MOV     CNTR.ERR,(SP)           ;                           7064
000050  012746  000001                MOV     #1,-(SP)
000054  010600                        MOV     SP,R0                   ; SP,*
000056  104415                        TRAP    15
000060  000475                        BR      3$                      ;                           7061
000062  010100                  1$:   MOV     R1,R0                   ;                           7068
000064  042700  174000                BIC     #174000,R0
000070  020027  000026                CMP     R0,#26
000074  101030                        BHI     2$
000076  012746  000000G               MOV     #CRLF,-(SP)             ;                           7071
000102  012746  000001                MOV     #1,-(SP)
000106  010600                        MOV     SP,R0                   ; SP,*
000110  104415                        TRAP    15
000112  012716  000000G               MOV     #ASTERISK,(SP)          ;                           7072
000116  012746  000001                MOV     #1,-(SP)
000122  010600                        MOV     SP,R0                   ; SP,*
000124  104415                        TRAP    15
000126  013700  000000G               MOV     SA.REG,R0               ;                           7073
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$                      ;                           7070
000156  010100                  2$:   MOV     R1,R0                   ;                           7077
000160  042700  174000                BIC     #174000,R0
000164  162700  000620                SUB     #620,R0
000170  020027  000006                CMP     R0,#6
000174  101031                        BHI     4$
000176  012746  000000G               MOV     #CRLF,-(SP)             ;                           7080
000202  012746  000001                MOV     #1,-(SP)
000206  010600                        MOV     SP,R0                   ; SP,*
000210  104415                        TRAP    15
000212  012716  000000G               MOV     #ASTERISK,(SP)          ;                           7081
000216  012746  000001                MOV     #1,-(SP)
000222  010600                        MOV     SP,R0                   ; SP,*
000224  104415                        TRAP    15
000226  013700  000000G               MOV     SA.REG,R0               ;                           7082
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$:   TST     (SP)+                   ;                           7079
000256  000407                        BR      5$                      ;                           7077
000260  010146                  4$:   MOV     R1,-(SP)                ;                           7085
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              ;                       7087
000302  062706  000006                      ADD     #6,SP                   ;                       7057
000306  012601                              MOV     (SP)+,R1                ;                       7050
000310  000207                              RTS     PC

Routine Size:  101 words,     Routine Base:  $CODE$ + 7712
Maximum stack depth per invocation:  7 words

   7089  1
```

```
;   1   routine EMS_SBC1 (addr): novalue =                                      !MMM
;   1
;   1   !+
;   1   !       THIS ROUTINE PRINTS (EXTENDED) THE  "SB_CODE" (SUB CODE) IF
;   1   !       EITHER THE STATUS CODE (ST_CODE) OR THE SUB CODE IS NON-ZERO. (A
;   1   !       NON-ZERO SUB-CODE ALWAYS HAS SIGNIFICANCE, WHEREAS A ZERO SUB-CODE ONLY
;   1   !       HAS MEANING WITH A NON-ZERO STATUS CODE).
;   1   !-
;   1
;   2       begin
;   2
;   2       local
;   2
;   2           ST_CODE : word,                                             ! MMM
;   2           SB_CODE : word,                                             ! MMM
;   2           RP_ADDR : ref block [RP_LEN, word] field (RP FIELDS);       ! RETURN PACKET ADDRESS  MMM
;   2
;   2       RP_ADDR = .addr;                            ! MMM
;   2       ST_CODE = .RP_ADDR [STSCOD];                ! MMM
;   2       SB_CODE = .RP_ADDR [SUBCOD];                ! MMM
;   2
;   2       if (.ST_CODE or .SB_CODE) neq 0                                 ! PRINT SUB-CODE ONLY ON ERROR
;   2       then
;   3           begin
;   3           PRINTX (EX_SB);                                             ! SUB-CODE :
;   3
;   3           selectoneu  .ST_CODE of                                     !MMM  ADDITIONAL SUBCODES
;   3               set                                                     !MMM
;   3
;   4               [ST_SUC]:           begin                               ! SUCCESS SUB-CODES  MMM
;   4                                   if .SB_CODE lequ 16                 ! SUCCESS SUB CODES MMM
;   4                                   then
;   4                                   PRINTX (.TBL SUC [.SB_CODE]);
;   4                                   if .SB_CODE eqlu 32
;   4                                   then
;   4                                   PRINTX (.TBL_SUC [17]);
;   4                                   if .SB_CODE eqlu 64
;   4                                   then
;   4                                   PRINTX (.TBL_SUC [18]);             !MMM
;   3                                   end;                                !MMM
;   3
;   3               [ST_CMD]:           PRINTX (EX_SBO, .SB CODE / 8);      ! INVALID COMMAND
;   3
;   3               [ST_ABO]:           ;                                   ! COMMAND ABORTED
;   3
;   3               [ST_OFL]:           if .SB_CODF lequ 8                  ! UNIT OFFLINE
;   3                                   then
;   3                                       PRINTY (.TBL_OFL [.SB CODE]);
;   3
;   3               [ST_AVL]:           ;                                   ! UNIT AVAILABLE
;   3
;   3               [ST_MFE]:           if .SB CODE lequ 10                 ! MEDIA FORMAT ERROR
```

```
                                then
                                    PRINTX (.TBL MFE [.SB_CODE]);

                [ST_WPT]:       begin                                   !MMM
                                if .SB_CODE eqlu 8                      !MMM
                                then
                                    PRINTX (.TBL WPT [3]);               !MMM
                                if (.SB_CODE / 128) lequ 2               ! WRITE PROTECTED
                                then
                                    PRINTX (.TBL_WPT [(.SB CODE / 128)]); !MMM
                                end;                                     !MMM

                [ST_CMP]:       ;                                        ! COMPARE ERROR

                [ST_DAT]:       if .SB_CODE lequ 15                      ! DATA ERROR
                                then
                                    PRINTX (.TBL_DAT [.SB_CODE]);

                [ST_HST]:       if .SB_CODE lequ 4                       ! HOST ACCESS ERROR
                                then
                                    PRINTX (.TBL_HST [.SB_CODE]);

                [ST_CNT]:       if .SB_CODE lequ 3                       ! CONTROLLER ERROR
                                then
                                    PRINTX (.TBL_CNT [.SB CODE]);

                [ST_DRV]:       if .SB_CODE lequ 8                       ! DRIVE ERROR
                                then
                                    PRINTX (.TBL DRV [.SB CODE]);

                [otherwise]:    PRINTX (EX_SBO, .SB_CODE);               ! JUST PRINT SUB CODE IF NO MATCH
                tes;

!MMM    case .ST_CODE from ST_SUC to ST_DRV of
!MMM            set

!MMM            [ST_SUC]:       if .SB_CODE lequ 16                      ! SUCCESS SUB-CODES
!MMM                            then
!MMM                                PRINTX (.TBL_SUC [.SB_CODE]);

!MMM            [ST_CMD]:       PRINTX (EX_SBO, .SB CODE / 8);           ! INVALID COMMAND

!MMM            [ST_ABO]:       ;                                        ! COMMAND ABORTED

!MMM            [ST_OFL]:       if .SB CODE lequ 8                       ! UNIT OFFLINE
!MMM                            then
!MMM                                PRINTX (.TBL OFL [.SB CODE]);

!MMM            [ST_AVL]:       ;                                        ! UNIT AVAILABLE

!MMM            [ST_MFE]:       if .SB_CODE lequ 10                      ! MEDIA FORMAT ERROR
!MMM                            then
```

```
;   3   !MMM                                   PRINTX (.TBL MFE [.SB_CODE]);
;   3
;   3   !MMM        [ST_WPT]:              if (.SB CODE / 128) lequ 2              ! WRITE PROTECTED
;   3   !MMM                               then
;   3   !MMM                                   PRINTX (.TBL WPT [(.SB CODE / 128)]);
;   3
;   3   !MMM        [ST_CMP]:              ;                                        ! COMPARE ERROR
;   3
;   3   !MMM        [ST_DAT]:              if .SB CODE lequ 15                      ! DATA ERROR
;   3   !MMM                               then
;   3   !MMM                                   PRINTX (.TBL_DAT [.SB CODE]);
;   3
;   3   !MMM        [ST_HST]:              if .SB_CODE lequ 4                       ! HOST ACCESS ERROR
;   3   !MMM                               then
;   3   !MMM                                   PRINTX (.TBL HST [.SB_CODE]);
;   3
;   3   !MMM        [ST_CNT]:              if .SB_CODE lequ 3                       ! CONTROLLER ERROR
;   3   !MMM                               then
;   3   !MMM                                   PRINTX (.TBL CNT [.SB_CODE]);
;   3
;   3   !MMM        [ST_DRV]:              if .SB_CODE lequ 8                       ! DRIVE ERROR
;   3   !MMM                               then
;   3   !MMM                                   PRINTX (.TBL DRV [.SB_CODE]);
;   3
;   3   !MMM        [outrange]:            PRINTX (EX_SBO, .SB CODF),               ! JUST PRINT SUB CODE IF NO MATCH
;   3   !MMM        tes;
;   3
;   2           end;
;   2
;   1       end;


                                           .SBTTL  EMS.SBC1 ERROR MESSAGE SUBROUTINES
000000  004137  000000G            EMS.SBC1:
                                           JSR     R1,$SAVE2                ;                              7090
000004  016600  000010                     MOV     10(SP),R0                ; ADDR,RP.ADDR                 7108
000010  116002  000016                     MOVB    16(R0),R2                ; *(RP.ADDR),ST.CODE           7109
000014  042702  177740                     BIC     #177740,R2               ; *,ST.CODE
000020  016001  000016                     MOV     16(R0),R1                ; *(RP.ADDR),SB.CODE           7110
000024  006201                             ASR     R1                       ; SB.CODE
000026  006201                             ASR     R1                       ; SB.CODE
000030  006201                             ASR     R1                       ; SB.CODE
000032  006201                             ASR     R1                       ; SB.CODE
000034  006201                             ASR     R1                       ; SB.CODE
000036  042701  174000                     BIC     #174000,R1               ; *,SB.CODE
000042  010100                             MOV     R1,R0                    ; SB.CODE,*                    7112
000044  050200                             BIS     R2,R0                    ; ST.CODE,*
000046  001001                             BNE     1$
000050  000207                             RTS     PC
000052  012746  000000G            1$:     MOV     #EX.SB,-(SP)             ;                              7115
000056  012746  000001                     MOV     #1,-(SP)
000062  010600                             MOV     SP,R0                    ; SP,*
000064  104415                             TRAP    15
```

```
000066 005702                    TST     R2                      ; ST.CODE          7120
000070 001040                    BNE     4$
000072 020127 000020             CMP     R1,#20                  ; SB.CODE,*        7121
000076 101011                    BHI     2$
000100 010100                    MOV     R1,R0                   ; SB.CODE,*        7123
000102 006300                    ASL     R0
000104 016016 000000'            MOV     TBL.SUC(R0),(SP)
000110 012746 000001             MOV     #1,-(SP)
000114 010600                    MOV     SP,R0                   ; SP,*
000116 104415                    TRAP    15
000120 005726                    TST     (SP)+
000122 020127 000040     2$:     CMP     R1,#40                  ; SB.CODE,*        7124
000126 001007                    BNE     3$
000130 013716 000042'            MOV     TBL.SUC+42,(SP)         ;                  7126
000134 012746 000001             MOV     #1,-(SP)
000140 010600                    MOV     SP,R0                   ; SP,*
000142 104415                    TRAP    15
000144 005726                    TST     (SP)+
000146 020127 000100     3$:     CMP     R1,#100                 ; SB.CODE,*        7127
000152 001030                    BNE     5$
000154 013716 000044'            MOV     TBL.SUC+44,(SP)         ;                  7129
000160 012746 000001             MOV     #1,-(SP)
000164 010600                    MOV     SP,R0                   ; SP,*
000166 104415                    TRAP    15
000170 000565                    BR      12$
000172 020227 000001     4$:     CMP     R2,#1                   ; ST.CODE,*        7132
000176 001020                    BNE     6$
000200 010116                    MOV     R1,(SP)                 ; SB.CODE,*
000202 012746 000010             MOV     #10,-(SP)
000206 004737 000000G            JSR     PC,BL$DIV
000212 010016                    MOV     R0,(SP)
000214 012746 000000G            MOV     #E.SBO,-(SP)
000220 012746 000002             MOV     #2,-(SP)
000224 010600                    MOV     SP,R0                   ; SP,*
000226 104415                    TRAP    15
000230 062706 000006             ADD     #6,SP
000234 000137 011110'    5$:     JMP     17$                     ;                  7117
000240 020227 000002     6$:     CMP     R2,#2                   ; ST.CODE,*        7134
000244 001773                    BEQ     5$                      ;                  7117
000246 020227 000003             CMP     R2,#3                   ; ST.CODE,*        7136
000252 001014                    BNE     7$
000254 020127 000010             CMP     R1,#10                  ; SB.CODE,*
000260 101365                    BHI     5$
000262 010100                    MOV     R1,R0                   ; SB.CODE,*        7138
000264 006300                    ASL     R0
000266 016016 000046'            MOV     TBL.OFL(R0),(SP)
000272 012746 000001             MOV     #1,-(SP)
000276 010600                    MOV     SP,R0                   ; SP,*
000300 104415                    TRAP    15
000302 000556                    BR      15$
000304 020227 000004     7$:     CMP     R2,#4                   ; ST.CODE,*        7140
000310 001565                    BEQ     17$                     ;                  7117
000312 020227 000005             CMP     R2,#5                   ; ST.CODE,*        7142
```

```
000316  001014                         BNE     8$
000320  020127  000012                 CMP     R1,#12                  ; SB.CODE,*
000324  101157                         BHI     17$
000326  010100                         MOV     R1,R0                   ; SB.CODE,*                7144
000330  006300                         ASL     R0
000332  016016  000070'                MOV     TBL.MFE(R0),(SP)
000336  012746  000001                 MOV     #1,-(SP)
000342  010600                         MOV     SP,R0                   ; SP,*
000344  104415                         TRAP    15
000346  000534                         BR      15$
000350  020227  000006         8$:     CMP     R2,#6                   ; ST.CODE,*                7146
000354  001033                         BNE     10$
000356  020127  000010                 CMP     R1,#10                  ; SB.CODE,*                7147
000362  001007                         BNE     9$
000364  013716  000124'                MOV     TBL.WPT+6,(SP)          ;                          7149
000370  012746  000001                 MOV     #1,-(SP)
000374  010600                         MOV     SP,R0                   ; SP,*
000376  104415                         TRAP    15
000400  005726                         TST     (SP)+
000402  010116                 9$:     MOV     R1,(SP)                 ; SB.CODE,*                7150
000404  012746  000200                 MOV     #200,-(SP)
000410  004737  000000G                JSR     PC,BL$DIV
000414  005726                         TST     (SP)+
000416  020027  000002                 CMP     R0,#2
000422  101120                         BHI     17$
000424  006300                         ASL     R0                      ;                          7152
000426  016016  000116'                MOV     TBL.WPT(R0),(SP)
000432  012746  000001                 MOV     #1,-(SP)
000436  010600                         MOV     SP,R0                   ; SP,*
000440  104415                         TRAP    15
000442  000476                         BR      15$
000444  020227  000007         10$:    CMP     R2,#7                   ; ST.CODE,*                7155
000450  001505                         BEQ     17$                     ;                          7117
000452  020227  000010                 CMP     R2,#10                  ; ST.CODE,*                7157
000456  001014                         BNE     11$
000460  020127  000017                 CMP     R1,#17                  ; SB.CODE,*
000464  101077                         BHI     17$
000466  010100                         MOV     R1,R0                   ; SB.CODE,*                7159
000470  006300                         ASL     R0
000472  016016  000126'                MOV     TBL.DAT(R0),(SP)
000476  012746  000001                 MOV     #1,-(SP)
000502  010600                         MOV     SP,R0                   ; SP,*
000504  104415                         TRAP    15
000506  000454                         BR      15$
000510  020227  000011         11$:    CMP     R2,#11                  ; ST.CODE,*                7161
000514  001014                         BNE     13$
000516  020127  000004                 CMP     R1,#4                   ; SB.CODE,*
000522  101060                         BHI     17$
000524  010100                         MOV     R1,R0                   ; SB.CODE,*                7163
000526  006300                         ASL     R0
000530  016016  000166'                MOV     TBL.HST(R0),(SP)
000534  012746  000001                 MOV     #1,-(SP)
000540  010600                         MOV     SP,R0                   ; SP,*
```

```
000542  104415                      TRAP    15
000544  000435             12$:     BR      15$
000546  020227  000012     13$:     CMP     R2,#12                  ; ST.CODE.*                 7165
000552  001014                      BNE     14$
000554  020127  000003              CMP     R1,#3                   ; SB.CODE.*
000560  101041                      BHI     17$
000562  010100                      MOV     R1,R0                   ; SB.CODE.*                 7167
000564  006300                      ASL     R0
000566  016016  000200'             MOV     TBL.CNT(R0),(SP)
000572  012746  000001              MOV     #1,-(SP)
000576  010600                      MOV     SP,R0                   ; SP,*
000600  104415                      TRAP    15
000602  000416                      BR      15$
000604  020227  000013     14$:     CMP     R2,#13                  ; ST.CODE.*                 7169
000610  001015                      BNE     16$
000612  020127  000010              CMP     R1,#10                  ; SB.CODE.*
000616  101022                      BHI     17$
000620  010100                      MOV     R1,R0                   ; SB.CODE.*                 7171
000622  006300                      ASL     R0
000624  016016  000210'             MOV     TBL.DRV(R0),(SP)
000630  012746  000001              MOV     #1,-(SP)
000634  010600                      MOV     SP,R0                   ; SP,*
000636  104415                      TRAP    15
000640  005726             15$:     TST     (SP)+
000642  000410                      BR      17$
000644  010116             16$:     MOV     R1,(SP)                 ;                           7117
000646  012746  000000G             MOV     #EX.SBO, (SP)           ; SB.CODE.*                 7173
000652  012746  000002              MOV     #2,-(SP)
000656  010600                      MOV     SP,R0                   ; SP,*
000660  104415                      TRAP    15
000662  022626                      CMP     (SP)+,(SP)+
000664  022626             17$:     CMP     (SP)+,(SP)+             ;                           7114
000666  000207                      RTS     PC                      ;                           7090

; Routine Size:  220 words,     Routine Base:  $CODE$ + 10224
; Maximum stack depth per invocation:  10 words


;       7226    1
;       7227    1
```

```
;  1
;  1   routine EMS_CMD1 (addr) : novalue =                               ! MMM
;  1
;  1   !+
;  1   !       THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
;  1   !       PRESENT) OF THE RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
;  1   !       INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
;  1   !
;  1   !-
;  1
;  2       begin
;  2
;  2       local
;  2           RP_ADDR : ref block [RP_LEN, word] field (RP FIELDS);           ! RETURN PACKET ADDRESS  MMM
;  2
;  2       RP_ADDR = .addr;                                               ! MMM
;  2       PRINTX (EX_CMD);                                               ! "COMMAND: "
;  2
;  2       selectoneu (.RP_ADDR [ENDCOD] and OP_MSK) of
;  2           set
;  2
;  2           [OP_ONL]:       PRINTX (EX_ONL);                           ! ONLINE
;  2
;  2           [OP_ACC]:       PRINTX (EX_ACC);                           ! ACCESS
;  2
;  3           [OP_RD]:        begin
;  3                           PRINTX (EX_RD);                            ! READ
;  3
;  3   !MMM                    if .RP_ADDR [CMDMOD] neq 0
;  3                           if (.RP_ADDR [CMDMOD] and MD_CMP) neq 0 !MMM
;  3                           then
;  3                               PRINTX (EX_CMP);                       ! COMPARE
;  3
;  2                           end;
;  2
;  3           [OP_WRT]:       begin
;  3                           PRINTX (EX_WRT);                           ! WRITE
;  3
;  3   !MMM                    if .RP_ADDR [CMDMOD] neq 0
;  3                           if (.RP_ADDR [CMDMOD] and MD_CMP) neq 0   !MMM
;  3                           then
;  3                               PRINTX (EX_CMP);                       ! COMPARE
;  3
;  2                           end;
;  2
;  2           [otherwise]:    PRINTX (EX_OP, .RP_ADDR [ENDCOD]);         ! ENDCODE VALUE IF NO MATCH
;  2           tes;
;  2
;  1       end;                                                           ! ROUTINE EMS_CMD1

                                             .SBTTL  EMS.CMD1 ERROR MESSAGE SUBROUTINES
000000  004137  000000G                EMS.CMD1:
```

```
                                          JSR     R1,$SAVE2               ;                       7229
000004  016601  000010                    MOV     10(SP),R1               ; ADDR,RP.ADDR          7243
000010  012746  000000G                   MOV     #EX.CMD,-(SP)           ;                       7244
000014  012746  000001                    MOV     #1,-(SP)
000020  010600                            MOV     SP,R0                   ; SP,*
000022  104415                            TRAP    15
000024  116102  000014                    MOVB    14(R1),R2               ; *(RP.ADDR),*          7246
000030  042702  177600                    BIC     #177600,R2
000034  020227  000011                    CMP     R2,#11                  ;                       7249
000040  001007                            BNE     1$
000042  012716  000000G                   MOV     #EX.ONL,(SP)
000046  012746  000001                    MOV     #1,-(SP)
000052  010600                            MOV     SP,R0                   ; SP,*
000054  104415                            TRAP    15
000056  000462                            BR      5$
000060  020227  000020          1$:       CMP     R2,#20                  ;                       7251
000064  001007                            BNE     2$
000066  012716  000000G                   MOV     #EX.ACC,(SP)
000072  012746  000001                    MOV     #1,-(SP)
000076  010600                            MOV     SP,R0                   ; SP,*
000100  104415                            TRAP    15
000102  000450                            BR      5$
000104  020227  000041          2$:       CMP     R2,#41                  ;                       7253
000110  001021                            BNE     3$
000112  012716  000000G                   MOV     #EX.RD,(SP)             ;                       7254
000116  012746  000001                    MOV     #1,-(SP)
000122  010600                            MOV     SP,R0                   ; SP,*
000124  104415                            TRAP    15
000126  032761  040000  000012            BIT     #40000,12(R1)           ; *,*(RP.ADDR)          7257
000134  001433                            BEQ     5$
000136  012716  000000G                   MOV     #EX.CMP,(SP)            ;                       7259
000142  012746  000001                    MOV     #1,-(SP)
000146  010600                            MOV     SP,R0                   ; SP,*
000150  104415                            TRAP    15
000152  000423                            BR      4$
000154  020227  000042          3$:       CMP     R2,#42                  ;                       7263
000160  001023                            BNE     6$
000162  012716  000000G                   MOV     #EX.WRT,(SP)            ;                       7264
000166  012746  000001                    MOV     #1,-(SP)
000172  010600                            MOV     SP,R0                   ; SP,*
000174  104415                            TRAP    15
000176  032761  040000  000012            BIT     #40000,12(R1)           ; *,*(RP.ADDR)          7267
000204  001407                            BEQ     5$
000206  012716  000000G                   MOV     #EX.CMP,(SP)            ;                       7269
000212  012746  000001                    MOV     #1,-(SP)
000216  010600                            MOV     SP,R0                   ; SP,*
000220  104415                            TRAP    15
000222  005726                  4$:       TST     (SP)+
000224  005726                  5$:       TST     (SP)+                   ;                       7263
000226  000412                            BR      7$                      ;                       7246
000230  005016                  6$:       CLR     (SP)                    ;                       7273
000232  116116  000014                    MOVB    14(R1),(SP)             ; *(RP.ADDR),*
000236  012746  000000G                   MOV     #EX.OP,-(SP)
```

```
000242  012746  000002              MOV     #2,-(SP)
000246  010600                      MOV     SP,R0                   ; SP,*
000250  104415                      TRAP    15
000252  022626                      CMP     (SP)+,(SP)+
000254  022626              7$:     CMP     (SP)+,(SP)+             ;                               7238
000256  000207                      RTS     PC                      ;                               7229

; Routine Size:  88 words,      Routine Base:  $CODE$ + 11114
; Maximum stack depth per invocation:  9 words


;    7277  1
```

```
!MMMGLOBAL ROUTINE EMS_DBN : NOVALUE =                                     !ZZZ
!+                                                                         !ZZZ
!       THIS ROUTINE PRINTS THE PRESENT DBN                                !ZZZ
!                                                                          !ZZZ
!       IMPLICIT IMPUTS:                                                   !ZZZ
!               CST_ADDR - ADDRESS OF CONTROLLER STATUS TABLE              !ZZZ
!-                                                                         !ZZZ
                                                                           !ZZZ
!BEGIN                                                                     !ZZZ
!PRINTB (XX13, .CDISK);                              !"DISK XXX'           !ZZZ
!PRINTB (XX23  .CST_ADDR [.CUOFF + OF_DBN, D_DBN], .CST_AD'                 !ZZZ
!                        [.CUOFF + OF_DBN, D_DBN]);     !"DBN, XXXXXX."    !ZZZ
!PRINTB (XX32, .S_DUPPKT - 2);                       !PRINT BYTE COUNT     !ZZZ
!PRINTB (XX33, .S_PATTERN);                          !PRINT THE PATTERN    !ZZZ
!PRINTB (XX34, .(DUPPKT + .S_DUPPKT), .(DUPPKT + .S_DUPPKT));   !PRINT THE WORD READ    !ZZZ
!EMS_BLK (DUPPKT +2, 256);                           !PRINT WHOLE BLOCK READ !ZZZ
!END;                                                !IN OCTAL             !ZZZ
!MMM
```

```
;   1
;   1      GLOBAL ROUTINE EMS_BLK (ADDR, LENGTH) : NOVALUE =                     !ZZZ
;   1                                                                            !ZZZ
;   1      !+                                                                    !ZZZ
;   1      !       THIS ROUTINE WILL PRINTX A BLOCK OF MEMORY, WHICH IS 'LENGTH'  !ZZZ
;   1      !       WORDS LONG STARTING AT ADDRESS 'ADDR'. PRINTING IS DONE IN OCTAL!ZZZ
;   1      !       8 WDS TO A LINE.                                              !ZZZ
;   1      !-                                                                    !ZZZ
;   1                                                                            !ZZZ
;   2      BEGIN                                                                 !ZZZ
;   2      LITERAL                                                               !ZZZ
;   2              MASK = %o'7';                                                 !ZZZ
;   2                                                                            !ZZZ
;   2      PRINTX (CRLF);                                                        !ZZZ
;   2              INCR COUNT FROM 1 TO .LENGTH DO          !FOR EACH WD TO PRINT  !ZZZ
;   3                      BEGIN                                                 !ZZZ
;   3                      IF ((.COUNT - 1) AND MASK) EQL 0 !IF START OF NEW LINE !ZZZ
;   3                      THEN                                                  !ZZZ
;   3                              PRINTX (SPACE4);         !PRINT 4 BLANKS       !ZZZ
;   3                                                                            !ZZZ
;   3                      PRINTX (EX_WRD, ..ADDR);         !PRINTX A WORD        !ZZZ
;   3                      ADDR = .ADDR +2;                 !TO NEXT ADDRESS      !ZZZ
;   3                                                                            !ZZZ
;   4                      IF (((.COUNT AND MASK) EQL 0) OR !END OF LINE OR       !ZZZ
;   4                          (.COUNT EQL .LENGTH))        !WHEN DONE            !ZZZ
;   3                      THEN                                                  !ZZZ
;   3                              PRINTX (CRLF);           !PRINT CR LF          !ZZZ
;   2                      END;                                                  !ZZZ
;   1              END;                                                          !ZZZ

                                                .SBTTL  EMS.BLK ERROR MESSAGE SUBROUTINES
000000  010146                          EMS.BLK::
                                                MOV     R1,-(SP)                ;                       7298
000002  012746  000000G                         MOV     #CRLF,-(SP)             ;                       7310
000006  012746  000001                          MOV     #1,-(SP)
000012  010600                                  MOV     SP,R0                   ; SP,*
000014  104415                                  TRAP    15
000016  005001                                  CLR     R1                      ; COUNT                 7311
000020  000445                                  BR      5$
000022  010100                          1$:     MOV     R1,R0                   ; COUNT,*               7313
000024  005300                                  DEC     R0
000026  032700  000007                          BIT     #7,R0
000032  001007                                  BNE     2$
000034  012716  000000G                         MOV     #SPACE4,(SP)            ;                       7315
000040  012746  000001                          MOV     #1,-(SP)
000044  010600                                  MOV     SP,R0                   ; SP,*
000046  104415                                  TRAP    15
000050  005726                                  TST     (SP)+
000052  017616  000012                  2$:     MOV     @12(SP),(SP)            ; ADDR,*                7317
000056  012746  000000G                         MOV     #EX.WRD,-(SP)
000062  012746  000002                          MOV     #2,-(SP)
00006   010600                                  MOV     SP,R0                   ; SP,*
```

```
000070  104415                               TRAP    15
000072  062766  000002  000016               ADD     #2.16(SP)                ; *.ADDR              7318
000100  032701  000007                       BIT     #7,R1                    ; *.COUNT             7320
000104  001403                               BEQ     3$
000106  020166  000014                       CMP     R1,14(SP)                ; COUNT,LENGTH        7321
000112  001007                               BNE     4$
000114  012716  000000G              3$:     MOV     #CRLF,(SP)               ;                     7323
000120  012746  000001                       MOV     #1,-(SP)
000124  010600                               MOV     SP,R0                    ; SP,*
000126  104415                               TRAP    15
000130  005726                               TST     (SP)+
000132  022626                       4$:     CMP     (SP)+,(SP)+              ;                     7312
000134  005201                       5$:     INC     R1                       ; COUNT               7311
000136  020166  000010                       CMP     R1,10(SP)                ; COUNT,LENGTH
000142  003727                               BLE     1$
000144  022626                               CMP     (SP)+,(SP)+              ;                     7306
000146  012601                               MOV     (SP)+,R1                 ;                     7298
000150  000207                               RTS     PC

; Routine Size:  53 words,     Routine Base:  $CODE$ + 11374
; Maximum stack depth per invocation:  8 words


;    7326  1
;    7327  1
```

```
;     7328  1
;     7329  1    routine EMS_LBN1 (addr) : novalue =                                  ! MMM
;     7330  1
;     7331  1    !+
;     7332  1    !       THIS ROUTINE PRINTS (EXTENDED) ONE OF TWO BLOCK NUMBERS APPEARING IN
;     7333  1    !       THE RETURN PACKET. NORMALLY, THE LBN FIELD IS PRINTED; THIS
;     7334  1    !       FIELD WAS COPIED INTO THE RETURN PACKET FROM THE ASSOCIATED COMMAND
;     7335  1    !       PACKET. HOWEVER, IF THE "FLAGS" FIELD OF THE RETURN PACKET
;     7336  1    !       INDICATES "BAD BLOCK REPORTED", THEN THE "FIRST BAD BLOCK" FIELD IS
;     7337  1    !       PRINTED.
;     7338  1    !
;     7339  1    !-
;     7340  1
;     7341  2        begin
;     7342  2        local
;     7343  2            RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);      ! RETURN PACKET ADDRESS  MMM
;     7344  2
;     7345  2        RP_ADDR = .addr;                                               ! MMM
;     7346  2
;     7347  2        if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                 ! IF NO BAD BLOCK FOUND
;     7348  3           (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;     7349  2        then
;     7350  2            PRINTX (EX_LBN2, .RP_ADDR [LBN_HI], .RP_ADDR [LBN_LO]);                      !ZZZ
;     7351  2
;     7352  2        if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                 ! IF BAD BLOCKS FOUND AND REPLACED
;     7353  3           (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;     7354  2        then
;     7355  2            PRINTX (EX_BBU2, .RP_ADDR [BBLK_HI], .RP_ADDR [BBLK_LO]);                    !ZZZ
;     7356  2
;     7357  2        if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                     ! IF HOST REPLACEABLE BAD BLOCK FOUND
;     7358  3           (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;     7359  2        then
;     7360  2            PRINTX (EX_BB2, .RP_ADDR [BBLK_HI], .RP_ADDR [BBLK_LO]);                     !ZZZ
;     7361  2
;     7362  2        if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                     ! IF MORE THAN 1 HOST REPLACEABLE BAD BLOCK FOUND
;     7363  3           (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;     7364  2        then
;     7365  2            PRINTX (EX_BB12, .RP_ADDR [BBLK_HI], .RP_ADDR [BBLK_LO]);                    !ZZZ
;     7366  1        end;


                                        .SBTTL  EMS.LBN1 ERROR MESSAGE SUBROUTINES
000000  004137  000000G         EMS.LBN1:
                                        JSR     R1,$SAVE3                  ;                       7329
000004  016601  000012                  MOV     12(SP),R1                  ; ADDR,RP.ADDR          7345
000010  005002                          CLR     R2                         ;                       7347
000012  156102  000015                  BISB    15(R1),R2                  ; *(RP.ADDR),*
000016  005003                          CLR     R3
000020  105702                          TSTB    R2
000022  100002                          BPL     1$
000024  005203                          INC     R3
000026  000417                          BR      2$
000030  032702  000100          1$:     BIT     #100,R2                    ;                       7348
```

```
000034  001014                         BNE     2$
000036  016146  000050                 MOV     50(R1),-(SP)        ; *(RP.ADDR),*         7350
000042  016146  000052                 MOV     52(R1),-(SP)        ; *(RP.ADDR),*
000046  012746  000000G                MOV     #EX.LBN2,-(SP)
000052  012746  000003                 MOV     #3,-(SP)
000056  010600                         MOV     SP,R0               ; SP,*
000060  104415                         TRAP    15
000062  0627.6  000010                 ADD     #10,SP
000066  032703  000001         2$:     BIT     #1,R3               ;                      7352
000072  001022                         BNE     4$
000074  032702  000100                 BIT     #100,R2             ;                      7353
000100  001414                         BEQ     3$
000102  016146  000040                 MOV     40(R1),-(SP)        ; *(RP.ADDR),*         7355
000106  016146  000042                 MOV     42(R1),-(SP)        ; *(RP.ADDR),*
000112  012746  000000G                MOV     #EX.BBU2,(SP)
000116  012746  000003                 MOV     #3,-(SP)
000122  010600                         MOV     SP,R0               ; SP,*
000124  104415                         TRAP    15
000126  062706  C00010                 ADD     #10,SP
000132  032703  000001         3$:     BIT     #1,R3               ;                      7357
000136  001417                         BEQ     5$
000140  032702  000100         4$:     BIT     #100,R2             ;                      7358
000144  001014                         BNE     5$
000146  016146  000040                 MOV     40(R1),(SP)         ; *(RP.ADDR),*         7360
000152  016146  000042                 MOV     42(R1),-(SP)        ; *(RP.ADDR),*
000156  012746  000000G                MOV     #EX.BB2,-(SP)
000162  012746  000003                 MOV     #3,(SP)
000166  010600                         MOV     SP,R0               ; SP,*
000170  104415                         TRAP    15
000172  062706  000010                 ADD     #10,SP
000176  006003                 5$:     ROR     R3                  ;                      7362
000200  103017                         BCC     6$
000202  032702  000100                 BIT     #100,R2             ;                      7363
000206  001414                         BEQ     6$
000210  016146  000040                 MOV     40(R1),(SP)         ; *(RP.ADDR),*         7365
000214  016146  000042                 MOV     42(R1),(SP)         ; *(RP.ADDR),*
000220  012746  000000G                MOV     #EX.BB12,-(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                 6$:     RTS     PC                  ;                      7329

; Routine Size:  81 words,      Routine Base:  $CODE$ + 11546
; Maximum stack depth per invocation:  10 words


;    7367  1
;    7368  1
;    7369  1
;    7370  1
```

```
;    7371  1
;    7372  1     routine EMS_BC1 (addr) : novalue =              ! MMM
;    7373  1
;    7374  1     !+
;    7375  1     !        THIS ROUTINE PRINTS (EXTENDED) BOTH BYTE COUNT FIELDS OF THE
;    7376  1     !        RETURN PACKET: THE BYTE COUNT FROM THE COMMAND PACKET AND THE
;    7377  1     !        ACTUAL NUMBER OF BYTES TRANSFERRED (FROM THE RESPONSE PACKET).
;    7378  1     !
;    7379  1     !-
;    7380  1
;    7381  2         begin
;    7382  2         local
;    7383  2             RP_ADDR : ref block [RP_LEN, word] field (RP FIELDS);          ! RETURN PACKET ADDRESS  MMM
;    7384  2
;    7385  2         RP_ADDR = .addr;                              ! MMM
;    7386  2         PRINTX (EX_CBC, .RP_ADDR [CBCNT LO]);       ! "BYTE COUNT IN COMMAND: XXXXX."
;    7387  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);         ! "ACTUAL # OF BYTES TRANSFERRED: XXXXX."
;    7388  1         end;                                        ! ROUTINE EMS_BC1


                                        .SBTTL  EMS.BC1 ERROR MESSAGE SUBROUTINES
000000  010146                 EMS.BC1:MOV      R1,-(SP)                    ;                            7372
000002  016601  000004                 MOV      4(SP),R1                    ; ADDR,RP.ADDR               7385
000006  016146  000044                 MOV      44(R1),-(SP)                ; *(RP.ADDR),*               7386
000012  012746  000000G                MOV      #EX.CBC,-(SP)
000016  012746  000002                 MOV      #2,-(SP)
000022  010600                         MOV      SP,R0                       ; SP,*
000024  104415                         TRAP     15
000026  016116  000020                 MOV      20(R1),(SP)                 ; *(RP.ADDR),*               7387
000032  012746  000000G                MOV      #EX.BC,-(SP)
000036  012746  000002                 MOV      #2,-(SP)
000042  010600                         MOV      SP,R0                       ; SP,*
000044  104415                         TRAP     15
000046  062706  000012                 ADD      #12,SP                      ;                            7381
000052  012601                         MOV      (SP)+,R1                    ;                            7372
000054  000207                         RTS      PC

; Routine Size:  23 words.      Routine Base:  $CODE$ + 12010
; Maximum stack depth per invocation:  8 words


;    7389  1
```

```
;      7390  1
;      7391  1     routine EMS_BD1 (addr) : novalue =                                   ! MMM
;      7392  1
;      7393  1     !+
;      7394  1     !
;      7395  1     !       THIS ROUTINE PRINTS (EXTENDED) THE TWO WORD I/O BUFFER DESCRIPTOR
;      7396  1     !       APPEARING IN THE RETURN PACKET.
;      7397  1     !
;      7398  2         begin
;      7399  2
;      7400  2         local
;      7401  2             RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);              ! RETURN PACKET ADDRESS  MMM
;      7402  2
;      7403  2         RP_ADDR = .addr;                                    ! MMM
;      7404  2         PRINTX (EX_BD, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);      ! "I/O BUFFER DESCRIPTOR: XXXXXX XXXXXX"
;      7405  2
;      7406  1         end;


                                                .SBTTL  EMS.BD1 ERROR MESSAGE SUBROUTINES
000000  016600  000002                  EMS.BD1:MOV     2(SP),R0                        ; ADDR,RP.ADDR                   7403
000004  016046  000024                          MOV     24(R0),-(SP)                    ; *(RP.ADDR),*                   7404
000010  016046  000026                          MOV     26(R0),-(SP)                    ; *(RP.ADDR),*
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                          ;                                7398
000034  000207                                  RTS     PC                              ;                                7391

; Routine Size:  15 words,      Routine Base:  $CODE$ + 12066
; Maximum stack depth per invocation:  6 words


;     7407  1
```

```
;  1
;  1    global routine EMS_R2 (addr) : novalue =                            ! MMM
;  1
;  1
;  1    !+
;  1    !       THIS ROUTINE IS RESPONSIBLE FOR PRINTING (EXTENDED) THE RELEVANT FIELDS
;  1    !       OF THE RETURN PACKET.
;  1    !-
;  1
;  1
;  2        begin
;  2
;  2        local
;  2            RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);       ! RETURN PACKET ADDRESS  MMM
;  2
;  2
;  2        RP_ADDR = .addr;                                  ! MMM
;  2        EMS_SBC1 (.RP_ADDR);                              ! MMM
;  2        EMS_CMD1 (.RP_ADDR);                              ! COMMAND (AND MODIFIER) MMM
;  2
;  2        if (.RP_ADDR [ENDCOD] and OP_MSK) neq OP_ONL
;  2
;  2        then
;  2            EMS_LBN1 (.RP_ADDR);                                 ! LBN OR BAD BLOCK NUMBER  MMM
;  2
;  2        if ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD) or
;  3           ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_WRT)
;  3
;  2        then
;  3            begin
;  3            EMS_BC1 (.RP_ADDR);                                  ! BYTE COUNTS MMM
;  3            EMS_BD1 (.RP_ADDR);                                  ! I/O BUFFER DESCRIPTOR MMM
;  2            end;
;  2
;  2        EMS_TIM ();                                       ! TIME
;  1        end;                                              ! ROUTINE EMS R2

                                .SBTTL  EMS.R2 ERROR MESSAGE SUBROUTINES
000000  010146                  EMS.R2::MOV     R1,-(SP)                ;                       7409
000002  016601  000004                  MOV     4(SP),R1                ; ADDR,RP.ADDR          7424
000006  010146                          MOV     R1,-(SP)                ; RP.ADDR,*             7425
000010  004737  010224'                 JSR     PC,EMS.SBC1
000014  010116                          MOV     R1,(SP)                 ; RP.ADDR,*             7426
000016  004737  011114'                 JSR     PC,EMS.CMD1
000022  116100  000014                  MOVB    14(R1),R0               ; *(RP.ADDR),*          7428
000026  042700  177600                  BIC     #177600,R0
000032  020027  000011                  CMP     R0,#11
000036  001403                          BEQ     1$
000040  010116                          MOV     R1,(SP)                 ; RP.ADDR,*             7431
000042  004737  011546'                 JSR     PC,EMS.LBN1
000046  116100  000014          1$:     MOVB    14(R1),R0               ; *(RP.ADDR),*          7433
000052  042700  177600                  BIC     #177600,R0
```

```
000056  020027  000041                  CMP     R0,#41
000062  001407                          BEQ     2$
000064  116100  000014                  MOVB    14(R1),R0               ; *(RP.ADDR),*                  7434
000070  042700  177600                  BIC     #177600,R0
000074  020027  000042                  CMP     R0,#42
000100  001006                          BNE     3$
000102  010116                  2$:     MOV     R1,(SP)                 ; RP.ADDR,*                     7438
000104  004737  012010'                 JSR     PC,EMS.BC1
000110  010116                          MOV     R1,(SP)                 ; RP.ADDR,*                     7439
000112  004737  012066'                 JSR     PC,EMS.BD1
000116  004737  000000V         3$:     JSR     PC,EMS.TIM              ;                               7442
000122  005726                          TST     (SP)+                   ;                               7418
000124  012601                          MOV     (SP)+,R1                ;                               7409
000126  000207                          RTS     PC

; Routine Size:  44 words,      Routine Base:  $CODE$ + 12124
; Maximum stack depth per invocation:  3 words


;       7444  1
```

```
;     7445  1       global routine EMS_R1 (addr) : novalue =
;     7446  1
;     7447  1       !+
;     7448  1       !        THIS  ROUTINE  IS  CALLED  TO  PRINT  THE  ENTIRE CONTENTS OF THE
;     7449  1       !        RETURN PACKET DESIGNATED.  HOWEVER, THE PRINTING WILL ONLY
;     7450  1       !        OCCUR IF EXTENDED ERROR PRINTING IS ENABLED.
;     7451  1       !-
;     7452  1
;     7453  2           begin
;     7454  2           local
;     7455  2               RP_ADDR : ref block [RP_LEN, word] field (RP FIELDS);          ! RETURN PACKET ADDRESS  MMM
;     7456  2
;     7457  2           RP_ADDR = .addr;                                  ! MMM
;     7458  2           PRINTX (EX_RP);                                   ! "CONTENTS OF RETURN PACKET:"
;     7459  2           EMS_BLK (.RP_ADDR, RP_LEN);                       ! PRINT BLOCK OF WORDS
;     7460  1           end;


                                               .SBTTL  EMS.R1 ERROR MESSAGE SUBROUTINES
000000  010146                          EMS.R1::MOV     R1,-(SP)                           ;                          7445
000002  016601  000004                          MOV     4(SP),R1                           ; ADDR,RP.ADDR             7457
000006  012746  000000G                         MOV     #EX.RP,-(SP)                       ;                          7458
000012  012746  000001                          MOV     #1,-(SP)
000016  010600                                  MOV     SP,R0                              ; SP,*
000020  104415                                  TRAP    15
000022  010116                                  MOV     R1,(SP)                            ; RP.ADDR,*                7459
000024  012746  000026                          MOV     #26,-(SP)
000030  004737  011374'                         JSR     PC,EMS.BLK
000034  062706  000006                          ADD     #6,SP
000040  012601                                  MOV     (SP)+,R1                           ;                          7453
000042  000207                                  RTS     PC                                 ;                          7445

; Routine Size:  18 words.      Routine Base:  $CODE$ + 12254
; Maximum stack depth per invocation:  5 words
```

```
1   global routine EMS_EL (index) : novalue =
1
1   !+
1   !       THIS ROUTINE IS CALLED FROM 'SEQUEN' AND 'DATAGM' AND PRINTS THE CONTENTS OF THE
1   !       ERROR-LOG PACKET
1   !-
1
2       begin
2
2       local
2           ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
2           REASON : word,
2           DISK_NUM : byte,
2           ELOG_CODE : byte,
2           ELOG_SUB : word;
2
2       ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                    ! ERROR LOG PACKET'S ADDRESS
2       REASON = .ELOG_ADDR [EL_FORMAT];                                 ! FORMAT
2       DISK_NUM = .ELOG_ADDR [EL_DK_NUM];                               ! DISK NUMBER
2       ELOG_CODE = .ELOG_ADDR [EL_CODE];                                ! CODE
2       ELOG_SUB = .ELOG_ADDR [EL_SUBCODE];                              ! SUBCODE
2       PRINTB (ELG_00);                                                 ! ERROR-LOG MESSAGE RECEIVED
2
2       if (.REASON eql FORMAT_CNTR) or
3          (.REASON eql FORMAT_HOST)
2       then
2           PRINTB (.ELG_FMT [.REASON]);                                 ! PRINT BASIC REASON
2
2       if (.REASON eql FORMAT_XFER) or                                  !MMM
2          (.REASON eql FORMAT_SDI) or                                   !MMM
3          (.REASON eql FORMAT_SDE)                                      !MMM
2       then
2           PRINTB (.ELG_FMT [.REASON], .DISK_NUM);                      ! PRINT BASIC REASON WITH DISK NUMBER
2
3       if (.REASON eql FORMAT_BRA)                                      ! BAD BLOCK REPLACEMENT ATTEMPT MMM
2       then
2           PRINTB (.ELG_FMT [5], .DISK_NUM);                            ! PRINT BASIC REASON WITH DISK NUMBER MMM
2
2       if (.ELOG_CODE gtru 0) and
3          (.ELOG_CODE lequ 11)
2       then
3           begin
3           PRINTX (ASTERISK);
3           PRINTX (.ERR_COD [.ELOG_CODE - 1]);                          ! CODE
3           end
2       else
2
2           if .ELOG_CODE eql ST_DIA                                     ! MESSAGE FROM INTERNAL DIAGNOSTICS
2           then
3               begin
3               PRINTX (ASTERISK);
3               PRINTX (.ERR_COD [11]);                                  !MMM
2               end;
```

```
	if .ELOG_CODE eql ST_BRC				! BAD BLOCK REPLACEMENT COMPLETION MMM
	then
	    begin
	    PRINTX (ASTERISK);
	    PRINTX (.ERR_COD [14]);				!MMM
	    end;

	if (.ELOG_CODE eql ST_MFE) and
	   (.ELOG_SUB lequ 10)
	then
	    begin
	    PRINTX (CRLF);
	    PRINTX (ASTERISK);
	    PRINTX (.TBL_MFE [.ELOG_SUB]);			! MEDIA FORMAT ERROR
	    end;

	if (.ELOG_CODE eql ST_DAT) and
	   (.ELOG_SUB lequ 15)
	then
	    begin
	    PRINTX (CRLF);
	    PRINTX (ASTERISK);
	    PRINTX (.TBL_DAT [.ELOG_SUB]);			! DATA ERROR
	    end;

	if (.ELOG_CODE eql ST_HST) and
	   (.ELOG_SUB lequ 4)
	then
	    begin
	    PRINTX (CRLF);
	    PRINTX (ASTERISK);
	    PRINTX (.TBL_HST [.ELOG_SUB]);			! HOST ACCESS ERROR
	    end;

	if (.ELOG_CODE eql ST_CNT) and
	   (.ELOG_SUB lequ 3)
	then
	    begin
	    PRINTX (CRLF);
	    PRINTX (ASTERISK);
	    PRINTX (.TBL_CNT [.ELOG_SUB]);			! CONTROLLER ERROR
	    end;

	if (.ELOG_CODE eql ST_DRV) and
	   (.ELOG_SUB lequ 8)
	then
	    begin
	    PRINTX (CRLF);
	    PRINTX (ASTERISK);
	    PRINTX (.TBL_DRV [.ELOG_SUB]);			! DRIVE ERROR
	    end;
```

```
;    7567  2          if (.ELOG_CODE eql ST_BRC) and                            ! BAD BLOCK REPLACEMENT COMPLETION MMM
;    7568  3             (.ELOG_SUB lequ 4)
;    7569  2          then
;    7570  3              begin
;    7571  3              PRINTX (CRLF);
;    7572  3              PRINTX (ASTERISK);
;    7573  3              PRINTX (.TBL_BRC [.ELOG SUB]);                        ! OUTCOME OF BAD BLOCK REPELACEMENT MMM
;    7574  2              end;
;    7575  2
;    7576  2          if .REASON eql FORMAT_XFER                                 ! IF DISK XFER INVOLVED
;    7577  2          then
;    7578  2
;    7579  2              if .ELOG_ADDR [EL_BLOCK_TYPE] eql TYPE_LBN              ! PRINT PBN OR RBN
;    7580  2              then
;    7581  3                  PRINTX (EX_PBN2, .ELOG_ADDR [EL_BLOCK_HI], .ELOG_ADDR [EL_BLOCK])          !ZZZ
;    7582  2              else
;    7583  2                  PRINTX (EX_RBN2, .ELOG_ADDR [EL_BLOCK_HI], .ELOG_ADDR [EL_BLOCK]);         !ZZZ
;    7584  2
;    7585  2          if .REASON eql FORMAT_BRA                                  ! IF BAD BLOCK REPLACEMENT INVOLVED MMM
;    7586  2          then
;    7587  2              PRINTX (EX_LB, .ELOG_ADDR [EL_BLOCK_HI], .ELOG_ADDR [EL_BLOCK]);           !MMM
;    7588  2
;    7589  2          EMS_TIM ();                                                ! TIME
;    7590  2          EMS_BLK ((.ELOG_ADDR + 2), ((.ELOG_ADDR [EL_MSGLEN] + 1) / 2) + 2); ! PRINTX CONTENTS OF PACKET
;    7591  2          ELOG_ADDR [EL_CONTENTS] = EMPTY;                           ! DECLARE SAVE AREA FREE
;    7592  2
;    7593  1          end;


                                         .SBTTL  EMS.EL ERROR MESSAGE SUBROUTINES
000000  004137  000000G        EMS.EL::JSR     R1,$SAVE5                   ;                              7461
000004  024646                         CMP     -(SP),-(SP)
000006  016646  000022                 MOV     22(SP),-(SP)                ; INDEX,*                      7477
000012  012746  000102                 MOV     #102,-(SP)
000016  004737  000C)0G                JSR     PC,BL$MUL
000022  062700  000000G                ADD     #ELOG.PKT,R0
000026  010001                         MOV     R0,R1                       ; *,ELOG.ADDR
000030  005002                         CLR     R2                          ; REASON                       7478
000032  156102  000016                 BISB    16(R1),R2                   ; *(ELOG.ADDR),REASON
000036  116105  000012                 MOVB    12(R1),R5                   ; *(ELOG.ADDR),DISK.NUM        7479
000042  116100  000020                 MOVB    20(R1),R0                   ; *(ELOG.ADDR),*
000046  042700  177740                 BIC     #177740,R0
000052  105004                         CLRB    R4                          ; ELOG.CODE
000054  050004                         BIS     R0,R4                       ; *,ELOG.CODE
000056  016103  000020                 MOV     20(R1),R3                   ; *(ELOG.ADDR),ELOG.SUB
000062  006203                         ASR     R3                          ; ELOG.SUB
000064  006203                         ASR     R3                          ; ELOG.SUB
000066  006203                         ASR     R3                          ; ELOG.SUB
000070  006203                         ASR     R3                          ; ELOG.SUB
000072  006203                         ASR     R3                          ; ELOG.SUB
000074  042703  174000                 BIC     #174000,R3                  ; *,ELOG.SUB
000100  012716  000000G                MOV     #ELG.00,(SP)                ;                              7482
000104  012746  000001                 MOV     #1,(SP)
```

```
000110  010600                    MOV     SP,RO                          ; SP,*
000112  104414                    TRAP    14
000114  005702                    TST     R2                             ; REASON                 7484
000116  001403                    BEQ     1$
000120  020227  000001            CMP     R2,#1                          ; REASON,*               7485
000124  001011                    BNE     2$
000126  010200            1$:     MOV     R2,RO                          ; REASON,*               7487
000130  006300                    ASL     RO
000132  016016  000000G           MOV     ELG.FMT(RO),(SP)
000136  012746  000001            MOV     #1,-(SP)
000142  010600                    MOV     SP,RO                          ; SP,*
000144  104414                    TRAP    14
000146  005726                    TST     (SP)+
000150  005066  000006    2$:     CLR     6(SP)                          ;                        7489
000154  020227  000002            CMP     R2,#2                          ; REASON,*
000160  001003                    BNE     3$
000162  005266  000006            INC     6(SP)
000166  000406                    BR      4$
000170  020227  000003    3$:     CMP     R2,#3                          ; REASON,*               7490
000174  001403                    BEQ     4$
000176  020227  000004            CMP     R2,#4                          ; REASON,*               7491
000202  001013                    BNE     5$
000204  005016            4$:     CLR     (SP)                           ;                        7493
000206  110516                    MOVB    R5,(SP)                        ; DISK.NUM,*
000210  010200                    MOV     R2,RO                          ; REASON,*
000212  006300                    ASL     RO
000214  016046  000000G           MOV     ELG.FMT(RO),-(SP)
000220  012746  000002            MOV     #2,-(SP)
000224  010600                    MOV     SP,RO                          ; SP,*
000226  104414                    TRAP    14
000230  022626                    CMP     (SP)+,(SP)+
000232  005066  000010    5$:     CLR     10(SP)                         ;                        7495
000236  020227  000011            CMP     R2,#11                         ; REASON,*
000242  001013                    BNE     6$
000244  005266  000010            INC     10(SP)
000250  005016                    CLR     (SP)                           ;                        7497
000252  110516                    MOVB    R5,(SP)                        ; DISK.NUM,*
000254  013746  000012G           MOV     ELG.FMT+12,-(SP)
000260  012746  000002            MOV     #2,-(SP)
000264  010600                    MOV     SP,RO                          ; SP,*
000266  104414                    TRAP    14
000270  022626                    CMP     (SP)+,(SP)+
000272  105704            6$:     TSTB    R4                             ; ELOG.CODE              7499
000274  001423                    BEQ     7$
000276  120427  000013            CMPB    R4,#13                         ; ELOG.CODE,*            7500
000302  101020                    BHI     7$
000304  012716  000000G           MOV     #ASTERISK,(SP)                 ;                        7503
000310  012746  000001            MOV     #1,-(SP)
000314  010600                    MOV     SP,RO                          ; SP,*
000316  104415                    TRAP    15
000320  005000                    CLR     RO                             ;                        7504
000322  150400                    BISB    R4,RO                          ; ELOG.CODE,*
000324  006300                    ASL     RO
```

```
000326  016016  177776G          MOV   ERR.COD 2(R0),(SP)
000332  012746  000001           MOV   #1,-(SP)
000336  010600                   MOV   SP,R0          ; SP,*
000340  104415                   TRAP  15
000342  000417                   BR    8$             ;
000344  120427  000037     7$:   CMPB  R4,#37         ; ELOG.CODE,*          7502
000350  001015                   BNE   9$                                    7508
000352  012716  000000G          MOV   #ASTERISK,(SP) ;                      7511
000356  012746  000001           MOV   #1,-(SP)
000362  010600                   MOV   SP,R0          ; SP,*
000364  104415                   TRAP  15
000366  013716  000026G          MOV   ERR.COD+26,(SP) ;                     7512
000372  012746  000001           MOV   #1,-(SP)
000376  010600                   MOV   SP,R0          ; SP,*
000400  104415                   TRAP  15
000402  022626             8$:   CMP   (SP)+,(SP)+    ;                      7510
000404  005002             9$:   CLR   R2             ;                      7515
000406  120427  000024           CMPB  R4,#24         ; ELOG.CODE,*
000412  001016                   BNE   10$
000414  005202                         R2
000416  012716  000000G                #ASTERISK,(SP) ;                      7518
000422  012746  000001                 #1,-(SP)
000426  010600                   MOV   SP,R0          ; SP,*
000430  104415                   TRAP  15
000432  013716  000034G          MOV   ERR.COD+34,(SP) ;                     7519
000436  012746  000001           MOV   #1,-(SP)
000442  010600                   MOV   SP,R0          ; SP,*
000444  104415                   TRAP  15
000446  022626                   CMP   (SP)+,(SP)+    ;                      7517
000450  120427  000005     10$:  CMPB  R4,#5          ; ELOG.CODE,*          7522
000454  001031                   BNE   11$
000456  020327  000012           CMP   R3,#12         ; ELOG.SUB,*           7523
000462  101026                   BHI   11$
000464  012716  000000G          MOV   #CRLF,(SP)     ;                      7526
000470  012746  000001           MOV   #1,-(SP)
000474  010600                   MOV   SP,R0          ; SP,*
000476  104415                   TRAP  15
000500  012716  000000G          MOV   #ASTERISK,(SP) ;                      7527
000504  012746  000001           MOV   #1,-(SP)
000510  010600                   MOV   SP,R0          ; SP,*
000512  104415                   TRAP  15
000514  010300                   MOV   R3,R0          ; ELOG.SUB,*           7528
000516  006300                   ASL   R0
000520  016016  000070'          MOV   TBL.MFE(R0),(SP)
000524  012746  000001           MOV   #1,-(SP)
000530  010600                   MOV   SP,R0          ; SP,*
000532  104415                   TRAP  15
000534  062706  000006           ADD   #6,SP          ;                      7525
000540  120427  000010     11$:  CMPB  R4,#10         ; ELOG.CODE,*          7531
000544  001031                   BNE   12$
000546  020327  000017           CMP   R3,#17         ; ELOG.SUB,*           7532
000552  101026                   BHI   12$
000554  012716  000000G          MOV   #CRLF,(SP)     ;                      7535
```

```
000560  012746  000001                 MOV     #1.-(SP)
000564  010600                         MOV     SP,R0                  ; SP,*
000566  104415                         TRAP    15
000570  012716  000000G                MOV     #ASTERISK,(SP)         ;
000574  012746  000001                 MOV     #1.-(SP)
000600  010600                         MOV     SP,R0                  ; SP,*
000602  104415                         TRAP    15
000604  010300                         MOV     R3,R0                  ; ELOG.SUB,*
000606  006300                         ASL     R0
000610  016016  000126'                MOV     TBL.DAT(R0),(SP)
000614  012746  000001                 MOV     #1.-(SP)
000620  010600                         MOV     SP,R0                  ; SP,*
000622  104415                         TRAP    15
000624  062706  000006                 ADD     #6,SP                  ;
000630  120427  000011         12$:    CMPB    R4,#11                 ; ELOG.CODE,*
000634  001031                         BNE     13$
000636  020327  000004                 CMP     R3,#4                  ; ELOG.SUB,*
000642  101026                         BHI     13$
000644  012716  000000G                MOV     #CRLF,(SP)             ;
000650  012746  000001                 MOV     #1.-(SP)
000654  010600                         MOV     SP,R0                  ; SP,*
000656  104415                         TRAP    15
000660  012716  000000G                MOV     #ASTERISK,(SP)         ;
000664  012746  000001                 MOV     #1.-(SP)
000670  010600                         MOV     SP,R0                  ; SP,*
000672  104415                         TRAP    15
000674  010300                         MOV     R3,R0                  ; ELOG.SUB,*
000676  006300                         ASL     R0
000700  016016  000166'                MOV     TBL.HST(R0),(SP)
000704  012746  000001                 MOV     #1.-(SP)
000710  010600                         MOV     SP,R0                  ; SP,*
000712  104415                         TRAP    15
000714  062706  000006                 ADD     #6,SP                  ;
000720  120427  000012         13$:    CMPB    R4,#12                 ; ELOG.CODE,*
000724  001031                         BNE     14$
000726  020327  000003                 CMP     R3,#3                  ; ELOG.SUB,*
000732  101026                         BHI     14$
000734  012716  000000G                MOV     #CRLF,(SP)             ;
000740  012746  000001                 MOV     #1.-(SP)
000744  010600                         MOV     SP,R0                  ; SP,*
000746  104415                         TRAP    15
000750  012716  000000G                MOV     #ASTERISK,(SP)         ;
000754  012746  000001                 MOV     #1.-(SP)
000760  010600                         MOV     SP,R0                  ; SP,*
000762  104415                         TRAP    15
000764  010300                         MOV     R3,R0                  ; ELOG.SUB,*
000766  006300                         ASL     R0
000770  016016  000200'                MOV     TBL.CNT(R0),(SP)
000774  012746  000001                 MOV     #1.-(SP)
001000  010600                         MOV     SP,R0                  ; SP,*
001002  104415                         TRAP    15
001004  062706  000006                 ADD     #6,SP                  ;
001010  120427  000013         14$:    CMPB    R4,#13                 ; ELOG.CODE,*
```

```
001014  001031                        BNE     15$
001016  020327  000010                CMP     R3,#10                   ; ELOG SUB.*                7559
001022  101026                        BHI     15$
001024  012716  000000G               MOV     #CRLF,(SP)               ;                           7562
001030  012746  000001                MOV     #1,-(SP)
001034  010600                        MOV     SP,R0                    ; SP,*
001036  104415                        TRAP    15
001040  012716  000000G               MOV     #ASTERISK,(SP)           ;                           7563
001044  012746  000001                MOV     #1,-(SP)
001050  010600                        MOV     SP,R0                    ; SP,*
001052  104415                        TRAP    15
001054  010300                        MOV     R3,R0                    ; ELOG.SUB.*                7564
001056  006300                        ASL     R0
001060  016016  000210'               MOV     TBL.DRV(R0),(SP)
001064  012746  000001                MOV     #1,-(SP)
001070  010600                        MOV     SP,R0                    ; SP,*
001072  104415                        TRAP    15
001074  062706  000006                ADD     #6,SP                    ;                           7561
001100  006002                15$:    ROR     R2                       ;                           7567
001102  103031                        BCC     16$
001104  020327  000004                CMP     R3,#4                    ; ELOG.SUB.*                7568
001110  101026                        BHI     16$
001112  012716  000000G               MOV     #CRLF,(SP)               ;                           7571
001116  012746  000001                MOV     #1,-(SP)
001122  010600                        MOV     SP,R0                    ; SP,*
001124  104415                        TRAP    15
001126  012716  000000G               MOV     #ASTERISK,(SP)           ;                           7572
001132  012746  000001                MOV     #1,-(SP)
001136  010600                        MOV     SP,R0                    ; SP,*
001140  104415                        TRAP    15
001142  010300                        MOV     R3,R0                    ; ELOG.SUB.*                7573
001144  006300                        ASL     R0
001146  016016  000232'               MOV     TBL.BRC(R0),(SP)
001152  012746  000001                MOV     #1,-(SP)
001156  010600                        MOV     SP,R0                    ; SP,*
001160  104415                        TRAP    15
001162  062706  000006                ADD     #6,SP                    ;                           7570
001166  032766  000001  000006  16$:  BIT     #1,6(SP)                 ;                           7576
001174  001437                        BEQ     19$
001176  012700  000060                MOV     #60,R0                   ;                           7579
001202  060100                        ADD     R1,R0                    ; ELOG.ADDR.*
001204  032710  170000                BIT     #170000,(R0)
001210  001014                        BNE     17$
001212  016116  000056                MOV     56(R1),(SP)              ; *(ELOG.ADDR).*            7581
001216  011046                        MOV     (R0),-(SP)
001220  042716  170000                BIC     #170000,(SP)
001224  012746  000000G               MOV     #EX.PBN2,-(SP)
001230  012746  000003                MOV     #3,-(SP)
001234  010600                        MOV     SP,R0                    ; SP,*
001236  104415                        TRAP    15
001240  000413                        BR      18$                      ;                           7579
001242  016116  000056        17$:    MOV     56(R1),(SP)              ; *(ELOG.ADDR).*            7583
001246  011046                        MOV     (R0),-(SP)
```

```
001250  042716  170000                    BIC     #170000,(SP)
001254  012746  000000G                   MOV     #EX.RBN2,-(SP)
001260  012746  000003                    MOV     #3,-(SP)
001264  010600                            MOV     SP,R0                  ; SP,*
001266  104415                            TRAP    15
001270  062706  000006          18$:      ADD     #6,SP                  ;                         7579
001274  032766  000001  000010  19$:      BIT     #1,10(SP)              ;                         7585
001302  001416                            BEQ     20$
001304  016116  000056                    MOV     56(R1),(SP)            ; *(ELOG.ADDR),*          7587
001310  016146  000060                    MOV     60(R1),-(SP)           ; *(ELOG.ADDR),*
001314  042716  170000                    BIC     #170000,(SP)
001320  012746  000000G                   MOV     #EX.LB,-(SP)
001324  012746  000003                    MOV     #3,-(SP)
001330  010600                            MOV     SP,R0                  ; SP,*
001332  104415                            TRAP    15
001334  062706  000006                    ADD     #6,SP
001340  004737  000000V         20$:      JSR     PC,EMS.TIM             ;                         7589
001344  012716  000002                    MOV     #2,(SP)                ;                         7590
001350  060116                            ADD     R1,(SP)                ; ELOG.ADDR,*
001352  016146  000002                    MOV     2(R1),-(SP)            ; *(ELOG.ADDR),*
001356  005216                            INC     (SP)
001360  012746  000002                    MOV     #2,-(SP)
001364  004737  000000G                   JSR     PC,BL$DIV
001370  010066  000002                    MOV     R0,2(SP)
001374  062766  000002  000002            ADD     #2,2(SP)
001402  005726                            TST     (SP)+
001404  004737  011374'                   JSR     PC,EMS.BLK
001410  105061  000001                    CLRB    1(R1)                  ; *(ELOG.ADDR)            7591
001414  062706  000014                    ADD     #14,SP                 ;                         7461
001420  000207                            RTS     PC

; Routine Size:  393 words,     Routine Base:  $CODE$ + 12320
; Maximum stack depth per invocation:  16 words
```

```
;   1   global routine EMS_CMP (ADDR) : novalue =
;   1
;   1   !+
;   1   ! THIS ROUTINE IS CALLED FROM 'HOST WPT CHK' AND PRINTS RELEVENT DATA ON A HOST
;   1   ! COMPARE ERROR
;   1   !-
;   1
;   2       begin
;   2
;   2       local
;   2           ORIG_ADDR : ref block [RP LEN, word] field (RP_FIELDS); ! MMM
;   2
;   2       ORIG_ADDR = .ADDR;                                          ! ADDRESS OF THE WRITE RETPKT
;   2       PRINTB (ERR_00, .CDISK);                                    ! "DISK XXX"
;   2       PRINTB (DASH);                                              !
;   2       PRINTB (.ERR_COD [12]);                                     !
;   2       PRINTX (EX_LBW2, .ORIG_ADDR [LBN_HI], .ORIG_ADDR [LBN_LO]); ! "   HOST COMPARE ERROR"
;   2       PRINTX (EX_LBR2, .RP_ADDR [LBN_HI], .RP_ADDR [LBN_LO]);     ! LBN (WRITTEN)             ZZZ
;   2       PRINTX (EX_CBW, .ORIG_ADDR [CBCNT_LO]);                     ! LBN (READ)                ZZZ
;   2       PRINTX (EX_BC, .ORIG_ADDR [BCNT_LO]);                       ! BYTE COUNT (WRITE)
;   2       PRINTX (EX_CBR, .RP_ADDR [CBCNT_LO]);                       ! BYTE COUNT XMITTED (WRITE)
;   2       PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);                         ! BYTE COUNT (READ);
;   2       PRINTX (EX_BDW, .ORIG_ADDR [BUFF_1], .ORIG_ADDR [BUFF_0]);  ! BYTE COUNT XMITTED (READ)
;   2       PRINTX (EX_BDR, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);      ! BUFFER ADDRESS (WRITE)
;   2       EMS_TIM ();                                                 ! BUFFER ADDRESS (READ)
;   2       EMS_R2 (.ORIG_ADDR);                                        ! TIME
;   2       EMS_R1 (.ORIG_ADDR);                                        ! MMM
;   2       EMS_R2 (.RP_ADDR);                                          ! MMM
;   2       EMS_R1 (.RP_ADDR);                                        ! MMM
;   1       end;                                                        ! MMM


                                    .SBTTL  EMS.CMP ERROR MESSAGE SUBROUTINES
000000  010146                  EMS.CMP::
                                    MOV     R1, (SP)                                    7594
000002  016601  000004              MOV     4(SP),R1            ;                       7606
000006  013746  000000G             MOV     CDISK,-(SP)         ; ADDR.ORIG.ADDR        7607
000012  012746  000000G             MOV     #ERR.00,-(SP)       ;
000016  012746  000002              MOV     #2,-(SP)
000022  010600                      MOV     SP,R0
000024  104414                      TRAP    14                  ; SP,*
000026  012716  000000G             MOV     #DASH,(SP)          ;                       7608
000032  012746  000001              MOV     #1,-(SP)
000036  010600                      MOV     SP,R0
000040  104414                      TRAP    14                  ; SP,*
000042  013716  000030G             MOV     ERR.COD+30,(SP)     ;                       7609
000046  012746  000001              MOV     #1,-(SP)
000052  010600                      MOV     SP,R0
000054  104414                      TRAP    14                  ; SP,*
000056  016116  000050              MOV     50(R1),(SP)         ; *(ORIG.ADDR),*        7610
000062  016146  000052              MOV     52(R1),-(SP)        ; *(ORIG.ADDR),*
000066  012746  000000G             MOV     #EX.LBW2,-(SP)
000072  012746  000003              MOV     #3,-(SP)
```

```
000076 010600              MOV     SP,R0                  ; SP,*
000100 104415              TRAP    15
000102 013700  000000G     MOV     RP.ADDR,R0             ;                                7611
000106 016016  000050      MOV     50(R0),(SP)
000112 016046  000052      MOV     52(R0),-(SP)
000116 012746  000000G     MOV     #EX.LBR2,-(SP)
000122 012746  000003      MOV     #3,-(SP)
000126 010600              MOV     SP,R0                  ; SP,*
000130 104415              TRAP    15
000132 016116  000044      MOV     44(R1),(SP)            ; *(ORIG.ADDR),*                 7612
000136 012746  000000G     MOV     #EX.CBW,-(SP)
000142 012746  000002      MOV     #2,-(SP)
000146 010600              MOV     SP,R0                  ; SP,*
000150 104415              TRAP    15
000152 016116  000020      MOV     20(R1),(SP)            ; *(ORIG.ADDR),*                 7613
000156 012746  000000G     MOV     #EX.BC,-(SP)
000162 012746  000002      MOV     #2,-(SP)
000166 010600              MOV     SP,R0                  ; SP,*
000170 104415              TRAP    15
000172 013700  000000G     MOV     RP.ADDR,R0             ;                                7614
000176 016016  000044      MOV     44(R0),(SP)
000202 012746  000000G     MOV     #EX.CBR,-(SP)
000206 012746  000002      MOV     #2,-(SP)
000212 010600              MOV     SP,R0                  ; SP,*
000214 104415              TRAP    15
000216 013700  000000G     MOV     RP.ADDR,R0             ;                                7615
000222 016016  000020      MOV     20(R0),(SP)
000226 012746  000000G     MOV     #EX.BC,-(SP)
000232 012746  000002      MOV     #2,-(SP)
000236 010600              MOV     SP,R0                  ; SP,*
000240 104415              TRAP    15
000242 016116  000024      MOV     24(R1),(SP)            ; *(ORIG.ADDR),*                 7616
000246 016146  000026      MOV     26(R1),-(SP)           ; *(ORIG.ADDR),*
000252 012746  000000G     MOV     #EX.BDW,-(SP)
000256 012746  000003      MOV     #3,-(SP)
000262 010600              MOV     SP,R0                  ; SP,*
000264 104415              TRAP    15
000266 013700  000000G     MOV     RP.ADDR,R0             ;                                7617
000272 016016  000024      MOV     24(R0),(SP)
000276 016046  000026      MOV     26(R0),-(SP)
000302 012746  000000G     MOV     #EX.BDR,-(SP)
000306 012746  000003      MOV     #3,-(SP)
000312 010600              MOV     SP,R0                  ; SP,*
000314 104415              TRAP    15
000316 004737  000000V     JSR     PC,EMS.TIM             ;                                7618
000322 010116              MOV     R1,(SP)                ; ORIG.ADDR,*                    7619
000324 004737  012124'     JSR     PC,EMS.R2
000330 010116              MOV     R1,(SP)                ; ORIG.ADDR,*                    7620
000332 004737  012254'     JSR     PC,EMS.R1
000336 013716  000000G     MOV     RP.ADDR,(SP)           ;                                7621
000342 004737  012124'     JSR     PC,EMS.R2
000346 013716  000000G     MOV     RP.ADDR,(SP)           ;                                7622
000352 004737  012254'     JSR     PC,EMS.R1
```

```
000356  062706  000062          ADD     #62,SP                              ;       7601
000362  012601                  MOV     (SP)+,R1                            ;       7594
000364  000207                  RTS     PC

; Routine Size:  123 words,     Routine Base:  $CODE$ + 13742
; Maximum stack depth per invocation:  28 words
```

```
;     7624  1    global rout'ne EMS_ERR : novalue =
;     7625  1
;     7626  2        begin
;     7627  2
;     7628  2    ! TABLE OF BASIC, HARD ERROR MESSAGE ADDRESSES, INDEXED BY STATUS CODE
;     7629  2    !
;     7630  2        PRINTB (ERR_00, .CDISK);                        ! "DISK XXX"
;     7631  2        PRINTB (DASH);                                  !
;     7632  2
;     7633  2        if (.ST_CODE gtru 0) and                        ! IF STATUS CODE IS WITHIN RANGE
;     7634  3          (.ST_CODE lequ 11)
;     7635  2        then
;     7636  3            PRINTB (.ERR_COD [.ST_CODE - 1])            ! PRINTB APPROPRIATE MESSAGE
;     7637  2        else
;     7638  2
;     7639  2            if .ST_CODE eql ST_DIA
;     7640  2            then
;     7641  3                PRINTB (.ERR_COD [11])                  ! MESSAGE FROM INTERNAL DIAGNOSTICS
;     7642  2            else
;     7643  2                PRINTB (EX_SC, .ST_CODE);               ! JUST PRINT STATUS CODE WHEN NO MATCH
;     7644  2
;     7645  2        EMS_R2 (.RP_ADDR);                                  ! PRINTX OTHER RETPKT FIELDS
;     7646  2
;     7647  1        end;

                                            .SBTTL  EMS.ERR ERROR MESSAGE SUBROUTINES
000000  013746  000000G          EMS.ERR::
                                            MOV     CDISK,-(SP)
000004  012746  000000G                     MOV     #ERR.00,-(SP)           ;              7630
000010  012746  000002                      MOV     #2,-(SP)
000014  010600                              MOV     SP,R0                   ; SP,*
000016  104414                              TRAP    14
000020  012716  000000G                     MOV     #DASH,(SP)              ;              7631
000024  012746  000001                      MOV     #1,-(SP)
000030  010600                              MOV     SP,R0                   ; SP,*
000032  104414                              TRAP    14
000034  013700  000000G                     MOV     ST.CODE,R0              ;              7633
000040  001413                              BEQ     1$
000042  020027  000013                      CMP     R0,#13                  ;              7634
000046  101010                              BHI     1$
000050  006300                              ASL     R0                      ;              7636
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$                      ;              7633
000070  020027  000037           1$:        CMP     R0,#37                  ;              7639
000074  001007                              BNE     2$
000076  013716  000026G                     MOV     ERR.COD+26,(SP)         ;              7641
000102  012746  000001                      MOV     #1,-(SP)
000106  010600                              MOV     SP,R0                   ; SP,*
000110  104414                              TRAP    14
```

```
000112  000410                     BR      3$                              ;       7639
000114  010016             2$:     MOV     R0,(SP)                         ;       7643
000116  012746  000000G            MOV     #EX.SC,-(SP)
000122  012746  000002             MOV     #2,-(SP)
000126  010600                     MOV     SP,R0
000130  104414                     TRAP    14                              ; SP,*
000132  005726                     TST     (SP)+
000134  013716  000000G    3$:     MOV     RP.ADDR,(SP)                    ;       7645
000140  004737  012124'            JSR     PC,EMS.R2
000144  062706  000012             ADD     #12,SP                          ;       7626
000150  000207                     RTS     PC                              ;       7624

; Routine Size:  53 words,      Routine Base:  $CODE$ + 14330
; Maximum stack depth per invocation:  8 words
```

```
;     7648  1      routine EMS_TIM : novalue =
;     7649  1
;     7650  1      !+
;     7651  1      ! THIS ROUTINE PRINTS THE TIME-OF DAY MESSAGE
;     7652  1      !-
;     7653  1
;     7654  1      PRINTX (EX_TIM, .HOURS, .MINUTES);


                                              .SBTTL  EMS.TIM ERROR MESSAGE SUBROUTINES
000000  005046                        EMS.TIM:CLR      (SP)                        ;              7654
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                           ;              7648

; Routine Size:  15 words,       Routine Base:  $CODE$ + 14502
; Maximum stack depth per invocation:  6 words
```

```
;     7655  1      BGNMSG (EMS_01);


                                                    .SBTTL  EMS.01 ERROR MESSAGE SUBROUTINES
000000  004737  000000V               EMS.01::JSR     PC,M$EMS.01                          ;                 7655
000004  104423                                TRAP    23
000006  000207                                RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14540
; Maximum stack depth per invocation:  2 words


;     7656  2      PRINTB (EBS_01, MAX_UNITS);                     ! "MORE THAN XX UNITS SPECIFIED"
;     7657  1      ENDMSG;


                                                    .SBTTL  M$EMS.01 ERROR MESSAGE SUBROUTINES
000000  012746  000004                M$EMS.01:
                                              MOV     #4,-(SP)
000004  012746  000000G                       MOV     #EBS.01,-(SP)                        ;                 7656
000010  012746  000002                        MOV     #2,-(SP)
000014  010600                                MOV     SP,R0                                ; SP,*
000016  104414                                TRAP    14
000020  062706  000006                        ADD     #6,SP                                ;                 7655
000024  000207                                RTS     PC

; Routine Size:  11 words,     Routine Base:  $CODE$ + 14550
; Maximum stack depth per invocation:  5 words
```

```
;    7658  1      BGNMSG (EMS_10);


                                                .SBTTL  EMS.10 ERROR MESSAGE SUBROUTINES
000000  004737  000000V             EMS.10::JSR     PC,M$EMS.10                                     7658
000004  104423                              TRAP    23                          ;
000006  000207                              RTS     PC

; Routine Size:  4 words,        Routine Base:  $CODE$ + 14576
; Maximum stack depth per invocation:  2 words


;    7659  2      PRINTB (EBD_10, .RDRX_ADDR + .OF_RC);          ! "NO RESPONSE AT ADDRESS XXXXXX"
;    7660  1      ENDMSG;


                                                .SBTTL  M$EMS.10 ERROR MESSAGE SUBROUTINES
000000  013746  000000G             M$EMS.10:
                                            MOV     RDRX.ADDR,-(SP)
000004  063716  000000G                     ADD     OF.RC,(SP)          ;                           7659
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               ;                           7658
000030  000207                              RTS     PC

; Routine Size:  13 words,       Routine Base:  $CODE$ + 14606
; Maximum stack depth per invocation:  5 words
```

```
;    7661  1      BGNMSG (EMS_12);


                                                  .SBTTL  EMS.12 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                   EMS.12::JSR     PC,M$EMS.12                          ;          7661
000004  104423                                    TRAP    23
000006  000207                                    RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14640
; Maximum stack depth per invocation:  2 words


;    7662  2      PRINTB (EBD_12, .RDRX_ADDR);             ! "INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX"
;    7663  1      ENDMSG;


                                                  .SBTTL  M$EMS.12 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                   M$EMS.12:
                                                  MOV     RDRX.ADDR,-(SP)                      ;          7662
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                                ;          7661
000024  000207                                    RTS     PC

; Routine Size:  11 words,     Routine Base:  $CODE$ + 14650
; Maximum stack depth per invocation:  5 words
```

```
;    7664  1      BGNMSG (EMS_13);


                                              .SBTTL  EMS.13 ERROR MESSAGE SUBROUTINES
000000  004737  000000V              EMS.13::JSR     PC,M$EMS.13                          ;              7664
000004  104423                               TRAP    23
000006  000207                               RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14676
; Maximum stack depth per invocation:  2 words


;    7665  2      PRINTB (EBD_13, .STEP);                              ! "STEP X READ ERROR"
;    7666  2      EMS_SA ();                                           ! PRINTX SA CONTENTS
;    7667  1      ENDMSG;


                                              .SBTTL  M$EMS.13 ERROR MESSAGE SUBROUTINES
000000  013746  000000G              M$EMS.13:
                                             MOV     STEP,-(SP)                           ;              7665
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  007712'                      JSR     PC,EMS.SA                            ;              7666
000024  062706  000006                       ADD     #6,SP                                ;              7664
000030  000207                               RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14706
; Maximum stack depth per invocation:  5 words
```

```
;    7668  1      BGNMSG (EMS_14);


                                                    .SBTTL  EMS.14 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                   EMS.14::JSR     PC,M$EMS.14                                   .                   7668
000004  104423                                    TRAP    23
000006  000207                                    RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14740
; Maximum stack depth per invocation:  2 words


;    7669  2      PRINTB (EBD_14, .IRDRX_ADDR);                    ! "BAD SA CODE FROM DEVICE XXXXXX"
;    7670  2      EMS_SA ();                                       ! PRINTX SA REGISTER CONTENTS
;    7671  1      ENDMSG;


                                                    .SBTTL  M$EMS.14 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                   M$EMS.14:
                                                    MOV     IRDRX.ADDR,-(SP)                    ;                   7669
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  007712'                             JSR     PC,EMS.SA                           ;                   7670
000024  062706  000006                              ADD     #6,SP                               ;                   7668
000030  000207                                      RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14750
; Maximum stack depth per invocation:  5 words
```

```
;     7672  1      BGNMSG (EMS_18);


                                                 .SBTTL  EMS.18 ERROR MESSAGE SUBROUTINES
000000  004737  000000V               EMS.18::JSR     PC,M$EMS.18                   ;                      7672
000004  104423                                 TRAP    23
000006  000207                                 RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 15002
; Maximum stack depth per invocation:  2 words


;     7673  2      PRINTB (EBD_18, .CDISK);                   ! "DISK XXX WENT OFFLINE"
;     7674  2      EMS_R2 (.RP_ADDR);                         ! PRINTX RELEVANT RETPKT FIELDS
;     7675  1      ENDMSG;


                                                 .SBTTL  M$EMS.18 ERROR MESSAGE SUBROUTINES
000000  013746  C00000G               M$EMS.18:
                                                 MOV     CDISK,-(SP)                   ;                      7673
000004  012746  000000G                          MOV     #EBD.18,-(SP)
000010  012746  000002                           MOV     #2,-(SP)
000014  010600                                   MOV     SP,R0                         ; SP,*
000016  104414                                   TRAP    14
000020  013716  000000G                          MOV     RP.ADDR,(SP)                  ;                      7674
000024  004737  012124'                          JSR     PC,EMS.R2
000030  062706  000006                           ADD     #6,SP                         ;                      7672
000034  000207                                   RTS     PC

; Routine Size:  15 words,      Routine Base:  $CODE$ + 15012
; Maximum stack depth per invocation:  5 words
```

```
;    7676  1      BGNMSG (EMS_21);

                                              .SBTTL  EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  000000V              EMS.21::JSR     PC,M$EMS.21                        ;          7676
000004  104423                               TRAP    23
000006  000207                               RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 15050
; Maximum stack depth per invocation:  2 words

;    7677  2      EMS_R1 (.RP_ADDR);                            ! CONTENTS OF RETURN PACKET
;    7678  1      ENDMSG;

                                              .SBTTL  M$EMS.21 ERROR MESSAGE SUBROUTINES
000000  013746  000000G              M$EMS.21:
                                             MOV     RP.ADDR,-(SP)                      ;          7677
000004  004737  012254'                      JSR     PC,EMS.R1
000010  005726                               TST     (SP)+                              ;          7676
000012  000207                               RTS     PC

; Routine Size:  6 words,       Routine Base:  $CODE$ + 15060
; Maximum stack depth per invocation:  2 words
```

```
;    7679  1     !MMMBGNMSG (EMS_22)                       !CONTENTS OF DUP BUFFER         ZZZ
;    7680  1     !MMMEMS_DBN ();                                               !ZZZ
;    7681  1     !MMMENDMSG;                                                               !ZZZ
```

```
;    7682  1        BGNMSG (EMS_24);


                                            .SBTTL  EMS.24 ERROR MESSAGE SUBROUTINES
000000  004737  000000V             EMS.24::JSR     PC,M$EMS.24                     ;               7682
000004  104423                              TRAP    23
000006  000207                              RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 15074
; Maximum stack depth per invocation:  2 words


;    7683  2        PRINTB (EBD_24, .CDISK);                      ! 'DISK XXX WENT TO THE AVAILABLE STATE"
;    7684  2        EMS_R2 (.RP_ADDR);                            ! PRINTX RELEVANT RETPKT FIELDS
;    7685  1        ENDMSG;


                                            .SBTTL  M$EMS.24 ERROR MESSAGE SUBROUTINES
000000  013746  000000G             M$EMS.24:
                                            MOV     CDISK,-(SP)                     ;               7683
000004  012746  000000G                     MOV     #EBD.24,-(SP)
000010  012746  000002                      MOV     #2,(SP)
000014  010600                              MOV     SP,R0                           ; SP,*
000016  104414                              TRAP    14
000020  013716  000000G                     MOV     RP.ADDR,(SP)                    ;               7684
000024  004737  012124'                     JSR     PC,EMS.R2
000030  062706  000006                      ADD     #6,SP                           ;               7682
000034  000207                              RTS     PC

; Routine Size:  15 words,     Routine Base:  $CODE$ + 15104
; Maximum stack depth per invocation:  5 words
```

```
;    7686  1      BGNMSG (EMS_30);


                                          .SBTTL   EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.30::JSR      PC,M$EMS.30                    ;                 7686
000004  104423                           TRAP     23
000006  000207                           RTS      PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 15142
; Maximum stack depth per invocation:  2 words


;    7687  2      EMS_ERR ();                          ! PRINT ALL RELEVANT DATA ON DETECTING AN ERROR
;    7688  1      ENDMSG;


                                          .SBTTL   M$EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  014330'          M$EMS.30:
                                          JSR      PC,EMS.ERR                     ;                 7687
000004  000207                           RTS      PC                             ;                 7686

; Routine Size:  3 words,       Routine Base:  $CODE$ + 15152
; Maximum stack depth per invocation:  1 word


;    7689  1
;    7690  1      end
;    7691  1
;    7692  0      eludom


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


;                                 PSECT SUMMARY
;
;       Psect Name                        Words     Attributes
;        $OWN$                               82       RW ,  D  ,  LCL,  REL,  CON
;        $CODE$                            3384       RO ,  I  ,  LCL,  REL,  CON


;                           Library Statistics
;
;                                              -  ---- Symbols  --          Pages       Processing
;       File                                 Total    Loaded    Percent     Mapped      Time
;
;  DISK$USER2:[MLYNAR.ZRQ]ZRQAHO.L16;10        411       295        71         21        00:00.1
```

```
;                                COMMAND QUALIFIERS

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

; Size:           3216 code + 6709 data words
; Run Time:          01:53.7
; Elapsed Time:      09:09.1
; Lines/CPU Min:      4060
; Lexemes/CPU-Min: 37129
; Memory Used:   727 pages
; Compilation Complete
```



```
;  0   module ZRQAM3 (
;  0
;  0   %title 'RD/RX EXERCISER'
;  0               ident = 'V02.3',
;  0               addressing_mode (absolute),
;  0               environment (noeis)
;  0               ) =
;  0
;  1   begin
;  1
;  1   %sbttl 'DECLARATIONS'
;  1
;  1   library 'ZRQAHO.L16';                      ! RDRX EXERCISER GLOBAL LIBRARY
;  1
;  1   !MMM require 'BLSMAC.REQ';                 ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;  1   require 'HSAXAO.BLB';                      ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;  1
;  1   EQUALS;
;  1
;  1   forward routine                            ! ROUTINES APPEAR IN THIS ORDER
;  1       INIT_TEST : novalue,                   !    INDENTATION IMPLIES CALLED SUBROUTINE
;  1           DRIVER_INIT : novalue,
;  1               CTLR_INIT : novalue,
;  1                   INI_CTLR_DAT : novalue,
;  1                   REG_EXIST,
;  1                   VEC_BR_TEST,
;  1                   INT_GEN,
;  1                   HARD_INIT,
;  1                   INI_RRING : novalue,
;  1                   SET_CTLR_CHAR,
;  1               UNIT_INIT : novalue,
;  1                   DR_ERR : novalue,
;  1           ACCESS : novalue,
;  1       MULTI_DRIVE : novalue,
;  1           MD_INIT : novalue,
;  1               INIT_IO_BUFF : novalue,
;  1           FATAL_ERROR : novalue,
;  1           QIO_OK,
;  1               QIO_OUT,
;  1               QIO_GEN : novalue,
;  1                   GET_RANDOM : novalue,
;  1                   QIO_UNIT : novalue,
;  1                   QIO_FUNC : novalue,
;  1   !MMM                    DUP : NOVALUE,                           !ZZZ
;  1   !MMM                          DUPWRTDBN : NOVALUE,                   !ZZZ
;  1   !MMM                          DUPREDDBN : NOVALUE,                   !ZZZ
;  1   !MMM                               DUPCOMMAND : NOVALUE,             !ZZZ
;  1   !MMM                               DUPIDLE : NOVALUE,                !ZZZ
;  1                   QIO_LBN : novalue,
;  1                   QIO_SIZE : novalue,
```

```
!MMM                    DUP_COMPARE : NOVALUE,                          !ZZZ
                IO_RETPKT : novalue,
                    FSET_UPAR : novalue,
                    HARD_ERROR : novalue,
                        ERR_HRD_RTNE : novalue,
                        ERR_HRD_RTNE_APT : novalue,
                    UPD_IO_TALLY : novalue,
                        OVF_CHK : novalue,
                        ROUND_OUTPUT : novalue,
                    HOST_WRT_CHK,
!                       ERR_HRD_RTNE : novalue,
!                       ERR_HRD_RTNE_APT : novalue,
                    SWEEP : novalue,
                        RPS_REM,
                DR_RETPKT : novalue,
    AZINTO : L$ISR novalue,
        AZINT : novalue,
!           FATAL_ERROR : novalue,
            POLL_CRING : novalue,
            POLL_RRING : novalue,
!MMM                DUP_RSP  : NOVALUE,
!ZZZ
                DISK_RSP : novalue,
                    SEQUEN : novalue,
!MMM                    SCAN_ERRLOG : novalue,
!MMM                        ERR_SOFT_RTNE : novalue,
!MMM                        ERR_SOFT_RTNE_APT : novalue,
!MMM                        SOFT_ERROR : novalue,
                    DATAGM : novalue,
                        ERR_SOFT_RTNE : novalue,
                        ERR_SOFT_RTNE APT : novalue,
                        SOFT_ERROR : novalue;

external
    CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
                        ! RUN-TIME CONTROLLER STATUS TABLES
    CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
                        ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
    DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
                        ! DRIVER CONTROLLER TABLES
    DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
                        ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
    RDRX_ADDR : ref rdrx field (RC_REG),
                        ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
    IRDRX_ADDR : ref rdrx field (RC_REG),
                        ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
    BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                     !ZZZ
                        !BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS      !ZZZ
                        !RANDOM SEEK) MODE                              !ZZZ
    TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
                        ! STATISTICS TABLES
    T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
                        ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
!MMM    DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),   !BUFFER FOR DUP  ZZZ
```

```
1       !MMM                   !INFO FROM RECEIVE AND SEND COMMANDS               ZZZ
1           TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TK DIRECTION     ZZZ
1           RDM_CNT : WORD,      !NO. OF RANDOM NOS.            KEEP\\            ZZZ
1           RANDOM : VECTOR [RDM_LEN, WORD],      !RAND NO TABLE TOGET//HER        ZZZ
1           C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR FIELDS),
1                              ! STATISTICS TABLE FOR CONTROLLER ERRORS
1           MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
1                              ! MSCP PACKET POOL
1           IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
1                              ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
1           PKT_USE : vector [PKT_CNT, byte, signed],
1                              ! MSCP PACKET POOL ALLOCATION TABLE
1           RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
1                              ! RETURN PACKET POOL
1           RP_USE : vector [RP_CNT, byte, signed],
1                              ! RETURN PACKET POOL ALLOCATION TABLE
1           RP_INDX : word,    ! CURRENT RETURN PACKET INDEX
1           RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
1                              ! CURRENT RETURN PACKET ADDRESS
1           ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
1                              ! ERROR-LOG PACKET SAVE AREA
1           BUFF_ADDR : vector [MAX_BUF_CNT],               ! TABLE OF I/O BUFFER DESCRIPTORS
1           BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],  ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
1           IODQ : vector [IODQ_LEN, byte],                 ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
1           IODQ_IN : word,                                 ! I/O DONE QUEUE IN POINTER
1           IODQ_OUT : word,                                ! I/O DONE QUEUE OUT POINTER
1           ENTRY_REASON : byte,                            ! CURRENT OPERATOR COMMAND
1           EOP_FLAG : byte,                                ! END-OF-PASS FLAG
1           DUP_FLAGS : WORD,                               !DUP FLAGS                  ZZZ
1           CCTLR : word,                                   ! NUMBER OF "CURRENT" CONTROLLER
1           CDISK : word,                                   ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
1           CUOFF : word,                                   ! CURRENT UNIT CST OFFSET
1           CTLR_CNT : word,                                ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
1           DUR : vector [MAX_UNITS, byte],                 ! DROP UNIT REASON
1           QIO : vector [MAX_CTLR, byte],                  ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
1           FREE_MEM_ADDR,                                  ! START OF FREE MEMORY
1           BYTS_PER_QIO : word,                            ! SIZE (BYTES) OF AN I/O BUFFER
1           ST_CODE : word,                                 ! CURRENT STATUS CODE
1           SB_CODE : word,                                 ! CURRENT SUB-CODE
1           STEP : word,                                    ! CURRENT STEP IN HARD_INIT
1           OF_RC : signed word,                            ! OFFSET (0 OR 2) TO READ IP OR SA
1           SA_REG : word,                                  ! STORAGE FOR SA REGISTER READS AND WRITES
1           CMD_TIME : word,                                ! COMMAND TIMEOUT VALUE (IN SECONDS)
1           NEX : word,                                     ! NON-EXISTENT MEMORY TRAP INDICATOR
1           CRN_LOW : word,                                 ! COMMAND REF NUMBER OF LAST COMMAND SENT
1           CRN_HIGH : word,                                ! COMMAND REF NUMBER (HI ORDER)
1           TEMP1 : WORD,                                   !TEMPORARY STORAGE WD USED IN BGNCLN      !ZZZ
1           TEMP2 : WORD,                                   !TEMPORARY STORAGE WD USED IN BGNCLN      !ZZZ
1           CREDIT_BAL : word,                              ! CREDIT BALANCE
1           NEXT_PKT_USE : byte,                            ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
1           HOURS : byte,                                   ! TIME OF DAY (HOURS)
1           MINUTES : byte,                                 ! TIME OF DAY (MINUTES)
1           CLK_TICKS : word,                               ! TIME OF DAY (LINE-CLOCK TICKS)
```

```
;  1        CLK_PRESENT : byte,                          ! FLAG INDICATES IF LINE-CLOCK PRESENT
;  1        HOE_FLAG : byte,                             ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;  1        FORCED_ERROR : byte,                         ! "FORCED ERROR" DETECTED IN LAST READ
;  1        FER0_LBN : word,                             !LO LBN ADR OF THE "FORCED ERROR" BLOCK
;  1        FER1_LBN : word,                             !HI LBN ADR OF THE "FORCED ERROR" BLOCK          ZZZ
;  1    !   FER_LBN : word,                              ! LBN OF THE "FORCED ERROR" BLOCK                 ZZZ
;  1        FER_BC : word,                               ! BYTE COUNT OF THE "FORCED ERROR" BLOCK          ZZZ
;  1        INIT_OCCURED : byte,                         ! EXERCISER INITIALIZATION COMPLETE
;  1        ADDR_VECT_OK : byte,                         ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;  1
;  1        TYPER : VECTOR [MAX_UNITS, WORD],     !READ I/O COUNTER                 ZZZ MMM
;  1        TYPEW : VECTOR [MAX_UNITS, WORD],     !WRITE I/O COUNTER                ZZZ MMM
;  1        BAL_IN_PROGRESS : VECTOR [MAX_UNITS, WORD], !FLAG SET TO BALANCE I/O TYPES      ZZZ MMM
;  1        FORCE_WR : VECTOR [MAX_UNITS, WORD],          !FLAG TO ISSUE MORE WRITES WHEN BALANCING MMM
;  1        CSR_MEM : WORD,                               !MMM
;  1        CSR_ADD : WORD,                               !MMM
;  1        TST_PAR : WORD,                               !MMM
;  1        MAN_TST : WORD,                               !MMM
;  1   !MMM     S_PATTERN : WORD,                      !PATTERN WRITTEN TO DBNS          ZZZ
;  1   !MMM     S_DUPPKT : WORD,                    !DBN BYTE COUNTER                 ZZZ
;  1        P_INDEX : SIGNED WORD,            !CURRENT MESSAGE PACKET INDEX        ZZZ
;  1        RD_COUNT : WORD,                  ! NUMBER OF WINCHESTER UNITS         ZZZ
;  1        BRLEVEL : word,                   ! CURRENT DEVICE'S BR LEVEL          ZZZ
;  1        D_FAIL : BYTE,                    !SIGNIFIES DUP TYPE ERROR            ZZZ
;  1        DBM12,
;  1        DBM18,
;  1        DBM19,
;  1        DBM20,
;  1        DBM21,
;  1        DBM22,
;  1        DBM23,
;  1        DBM25,
;  1        DBM26,
;  1        DBM27,
;  1        DBM29,
;  1        DBM108,
;  1        DBM109,
;  1        DBM111,
;  1        DBM112,
;  1        DBM120,
;  1        DBM121,
;  1        DBM123,                                              !MMM
;  1        DBM125,                                              !MMM
;  1        DBM126,                                              !MMM
;  1        DBM127,                                              !MMM
;  1        DBM128,                                              !MMM
;  1        EH_0,                                                !ZZZ
;  1        EH_1,                                                !ZZZ
;  1        EH_2,                                                !ZZZ
;  1        EH_3,                                                !ZZZ
;  1        EH_4,                                                !ZZZ
;  1        EH_5,                                                !ZZZ
;  1        EH_6,                                                !ZZZ
```

```
        EH_7,                                                          !ZZZ
        EH_8,                                                          !ZZZ
        EH_9,                                                          !ZZZ
        EH_10,                                                         !ZZZ
        EH_12,                                                         !ZZZ
        EH_13,                                                         !ZZZ
        MSG_02,
        MSG_03,
        EGS_02,
        EGD_10,
        EGD_11,
        EGD_12,
        EGD_13,
        EGD_14,
        EGD_15,
        EGD_16,
        EGD_17,
        EGD_18,
        EGD_19,
        EGD_20,
        EGD_21,
        EGD_22,
        EGD_23,
        EGD_24,
        EGH_30,
        DF_MSG,
        HRD_MSG,
        SFT_MSG,
        HRD_SUB,
        CRLF,
        SWP_ERROR : word,                          ! HARD ERROR LIMIT FOR DROPPING UNIT
        SWP_XFER : word,                           ! TRANSFER LIMIT FOR DROPPING UNIT
        SWP_FLAGS : word,                          ! FLAGS (SEE DOCUMENTATION)
        !MMM DUPROUND  : WORD,                            !DUP TESTING RATIO          ZZZ
        SWP_RAT : word,                            ! RD51/52 OPERATION RATIO
        SWP_DPAT : word,                           ! DATA PATTERN NUMBER
        SWP_UCNT : word,                           ! USER DATA PATTERN COUNT
        SWP_TIME : word,                           ! TIME OF DAY
        SWP_UDPAT : vector [MAX_UDP_CNT, word],    ! USER DATA PATTERN
        L$LUN,
        L$UNIT;

    psect
        own = $GGG$(read, nowrite, execute, local, concatenate);

    own
        COMM_AREA : blockvector [MAX_CTLR, COMM_LEN, word] field (COM_FIELDS),
                        ! COMMUNICATIONS AREA BETWEEN HOST AND AZTEC CONTROLLERS
    !!ZZZ    BST : vector [MAX_UNITS, word, signed],
                        ! BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
        DPST : vector [MAX_UNITS, byte],           ! DATA PATTERN SEQUENCE TABLE
        MAX_LBN : vector [MAX_UNITS, word],        ! LARGEST LBN ALLOWED
        STORAGE : vector [MAX_UNITS, word],        ! DUMMY STORAGE
```

```
;       1       ICOM_ADDR : ref block [COMM_LEN, word] field (COM_FIELDS),
;       1                                   ! ADDRESS OF INTERRUPTING CONTROLLER'S COMMUNICATION AREA
;       1       ICST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;       1                                   ! ADDRESS OF INTERRUPTING CONTROLLER'S CST
;       1       IDCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;       1                                   ! ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;       1       INT_ADDR : vector [MAX_CTLR] initial (AZINT0 %(, AZINT1, AZINT2, AZINT3)%),
;       1                                   ! INTERRUPT SERVICE ROUTINE ADDRESS TABLE
;       1   !!ZZZ    RDM_CNT : word initial (RDM_LEN),              ! NUMBER OF RANDOM NUMBERS   \ KEEP
;       1   !!ZZZ    RANDOM : vector [RDM_LEN, word],               ! RANDOM NUMBER TABLE         / TOGETHER
;       1       ICTLR : word,                         ! INTERRUPTING CONTROLLING NUMBER
;       1       EL_FLUSH : vector [MAX_CTLR, word],   ! STOP QIO TO PROCESS ERROR LOGS     MMM
;       1       MX1 : signed word,                    ! MSCP PKT INDEX FOR FIRST QIO
;       1       MX2 : signed word,                    ! MSCP PKT INDEX FOR SECOND QIO
;       1       MAD1 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;       1                                   ! ADDRESS OF MSCP PACKET FOR FIRST QIO
;       1       MAD2 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;       1                                   ! ADDRESS OF MSCP PACKET FOR SECOND QIO
;       1       LAST_PKT : blockvector [MAX_CTLR, LAST_PKT_LEN, word] field (LAST_PKT_FIELDS),
;       1                                   ! SAVE AREA FOR INFO ABOUT LAST RESPONSE PACKET
;       1
;       1       RNDY0 : WORD,                                   !32-BIT RANDOM PATTERN LO WD          ZZZ
;       1       RNDY1 : WORD,                                   !32-BIT RANDOM PATTERN HI WD          ZZZ
;       1       FRAME_CNT : WORD,                               !WHICH 7-BIT FRAME OF R_STRING IN USE  ZZZ
;       1       R_STRING: WORD,                                 !BITS USED IN PATTERN SELECTION       ZZZ
;       1       RNDYIN :  vector [9, word] initial (%o'127102',  !NINE SEED WORDS                      ZZZ
;       1                 %o'143662', %o'036750', %o'121624', %o'023267',                             !ZZZ
;       1                 %o'036561', %o'063714', %o'560255', %o'134230'),                            !ZZZ
;       1       RNDMS0 :  vector [8, word] initial (%o'17',     !MASK FOR LOW WORD                     ZZZ
;       1                 %o'377', %o'7777', %o'177777', %o'177777',                                  !ZZZ
;       1                 %o'177777', %o'177777', %o'177777'),                                        !ZZZ
;       1       RNDMS1 :  vector [8, word] initial (%o'0000',   !MASK FOR HIGH WORD                    ZZZ
;       1                 %o'0000', %o'0000', %o'0000', %o'17',                                       !ZZZ
;       1                 %o'377', %o'7777', %o'177777'),                                             !ZZZ
;       1
;       1       PAT02 : vector [2] initial (1,                          ! PATTERN 2
;       1               %o'000000'),
;       1       PAT03 : vector [2] initial (1,                          ! PATTERN 3
;       1               %o'177777'),
;       1       PAT04 : vector [2] initial (1,                          ! PATTERN 4
;       1               %o'105613'),
;       1       PAT05 : vector [2] initial (1,                          ! PATTERN 5
;       1               %o'031463'),
;       1       PAT06 : vector [2] initial (1,                          ! PATTERN 6
;       1               %o'030221'),
;       1       PAT07 : vector [17] initial (16,                        ! PATTERN 7
;       1               %o'000001', %o'000003', %o'000007', %o'000017',
;       1               %o'000037', %o'000077', %o'000177', %o'000377',
;       1               %o'000777', %o'001777', %o'003777', %o'007777',
;       1               %o'017777', %o'037777', %o'077777', %o'177777'),
;       1       PAT08 : vector [17] initial (16,                        ! PATTERN 8
;       1               %o'177776', %o'177774', %o'177770', %o'177760',
;       1               %o'177740', %o'177700', %o'177600', %o'177400',
```

```
                %o'177000', %o'176000', %o'174000', %o'170000',
                %o'160000', %o'140000', %o'100000', %o'000000' ),
        PAT09 : vector [17] initial (16,                          ! PATTERN 9
                rep 3 of (%o'000000'), rep 3 of (%o'177777'),
                rep 2 of (%o'000000'), rep 2 of (%o'177777'),
                %o'000000', %o'177777', %o'000000', %o'177777',
                %o'000000', %o'177777'),
        PAT10 : vector [2] initial (1,                            ! PATTERN 10
                %o'133331'),
        PAT11 : vector [17] initial (16,                          ! PATTERN 11
                rep 3 of (%o'052525'), rep 3 of (%o'125252'),
                rep 2 of (%o'052525'), rep 2 of (%o'125252'),
                %o'052525', %o'125252', %o'052525', %o'125252',
                %o'052525', %o'125252'),
        PAT12 : vector [21] initial (20,                          ! PATTERN 12
                rep 3 of (%o'026455'), rep 3 of (%o'151322'),
                rep 2 of (%o'026455'), rep 2 of (%o'151322'),
                rep 2 of (%o'026455'),
                %o'151322', %o'026455', %o'151322', %o'026455',
                %o'151322', %o'026455', %o'151322', %o'026455' ),
        PAT13 : vector [2] initial (1,                            ! PATTERN 13
                %o'066666'),
        PAT14 : vector [17] initial (16,                          ! PATTERN 14
                %o'000001', %o'000002', %o'000004', %o'000010',
                %o'000020', %o'000040', %o'000100', %o'000200',
                %o'000400', %o'001000', %o'002000', %o'004000',
                %o'010000', %o'020000', %o'040000', %o'100000' ),
        PAT15 : vector [17] initial (16,                          ! PATTERN 15
                %o'177776', %o'177775', %o'177773', %o'177767',
                %o'177757', %o'177737', %o'177677', %o'177577',
                %o'177377', %o'176777', %o'175777', %o'173777',
                %o'167777', %o'157777', %o'137777', %o'077777' ),
        PAT16 : vector [17] initial (16,                          ! PATTERN 16
                rep 3 of (%o'133331'), rep 3 of (%o'155554'),
                rep 2 of (%o'133331'), rep 2 of (%o'155554'),
                %o'133331', %o'155554', %o'133331', %o'155554',
                %o'133331', %o'155554'),
        PAT17 : vector [22] initial (21,                          ! PATTERN 17
                %o'000000', rep 2 of (%o'106466'),
                rep 3 of (%o'071311'), rep 4 of (%o'106466'),
                rep 5 of (%o'071311'), rep 6 of (%o'106466')),
        PAT18 : vector [22] initial (21,                          ! PATTERN 18
                %o'106466', %o'000000', %o'071311',
                rep 3 of (%o'106466'), rep 4 of (%o'071311'),
                rep 5 of (%o'106466'), rep 6 of (%o'071311')),
        PAT19 : vector [22] initial (21,                          ! PATTERN 19
                %o'000000', rep 2 of (%o'134631'),
                rep 3 of (%o'043146'), rep 4 of (%o'134631'),
                rep 5 of (%o'043146'), rep 6 of (%o'134631')),
        PAT20 : vector [22] initial (21,                          ! PATTERN 20
                %o'134631', %o'000000', %o'043146',
                rep 3 of (%o'134631'), rep 4 of (%o'043146'),
                rep 5 of (%o'134631'), rep 6 of (%o'043146')),
```

```
        PAT21 : vector [2] initial (1,                          ! PATTERN 21
                %'000000'),                                     !    (LBN)
        DPA_TBL : vector [DP_CNT] initial                       ! DATA PATTERN ADDRESS TABLE
                (RDM_CNT, PAT02, PAT03, PAT04, PAT05,
                PAT06, PAT07, PAT08, PAT09, PAT10, PAT11,
                PAT12, PAT13, PAT14, PAT15, PAT16, PAT17,
                PAT18, PAT19, PAT20, PAT21),
        BST_CNT : word initial (0),                     ! CURRENT SEQUENTIAL BLOCK COUNT
        BST_DEV : word initial (0),                     ! CURRENT SEQUENTIAL BLOCK DEVICE
        CURRENT_VECTOR : word,                          ! CURRENT DEVICE'S VECTOR ADDRESS
!ZZZ    BRLEVEL : word,                                 ! CURRENT DEVICE'S BR LEVEL            ZZZ
!MMM    DUOFF   : WORD,                                 !DUP OFFSET INTO CST                   ZZZ
        DRS_START,                                      ! START OF THE SUPERVISOR
        PAR_TSD : word,                                 !MMM
        APT_MODE : byte initial (byte (FALSE)),         ! FLAG SET IF EXERCISER RUNNING UNDER APT
        MAIL_BOX_TESTNUM,                               ! ADDRESS OF TEST NUMBER LOCATION IN APT MAIL-BOX
        MAIL_BOX_SUBTST,                                ! ADDRESS OF SUB-TEST NUMBER LOCATION IN APT MAIL BOX
        COMPARE_DATA : byte,                            ! FLAG CLEARED TO BYPASS HOST COMPARES
        DRS_FLAGS: word,                                ! FLAGS USED IN START/RESTART OF THE EXERCISER
        RD_MAX_SEQ_CNT : word,                          ! COUNT USED IN SEQUENTIAL ACCESS OPERATIONS
        RX_MAX_SEQ_CNT : word;                          !

external routine
        NEX_TRAP : L$ISR novalue,
        PARITY : novalue,                               !MMM
        TIME : L$ISR novalue,
        SET_CPAR : novalue,
        SET_UPAR : novalue,
        OUT_IODQ,
        IN_IODQ : novalue,
        GET_PKT,
        PUT_PKT : novalue,
        GET_RETPKT,
        PUT_RETPKT : novalue,
        GET_IO_BUFF : novalue,
        PUT_IO_BUFF : novalue,
        PUTA_BUFF : novalue,
        SEND,
        WAIT : novalue,
        MODULAS,                                                        !ZZZ
        DROP_CTLR : novalue,
        DRV_CTLERR : novalue,
        EMS_R2 : novalue,                                               !MMM
        EMS_R1 : novalue,                                               !MMM
        EMS_EL : novalue,
        EMS_CMP : novalue,
        EMS_ERR : novalue,
        EMS_10 : novalue,
        EMS_12 : novalue,
        EMS_13 : novalue,
        EMS_14 : novalue,
        EMS_18 : novalue,
        EMS_21 : novalue,
```

```
;    2165  1        !MMMEMS_22 : NOVALUE,                                  !ZZZ
;    2166  1        EMS_24 : novalue,
;    2167  1        EMS_30 : novalue;
```

```
;  1   %sbttl 'TEST SECTION'
;  1
;  1
;  1   !+
;  1   !       THIS SECTION CONTAINS THE TOP LEVEL TEST CODE FOR THE RDRX EXERCISER.
;  1   !       THE EXERCISER CONSISTS OF ONE TEST WHICH IS SUBDIVIDED INTO A NUMBER OF
;  1   !       SUBTESTS. ALL SUBTESTS ARE DECLARED WITHIN THIS BLOCK.
;  1   !-
;  1
;  1
;  3   BGNTST;
;  3
;  3   local
;  3       DUMMY_0 : word,
;  3       DUMMY_1 : word;
;  3
;  3
;  3   EOP_FLAG = TRUE;                                   ! ASSUME NO UNIT AVAILABLE
;  3   COMPARE_DATA = TRUE;                               ! ALLOW HOST COMAPRES IF ASKED FOR
;  3   DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT);         !CLEAR DUP INIT FLAG     ZZZ
;  3   HOE_FLAG = FALSE;                                  ! ASSUME 'HOE' FLAG NOT SET
;  3   FORCED_ERROR = FALSE;                              ! INITIALIZE "FORCED ERROR" FLAG
;  3
;  3
;  3   incr I from 0 to PKT_CNT - 1 do                    ! INITIALIZE PACKET AREA
;  4       begin
;  4
;  4       incr J from 0 to PKT_LEN - 1 do
;  4           MSCP PKT [.I, .J, 0, 16, 0] = 0;
;  4
;  4       MSCP_PKT [.I, RSP_RECEIVED] = FALSE;
;  3       end;
;  3
;  3   incr I from 0 to RP_CNT   1 do                     ! INITIALIZE RESPONSE SAVE AREA
;  3       incr J from 0 to RP_LEN - 1 do
;  3           RETPKT [.I, .J, 0, 16, 0] = 0;
;  3
;  3   incr I from 0 to EP_CNT do                         ! INITIALIZE ERROR LOG SAVE AREA
;  4       begin
;  4
;  4       incr J from 0 to EP_LEN - 1 do
;  4           ELOG_PKT [.I, .J, 0, 16, 0] = 0;
;  4
;  4       ELOG_PKT [.I, EL CONTENTS] = EMPTY;
;  3       end;
;  3
;  4   if BIT_TST (SWP_FLAGS, SWF_CWC)                    ! NO SIMULTANEOUS CNTR/HOST WRIE CHECKS
;  3   then
;  3       SWP_FLAGS = .SWP_FLAGS and (not SWF_HWC);
;  3
;  4   if BIT_TST (SWP_FLAGS, SWF_RDM)                    ! NO SIMULTANEOUS RANDOM/SEQUENTIAL SELECTS
;  3   then
;  3       SWP_FLAGS = .SWP_FLAGS and (not SWF_SEQ);
```

```
;      3
;      3   if not .INIT_OCCURED
;      3   then
;      4       begin
;      4       DRS_START = .FREE MEM ADDR + 2 + (..FREE MEM_ADDR * 2);          ! START OF SUPERVISOR
;      4
;      4
;      4
;      4   !   THE FOLLOWING DETERMINES WH-THER THE TEST IS TO BE RUN IN APT MODE:         !Z
ZZ
;      4                                                                                  !Z
ZZ
;      5   IF BIT_TST (SWP FLAGS, SWF APT)                                !IF APT         !Z
ZZ
;      4       then
;      5           begin
;      5           APT_MODE = TRUE;
;      5           MAIL BOX_TESTNUM = .DRS START + %o 62  + %o'6 ;       ! APT MAIL BOX IS OFFSET AT OCTAL 62 FROM
;      5           MAIL_BOX_SUBTST = .DRS START + %o 62  + %o 4 ;        ! BEGINNING OF SUPERVISOR
;      4           end;
;      4
;      4
;      4       NEX = FALSE;                                               ! CHECK IF LINE-CLOCK PRESENT
;      4       CLK_PRESENT = FALSE;                                       !
;      4       SETVEC (4, NEX_TRAP, PRIO7);                               ! SET TRAP CATCHER ADDRESS
;      4       DUMMY_0 = .LINE_CLOCK;                                     ! TRY TO ADDRESS THE CLOCK
;      4       DUMMY_1 = 0;                                               ! DUMMY INSTRUCTION
;      4       CLRVEC (4);                                                ! RETURN LOC 4 TO THE SUPERVISOR
;      4
;      4
;      4       if not .NEX
;      4       then
;      5           begin
;      5           CLK_PRESENT = TRUE;                                    ! SET FLAG IF CLOCK PRESENT
;      5           CLK_TICKS = 0;                                         ! INITIALIZE THE LINE CLOCK TICK COUNT
;      5           HOURS = .SWP_TIME / 100;                               ! TIME OF DAY (HOURS)
;      5           MINUTES = (.SWP TIME mod 100) + 1;                     ! TIME OF DAY (MINUTES)
;      5
;      5           while .MINUTES gequ 60 do                              ! NORMALIZE MINUTES
;      6               begin
;      6               MINUTES = .MINUTES   60;
;      6               HOURS = .HOURS + 1;
;      5               end;
;      5
;      5           HOURS = .HOURS mod 24;                                 ! NORMALIZE HOURS
;      4           end;
;      4
;      4       NEX = FALSE;                                               ! CHECK IF MEMORY CSR PRESENT MMM
;      4       CSR_MEM = FALSE;                                           ! MMM
;      4       SETVEC (4, NEX TRAP, PRIO7);                               ! SET TRAP CATCHER ADDRESS
;      4       DUMMY_0 = ..CSR ADD;                                       ! TRY TO ADDRESS THE MEMORY CSR MMM
;      4       DUMMY_1 = 0;                                               ! DUMMY INSTRUCTION MMM
;      4       CLRVEC (4);                                                ! RETURN LOC 4 TO THE SUPERVISOR MMM
;      4
;      4
;      4       if not .NEX                                                ! MMM
```

```
;  4            then                                                         ! MMM
;  5                begin                                                    ! MMM
;  5                CSR_MEM = TRUE;                                          ! SET FLAG IF CSR PRESENT  MMM
;  4                end;                                                     ! MMM
;  4
;  3            end;
;  3
;  3
.  3        if .CLK_PRESENT
;  3        then
;  4            begin
;  4            SETVEC (%o'100', TIME, PRI06);                              ! LINE CLOCK VECTOR
;  4            LINE_CLOCK = BIT6;                                           ! START THE CLOCK
;  3            end;
;  3
;  3        if .CSR_MEM and .TST_PAR                                         ! MEMORY CSR PRESENT AND MMM
;  3        then                                                             ! ENABLE MEMORY PARITY REQUESTED MMM
;  4            begin                                                        ! THEN ENABLE MEMORY PARITY MMM
;  4            SETVEC (%o'114', PARITY, PRI07);                            ! MEMORY PARITY VECTOR MMM
;  4             CSR_ADD = %o'1';                                            ! ENABLE MEMORY PARITY MMM
;  3            end;
;  3
;  3        RFLAGS (DRS_FLAGS);                                              ! READ DRS FLAGS INTO LOC DRS_FLAGS
;  3
;  3        if BIT_TST (DRS_FLAGS, HOE) eql HOF
;  3        then
;  3            HOE_FLAG = TRUE;                                             ! SET FLAG IF  HOE' SET
;  3
;  3
;  3        INIT_TEST ();                                                    ! INITIALIZE TEST ENVIRONMENT
;  3
;  3        incr CTLR from 0 to (MAX_CTLR - 1) do                            ! FOR EVERY CONTROLLER
;  3
;  3            if (.CST [.CTLR, STATE] eql ONLINE) and                      ! IF CONTROLLER ONLINE
;  3               (.DCT [.CTLR, STAT] eql ONLINE) and
:  4               (.CST [.CTLR, U CNT] gequ 0)
;  3            then
;  3                incr OFFSET from (0 + OF UN) to ((UNITS_PER_CNTR   1) * UNIT_SIZE + 4) by UNIT SIZE do
;  3
;  3                    if .CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE  ! IF AT LEAST ONE UNIT ALIVE
;  3                    then
;  4                        begin
;  4                        EOP_FLAG = FALSE;                                ! NOT END OF PASS
;  4                        exitloop;
;  3                        end;
;  3
;  3        if not .EOP_FLAG
;  3        then
;  3            MULTI_DRIVE ();                                              ! RUN MULTI DRIVE TEST
;  1        ENDTST;
```

.TITLE ZRQAM3 RD/RX EXERCISER

```
                                        .IDENT  /V02.3/
                                        .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                          EL.FLUSH:
                                        .BLKW   1
000110                          MX1:    .BLKW   1
000112                          MX2:    .BLKW   1
000114                          MAD1:   .BLKW   1
000116                          MAD2:   .BLKW   1
000120                          LAST.PKT:
                                        .BLKW   3
000126                          RNDY0:  .BLKW   1
000130                          RNDY1:  .BLKW   1
000132                          FRAME.CNT:
                                        .BLKW   1
000134                          R.STRING:
                                        .BLKW   1
000136  127102                  RNDYIN: .WORD   -50676
000140  143662                          .WORD   -34116
000142  036750                          .WORD   36750
000144  121624                          .WORD   -56154
000146  023267                          .WORD   23267
000150  036561                          .WORD   36561
000152  063714                          .WORD   63714
000154  160255                          .WORD   -17523
000156  134230                          .WORD   -43550
000160  000017                  RNDMS0: .WORD   17
000162  000377                          .WORD   377
000164  007777                          .WORD   7777
000166  177777                          .WORD   -1
000170  177777                          .WORD   -1
000172  177777                          .WORD   -1
000174  177777                          .WORD   -1
000176  177777                          .WORD   -1
000200  000000                  RNDMS1: .WORD   0
000202  000000                          .WORD   0
000204  000000                          .WORD   0
```

```
000206  000000              .WORD   0
000210  000017              .WORD   17
000212  000377              .WORD   377
000214  007777              .WORD   7777
000216  177777              .WORD    1
000220  000001     PAT02:   .WORD   1
000222  000000              .WORD   0
000224  000001     PAT03:   .WORD   1
000226  177777              .WORD   -1
000230  000001     PAT04:   .WORD   1
000232  105613              .WORD   -72165
000234  000001     PAT05:   .WORD   1
000236  031463              .WORD   31463
000240  000001     PAT06:   .WORD   1
000242  030221              .WORD   30221
000244  000020     PAT07:   .WORD   20
000246  000001              .WORD   1
000250  000003              .WORD   3
000252  000007              .WORD   7
000254  000017              .WORD   17
000256  000037              .WORD   37
000260  000077              .WORD   77
000262  000177              .WORD   177
000264  000377              .WORD   377
000266  000777              .WORD   777
000270  001777              .WORD   1777
000272  003777              .WORD   3777
000274  007777              .WORD   7777
000276  017777              .WORD   17777
000300  037777              .WORD   37777
000302  077777              .WORD   77777
000304  177777              .WORD    1
000306  000020     PAT08:   .WORD   20
000310  177776              .WORD   -2
000312  177774              .WORD   -4
000314  177770              .WORD   -10
000316  177760              .WORD   -20
000320  177740              .WORD    40
000322  177700              .WORD   -100
000324  177600              .WORD   -200
000326  177400              .WORD   -400
000330  177000              .WORD   -1000
000332  176000              .WORD   -2000
000334  174000              .WORD   -4000
000336  170000              .WORD   -10000
000340  160000              .WORD   -20000
000342  140000              .WORD   -40000
000344  100000              .WORD   -100000
000346  000000              .WORD   0
000350  000020     PAT09:   .WORD   20
000352  000000              .WORD   0
000354  000000              .WORD   0
000356  000000              .WORD   0
```

```
000360  177777             .WORD   -1
000362  177777             .WORD   -1
000364  177777             .WORD   -1
000366  000000             .WORD   0
000370  000000             .WORD   0
000372  177777             .WORD   -1
000374  177777             .WORD   -1
000376  000000             .WORD   0
000400  177777             .WORD   -1
000402  000000             .WORD   0
000404  177777             .WORD   -1
000406  000000             .WORD   0
000410  177777             .WORD   -1
000412  000001     PAT10:  .WORD   1
000414  133331             .WORD   -44447
000416  000020     PAT11:  .WORD   20
000420  052525             .WORD   52525
000422  052525             .WORD   52525
000424  052525             .WORD   52525
000426  125252             .WORD   -52526
000430  125252             .WORD   -52526
000432  125252             .WORD   -52526
000434  052525             .WORD   52525
000436  052525             .WORD   52525
000440  125252             .WORD   -52526
000442  125252             .WORD   -52526
000444  052525             .WORD   52525
000446  125252             .WORD   -52526
000450  052525             .WORD   52525
000452  125252             .WORD   -52526
000454  052525             .WORD   52525
000456  125252             .WORD   -52526
000460  000024     PAT12:  .WORD   24
000462  026455             .WORD   26455
000464  026455             .WORD   26455
000466  026455             .WORD   26455
000470  151322             .WORD   -26456
000472  151322             .WORD   -26456
000474  151322             .WORD   -26456
000476  026455             .WORD   26455
000500  026455             .WORD   26455
000502  151322             .WORD   -26456
000504  151322             .WORD   -26456
000506  026455             .WORD   26455
000510  026455             .WORD   26455
000512  151322             .WORD   -26456
000514  026455             .WORD   26455
000516  151322             .WORD   -26456
000520  026455             .WORD   26455
000522  151322             .WORD   -26456
000524  026455             .WORD   26455
000526  151322             .WORD   -26456
000530  026455             .WORD   26455
```

```
000532  000001         PAT13:  .WORD   1
000534  066666                 .WORD   66666
000536  000020         PAT14:  .WORD   20
000540  000001                 .WORD   1
000542  000002                 .WORD   2
000544  000004                 .WORD   4
000546  000010                 .WORD   10
000550  000020                 .WORD   20
000552  000040                 .WORD   40
000554  000100                 .WORD   100
000556  000200                 .WORD   200
000560  000400                 .WORD   400
000562  001000                 .WORD   1000
000564  002000                 .WORD   2000
000566  004000                 .WORD   4000
000570  010000                 .WORD   10000
000572  020000                 .WORD   20000
000574  040000                 .WORD   40000
000576  100000                 .WORD   -100000
000600  000020         PAT15:  .WORD   20
000602  177776                 .WORD   -2
000604  177775                 .WORD   -3
000606  177773                 .WORD   -5
000610  177767                 .WORD    11
000612  177757                 .WORD   -21
000614  177737                 .WORD   -41
000616  177677                 WORD     101
000620  177577                 .WORD   -201
000622  177377                 .WORD   -401
000624  176777                 .WORD    1001
000626  175777                 .WORD   -2001
000630  173777                 .WORD   -4001
000632  167777                 .WORD   -10001
000634  157777                 .WORD   -20001
000636  137777                 .WORD   -40001
000640  077777                 .WORD   77777
000642  000020         PAT16:  .WORD   20
000644  133331                 .WORD   -44447
000646  133331                 .WORD   -44447
000650  133331                 .WORD   -44447
000652  155554                 .WORD   -22224
000654  155554                 .WORD   -22224
000656  155554                 .WORD   -22224
000660  133331                 .WORD   -44447
000662  133331                 .WORD   -44447
000664  155554                 .WORD   -22224
000666  155554                 .WORD   -22224
000670  133331                 .WORD   -44447
000672  155554                 .WORD   -22224
000674  133331                 .WORD   -44447
000676  155554                 .WORD   -22224
000700  133331                 .WORD   -44447
000702  155554                 .WORD   -22224
```

```
000704  000025        PAT17:  .WORD   25
000706  000000                .WORD   0
000710  106466                .WORD   -71312
000712  106466                .WORD   -71312
000714  071311                .WORD   71311
000716  071311                .WORD   71311
000720  071311                .WORD   71311
000722  106466                .WORD   -71312
000724  106466                .WORD   -71312
000726  106466                .WORD   -71312
000730  106466                .WORD   -71312
000732  071311                .WORD   71311
000734  071311                .WORD   71311
000736  071311                .WORD   71311
000740  071311                .WORD   71311
000742  071311                .WORD   71311
000744  106466                .WORD   71312
000746  106466                .WORD   -71312
000750  106466                .WORD   -71312
000752  106466                .WORD   -71312
000754  106466                .WORD   -71312
000756  106466                .WORD   -71312
000760  000025        PAT18:  .WORD   25
000762  106466                .WORD   -71312
000764  000000                .WORD   0
000766  071311                .WORD   71311
000770  106466                .WORD   -71312
000772  106466                .WORD   -71312
000774  106466                .WORD   -71312
000776  071311                .WORD   71311
001000  071311                .WORD   71311
001002  071311                .WORD   71311
001004  071311                .WORD   71311
001006  106466                .WORD   -71312
001010  106466                .WORD   -71312
001012  106466                .WORD   -71312
001014  106466                .WORD   -71312
001016  106466                .WORD   -71312
001020  071311                .WORD   71311
001022  071311                .WORD   71311
001024  071311                .WORD   71311
001026  071311                .WORD   71311
001030  071311                .WORD   71311
001032  071311                .WORD   71311
001034  000025        PAT19:  .WORD   25
001036  000000                .WORD   0
001040  134631                .WORD   -43147
001042  134631                .WORD   -43147
001044  043146                .WORD   43146
001046  043146                .WORD   43146
001050  043146                .WORD   43146
001052  134631                .WORD   -43147
001054  134631                .WORD   -43147
```

```
001056  134631                 .WORD    43147
001060  134631                 .WORD   -43147
001062  043146                 .WORD   43146
001064  043146                 .WORD   43146
001066  043146                 .WORD   43146
001070  043146                 .WORD   43146
001072  043146                 .WORD   43146
001074  134631                 .WORD   -43147
001076  134631                 .WORD   -43147
001100  134631                 .WORD   -43147
001102  134631                 .WORD   -43147
001104  134631                 .WORD   -43147
001106  134631                 .WORD   -43147
001110  000025        PAT20:   .WORD   25
001112  134631                 .WORD   -43147
001114  000000                 .WORD   0
001116  043146                 .WORD   43146
001120  134631                 .WORD   -43147
001122  134631                 .WORD   -43147
001124  134631                 .WORD   -43147
001126  043146                 .WORD   43146
001130  043146                 .WORD   43146
001132  043146                 .WORD   43146
001134  043146                 .WORD   43146
001136  134631                 .WORD   -43147
001140  134631                 .WORD   -43147
001142  134631                 .WORD   -43147
001144  134631                 .WORD   -43147
001146  134631                 .WORD   -43147
001150  043146                 .WORD   43146
001152  043146                 .WORD   43146
001154  043146                 .WORD   43146
001156  043146                 .WORD   43146
001160  043146                 .WORD   43146
001162  043146                 .WORD   43146
001164  000001        PAT21:   .WORD   1
001166  000000                 .WORD   0
001170  000000G       DPA TBL: .WORD   RDM.CNT
001172  000220'                .WORD   PAT02
001174  000224'                .WORD   PAT03
001176  000230'                .WORD   PAT04
001200  000234'                .WORD   PAT05
001202  000240'                .WORD   PAT06
001204  000244'                .WORD   PAT07
001206  000306'                .WORD   PAT08
001210  000350'                .WORD   PAT09
001212  000412'                .WORD   PAT10
001214  000416'                .WORD   PAT11
001216  000460'                .WORD   PAT12
001220  000532'                .WORD   PAT13
001222  000536                 .WORD   PAT14
001224  000600'                .WORD   PAT15
001226  000642'                .WORD   PAT16
```

```
001230  000704'                         .WORD   PAT17
001232  000760'                         .WORD   PAT18
001234  001034'                         .WORD   PAT19
001236  001110'                         .WORD   PAT20
001240  001164'                         .WORD   PAT21
001242  000000          BST.CNT::.WORD  0
001244  000000          BST.DEV::.WORD  0
001246                  CURRENT.VECTOR:
                                .BLKW   1
001250                  DRS.START:
                                .BLKW   1
001252                  PAR.TSD::.BLKW  1
001254                  APT.MODE:
001254     000                  .BYTE   0
                                .EVEN
001256                  MAIL.BOX.TESTNUM:
                                .BLKW   1
001260                  MAIL.BOX.SUBTST:
                                .BLKW   1
001262                  COMPARE.DATA:
                                .BLKB   1
                                .EVEN
001264                  DRS.FLAGS:
                                .BLKW   1
001266                  RD.MAX.SEQ.CNT:
                                .BLKW   1
001270                  RX.MAX.SEQ.CNT:
                                .BLKW   1

                                .GLOBL  CST, CST.ADDR, DCT, DCT.ADDR, RDRX.ADDR
                                .GLOBL  IRDRX.ADDR, BST, TALLY, T.ADDR
                                .GLOBL  TRK.SGN, RDM.CNT, RANDOM, C.ERR.TBL
                                .GLOBL  MSCP.PKT, IPKT ADDR, PKT.USE, RETPKT
                                .GLOBL  RP.USE, RP.INDX, RP.ADDR, ELOG.PKT
                                .GLOBL  BUFF.ADDR, BUFF.OWN, IODQ, IODQ.IN
                                .GLOBL  IODQ.OUT, ENTRY.REASON, EOP.FLAG
                                .GLOBL  DUP.FLAGS, CCTLR, CDISK, CUOFF
                                .GLOBL  CTLR.CNT, DUR, QIO, FREE.MEM.ADDR
                                .GLOBL  BYTS.PER.QIO, ST.CODE, SB.CODE
                                .GLOBL  STEP, OF.RC, SA.REG, CMD.TIME
                                .GLOBL  NEX, CRN.LOW, CRN.HIGH, TEMP1
                                .GLOBL  TEMP2, CREDIT.BAL, NEXT.PKT.USE
                                .GLOBL  HOURS, MINUTES, CLK.TICKS, CLK.PRESENT
                                .GLOBL  HOE.FLAG, FORCED.ERROR, FERO.LBN
                                .GLOBL  FER1.LBN, FER.BC, INIT.OCCURED
                                .GLOBL  ADDR.VECT.OK, TYPER, TYPEW, BAL.IN.PROGRESS
                                .GLOBL  FORCE.WR, CSR.MEM, CSR.ADD, TST.PAR
                                .GLOBL  MAN.TST, P.INDEX, RD.COUNT, BRLEVEL
                                .GLOBL  D.FAIL, DBM12, DBM18, DBM19, DBM20
                                .GLOBL  DBM21, DBM22, DBM23, DBM25, DBM26
                                .GLOBL  DBM27, DBM29, DBM108, DBM109, DBM111
                                .GLOBL  DBM112, DBM120, DBM121, DBM123
```

```
                        .GLOBL  DBM125, DBM126, DBM127, DBM128
                        .GLOBL  EH.0, EH.1, EH.2, EH.3, EH.4, EH.5
                        .GLOBL  EH.6  EH 7, EH.8, EH.9, EH.10
                        .GLOBL  EH.1   EH.13, MSG.02, MSG.03, EGS.02
                        .GLOBL  EGD.10, EGD.11, EGD.12, EGD.13
                        .GLOBL  EGD.14, EGD.15, EGD.16, EGD.17
                        .GLOBL  EGD.18, EGD.19, EGD.20, EGD.21
                        .GLOBL  EGD.22, EGD.23, EGD.24, EGH.30
                        .GLOBL  DF.MSG, HRD.MSG, SFT.MSG, HRD.SUB
                        .GLOBL  CRLF, SWP.ERROR, SWP.XFER, SWP.FLAGS
                        .GLOBL  SWP.RAT, SWP.DPAT, SWP.UCNT, SWP.TIME
                        .GLOBL  SWP.UDPAT, L$LUN, L$UNIT, NEX.TRAP
                        .GLOBL  PARITY, TIME, SET.CPAR, SET.UPAR
                        .GLOBL  OUT.IODQ, IN.IODQ, GET.PKT, PUT.PKT
                        .GLOBL  GET.RETPKT, PUT.RETPKT, GET.IO.BUFF
                         GLOBL  PUT.IO.BUFF, PUTA.BUFF, SEND, WAIT
                        .GLOBL  MODULAS, DROP.CTLR, DRV.CTLERR
                        .GLOBL  EMS.R2, EMS.R1, EMS.EL, EMS.CMP
                        .GLOBL  EMS.ERR, EMS.10, EMS.12, EMS.13
                        .GLOBL  EMS.14, EMS.18, EMS.21, EMS.24
                        .GLOBL  EMS.30

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

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

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

000000  004137  000000G             $T1:    JSR     R1,$SAVE3                       ;          2167
000004  112737  000001  000000G             MOVB    #1,EOP.FLAG                     ;          2185
000012  112737  000001  001262'             MOVB    #1,COMPARE.DATA                 ;          2186
000020  042737  000002  000000G             BIC     #2,DUP.FLAGS                    ;          2187
000026  105037  000000G                     CLRB    HOE.FLAG                        ;          2188
000032  105037  000000G                     CLRB    FORCED.ERROR                    ;          2189
000036  005002                              CLR     R2                              ; I        2192
000040  010246                      1$:     MOV     R2,-(SP)                        ; I,*      2196
000042  012746  000043                      MOV     #43,-(SP)
000046  004737  000000G                     JSR     PC,BL$MUL
000052  005001                              CLR     R1                              ; J        2195
000054  010003                      2$:     MOV     R0,R3                           ;          2196
000056  060103                              ADD     R1,R3                           ; J,*
000060  006303                              ASL     R3
000062  005063  000000G                     CLR     MSCP.PKT(R3)
000066  005201                              INC     R1                              ; J        2195
000070  020127  000042                      CMP     R1,#42                          ; J,*
000074  003767                              BLE     2$
000076  010216                              MOV     R2,(SP)                         ; I,*      2198
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                           ;          2193
```

```
000120  005202                            INC    R2                          ; I              2192
000122  020227  000013                    CMP    R2,#13                      ; I,*
000126  003744                            BLE    1$
000130  005002                            CLR    R2                          ; I              2201
000132  005001                    3$:     CLR    R1                          ; J              2202
000134  010200                    4$:     MOV    R2,R0                       ; I,*            2203
000136  060100                            ADD    R1,R0                       ; J,*
000140  006300                            ASL    R0
000142  005060  000000G                   CLR    RETPKT(R0)
000146  005201                            INC    R1                          ; J              2202
000150  020127  000025                    CMP    R1,#25                      ; J,*
000154  003767                            BLE    4$
000156  062702  000026                    ADD    #26,R2                      ; *,I            2201
000162  020227  000232                    CMP    R2,#232                     ; I,*
000166  003761                            BLE    3$
000170  005002                            CLR    R2                          ; I              2205
000172  010246                    5$:     MOV    R2,-(SP)                    ; I,*            2209
000174  012746  000041                    MOV    #41,-(SP)
000200  004737  000000G                   JSR    PC,BL$MUL
000204  005001                            CLR    R1                          ; J              2208
000206  010003                    6$:     MOV    R0,R3                       ;                2209
000210  060103                            ADD    R1,R3                       ; J,*
000212  006303                            ASL    R3
000214  005063  000000G                   CLR    ELOG.PKT(R3)
000220  005201                            INC    R1                          ; J              2208
000222  020127  000040                    CMP    R1,#40                      ; J,*
000226  003767                            BLE    6$
000230  010216                            MOV    R2,(SP)                     ; I,*            2211
000232  012746  000102                    MOV    #102,-(SP)
000236  004737  000000G                   JSR    PC,BL$MUL
000242  105060  000001G                   CLRB   ELOG.PKT+1(R0)
000246  062706  000006                    ADD    #6,SP                       ;                2206
000252  005202                            INC    R2                          ; I              2205
000254  020227  000014                    CMP    R2,#14                      ; I,*
000260  003744                            BLE    5$
000262  032737  000020  000000G           BIT    #20,SWP.FLAGS               ;                2214
000270  001403                            BEQ    7$
000272  042737  000040  000000G           BIC    #40,SWP.FLAGS               ;                2216
000300  032737  000002  000000G   7$:     BIT    #2,SWP.FLAGS                ;                2218
000306  001403                            BEQ    8$
000310  042737  001000  000000G           BIC    #1000,SWP.FLAGS             ;                2220
000316  132737  000001  000000G   8$:     BITB   #1,INIT.OCCURED             ;                2222
000324  001402                            BEQ    9$
000326  000137  000732'                   JMP    15$
000332  017700  000000G           9$:     MOV    @FREE.MEM.ADDR,R0           ;                2225
000336  006300                            ASL    R0
000340  063700  000000G                   ADD    FREE.MEM.ADDR,R0
000344  010037  001250                    MOV    R0,DRS.START
000350  062737  000002  001250'           ADD    #2,DRS.START
000356  032737  000001  000000G           BIT    #1,SWP.FLAGS                ;                2231
000364  001417                            BEQ    10$
000366  112737  000001  001254'           MOVB   #1,APT.MODE                 ;                2234
000374  013737  001250' 001256'           MOV    DRS.START,MAIL.BOX.TESTNUM  ;                2235
```

```
000402  062737  000070  001256'            ADD     #70,MAIL.BOX.TESTNUM
000410  013737  001250' 001260'            MOV     DRS.START,MAIL.BOX.SUBTST
000416  062737  000066  001260'            ADD     #66,MAIL.BOX.SUBTST             ;                       2236
000424  005037  000000G            10$:    CLR     NEX
000430  105037  000000G                    CLRB    CLK.PRESENT                     ;                       2240
000434  012746  000340                     MOV     #340,-(SP)                      ;                       2241
000440  012746  000000G                    MOV     #NEX.TRAP,-(SP)                 ;                       2242
000444  012746  000004                     MOV     #4,-(SP)
000450  012746  000003                     MOV     #3,-(SP)
000454  104437                             TRAP    37
000456  013701  177546                     MOV     @#177546,R1
000462  005002                             CLR     R2                              ; *,DUMMY.0             2243
000464  012700  000004                     MOV     #4,R0                           ; DUMMY.1               2244
000470  104436                             TRAP    36                              ;                       2245
000472  032737  000001  000000G            BIT     #1,NEX
000500  001060                             BNE     13$                             ;                       2248
000502  112737  000001  000000G            MOVB    #1,CLK.PRESENT
000510  005037  000000G                    CLR     CLK.TICKS                       ;                       2251
000514  013716  000000G                    MOV     SWP.TIME,(SP)                   ;                       2252
000520  012746  000144                     MOV     #144,-(SP)                      ;                       2253
000524  004737  000000G                    JSR     PC,BL$DIV
000530  110037  000000G                    MOVB    R0,HOURS
000534  013716  000000G                    MOV     SWP.TIME,(SP)
000540  012746  000144                     MOV     #144,-(SP)                      ;                       2254
000544  004737  000000G                    JSR     PC,BL$MOD
000550  010003                             MOV     R0,R3
000552  005203                             INC     R3
000554  110337  000000G                    MOVB    R3,MINUTES
000560  123727  000000G 000074     11$:    CMPB    MINUTES,#74
000566  103412                             BLO     12$                             ;                       2256
000570  005000                             CLR     R0
000572  153700  000000G                    BISB    MINUTES,R0                      ;                       2258
000576  162700  000074                     SUB     #74,R0
000602  110037  000000G                    MOVB    R0,MINUTES
000606  105237  000000G                    INCB    HOURS
000612  000762                             BR      11$                             ;                       2259
000614  005016                     12$:    CLR     (SP)                            ;                       2256
000616  113716  000000G                    MOVB    HOURS,(SP)                      ;                       2262
000622  012746  000030                     MOV     #30,-(SP)
000626  004737  000000G                    JSR     PC,BL$MOD
000632  110037  000000G                    MOVB    R0,HOURS
000636  062706  000006                     ADD     #6,SP
000642  005037  000000G            13$:    CLR     NEX                             ;                       2250
000646  005037  000000G                    CLR     CSR.MEM                         ;                       2265
000652  012716  000340                     MOV     #340,(SP)                       ;                       2266
000656  012746  000000G                    MOV     #NEX.TRAP,-(SP)                 ;                       2267
000662  012746  000004                     MOV     #4,-(SP)
000666  012746  000003                     MOV     #3,-(SP)
000672  104437                             TRAP    37
000674  017701  000000G                    MOV     @CSR.ADD,R1
000700  005002                             CLR     R2                              ; *,DUMMY.0             2268
000702  012700  000004                     MOV     #4,R0                           ; DUMMY.1               2269
000706  104436                             TRAP    36                              ;                       2270
```

```
000710  032737  000001  000000G          BIT     #1,NEX                  ;                2273
000716  001003                           BNE     14$
000720  012737  000001  000000G          MOV     #1,CSR.MEM              ;                2276
000726  062706  000016           14$:    ADD     #16,SP                  ;                2224
000732  132737  000001  000000G  15$:    BITB    #1,CLK.PRESENT          ;                2282
000740  001416                           BEQ     16$
000742  012746  000300                   MOV     #300,-(SP)              ;                2285
000746  012746  000000G                  MOV     #TIME,-(SP)
000752  012746  000100                   MOV     #100,-(SP)
000756  012746  000003                   MOV     #3,-(SP)
000762  104437                           TRAP    37
000764  012737  000100  177546           MOV     #100,@#177546           ;                2286
000772  062706  000010                   ADD     #10,SP                  ;                2284
000776  032737  000001  000000G  16$:    BIT     #1,CSR.MEM              ;                2289
001004  001422                           BEQ     17$
001006  032737  000001  000000G          BIT     #1,TST.PAR
001014  001416                           BEQ     17$
001016  012746  000340                   MOV     #340,-(SP)              ;                2292
001022  012746  000000G                  MOV     #PARITY,-(SP)
001026  012746  000114                   MOV     #114,-(SP)
001032  012746  000003                   MOV     #3,-(SP)
001036  104437                           TRAP    37
001040  012777  000001  000000G          MOV     #1,@CSR.ADD             ;                2293
001046  062706  000010                   ADD     #10,SP                  ;                2291
001052  104421                   17$:    TRAP    21                      ;                2296
001054  010037  001264'                  MOV     R0,DRS.FLAGS
001060  042700  077777                   BIC     #77777,R0               ;                2298
001064  020027  100000                   CMP     R0,#-100000
001070  001003                           BNE     18$
001072  012700  000001                   MOV     #1,R0
001076  000401                           BR      19$
001100  005000                   18$:    CLR     R0
001102  020027  100000           19$:    CMP     R0,#-100000
001106  001003                           BNE     20$
001110  112737  000001  000000G          MOVB    #1,HOE.FLAG             ;                2300
001116  004737  000000V          20$:    JSR     PC,INIT.TEST            ;                2303
001122  005002                           CLR     R2                      ; CTLR           2305
001124  010246                   21$:    MOV     R2,-(SP)                ; CTLR,*         2307
001126  012746  000126                   MOV     #126,-(SP)
001132  004737  000000G                  JSR     PC,BL$MUL
001136  022626                           CMP     (SP)+,(SP)+
001140  005760  000002G                  TST     CST+2(R0)
001144  100040                           BPL     25$
001146  010246                           MOV     R2,-(SP)                ; CTLR,*         2308
001150  012746  000022                   MOV     #22,-(SP)
001154  004737  000000G                  JSR     PC,BL$MUL
001160  022626                           CMP     (SP)+,(SP)+
001162  005760  000000G                  TST     DCT(R0)
001166  100027                           BPL     25$
001170  010246                           MOV     R2,-(SP)                ; CTLR,*         2313
001172  012746  000053                   MOV     #53,-(SP)
001176  004737  000000G                  JSR     PC,BL$MUL
001202  012701  000003                   MOV     #3,R1                   ; *,OFFSET       2311
```

```
001206  010003                          22$:    MOV     R0,R3                       ;                       2313
001210  060103                                  ADD     R1,R3                       ; OFFSET,*
001212  006303                                  ASL     R3
001214  032763  020000  000000G                 BIT     #20000,CST(R3)
001222  001403                                  BEQ     23$
001224  105037  000000G                         CLRB    EOP.FLAG                    ;                       2316
001230  000405                                  BR      24$                         ;                       2315
001232  062701  000012                  23$:    ADD     #12,R1                      ; *,OFFSET              2311
001236  020127  000042                          CMP     R1,#42                      ; OFFSET,*
001242  003761                                  BLE     22$
001244  022626                          24$:    CMP     (SP)+,(SP)+
001246  005202                          25$:    INC     R2                          ; CTLR                  2305
001250  000243                                  .WORD CLV!CLC

001252  003724                                  BLE     21$
001254  132737  000001  000000G                 BITB    #1,EOP.FLAG                 ;                       2320
001262  001002                                  BNE     26$
001264  004737  000000V                         JSR     PC,MULTI.DRIVE              ;                       2322
001270  000207                          26$:    RTS     PC                          ;                       2167

; Routine Size:  349 words,     Routine Base:  $CODE$ + 0000
; Maximum stack depth per invocation:  13 words


                                                .SBTTL  T1 TEST SECTION
000000  004737  000000'                 T1::
000000                                  1$:     JSR     PC,$T1                      ;                       2322
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

; Routine Size:  6 words,       Routine Base:  $CODE$ + 1272
; Maximum stack depth per invocation:  2 words
```

```
;    2324  1      %sbttl 'INITIALIZATION TEST ROUTINES'
;    2325  1
;    2326  1      GLOBAL routine INIT_TEST : novalue =
;    2327  1
;    2328  1      !+
;    2329  1      !       THE INITIALIZATION TEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
;    2330  1      !       DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
;    2331  1      !       BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI DRIVE TEST
;    2332  1      !       OR THE DM EXERCISER.
;    2333  1      !
;    2334  1      !       BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
;    2335  1      !       INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
;    2336  1      !       MARKED OFFLINE IN THEIR DCT AND CST. FOR THOSE DEVICES WHICH SURVIVE
;    2337  1      !       THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
;    2338  1      !       EACH DISK VIA ROUTINE "ACCESS'. THE INITIALIZATION TEST IS DEEMED A
;    2339  1      !       SUCCESS IF A BLOCK OF EACH DISK CAN BE ACCESSED.
;    2340  1      !-
;    2341  1
;    2342  2          begin
;    2343  2          DRIVER_INIT ();                          ! INIT DRIVER DATA AND DEVICES
;    2344  2
;    2345  2          incr CTLR from 0 to (MAX_CTLR   1) do    ! FOR EACH CONTROLLER
;    2346  3              begin
;    2347  3              SET_CPAR (.CTLR);                    ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    2348  3
;    2349  3              if .CST_ADDR [STATE] eql ONLINE      ! IF CONTROLLER IS STILL ALIVE
;    2350  3              then                                 ! FOR EACH DISK
;    2351  3
;    2352  3                  incr OFFSET from (0 + OF UN) to ((UNITS_PER_CNTR - 1) * UNIT SIZE + OF_UN) by UNIT SIZF do
;    2353  3
;    2354  3                      if (.CST ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    2355  3                         (.CST ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    2356  4                         (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2357  3                      then
;    2358  4                          begin
;    2359  4                          SET_UPAR (.OFFSET);      ! SET UP UNIT-RELATED DATA ITEMS
;    2360  5                          IF SWP DINT NEQ (.DUP FLAGS AND SWP_DINT)        !ZZZ
;    2361  4                              THEN ACCESS ();                              !ZZZ
;    2362  4                                               !SKIP IF DUP CAUSED INIT ZZZ
;    2363  4
;    2364  3                          end;                 ! IF UNIT IS PRESENT AND ONLINE
;    2365  3
;    2366  2              end;                             ! CONTROLLER LOOP
;    2367  2
;    2368  1          end;                                 ! ROUTINE INIT TEST

                                           .SBTTL  INIT.TEST INITIALIZATION TEST ROUTINES
000000  004137  000000G         INIT.TEST::
                                           JSR     R1,$SAVE2                        ;           2326
000004  004737  000000V                    JSR     PC,DRIVER.INIT                   ;           2343
000010  005002                             CLR     R2                               ; CTLR      2345
000012  010246                  1$:        MOV     R2,-(SP)                         ; CTLR,*    2347
```

```
000014  004737  000000G                JSR     PC,SET.CPAR
000020  013700  000000G                MOV     CST.ADDR,R0
000024  005760  000002                 TST     2(R0)                 ;                     2349
000030  100035                         BPL     4$
000032  012701  000003                 MOV     #3,R1
000036  010100               2$:       MOV     R1,R0                 ; *,OFFSET            2352
000040  006300                         ASL     R0                    ; OFFSET,*            2354
000042  063700  000000G                ADD     CST.ADDR,R0
000046  032710  040000                 BIT     #40000,(R0)
000052  001417                         BEQ     3$
000054  032710  020000                 BIT     #20000,(R0)
000060  001414                         BEQ     3$                    ;                     2355
000062  032710  010000                 BIT     #10000,(R0)
000066  001011                         BNE     3$                    ;                     2356
000070  010116                         MOV     R1,(SP)
000072  004737  000000G                JSR     PC,SET.UPAR           ; OFFSET,*            2359
000076  032737  000002  000000G        BIT     #2,DUP.FLAGS
000104  001002                         BNE     3$                    ;                     2360
000106  004737  000000V                JSR     PC,ACCESS
000112  062701  000012       3$:       ADD     #12,R1                ;                     2361
000116  020127  000041                 CMP     R1,#41                ; *,OFFSET            2352
000122  003745                         BLE     2$                    ; OFFSET,*
000124  005726               4$.       TST     (SP)+
000126  005202                         INC     R2                    ;                     2346
000130  000243                         .WORD CLV!CLC                 ; CTLR                2345

000132  003727                         BLE     1$
000134  000207                         RTS     PC                    ;                     2326

; Routine Size:  47 words,      Routine Base:  $CODE$ + 1306
; Maximum stack depth per invocation:  5 words
```

```
GLOBAL routine DRIVER_INIT : novalue =

!+
!       THIS ROUTINE IS EQUIVALENT IN FUNCTION TO THE INITIALIZATION ENTRY
!       POINT OF A STANDARD DEVICE DRIVER. ITS RESPONSIBILITY IS TO INITIALIZE
!       DRIVER DATA, AND TO BRING EACH RDRX CONTROLLER AND UNIT (DISK)
!       ONLINE.
!-

    begin

    local
        PKT_ADDR;

    PKT_ADDR = MSCP_PKT + 10;                                    ! ADDR (TEXT + 0) OF 1ST MSCP PKT
    NEXT_PKT_USE = 0;                                            ! NEXT PACKET TO ALLOCATE

    incr COUNT from 0 to (PKT_CNT - 1) do                        ! FOR EACH MSCP PACKET
        begin
        PKT_USE [.COUNT] = -1;                                   ! MARK PACKET FREE
        MSCP_PKT [.COUNT, PKT_LO] = .PKT_ADDR;                   ! LOAD ADDR INTO BUFFER DESCRIPTOR
        MSCP_PKT [.COUNT, PKT_HI] = 0;
        MSCP_PKT [.COUNT, CONNID] = CID_DISK;                    ! SET CONNECTION ID TO MSCP ID
        PKT_ADDR = .PKT_ADDR + (PKT_LEN  *  2);                  ! ADVANCE ADDR TO NEXT PACKET
        end;

    incr CTLR from 0 to (MAX_CTLR - 1) do                        ! FOR EACH CONTROLLER

        if .CST [.CTLR, IP_ADDR] neqa 0                          ! IF CONTROLLER IS PRESENT
        then
            begin
            SET_CPAR (.CTLR);                                    ! CURRENT CONTROLLER PARAMETERS
            CURRENT_VECTOR = .CST_ADDR [VEC_ADDR];               ! CURRENT CONTROLLER'S VECTOR
            BRLEVEL = .CST_ADDR [BR_LEV] ^ 5;                    ! SET CURRENT CONTROLLER'S BR LEVEL
            CTLR_INIT ();                                        ! INIT DEVICE AND CTLR DATA

            if .DCT_ADDR [STAT] eql ONLINE                       ! IF CONTROLLER IS STILL ALIVE
            then                                                 ! FOR EACH DIAK UNIT

                incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT SIZE + OF UN) by UNIT_SIZE do

                    if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and   ! IF UNIT EXISTS
                       (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
                    then
                        begin
                        CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_0] = %o'40';       ! BLANK DEVICE NAME
                        CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_1] = %o'40';       !
                        CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_2] = %o'40';       !
                        CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_3] = %o'40';       !
                        SET_UPAR (.OFFSET);                                      ! SET UP UNIT RELATED DATA ITEMS
                        UNIT_INIT ();                                            ! BRING UNIT ONLINE
                        end;                                                     ! IF UNIT EXISTS
```

```
;    2422  2            end;                                          ! IF CONTROLLER IS PRESENT
;    2423  2
;    2424  1        end;                                              ! ROUTINE DRIVER INIT

                                        .SBTTL  DRIVER.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G         DRIVER.INIT::
                                        JSR     R1,$SAVE3                       ;                       2369
000004  012702  000012G                 MOV     @MSCP.PKT+12,R2                 ; *,PKT.ADDR            2383
000010  105037  000000G                 CLRB    NEXT.PKT.USE                    ;                       2384
000014  005001                          CLR     R1                              ; COUNT                 2386
000016  112761  000377  000000G  1$:    MOVB    #377,PKT.USE(R1)                ; *,*(COUNT)            2388
000024  010146                          MOV     R1,-(SP)                        ; COUNT,*               2389
000026  012746  000106                  MOV     #106,-(SP)
000032  004737  000000G                 JSR     PC,BL$MUL
000036  010260  000000G                 MOV     R2,MSCP.PKT(R0)                 ; PKT.ADDR,*
000042  005060  000002G                 CLR     MSCP.PKT+2(R0)                  ;                       2390
000046  105060  000011G                 CLRB    MSCP.PKT+11(R0)                 ;                       2391
000052  062702  000106                  ADD     #106,R2                         ; *,PKT.ADDR            2392
000056  022626                          CMP     (SP)+,(SP)+                     ;                       2387
000060  005201                          INC     R1                              ; COUNT                 2386
000062  020127  000013                  CMP     R1,#13                          ; COUNT,*
000066  003753                          BLE     1$
000070  005003                          CLR     R3                              ; CTLR                  2395
000072  010346                   2$:    MOV     R3,-(SP)                        ; CTLR,*                2397
000074  012746  000126                  MOV     #126,-(SP)
000100  004737  000000G                 JSR     PC,BL$MUL
000104  022626                          CMP     (SP)+,(SP)+
000106  005760  000000G                 TST     CST(R0)
000112  001503                          BEQ     6$
000114  010346                          MOV     R3,-(SP)                        ; CTLR,*                2400
000116  004737  000000G                 JSR     PC,SET.CPAR
000122  013700  000000G                 MOV     CST.ADDR,R0                     ;                       2401
000126  016037  000002  001246'         MOV     2(R0),CURRENT.VECTOR
000134  042737  177000  001246'         BIC     #177000,CURRENT.VECTOR
000142  005016                          CLR     (SP)                            ;                       2402
000144  116016  000004                  MOVB    4(R0),(SP)
000150  012746  000005                  MOV     #5,-(SP)
000154  004737  000000G                 JSR     PC,BL$SHF
000160  010037  000000G                 MOV     R0,BRLEVEL
000164  004737  000000V                 JSR     PC,CTLR.INIT                    ;                       2403
000170  005777  000000G                 TST     @DCT.ADDR                       ;                       2405
000174  100051                          BPL     5$
000176  012701  000003                  MOV     #3,R1                           ; *,OFFSET              2408
000202  013702  000000G          3$:    MOV     CST.ADDR,R2                     ;                       2410
000206  010100                          MOV     R1,R0                           ; OFFSET,*
000210  006300                          ASL     R0
000212  060200                          ADD     R2,R0
000214  032710  040000                  BIT     #40000,(R0)
000220  001432                          BEQ     4$
000222  032710  010000                  BIT     #10000,(R0)                     ;                       2411
000226  001027                          BNE     4$
000230  010100                          MOV     R1,R0                           ; OFFSET,*              2414
```

```
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)
000252  010100                          MOV     R1,R0                   ;                       2415
000254  006300                          ASL     R0                      ; OFFSET,*              2416
000256  060200                          ADD     R2,R0
000260  112760  000040  000014          MOVB    #40,14(R0)
000266  112760  000040  000015          MOVB    #40,15(R0)
000274  010116                          MOV     R1,(SP)                 ;                       2417
000276  004737  000000G                 JSR     PC,SET.UPAR             ; OFFSET,*              2418
000302  004737  000000V                 JSR     PC,UNIT.INIT
000306  062701  000012          4$:     ADD     #12,R1                  ;                       2419
000312  020127  000041                  CMP     R1,#41                  ; *,OFFSET              2408
000316  003731                          BLE     3$                      ; OFFSET,*
000320  022626                  5$:     CMP     (SP)+,(SP)+
000322  005203                  6$:     INC     R3                      ;                       2399
000324  000243                          .WORD CLV!CLC                   ; CTLR                  2395

000326  003661                          BLE     2$
000330  000207                          RTS     PC                      ;                       2369

; Routine Size:  109 words,     Routine Base:  $CODE$ + 1444
; Maximum stack depth per invocation:  7 words
```

```
;  1   GLOBAL routine CTLR_INIT : novalue =
;  1
;  1   !+
;  1   !       THIS "DRIVER' ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONTROLLER
;  1   !       CONFIGURED FOR TESTING. ITS GENERAL PURPOSE IS TO BRING THE RDRX ONLINE
;  1   !       TO THE HOST. SPECIFICALLY, IT IS WRITTEN TO:
;  1   !
;  1   !       1. INITIALIZE DRIVER CONTROLLER DATA, INCLUDING THE DCT,
;  1   !       2. SET UP THE DEVICE'S INTERRUPT VECTOR ADDRESS,
;  1   !       3. PERFORM A REGISTER EXISTENCE TEST TO VERIFY THE DEVICE'S PRESENCE,
;  1   !       4. PERFORM A VECTOR AND BR LEVEL TEST TO VERIFY THE DEVICE'S VECTOR
;  1   !             ADDRESS AND INTERRUPT REQUEST LEVEL,
;  1   !       5. DO A HARD INITIALIZATION (FOUR STEPS) ON THE DEVICE.
;  1   !
;  1   !       IF ANY OF THESE INITIAL TESTS FAIL, THEN ALL UNITS ASSOCIATED WITH THE
;  1   !       DEVICE ARE DROPPED.
;  1   !
;  1
;  2       begin
;  2
;  2       local
;  2           RESULT : byte;
;  2
;  2       INI_CTLR_DAT ();                                         ! INITIALIZE CONTROLLER DATA
;  2   !ZZZ    SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], PRIO4); ! SET DEVICE'S ASSUMED VECTOR ADDRESS
;  2       SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], .BRLEVEL);  ! SET DEVICE'S ASSUMED VECTOR ADDRESS    ZZZ
;  2       DCT_ADDR [IG_INT] = TRUE;                                ! SET "IGNORE INTERRUPT" BIT
;  2       L$LUN = .CST_ADDR [OF_UN + OF_DATA, D_UNIT];             ! GET FIRST UNIT NUMBER OF CONTROLLER
;  2                                                                ! (USED BY DRS FOR DEVICE FATAL CTLR ERRORS)
;  3   IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                    !IF DUP  ZZZ
;  2   THEN                                  !CAUSED INIT, SKIP THIS CODE  ZZZ
;  2
;  2       if REG_EXIST () eql FAILURE                              ! REGISTER EXISTENCE TEST
;  2       then
;  3           begin
;  3           DROP_CTLR (.CCTLR, DU_INIT);                         ! DROP ALL CONTROLLER'S UNITS
;  3           return;
;  2           end;
;  2
;  3   IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                    !IF DUP  ZZZ
;  2   THEN                                  !CAUSED INIT, SKIP THIS CODE  ZZZ
;  2
;  2       if VEC_BR_TEST () eql FAILURE                            ! VECTOR ADDR AND BR LEVEL TEST
;  2       then
;  3           begin
;  3           DROP_CTLR (.CCTLR, DU_INIT);                         ! DROP ALL CONTROLLER'S UNITS
;  3           return;
;  2           end;
;  2
;  2       RESULT = HARD_INIT ();                                   ! ATTEMPT HARD DEVICE INIT
;  2       DCT_ADDR [IG_INT] = FALSE;                               ! CLAER "IGNORE INTERRUPT" BIT
;  2
;  2       if .RESULT eql SUCCESS                                   ! IF HARD INIT WAS SUCCESSFUL
```

```
;   2        then
;   3            begin
;   3            ADDR_VECT_OK = TRUE;                          ! ADDRESS/VECTOR TEST PASSED
;   3            INI_RRING ();                                 ! INITIALIZE RESPONSE RING
;   3            WRT_RDRX (RCSA, RC_ALL, SA_GO);               ! SET "GO" BIT (START CTLR POLLING)
;   3
;   3            if SET_CTLR_CHAR () eql SUCCESS               ! SET CONTROLLER CHARACTERISTICS
;   3            then
;   4                begin
;   4                DCT_ADDR [STAT] = ONLINE;                 ! MARK CONTROLLER ONLINE IN "DRIVER"
;   4                CST_ADDR [STATE] = ONLINE;                ! MARK CONTROLLER ONLINE IN "PROGRAM"
;   3                end;
;   3            end
;   3
;   2        else                                              ! HARD INIT FAILED
;   3            begin
;   3            DROP_CTLR (.CCTLR, DU_INIT);                  ! DROP ALL CONTROLLER'S UNITS
;   2            end;
;   2
;   1        end;                                              ! ROUTINE CTLR_INIT

                                        .SBTTL  CTLR.INIT INITIALIZATION TEST ROUTINES
000000  010146                    CTLR.INIT::
                                        MOV     R1,-(SP)
000002  004737  000000V                 JSR     PC,INI.CTLR.DAT                 ;       2425
000006  013746  000000G                 MOV     BRLEVEL,-(SP)                   ;       2448
000012  013700  000000G                 MOV     CCTLR,R0                        ;       2450
000016  006300                          ASL     R0
000020  016046  000102'                 MOV     INT.ADDR(R0),-(SP)
000024  013746  001246'                 MOV     CURRENT.VECTOR,-(SP)
000030  012746  000003                  MOV     #3,-(SP)
000034  104437                          TRAP    37
000036  052777  040000  000000G         BIS     #40000,@DCT.ADDR                ;       2451
000044  013700  000000G                 MOV     CST.ADDR,R0                     ;       2452
000050  016001  000006                  MOV     6(R0),R1
000054  000301                          SWAB    R1
000056  042701  177760                  BIC     #177760,R1
000062  010137  000000G                 MOV     R1,L$LUN
000066  032737  000002  000000G         BIT     #2,DUP.FLAGS                    ;       2454
000074  001025                          BNE     2$
000076  004737  000000V                 JSR     PC,REG.EXIST                    ;       2457
000102  005700                          TST     R0
000104  001410                          BEQ     1$                              ;       2460
000106  032737  000002  000000G         BIT     #2,DUP.FLAGS                    ;       2464
000114  001015                          BNE     2$
000116  004737  000000V                 JSR     PC,VEC.BR.TEST                  ;       2467
000122  005700                          TST     R0
000124  001011                          BNE     2$
000126  013716  000000G           1$:   MOV     CCTLR,(SP)                      ;       2470
000132  012746  000002                  MOV     #2,-(SP)
000136  004737  000000G                 JSR     PC,DROP.CTLR
000142  062706  000012                  ADD     #12,SP                          ;       2471
```

```
000146 000453                              BR      5$
000150 004737 000000V               2$:    JSR     PC,HARD.INIT            ;                    2469
000154 110001                              MOVB    R0,R1                   ;                    2474
000156 042777 040000  000000G              BIC     #40000,@DCT.ADDR        ; *,RESULT
000164 120127 000001                       CMPB    R1,#1                   ;                    2475
000170 001031                              BNE     3$                      ; RESULT,*           2477
000172 112737 000001  000000G              MOVB    #1,ADDR.VECT.OK
000200 004737 000000V                      JSR     PC,INI.RRING            ;                    2480
000204 012701 000001                       MOV     #1,R1                   ;                    2481
000210 013700 000000G                      MOV     RDRX.ADDR,R0            ; *,RC.REG           2482
000214 010160 000002                       MOV     R1,2(R0)
000220 004737 000000V                      JSR     PC,SET.CTLR.CHAR        ; RC.REG,*
000224 020027 000001                       CMP     R0,#1                   ;                    2484
000230 001020                              BNE     4$
000232 052777 100000  000000G              BIS     #100000,@DCT.ADDR
000240 013700 000000G                      MOV     CST.ADDR,R0             ;                    2487
000244 052760 100000  000002               BIS     #100000,2(R0)           ;                    2488
000252 000407                              BR      4$
000254 013716 C00000G               3$:    MOV     CCTLR,(SP)              ;                    2477
000260 012746 000002                       MOV     #2,-(SP)                ;                    2494
000264 004737 000000G                      JSR     PC,DROP.CTLR
000270 005726                              TST     (SP)+
000272 062706 000010                4$:    ADD     #10,SP                  ;                    2493
000276 012601                       5$:    MOV     (SP)+,R1                ;                    2443
000300 000207                              RTS     PC                      ;                    2425

; Routine Size:  97 words,      Routine Base:  $CODE$ + 1776
; Maximum stack depth per invocation:  7 words
```

```
;    2498  1        GLOBAL routine INI_CTLR_DAT : novalue =
;    2499  1
;    2500  1        !+
;    2501  1        !       THIS ROUTINE IS RESPONSIBLE FOR INITIALIZING ALL CONTROLLER RELATED
;    2502  1        !       DATA IN THE "DRIVER" PORTION OF THE EXERCISER. THIS INCLUDES THE
;    2503  1        !       CONTROLLER'S DCT AND OUTSTANDING COMMAND LIST.
;    2504  1        !
;    2505  1        !       IMPLICIT INPUTS:
;    2506  1        !               CCTLR - CURREN. CONTROLLER NUMBER
;    2507  1        !               DCT_ADDR - ADDRESS OF CURENT CONTROLLER'S DCT
;    2508  1        !-
;    2509  1
;    2510  2            begin
;    2511  2            DCT_ADDR [WORD0] = 0;                                          ! CLEAR FIRST DCT WORD
;    2512  2            DCT_ADDR [RR_BEG] = COMM_AREA + 8 + (.CCTLR * COMM_LEN * 2);   ! START OF RESPONSE RING
;    2513  2            DCT_ADDR [RR_END] = .DCT_ADDR [RR_BEG] + ((RRING_LEN - 1) * 4); ! LAST SLOT IN RESPONSE RING
;    2514  2            DCT_ADDR [CR_BEG] = .DCT_ADDR [RR_END] + 4;                    ! START OF COMMAND RING
;    2515  2            DCT_ADDR [CR_END] = .DCT_ADDR [CR_BEG] + ((CRING LEN - 1) * 4); ! LAST SLOT IN COMMAND RING
;    2516  2            DCT_ADDR [PR_POLL] = .DCT_ADDR [RR_BEG];                       ! FIRST RRING SLOT TO POLL
;    2517  2            DCT_ADDR [CR_POLL] = DCT_ADDR [CR_NEXT] = .DCT_ADDR [CR_BEG];  ! CRING POLL AND NEXT COMMAND POINTERS
;    2518  1            end;

                                                 .SBTTL  INI.CTLR.DAT INITIALIZATION TEST ROUTINES
000000  004137  000000G                  INI.CTLR.DAT::
                                                 JSR     R1,$SAVE2                    ;          2498
000004  013701  000000G                          MOV     DCT.ADDR,R1                  ;          2511
000010  005011                                   CLR     (R1)
000012  012702  000004                           MOV     #4,R2                        ;          2512
000016  060102                                   ADD     R1,R2
000020  013746  000000G                          MOV     CCTLR,-(SP)
000024  012746  000050                           MOV     #50,-(SP)
000030  004737  000000G                          JSR     PC,BL$MUL
000034  062700  000010'                          ADD     #COMM.AREA+10,R0
000040  010012                                   MOV     R0,(R2)
000042  010061  000006                           MOV     R0,6(R1)
000046  062761  000014  000006                   ADD     #14,6(R1)                    ;          2513
000054  012700  000010                           MOV     #10,R0
000060  060100                                   ADD     R1,R0                        ;          2514
000062  016110  000006                           MOV     6(R1),(R0)
000066  062710  000004                           ADD     #4,(R0)
000072  011061  000012                           MOV     (R0),12(R1)
000076  062761  000014  000012                   ADD     #14,12(R1)                   ;          2515
000104  011261  000014                           MOV     (R2),14(R1)
000110  011061  000020                           MOV     (R0),20(R1)                  ;          2516
000114  011061  000016                           MOV     (R0),16(R1)                  ;          2517
000120  022626                                   CMP     (SP)+,(SP)+
000122  000207                                   RTS     PC                           ;          2510
                                                                                      ;          2498

; Routine Size:  42 words,       Routine Base:  $CODE$ + 2300
; Maximum stack depth per invocation:  6 words
```

```
1    GLOBAL routine REG_EXIST =
1    !+
1    !        THIS IS THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VERIFY
1    !        THE PRESENCE OF AN RDRX DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
1    !        SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
1    !        ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE'S SA AND IP
1    !        REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
1    !        THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
1    !        DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
1    !-
2        begin
2
2        local
2            DUMMY_0 : word,                          ! TEMP FOR READING SA AND IP
2            DUMMY_1 : word;                          !
2
2        if .ENTRY_REASON eql NEW_PASS
2        then
2            return SUCCESS;                          ! SKIP TEST FOR NEXT PASS
2
2        OF_RC = 2;                                   ! SET UP TO READ SA FIRST
2
2        do
3            begin
3            NEX = FALSE,                             ! SET TO "TRAP NOT RECEIVED"
3            SETVEC (4, NEX_TRAP, PRIO7);             ! SET LOCATION 4 TRAP VECTOR ADDRESS
3            DUMMY_0 = .(.RDRX_ADDR + .OF_RC);        ! READ REGISTER (THEN TRAP OR CONTINUE)
3            DUMMY_1 = 0;                             ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
3                                                     !       (TRAP RETURNS TO NEXT INSTRUCTION)
3            CLRVEC (4);                              ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS
3
3            if .NEX                                  ! IF NEX TRAP OCCURRED
3            then
4                begin
4                C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
4
4                if .APT_MODE
4                then
5                    begin
5                    .MAIL_BOX_TESTNUM = 1;
5                    .MAIL_BOX_SUBTST = 0;
4                    end;
4
4                ERRDF (10, EGD_10, EMS_10);          ! REGISTER EXISTENCE TEST FAILED
4                SETPRI (PRIO0);                      ! LOWER PRIORITY
4                return FAILURE;
4                end
3            else
3                OF_RC = .OF_RC - 2;                  ! SET UP FOR IP REG OR QUIT
3
3            end
2        until .OF_RC leq 0;
2
```

```
;   2572  2        return SUCCESS;
;   2573  1        end;

                                         .SBTTL  REG.EXIST INITIALIZATION TEST ROUTINES
000000  004137  000000G          REG.EXIST::
                                         JSR     R1,$SAVE2                          ;                 2519
000004  123727  000000G 000005           CMPB    ENTRY.REASON,#5                    ;                 2535
000012  001472                           BEQ     4$                                 ;                 2537
000014  012737  000002  000000G          MOV     #2,OF.RC                           ;                 2539
000022  005037  000000G          1$:     CLR     NEX                                ;                 2543
000026  012746  000340                   MOV     #340,-(SP)                         ;                 2544
000032  012746  000000G                  MOV     #NEX.TRAP,-(SP)
000036  012746  000004                   MOV     #4,-(SP)
000042  012746  000003                   MOV     #3,-(SP)
000046  104437                           TRAP    37
000050  013700  000000G                  MOV     RDRX.ADDR,R0                       ;                 2545
000054  063700  000000G                  ADD     OF.RC,R0
000060  011001                           MOV     (R0),R1                            ; *,DUMMY.0
000062  005002                           CLR     R2                                 ; DUMMY.1         2546
000064  012700  000004                   MOV     #4,R0                              ;                 2548
000070  104436                           TRAP    36
000072  032737  000001  000000G          BIT     #1,NEX                             ;                 2550
000100  001427                           BEQ     3$
000102  013700  000000G                  MOV     CCTLR,R0                           ;                 2553
000106  006300                           ASL     R0
000110  105260  000000G                  INCB    C.ERR.TBL(R0)
000114  032737  000001  001254'          BIT     #1,APT.MODE                        ;                 2555
000122  001405                           BEQ     2$
000124  012777  000001  001256'          MOV     #1,@MAIL.BOX.TESTNUM               ;                 2558
000132  005077  001260'                  CLR     @MAIL.BOX.SUBTST                   ;                 2559
000136  104455                   2$:     TRAP    55                                 ;                 2562
000140  000012                           .WORD   12
000142  000000G                          .WORD   EGD.10
000144  000000G                          .WORD   EMS.10
000146  005000                           CLR     R0                                 ;                 2563
000150  104441                           TRAP    41
000152  062706  000010                   ADD     #10,SP                             ;                 2564
000156  000413                           BR      5$                                 ;                 2552
000160  162737  000002  000000G  3$:     SUB     #2,OF.RC                           ;                 2567
000166  062706  000010                   ADD     #10,SP                             ;                 2542
000172  005737  000000G                  TST     OF.RC                              ;                 2570
000176  002311                           BGE     1$
000200  012700  000001           4$:     MOV     #1,R0                              ;                 2529
000204  000207                           RTS     PC
000206  005000                   5$:     CLR     R0                                 ;                 2519
000210  000207                           RTS     PC

; Routine Size:  69 words,     Routine Base:  $CODE$ + 2424
; Maximum stack depth per invocation.  9 words
```

```
;      1     GLOBAL routine VEC_BR TEST =
;      1
;      1     !+
;      1     !        THIS ROUTINE ATTEMPTS TO VERIFY (A) THAT THE RDRX VECTOR ADDRESS GIVEN
;      1     !        BY THE USER DURING THE HW DIALOG IS VALID, AND (B) THAT THE
;      1     !        USER SPECIFIED BUS REQUEST LEVEL FOR THE DEVICE IS CORRECT. THE FIRST
;      1     !        OBJECTIVE IS ACCOMPLISHED BY SETTING THE CPU PRIORITY TO 0 AND FORCING
;      1     !        AN RDRX INTERRUPT. IF THE USER SPECIFIED AN INCORRECT VECTOR ADDRESS,
;      1     !        THEN THE RESULT MAY BE UNPREDICTABLE. FOR THIS REASON, THE MESSAGE
;      1     !        "FUNCTIONAL TEST STARTED" IS PRINTED BEFORE THE TEST, AND
;      1     !        "EXERCISER STARTED" IS PRINTED AT ITS SUCCESSFUL CONCLUSION. IF
;      1     !        EITHER "FUNCTIONAL TEST ..." OR "EXERCISER  ..  DOES NOT APPEAR, THEN
;      1     !        PROGRAM CONTROL IS ASSUMED LOST AND A FATAL TRAP IS LIKELY TO OCCUR. AT
;      1     !        THIS POINT, THE EXERCISER MUST BE STARTED AGAIN.
;      1     !
;      1     !        IF THIS TEST SUCCEEDS, THEN THE BR LEVEL TEST IS RUN BY SETTING THE
;      1     !        PROCESSOR PRIORITY TO THE ASSUMED INTERRUPT PRIORITY GIVEN BY THE
;      1     !        USER. A FORCED INTERRUPT SHOULD NOT OCCURR. THEN, BY LOWERING THE
;      1     !        PRIORITY BY ONE, THE DELAYED INTERRUPT SHOULD OCCUR.
;      1     !
;      1
;      2         begin
;      2
;      2         if .ENTRY_REASON eql NEW_PASS
;      2         then
;      3             begin
;      3             SETPRI (PRIO0);                           ! LOWER PRIORITY
;      3             return SUCCESS;                           ! SKIP TEST IF NEXT PASS
;      2             end;
;      2
;      2         PRINTF (MSG_02);                              ! 'FUNCTIONAL TEST STARTED"
;      2
;      2         if INT_GEN () eql FALSE                       ! FORCE AN INTERRUPT
;      2         then
;      3             begin                                     ! IF INTERRUPT DID NOT OCCUR
;      3             C_ERR_TBL [.CCTLR, C ERR_HRD] = .C_ERR TBL [.CCTLR, C ERR_HRD] + 1;
;      3
;      3             if .APT_MODE
;      3             then
;      4                 begin
;      4                 .MAIL BOX TESTNUM = 1;
;      4                 .MAIL BOX_SUBTST = 0;
;      3                 end;
;      3
;      3             ERRDF (11, EGD_11, 0);                    ! VECTOR TEST FAILED
;      3             return FAILURE;
;      3             end
;      2         else
;      3             begin                                     ! INTERRUPT DID OCCUR
;      3             PRINTF (MSG 03);                          ! "EXERCISER STARTED"
;      3             SETPRI (.BRLEVEL);                        ! SET PRIORITY TO ASSUMED BR LEVEL
;      3
;      3             if INT GEN () eql FALSE                   ! FORCE AN INTERRUPT (SHOULD NOT OCCUR)
```

```
;     2627  3             then
;     2628  4                 begin                              ! IF INTERRUPT DID NOT OCCUR
;     2629  4                 SETPRI (.BRLEVEL   %o 40');        ! LOWER PRIORITY BY 1
;     2630  4                 DELAY (1);                         ! WAIT
;     2631  4                 BREAK;                             ! MMM
;     2632  4
;     2633  4                 if .DCT_ADDR [SA SAVE] neq 0       ! IF INTERRUPT DID OCCUR (SA SAVE WOULD BE NON-ZERO)
;     2634  4                 then
;     2635  5                     begin
;     2636  5                     SETPRI (PRIO0);                ! RESTORE PROCESSOR PRIORITY TO 0
;     2637  5                     return SUCCESS;                ! ONLY SUCCESSFUL EXIT POINT
;     2638  4                     end;
;     2639  4
;     2640  3                 end;
;     2641  3
;     2642  2             end;
;     2643  2
;     2644  2         SETPRI (PRIO0);                            ! COME HERE ONLY FOR BR TEST FAILURE
;     2645  2         C_ERR_TBL [.CCTLR, C ERR HRD] = .C ERR TBL [.CCTLR, C ERR HRD] + 1;
;     2646  2
;     2647  2         if .APT_MODE
;     2648  2         then
;     2649  3             begin
;     2650  3             .MAIL_BOX TESTNUM = 1;
;     2651  3             .MAIL BOX SUBTST = 0;
;     2652  2             end;
;     2653  2
;     2654  2         ERRDF (12, EGD_12, EMS 12);
;     2655  2         return FAILURE;
;     2656  1         end;


                                                 .GLOBL  L$DLY


                                                 .SBTTL  VEC.BR.TEST INITIALIZATION TEST ROUTINES
000000  010146                          VEC.BR.TEST::
                                                 MOV     R1,-(SP)                                                 2574
000002  005746                                   TST     (SP)                       ;
000004  123727  000000G 000005                   CMPB    ENTRY.REASON,#5                                          2597
000012  001003                                   BNE     1$                         ;
000014  005000                                   CLR     R0                                                       2600
000016  104441                                   TRAP    41                         ;
000020  000505                                   BR      8$
000022  012746  000000G                 1$:      MOV     #MSG.02,-(SP)              ;                             2599
000026  012746  000001                           MOV     #1,-(SP)                   ;                             2604
000032  010600                                   MOV     SP,R0                      ; SP,*
000034  104417                                   TRAP    17
000036  004737  000000V                          JSR     PC,INT.GEN                 ;                             2606
000042  005700                                   TST     R0
000044  001023                                   BNE     3$
000046  013700  000000G                          MOV     CCTLR,R0                   ;                             2609
000052  006300                                   ASL     R0
```

```
000054  105260  000000G                  INCB    C.ERR.TBL(R0)
000060  032737  000001  001254'          BIT     #1,APT.MODE              ;            2611
000066  001405                           BEQ     2$
000070  012777  000001  001256'          MOV     #1,@MAIL.BOX.TESTNUM     ;            2614
000076  005077  001260'                  CLR     @MAIL.BOX.SUBTST         ;            2615
000102  104455                   2$:     TRAP    55                       ;            2618
000104  000013                           .WORD   13
000106  000000G                          .WORD   EGD.11
000110  000000                           .WORD   0
000112  000500                           BR      11$
000114  012716  000000G          3$:     MOV     #MSG.03,(SP)             ;            2619
000120  012746  000001                   MOV     #1,-(SP)                 ;            2623
000124  010600                           MOV     SP,R0                    ; SP,*
000126  104417                           TRAP    17
000130  013700  000000G                  MOV     BRLEVEL,R0               ;            2624
000134  104441                           TRAP    41
000136  004737  000000V                  JSR     PC,INT.GEN               ;            2626
000142  005700                           TST     R0
000144  001036                           BNE     9$
000146  013700  000000G                  MOV     BRLEVEL,R0               ;            2629
000152  162700  000040                   SUB     #40,R0
000156  104441                           TRAP    41
000160  012701  000001                   MOV     #1,R1                    ; *,$$TMP2   2630
000164  001411                   4$:     BEQ     7$
000166  013700  000000G                  MOV     L$DLY,R0                 ; *,$$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  104422                   7$:     TRAP    22
000212  013700  000000G                  MOV     DCT.ADDR,R0              ;            2633
000216  005760  000002                   TST     2(R0)
000222  001407                           BEQ     9$
000224  005000                           CLR     R0                       ;            2636
000226  104441                           TRAP    41
000230  062706  000006                   ADD     #6,SP                    ;            2637
000234  012700  000001           8$:     MOV     #1,R0                    ;            2635
000240  000427                           BR      12$
000242  005726                   9$:     TST     (SP)+                    ;            2622
000244  005000                           CLR     R0                       ;            2644
000246  104441                           TRAP    41
000250  013700  000000G                  MOV     CCTLR,R0                 ;            2645
000254  006300                           ASL     R0
000256  105260  000000G                  INCB    C.ERR.TBL(R0)
000262  032737  000001  001254'          BIT     #1,APT.MODE              ;            2647
000270  001405                           BEQ     10$
000272  012777  000001  001256'          MOV     #1,@MAIL.BOX.TESTNUM     ;            2650
000300  005077  001260'                  CLR     @MAIL.BOX.SUBTST         ;            2651
000304  104455                   10$:    TRAP    55                       ;            2654
000306  000014                           .WORD   14
000310  000000G                          .WORD   EGD.12
```

```
000312  000000G                 .WORD   EMS.12
000314  022626          11$:    CMP     (SP)+,(SP)+                     ;       2655
000316  005000                  CLR     R0
000320  005726          12$:    TST     (SP)+                           ;       2574
000322  012601                  MOV     (SP)+,R1
000324  000207                  RTS     PC

; Routine Size:  107 words,     Routine Base:  $CODE$ + 2636
; Maximum stack depth per invocation:  7 words
```

```
;  1   GLOBAL routine INT_GEN =
;  1
;  1   !+
;  1   !       THIS ROUTINE BEGINS AN RDRX INITIALIZATION SEQUENCE, BUT ONLY
;  1   !       COMPLETES THROUGH THE STEP 1 WRITE. ITS PURPOSE IS TO CREATE AN RDRX
;  1   !       INTERRUPT (AT THE COMPLETEION OF STEP 1) IN ORDER TO HELP VERIFY THE
;  1   !       THE USER-SPECIFIED VECTOR ADDRESS AND BUS REQUEST INTERRUPT LEVEL.
;  1   !       A VALUE OF "TRUE" IS RETURNED TO THE CALLER IF AN INTERRUPT OCCURS,
;  1   !       AND "FALSE" OTHERWISE. THE INTERRUPT IS VERIFIED BY A NON-ZERO VALUE
;  1   !       IN THE "SA SAVE" WORD IN THE DEVICE'S DCT.
;  1   !
;  1
;  2       begin
;  2
;  2       local
;  2           SA : word;                                       ! STORAGE FOR STEP 1 READ AND WRITE
;  2
;  2       DCT_ADDR [SA_SAVE] = 0;                              ! ZERO OUT SA SAVE WORD IN DCT
;  2       WRT_RDRX (RCIP, RC_ALL, ALL_ONES);                   ! WRITE IP TO START INIT SEQUENCE
;  2       DELAY (2);                                           ! WAIT
;  2       BREAK;                                               ! MMM
;  2       INCR COUNT FROM 1 TO 500 DO                          !MAKE SURE WE GET INTO STEP 1  ZZZ
;  3           BEGIN                                            !BEFORE STEP 1 WRITE           ZZZ
;  3           SA = .RDRX_ADDR [RCSA, RC_ALL];                  ! STEP 1 READ
;  3             IF (.SA AND S1_MASK) EQL SA_S1                 !DID WE GET THE S1 BIT?        ZZZ
;  3             THEN                                           !                              ZZZ
;  3             EXITLOOP;                                      !EXIT IF SO                    ZZZ
;  3           DELAY (1);                                       !                              ZZZ
;  3           BREAK;                                           !                              MMM
;  2           END;                                             !                              ZZZ
;  2
;  2     SA = (WR_RING ^ 8) or (.CURRENT_VECTOR ^ -2) or SA_INT;  ! STEP 1 WRITE VALUE
;  2      WRT_RDRX (RCSA, RC_ALL, .SA);                         ! STEP 1 WRITE
;  2
;  2       incr COUNT from 1 to 8000 do
;  3           begin
;  3           DELAY (1);                                       ! TOTAL DELAY COUNT OF 8,000
;  3
;  3           if .DCT_ADDR [SA_SAVE] neq 0                     ! IF SA WAS CHANGED
;  3           then
;  3               return TRUE;                                 ! INTERRUPT OCCURED
;  3
;  3           BREAK;
;  2           end;
;  2
;  2       return FALSE;                                        ! IF INTERRUPT DID NOT OCCUR
;  1       end;


                                              .SBTTL  INT.GEN INITIALIZATION TEST ROUTINES
000000  004137  000000G             INT.GEN::
                                              JSR     R1,$SAVE3                     ;        2657
000004  024646                                CMP     -(SP),-(SP)
```

```
000006  013700  000000G                 MOV     DCT.ADDR,R0             ;                       2674
000012  005060  000002                  CLR     2(R0)
000016  012700  177777                  MOV     #-1,R0                  ; *,RC.REG              2675
000022  010077  000000G                 MOV     R0,@RDRX.ADDR           ; RC.REG,*
000026  012701  000002                  MOV     #2,R1                   ; *,$$TMP2              2676
000032  001411                  1$:     BEQ     4$
000034  013700  000000G                 MOV     L$DLY,R0                ; *,$$TMP1
000040  001404                          BEQ     3$
000042  005066  000002          2$:     CLR     2(SP)                   ; $$TMP
000046  005300                          DEC     R0                      ; $$TMP1
000050  001374                          BNE     2$
000052  005301                  3$:     DEC     R1                      ; $$TMP2
000054  000766                          BR      1$
000056  104422                  4$:     TRAP    22
000060  012703  000764                  MOV     #764,R3                 ; *,COUNT               2678
000064  013700  000000G         5$:     MOV     RDRX.ADDR,R0            ;                       2680
000070  016016  000002                  MOV     2(R0),(SP)              ; *,RC.REG
000074  011602                          MOV     (SP),R2                 ; RC.REG,SA
000076  010200                          MOV     R2,R0                   ; SA,*                  2681
000100  042700  001777                  BIC     #1777,R0
000104  020027  004000                  CMP     R0,#4000
000110  001417                          BEQ     10$                     ;                       2683
000112  012701  000001                  MOV     #1,R1                   ; *,$$TMP2              2684
000116  001411                  6$:     BEQ     9$
000120  013700  000000G                 MOV     L$DLY,R0                ; *,$$TMP1
000124  001404                          BEQ     8$
000126  005066  000002          7$:     CLR     2(SP)                   ; $$TMP
000132  005300                          DEC     R0                      ; $$TMP1
000134  001374                          BNE     7$
000136  005301                  8$:     DEC     R1                      ; $$TMP2
000140  000766                          BR      6$
000142  104422                  9$:     TRAP    22
000144  005303                          DEC     R3                      ; COUNT                 2678
000146  001346                          BNE     5$
000150  013700  001246'         10$:    MOV     CURRENT.VECTOR,R0       ;                       2688
000154  006200                          ASR     R0
000156  006200                          ASR     R0
000160  010002                          MOV     R0,R2                   ; *,SA
000162  052702  111200                  BIS     #111200,R2              ; *,SA
000166  010201                          MOV     R2,R1                   ; SA,RC.REG             2689
000170  013700  000000G                 MOV     RDRX.ADDR,R0
000174  010160  000002                  MOV     R1,2(R0)                ; RC.REG,*
000200  012702  017500                  MOV     #17500,R2               ; *,COUNT               2691
000204  012701  000001          11$:    MOV     #1,R1                   ; *,$$TMP2              2693
000210  001411                  12$:    BEQ     15$
000212  013700  000000G                 MOV     L$DLY,R0                ; *,$$TMP1
000216  001404                          BEQ     14$
000220  005066  000002          13$:    CLR     2(SP)                   ; $$TMP
000224  005300                          DEC     R0                      ; $$TMP1
000226  001374                          BNE     13$
000230  005301                  14$:    DEC     R1                      ; $$TMP2
000232  000766                          BR      12$
000234  013700  000000G         15$:    MOV     DCT.ADDR,R0             ;                       2695
```

```
000240  005760  000002          TST     2(RO)
000244  001403                  BEQ     16$
000246  012700  000001          MOV     #1,RO                     ;                    2697
000252  000404                  BR      17$
000254  104422          16$:    TRAP    22
000256  005302                  DEC     R2                        ; COUNT              2691
000260  001351                  BNE     11$
000262  005000                  CLR     RO                        ;                    2669
000264  022626          17$:    CMP     (SP)+,(SP)+               ;                    2657
000266  000207                  RTS     PC

; Routine Size:  92 words,      Routine Base:  $CODE$ + 3164
; Maximum stack depth per invocation:  8 words
```

```
;  1  GLOBAL routine HARD_INIT =
;  1
;  1  !+
;  1  !      THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
;  1  !      INITIALIZE AN RDRX DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
;  1  !      THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
;  1  !      OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
;  1  !      IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
;  1  !-
;  1
;  2      begin
;  2
;  2      local
;  2          IE_VEC : word;                                  ! IE-BIT AND VECTOR-ADDRESS/4 BYTE
;  2                                                          !   (USED IN STEP 1 WRITE AND STEP 3 READ)
;  2
;  2      IE_VEC = .CURRENT_VECTOR ^  2;                      ! GET VECTOR ADDR/4 (IE = 0)
;  2
;  2      incr ATTEMPTS from 1 to INI_ATT do
;  3          begin
;  3
;  3          label
;  3              STEP_1_READ,
;  3              STEP_2_READ,
;  3              STEP_3_READ,
;  3              STEP_4_READ;
;  3
;  3          WRT_RDRX (RCIP, RC_ALL, ALL ONES);              ! WRITE IP TO START INIT SEQUENCE
;  3  !
;  3  !       STEP 1 READ
;  3  !
;  3          STEP = 1;
;  3          STEP_1_READ:
;  4              begin
;  4
;  4              incr COUNT from 1 to 500 do
;  5                  begin
;  5                  DELAY (1);                              ! TOTAL DELAY COUNT OF 500 FOR STEP 1
;  5                  SA REG = .RDRX_ADDR [RCSA, RC_ALL];     ! READ SA
;  5
;  5                  if (.SA_REG and S1_MASK) eql SA_S1      ! IF STEP 1 READ IS O.K.
;  5                  then
;  5                      leave STEP 1_READ;
;  5
;  5                  BREAK;
;  4                  end;
;  4
;  4              exitloop;
;  3              end;
;  3
;  3  !
;  3  !       STEP 1 WRITE
;  3  !
```

```
3           SA_REG = (WR_RING ↑ 8) or .IE_VEC;                          ! STEP 1 WRITE VALUE
3           WRT_RDRX (RCSA, RC_ALL, .SA_REG);                           ! STEP 1 WRITE
3       !
3       !   STEP 2 READ
3       !
3           STEP = .STEP + 1;
3           STEP_2_READ:
4               begin
4
4               incr COUNT from 1 to 10000 do
5                   begin
5                   DELAY (1);                                          ! TOTAL DELAY COUNT OF 10,000 FOR STEP 2
5                   SA_REG = .RDRX_ADDR [RCSA, RC_ALL];                 ! READ SA
5
6                   if (.SA_REG and S2_MASK) eql (SA_S2 or WR_RING) ! IF STEP 2 READ IS O.K.
5                   then
5                       leave STEP_2_READ;
5
5                   BREAK;
4                   end;
4
4               exitloop;
3               end;
3
3       !
3       !   STEP 2 WRITE
3       !
3           WRT_RDRX (RCSA, RC_ALL, .DCT_ADDR [RR BEG]);                ! RINGBASE LO, PI = 0
3       !
3       !   STEP 3 READ
3       !
3           STEP = .STEP + 1;
3           STEP_3_READ:
4               begin
4
4               incr COUNT from 1 to 10000 do
5                   begin
5                   DELAY (1);                                          ! TOTAL DELAY COUNT OF 10,000 FOR STEP 3 READ
5                   SA_REG = .RDRX_ADDR [RCSA, RC_ALL];                 ! READ SA
5
6                   if (.SA_REG and S3_MASK) eql (SA_S3 or .IE_VEC) ! IF STEP 3 READ IS O.K.
5                   then
5                       leave STEP 3_READ;
5
5                   BREAK;
4                   end;
4
4               exitloop;
3               end;
3
3       !
3       !   STEP 3 WRITE
3       !
```

```
;   3               WRT_RDRX (RCSA, RC_ALL, 0);                        ! PP, RINGBASE-HI = 0
;   3       !
;   3       !       STEP 4 READ
;   3       !
;   3               STEP = .STEP + 1;
;   3               STEP_4_READ:
;   4                   begin
;   4
;   4                   incr COUNT from 1 to 10000 do
;   5                       begin
;   5                       DELAY (1);                                 ! TOTAL DELAY COUNT OF 10,000 FOR STEP 4 READ
;   5                       SA_REG = .RDRX_ADDR [RCSA, RC_ALL];        ! READ SA
;   5
;   5                       if (.SA_REG and S4 MASK) eql SA_S4         ! IF STEP 4 READ IS O.K.
;   5                       then
;   5                           leave STEP_4_READ;
;   5
;   5                       BREAK;
;   4                       end;
;   4
;   4                   exitloop;
;   3                   end;
;   3       !
;   3       !       STEP 4 WRITE
;   3       !
;   3               CREDIT_BAL = 1;                                    ! START WITH A CREDIT BALANCE = 1
;   3               WRT_RDRX (RCSA, RC_ALL, 0);                        ! BURST, LF, GO = 0
;   3               return SUCCESS;                                    ! SUCCESS EXIT POINT
;   3
;   2               end;                                               ! TRY AGAIN OR GIVE UP
;   2
;   2           CREDIT_BAL = 0;                                        ! NO CREDIT BALANCE
;   2           C_ERR_TBL [.CCTLR, C_ERR HRD] = .C_ERR TBL [.CCTLR, C ERR_HRD] + 1;
;   2
;   2           if .APT_MODE
;   2           then
;   3               begin
;   3               .MAIL_BOX_TESTNUM = 1;
;   3               .MAIL_BOX_SUBTST = 0;
;   2               end;
;   2
;   2           ERRDF (13, EGD_13, EMS_13);                            ! INIT SEQUENCE FAILED
;   2           return FAILURE;
;   1           end;                                                   ! ROUTINE HARD_INIT


                                                  .SBTTL  HARD.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G                 HARD.INIT::
                                                  JSR     R1,$SAVE5                          ;          2704
000004  162706  000012                            SUB     #12,SP
000010  013704  001246'                           MOV     CURRENT.VECTOR,R4                  ; *,IE.VEC 2720
000014  006204                                    ASR     R4                                 ; IE.VEC
000016  006204                                    ASR     R4                                 ; IE.VEC
```

```
000020  012705  000002                    MOV   #2,R5                 ; *.ATTEMPTS       2722
000024  012700  177777                    MOV   #-1,R0                ; *.RC.REG         2731
000030  010077  000000G                   MOV   R0,@RDRX.ADDR         ; RC.REG.*
000034  012737  000001  000000G           MOV   #1,STEP               ;                  2735
000042  012702  000764                    MOV   #764,R2               ; *.COUNT          2739
000046  012701  000001            1$:     MOV   #1,R1                 ; *.$$TMP2         2741
000052  001411                    2$:     BEQ   5$
000054  013700  000000G                   MOV   L$DLY,R0              ; *.$$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                    ; $$TMP2
000074  000766                            BR    2$
000076  013700  000000G           5$:     MOV   RDRX.ADDR,R0          ;                  2742
000102  016016  000002                    MOV   2(R0),(SP)            ; *.RC.REG
000106  011637  000000G                   MOV   (SP),SA.REG           ; RC.REG.*
000112  011600                            MOV   (SP),R0               ; SA.REG.*         2744
000114  042700  001777                    BIC   #1777,R0
000120  020027  004000                    CMP   R0,#4000
000124  001404                            BEQ   6$                    ;                  2746
000126  104422                            TRAP  22
000130  005302                            DEC   R2                    ; COUNT            2739
000132  001345                            BNE   1$
000134  000532                            BR    18$                   ;                  2723
000136  010437  000000G           6$:     MOV   R4,SA.REG             ; IE.VEC.*         2757
000142  052737  111000  000000G           BIS   #111000,SA.REG
000150  013701  000000G                   MOV   SA.REG,R1             ; *.RC.REG         2758
000154  013700  000000G                   MOV   RDRX.ADDR,R0
000160  010160  000002                    MOV   R1,2(R0)              ; RC.REG.*
000164  005237  000000G                   INC   STEP                  ;                  2762
000170  012702  023420                    MOV   #23420,R2             ; *.COUNT          2766
000174  012701  000001            7$:     MOV   #1,R1                 ; *.$$TMP2         2768
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$:    MOV   RDRX.ADDR,R0          ;                  2769
000230  016066  000002  000002            MOV   2(R0),2(SP)           ; *.RC.REG
000236  016637  000002  000000G           MOV   2(SP),SA.REG          ; RC.REG.*
000244  016600  000002                    MOV   2(SP),R0              ; SA.REG.*         2771
000250  042700  003400                    BIC   #3400,R0
000254  020027  010222                    CMP   R0,#10222
000260  001404                            BEQ   12$                   ;                  2773
000262  104422                            TRAP  22
000264  005302                            DEC   R2                    ; COUNT            2766
000266  001342                            BNE   7$
000270  000537                            BR    26$                   ;                  2723
000272  013700  000000G           12$:    MOV   DCT.ADDR,R0           ;                  2784
```

```
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                ;                   2788
000316  010403                            MOV   R4,R3               ; IE.VEC,*          2797
000320  052703  020000                    BIS   #20000,R3
000324  012702  023420                    MG.   #23420,R2           ; *,COUNT           2792
000330  012701  000001           13$:     MOV   #1,R1               ; *,$$TMP2          2794
000334  001411                   14$:     BEQ   17$
000336  013700  000000G                   MOV   L$DLY,R0            ; *,$$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                  ; $$TMP2
000356  000766                            BR    14$
000360  013700  000000G          17$:     MOV   RDRX.ADDR,R0        ;                   2795
000364  016066  000002  000004            MOV   2(R0),4(SP)         ; *,RC.REG
000372  016637  C00004  000000G           MOV   4(SP),SA.REG        ; RC.REG,*
000400  016600  000004                    MOV   4(SP),R0            ; SA.REG,*          2797
000404  042700  003400                    BIC   #3400,R0
000410  020003                            CMP   R0,R3
000412  001404                            BEQ   19$
000414  104422                            TRAP  22                  ;                   2799
000416  005302                            DEC   R2                  ; COUNT             2792
000420  001343                            BNE   13$
000422  000462                   18$:     BR    26$                 ;                   2723
000424  013700  000000G          19$:     MOV   RDRX.ADDR,R0        ;                   2810
000430  005060  000002                    CLR   2(R0)
000434  005237  000000G                   INC   STEP                ;                   2814
000440  012703  023420                    MOV   #23420,R3           ; *,COUNT           2818
000444  012701  000001           20$:     MOV   #1,R1               ; *,$$TMP2          2820
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$:     MOV   RDRX.ADDR,R0        ;                   2821
000500  016066  000002  000006            MOV   2(R0),6(SP)         ; *,RC.REG
000506  016637  000006  000000G           MOV   6(SP),SA.REG        ; RC.REG,*
000514  016600  000006                    MOV   6(SP),R0            ; SA.REG,*          2823
000520  042700  003777                    BIC   #3777,R0
000524  020027  040000                    CMP   R0,#40000
000530  001404                            BEQ   25$
000532  104422                            TRAP  22                  ;                   2825
000534  005303                            DEC   R3                  ; COUNT             2818
000536  001342                            BNE   20$
000540  000413                            BR    26$                 ;                   2723
000542  012737  000001  000000G  25$.     MOV   #1,CREDIT.BAL       ;                   2835
000550  005001                            CLR   R1                  ; RC.REG            2836
```

```
000552  013700  000000G                  MOV     RDRX.ADDR,R0
000556  005060  000002                   CLR     2(R0)
000562  012700  000001                   MOV     #1,R0                                    ;        2723
000566  000425                           BR      28$
000570  005037  000000G          26$:    CLR     CREDIT.BAL                               ;        2841
000574  013700  000000G                  MOV     CCTLR,R0                                 ;        2842
000600  006300                           ASL     R0
000602  105260  000000G                  INCB    C.ERR.TBL(R0)
000606  032737  000001  001254'          BIT     #1,APT.MODE                              ;        2844
000614  001405                           BEQ     27$
000616  012777  000001  001256           MOV     #1,@MAIL.BOX.TESTNUM                     ;        2847
000624  005077  001260'                  CLR     @MAIL.BOX.SUBTST                         ;        2848
000630  104455                   27$:    TRAP    55                                       ;        2851
000632  000015                           .WORD   15
000634  000000G                          .WORD   EGD.13
000636  000000G                          .WORD   EMS.13
000640  005000                           CLR     R0                                       ;        2714
000642  062706  000012           28$:    ADD     #12,SP                                   ;        2704
000646  000207                           RTS     PC

; Routine Size:  212 words,     Routine Base:  $CODE$ + 3454
; Maximum stack depth per invocation:  13 words
```

```
;    2854  1        GLOBAL rout ne INI RRING . novalue =
;    2855  1
;    2856  1        !+
;    2857  1        !         THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP PACKETS TO
;    2858  1        !         FILL AN RDRX RESPONSE RING. THE BUFFER DESCRIPTOR OF EACH PACKET
;    2859  1        !         (LOCATED IN FRONT OF THE PACKET ITSELF) IS LOADED INTO SUCCESSIVE
;    2860  1        !         RRING SLOTS. NOTE THAT THE BUFFER DESCRIPTORS HAVE BEEN INITIALIZED
;    2861  1        !         WITH THE FLAG AND OWNERSHIP BITS SET TO "1', MAKING EACH SLOT
;    2862  1        !         CONTROLLER OWNED.
;    2863  1        !
;    2864  1        !         IMPLICIT INPUTS:
;    2865  1        !                 CCTLR    CURRENT CONTROLLER NUMBER
;    2866  1        !                 DCT_ADDR    ADDRESS OF CURRENT CONTROLLER S DCT
;    2867  1        !
;    2868  1
;    2869  2            begin
;    2870  2
;    2871  2            local
;    2872  2                index : word,
;    2873  2                RRING_ADDR;
;    2874  2
;    2875  2            RRING_ADDR = .DCT ADDR [RR BEG];                       ! FIRST RESPONSE RING SLOT
;    2876  2
;    2877  2            incr COUNT from 1 to RRING_LEN do
;    2878  3                begin
;    2879  3                index = GET PKT (.CCTLR);                          ! GET AN MSCP PACKET
;    2880  3                .RRING_ADDR = .MSCP_PKT [.index, PKT LO];          ! LOAD LO ORDER BUFF DESC INTO SLOT
;    2881  3                RRING_ADDR = .RRING_ADDR + 2;                      ! ADVANCE TO SECOND WORD
;    2882  3                .RRING_ADDR = .MSCP_PKT [.index, PKT_HI];          ! LOAD HI ORDER BUFF DESC INTO SLOT
;    2883  3                PKT_USE [.index] = .CCTLR;                         ! PACKET IN USE
;    2884  3                .RRING_ADDR = ..RRING_ADDR or ED OWN or ED FLAG;   ! GIVE OWNERSHIP TO CONTRLLER
;    2885  3                RRING_ADDR = .RRING_ADDR + 2;                      ! ADVANCE TO NEXT SLOT
;    2886  2                end;
;    2887  2
;    2888  1            end;


                                          .SBTT    INI.RRING INITIALIZATION TEST ROUTINES
000000  004137  000000G           INI.RRING::
                                          JSR      R1,$SAVE4                     ;
000004  013700  000000G                   MOV      DCT.ADDR,R0                   ;                    2854
000010  016001  000004                    MOV      4(R0),R1                      ;                    2875
000014  013703  000000G                   MOV      CCTLR,R3                      ; *,RRING.ADDR
000020  012704  000004                    MOV      #4,R4                         ;                    2879
000024  010346                    1$:     MOV      R3,-(SP)                      ; *,COUNT            2877
000026  004737  000000G                   JSR      PC,GET.PKT                    ;                    2879
000032  010002                            MOV      R0,R2                         ; *,INDEX
000034  010216                            MOV      R2,(SP)                       ; INDEX,*
000036  012746  000106                    MOV      #106,-(SP)                    ;                    2880
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       2882
000056  013703  000000G                   MOV      CCTLR,R3                      ;                    2883
```

```
000062  110362  000000G            MOVB    R3,PKT.USE(R2)            ; .,.(INDEX)
000066  052721  140000             BIS     #140000,(R1)+             ; .,RRING.ADDR                 2894
000072  022626                     CMP     (SP)+,(SP)+               ;                              2878
000074  005304                     DEC     R4                        ; COUNT                        2877
000076  001352                     BNE     1$
000100  000207                     RTS     PC                        ;                              2854

; Routine Size:  33 words,      Routine Base:  $CODE$ + 4324
; Maximum stack depth per invocation:  8 words
```

```
GLOBAL routine SET_CTLR_CHAR =

!+
!       THIS ROUTINE IS CALLED BY CTLR INIT AFTER THE RDRX HAS BEEN HARD-
!       INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
!       CHARACTERISTICS" COMMAND, AND TO VALIDATE THE RESPONSE (END MESSAGE).
!
!       IMPLICIT INPUTS:
!               CCTLR - CURRENT CONTROLLER NUMBER
!-

    begin

    local
        P_INDEX : word;

! MISCELLANEOUS INITIALIZATION                                          !ZZZ
                                                                        !ZZZ
QIO [.CCTLR] = 0;                  !INIT NO OF OUTSTANDING QIOS         !ZZZ
CST [.CCTLR, U_CNT] = 0;           !CLEAR UNITS IN CST TABLE            !ZZZ
INCR COUNT FROM 0 TO (RP CNT   1) DO    !INIT RETURN PACKET POOL        !ZZZ
    RP_USE [.COUNT] = -1;                                               !ZZZ
                                                                        !ZZZ
IODQ_IN = IODQ_OUT = 0;            !INIT I/O DONE QUEUE POINTERS        !ZZZ


    P_INDEX = GET_PKT (.CCTLR);                             ! GET AN MSCP PACKET
    MSCP_PKT [.P_INDEX, MSGLEN] = SZ_SCC;                   ! PACKET SIZE
    MSCP_PKT [.P_INDEX, OPCODE] = OP_SCC;                   ! OPCODE = SET CTLR CHAR
    MSCP_PKT [.P_INDEX, C_FLAGS] = CF_MASK;                 ! CONTROLLER FLAGS
    MSCP_PKT [.P_INDEX, CMD TYPE] = IMM_CMD;                ! IMMEDIATE COMMAND

    if SEND (.P_INDEX) eql FAILURE                          ! ATTEMPT SEND
    then
        begin                                               ! IF SEND WAS UNSUCCESSFUL
        C_ERR TBL [.CCTLR, C ERR HRD] = .C ERR_TBL [.CCTLR, C_ERR_HRD] + 1;

        if .APT_MODE
        then
            begin
            .MAIL_BOX_TESTNUM = 1;
            .MAIL_BOX_SUBTST = 0;
            end;

        ERRDF (20, EGD_20, 0);                              ! FATAL ERROR
        PUT_PKT (.P_INDEX);                                 ! RETURN PACKET TO POOL
        DROP_CTLR (.CCTLR, DU CFATAL);                      ! DROP CONTROLLER
        return FAILURE;
        end
    else
        begin                                               ! IF SEND WAS SUCCESSFUL
```

```
;  3          do
;  4              begin
;  4              WAIT ();                                          ! WAIT FOR RETPKT RESPONSE
;  4              RP_INDX = OUT_IODQ ();                            ! GET INDEX OF RETPKT
;  4              RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);      ! CALCULATE RETPKT ADDRESS
;  4
;  4              if .RP_ADDR [MESTYP] neq MT_SEQ                   ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;  4              then
;  4                  PUT_RETPKT (.RP_INDX);
;  4
;  4              end
;  3          until (.RP_ADDR [CONID] eql CID_DRIVER) or
;  4              ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;  3               ((.RP_ADDR [ENDCOD] and OP_END) eql OP END));
;  3
;  3          if .RP_ADDR [CONID] eql CID_DRIVER                    ! IF RETPKT IS FROM "DRIVER"
;  3          then
;  4              begin
;  4              PRINTF (DBM23);                                   ! "ERROR IN SET_CTLR_CHAR"
;  4              PUT_RETPKT (.RP_INDX);                            ! RELEASE RETURN PACKET
;  4              DR_ERR ();                                        ! DROP CONTROLLER
;  4              return FAILURE;
;  4              end
;  3          else
;  4              begin                                             ! ELSE - RETPKT IS FROM DISK MSCP
;  4
;  4              if (.RP_ADDR [ENDCOD] neq (OP_SCC or OP_END)) or  ! IF WRONG ENDCODE
;  5                 ((.RP_ADDR [C_FLGS] and CF_MASK) neq CF_MASK)  ! OR FLAGS IN ERROR
;  4              then
;  5                  begin
;  5                  C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR TBL [.CCTLR, C_ERR_HRD] + 1;
;  5
;  5                  if .APT_MODE
;  5                  then
;  6                      begin
;  6                      .MAIL_BOX_TESTNUM = 1;
;  5                      .MAIL_BOX_SUBTST = 0;
;  5                      end;
;  5
;  5                  ERRDF (21, EGD_21, EMS_21);                   ! FATAL ERROR
;  5                  DROP_CTLR (.CCTLR, DU_CFATAL);                ! DROP CONTROLLER
;  5                  PUT_RETPKT (.RP_INDX);                        ! RELEASE RETURN PACKET
;  5                  return FAILURE;
;  5                  end
;  4              else
;  5                  begin                                         ! RETPKT HAS CORRECT ENDCODE
;  5                  CMD TIME = .RP_ADDR [C_TIME] * 2;
;  5
;  5  !ZZZ             if BIT_TST (SWP FLAGS, SWF_TRC)
;  5  !ZZZ             then
;  5  !ZZZ                 PRINTF (DBM25, .RP_ADDR [C_TIME]);
;  5
;  4                  end;                                          ! RETPKT HAS CORRECT ENDCODE
```

```
;    2995  4
;    2996  3            end;                                  ! IF RETPKT WAS SENT BY DISK MSCP
;    2997  3
;    2998  3         PUT_RETPKT (.RP_INDX);
;    2999  3         return SUCCESS;                          ! IF SEND WAS SUCCESSFUL
;    3000  2         end;
;    3001  2
;    3002  1      end;                                        ! ROUTINE SET_CTLR_CHAR


                                          .SBTTL  SET.CTLR.CHAR INITIALIZATION TEST ROUTINES
000000  010146                    SET.CTLR.CHAR::
                                          MOV     R1,-(SP)
000002  013701  000000G                   MOV     CCTLR,R1                  ;                  2889
000006  105061  000000G                   CLRB    QIO(R1)                   ;                  2908
000012  010146                            MOV     R1,-(SP)
000014  012746  000126                    MOV     #126,-(SP)                ;                  2909
000020  004737  000000G                   JSR     PC,BL$MUL
000024  105060  000005G                   CLRB    CST+5(R0)
000030  005000                            CLR     R0
000032  112760  000377  000000G   1$:     MOVB    #377,RP.USE(R0)           ; COUNT            2910
000040  005200                            INC     R0                        ; *,*(COUNT)       2911
000042  020027  000007                    CMP     R0,#7                     ; COUNT            2910
000046  003771                            BLE     1$                        ; COUNT,*
000050  005037  000000G                   CLR     IODQ.OUT
000054  005037  000000G                   CLR     IODQ.IN                   ;                  2913
000060  010116                            MOV     R1,(SP)
000062  004737  000000G                   JSR     PC,GET.PKT                ;                  2916
000066  010001                            MOV     R0,R1
000070  010116                            MOV     R1,(SP)                   ; *,P.INDEX
000072  012746  000106                    MOV     #106,-(SP)                ; P.INDEX,*        2917
000076  004737  000000G                   JSR     PC,BL$MUL
000102  012760  000040  000006G           MOV     #40,MSCP.PKT+6(R0)
000110  112760  000004  000022G           MOVB    #4,MSCP.PKT+22(R0)
000116  012760  000120  000030G           MOV     #120,MSCP.PKT+30(R0)      ;                  2918
000124  105060  000004G                   CLRB    MSCP.PKT+4(R0)            ;                  2919
000130  010116                            MOV     R1,(SP)                   ;                  2920
000132  004737  000000G                   JSR     PC,SEND                   ; P.INDEX,*        2922
000136  005700                            TST     R0
000140  001036                            BNE     3$
000142  013700  000000G                   MOV     CCTLR,R0
000146  006300                            ASL     R0                        ;                  2925
000150  105260  000000G                   INCB    C.ERR.TBL(R0)
000154  032737  000001  001254'           BIT     #1,APT.MODE
000162  001405                            BEQ     2$                        ;                  2927
000164  012777  000001  001256'           MOV     #1,@MAIL.BOX.TESTNUM
000172  005077  001260'                   CLR     @MAIL.BOX.SUBTST          ;                  2930
000176  104455                    2$:     TRAP    55                        ;                  2931
000200  000024                            .WORD   24                        ;                  2934
000202  000000G                           .WORD   EGD.20
000204  000000                            .WORD   0
000206  010116                            MOV     R1,(SP)                   ; P.INDEX,*        2935
000210  004737  000000G                   JSR     PC,PUT.PKT
```

```
000214  013716  000000G                  MOV     CCTLR,(SP)            ;              2936
000220  012746  000006                   MOV     #6.-(SP)
000224  004737  000000G                  JSR     PC,DROP.CTLR
000230  005726                           TST     (SP)+                 ;              2924
000232  005000                           CLR     R0                    ;              2940
000234  000554                           BR      12$
000236  004737  000000G          3$:     JSR     PC,WAIT               ;              2944
000242  004737  000000G                  JSR     PC,OUT.IODQ           ;              2945
000246  010037  000000G                  MOV     R0,RP.INDX
000252  010016                           MOV     R0,(SP)               ; RP.INDX,*    2946
000254  012746  000054                   MOV     #54,-(SP)
000260  004737  000000G                  JSR     PC,BL$MUL
000264  062700  000000G                  ADD     #RETPKT,R0
000270  010037  000000G                  MOV     R0,RP.ADDR
000274  132760  000360  000002           BITB    #360,2(R0)            ;              2948
000302  001404                           BEQ     4$
000304  013716  000000G                  MOV     RP.INDX,(SP)          ;              2950
000310  004737  000000G                  JSR     PC,PUT.RETPKT
000314  005726                   4$:     TST     (SP)+                 ;              2943
000316  013701  000000G                  MOV     RP.ADDR,R1            ;              2953
000322  005000                           CLR     R0
000324  126127  000003  000003           CMPB    3(R1),#3
000332  001002                           BNE     5$
000334  005200                           INC     R0
000336  000407                           BR      6$
000340  132761  000360  000002   5$:     BITB    #360,2(R1)            ;              2954
000346  001333                           BNE     3$
000350  105761  000014                   TSTB    14(R1)                ;              2955
000354  100330                           BPL     3$
000356  006000                   6$:     ROR     R0                    ;              2957
000360  103015                           BCC     7$
000362  012716  000000G                  MOV     #DBM23,(SP)           ;              2960
000366  012746  000001                   MOV     #1,-(SP)
000372  010600                           MOV     SP,R0                 ; SP,*
000374  104417                           TRAP    17
000376  013716  000000G                  MOV     RP.INDX,(SP)          ;              2961
000402  004737  000000G                  JSR     PC,PUT.RETPKT
000406  004737  000000V                  JSR     PC,DR.ERR             ;              2962
000412  000447                           BR      10$                   ;              2963
000414  126127  000014  000204   7$:     CMPB    14(R1),#204           ;              2968
000422  001007                           BNE     8$
000424  016100  000022                   MOV     22(R1),R0             ;              2969
000430  042700  177657                   BIC     #177657,R0
000434  020027  000120                   CMP     R0,#120
000440  001437                           BEQ     11$
000442  013700  000000G          8$:     MOV     CCTLR,R0              ;              2972
000446  006300                           ASL     R0
000450  105260  000000G                  INCB    C.ERR.TBL(R0)
000454  032737  000001  001254'          BIT     #1,APT.MODE           ;              2974
000462  001405                           BEQ     9$
000464  012777  000001  001256'          MOV     #1,@MAIL.BOX.TESTNUM  ;              2977
000472  005077  001260'                  CLR     @MAIL.BOX.SUBTST      ;              2978
000476  104455                   9$:     TRAP    55                    ;              2981
```

```
000500  000025                              .WORD   25
000502  000000G                             .WORD   EGD.21
000504  000000G                             .WORD   EMS.21
000506  013716  000000G                     MOV     CCTLR,(SP)                  ;        2982
000512  012746  000006                      MOV     #6,-(SP)
000516  004737  000000G                     JSR     PC,DROP.CTLR
000522  013716  000000G                     MOV     RP.INDX,(SP)                ;        2983
000526  004737  000000G                     JSR     PC,PUT.RETPKT
000532  062706  000010            10$:      ADD     #10,SP
000536  000416                              BR      13$                         ;        2984
000540  016137  000024  000000G   11$:      MOV     24(R1),CMD.TIME             ;        2971
000546  006337  000000G                     ASL     CMD.TIME                    ;        2988
000552  013716  000000G                     MOV     RP.INDX,(SP)
000556  004737  000000G                     JSR     PC,PUT.RETPKT               ;        2998
000562  012700  000001                      MOV     #1,R0
000566  062706  000006            12$:      ADD     #6,SP                       ;        2940
000572  000401                              BR      14$                         ;        2922
000574  005000                    13$:      CLR     R0                          ;        2900
000576  012601                    14$:      MOV     (SP)+,R1                    ;        2889
000600  000207                              RTS     PC

; Routine Size:  193 words,     Routine Base:  $CODE$ + 4426
; Maximum stack depth per invocation:  7 words
```

```
;   1   routine UNIT_INIT : novalue =
;   1
;   1   !+
;   1   !       THIS ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONFIGURED UNIT
;   1   !       (DISK) WHICH IS ATTACHED TO A CONTROLLER THAT SURVIVED
;   1   !       INITIALIZATION. ITS PURPOSE IS TO FORMAT AND SEND AN "ONLINE"
;   1   !       MESSAGE, AND TO VERIFY THE RESPONSE.
;   1   !
;   1   !       IMPLICIT INPUTS:
;   1   !               CCTLR - CURRENT CONTROLLER NUMBER
;   1   !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;   1   !               L$LUN - CURRENT (DRS) UNIT NUMBER
;   1   !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;   1   !-
;   1
;   2       begin
;   2       local
;   2           MAX0_LBNS : WORD UNSIGNED,              ! UNIT'S MAXIMUM LO WORD LBN
;   2           MAX1_LBNS : WORD UNSIGNED;              ! UNIT'S MAXIMUM HI WORD LBN
;   2
;   2
;   2       P_INDEX = GET_PKT (.CCTLR);                 ! GET AN MSCP PACKET
;   2       MSCP_PKT [.P_INDEX, MSGLEN] = SZ_ONL;       ! PACKET SIZE
;   2       MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;       ! SET DISK ADDRESS (RD/RX DISK NUMBER)
;   2       MSCP_PKT [.P_INDEX, OPCODE] = OP_ONL;       ! OPCODE FOR "ONLINE"
;   2   !ZZZ MSCP_PKT [.P_INDEX, DDPAR] = BIT00;        ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
;   2       MSCP_PKT [.P_INDEX, CMD TYPE] = SEQ_CMD;    ! SEQUENTIAL COMMAND
;   2
;   2       if SEND (.P_INDEX) eql FAILURE              ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
;   2       then
;   3           begin
;   3           T_ADDR [ERR HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;   3
;   3           if .APT_MODE                            !ZZZ
;   3           then
;   4               begin
;   4               .MAIL_BOX TESTNUM = 1;
;   4               .MAIL_BOX SUBTST = .CST ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;   3               end;
;   3
;   3           CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
;   3           ERRDF (22, EGD_22, 0);                  !
;   3           DUR [.L$LUN] = DU ONLINE;               ! SETUP REASON TO DROP UNIT
;   3           DODU (.L$LUN);                          ! DROP UNIT
;   3           PUT_PKT (.P INDEX);                     ! RETURN PACKET TO POOL
;   3           end
;   2       else
;   3           begin                                   ! OTHERWISE (SEND WAS SUCCESSFUL)
;   3
;   3           do
;   4               begin
;   4               WAIT ();                            ! WAIT FOR RETPKT RESPONSE
;   4               RP_INDX = OUT IODQ ();              ! GET INDEX OF RETPKT
```

```
                RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS

                if .RP_ADDR [MESTYP] neq MT_SEQ     ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
                then
                    PUT_RETPKT (.RP_INDX);

                end
            until (.RP_ADDR [CONID] eql CID_DRIVER) or
                ((.RP_ADDR [MESTYP] eql MT_SEQ) and
                 ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));

            if .RP_ADDR [CONID] eql CID_DRIVER      ! IF RETPKT IS FROM "DRIVER"
            then
                begin
                PRINTF (DBM26);                     ! "ERROR IN UNIT_INIT"
                DR_ERR ();                          ! DROP CONTROLLER
                end
            else

                if .RP_ADDR [ENDCOD] neq (OP_ONL or OP_END) ! IF RETPKT IS FROM DISK MSCP
                then
                    begin
                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;

                    if .APT_MODE                    !ZZZ
                    then
                        begin
                        .MAIL_BOX_TESTNUM = 1;
                        .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
                        end;

                    CST_ADDR [.CUOFF, D_FATAL] = TRUE;
                    ERRDF (23, EGD_23, EMS_21);     ! FATAL ERROR
                    DUR [.L$LUN] = DU_ONLINE;       ! SETUP REASON TO DROP UNIT
                    DODU (.L$LUN);                  ! DROP UNIT
                    end
                else
                    begin                           ! RETPKT HAS GOOD ENDCODE
                    ST_CODE = .RP_ADDR [STSCOD];    ! GET STATUS CODE
                    SB_CODE = .RP_ADDR [SUBCOD];    ! GET SUB-CODE

                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_0] = .RP_ADDR [NAME_0] + %o'100';        ! UNIT NAME
ZZZ
                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .RP_ADDR [NAME_1_HI] * 16;          !ZZZ
                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] +  !ZZZ
                                                              .RP_ADDR [NAME_1_LO] + %o'100';     !ZZZ
                    CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_2] = .RP_ADDR [NAME_NUM] / 10 + %o'60';  !ZZZ
                    CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_3] = (.RP_ADDR [NAME_NUM] mod 10) + %o'60';  !ZZZ



                    IF .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] EQL %O'104'     !IF NAME IS _D    !ZZZ
                    THEN                                                                          !ZZZ
                    CST_ADDR [.CUOFF, D_TYPE] = FIXED                           !ITS FIXED.       !ZZZ
```

```
;  4                    ELSE
;  4                    CST_ADDR [.CUOFF, D TYPE] = REMOVABLE;                       !OTHERWISE REMOVABLE     !ZZZ
;  4                                                                                                         !ZZZ
;  4
;  4
;  4                    if .ST_CODE neq ST SUC          ! IF STATUS CODE IS NOT SUCCESSFUL
;  4                    then
;  5                        begin
;  5                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5
;  5                        if .APT_MODE                 !ZZZ
;  5                        then
;  6                            begin
;  6                            .MAIL_BOX_TESTNUM = 1;
;  6                            .MAIL BOX_SUBTST = .CST ADDR [.CUGFF + OF_DATA, D_DISK_NUM];
;  5                            end;
;  5
;  5                        CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;  5                        ERRDF (15, EGD_15, EMS_30); ! ONLINE FAILED
;  5                        DUR [.L$LUN] = DU ONLINE;    ! SET UP REASON FOR DROPPING UNIT
;  5                        DODU (.L$LUN);               ! DROP UNIT
;  5                        end
;  4                    else
;  5                        begin                        ! SUCCESFUL OPERATION
;  5
;  5                        MAXO_LBNS = .RP_ADDR [SIZEO];                      ! LOAD LOWER WORD OF UNIT SIZE
;  5                        MAX1_LBNS = .RP_ADDR [SIZE1];                      ! LOAD UPPER WORD OF UNIT SIZE
;  5
;  6                        if (.MAXO_LBNS eql 0)                              ! THIS SUBTRACTS ONE FROM THE TOTAL
;  5                        then                                               ! BECAUSE EVERYTHING STARTS AT 0
;  6                            begin                                          ! THROUGH (MAXIMUM   1)
;  6                            MAXO LBNS = %O'177777';
;  6                            MAX1_LBNS = .MAX1_LBNS - 1;
;  6                            end
;  5                        else
;  5                            MAXO_LBNS = .MAXO_LBNS - 1;
;  5
;  5                        if  (.CST_ADDR [.CUOFF + 2, D_BEG1] gtru .MAX1_LBNS) or        ! THIS SECTION CHECKS TO SEE
 IF LBNs LISTED
;  5
;  6                            ((.CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .MAX1_LBNS) and      ! IN SOFTWARE QUESTIONS WERE
 TO LARGE FOR
;  6
;  6                            (.CST_ADDR [.CUOFF + 1, D_BEGO] gtru (.MAXO_LBNS - 1)))    ! DEVICE SPECIFIED
                                                                                 ! note 1 less then max. or diagnosti
c  will error
;  5                        then                                               ! operator error
;  6                            begin
;  6                            CST ADDR [.CUOFF + 2, D_BEG1] = 0;
;  6                            CST_ADDR [.CUOFF + 1, D_BEGO] = 0;             ! change beginning lbn to 0
;  5                            end;
;  5
;  5                         if
;  5                            (.CST_ADDR [.CUOFF + 4, D_END1] gtru .MAX1_LBNS) or
;  5
;  6                            ((.CST_ADDR [.CUOFF + 4, D_END1] eqlu .MAX1_LBNS) and
;  6                            (.CST_ADDR [.CUOFF + 3, D FNDO] gtru .MAXO_LBNS))
```

```
                        then
                           begin
                           CST_ADDR [.CUOFF + 4, D END1] = .MAX1_LBNS;
                           CST_ADDR [.CUOFF + 3, D END0] = .MAX0 LBNS;          ! and ending lbn to max_lbn
                           end;
                     if  (.CST_ADDR [.CUOFF + OF BEG1, D_BEG1] gtru             !MAKE SURE START ADDRESS
!ZZZ
                        .CST_ADDR [.CUOFF + OF_END1, D_END1]) or                !IS NO LARGER THAN END ADDRESS
!ZZZ
                                                                                !
!ZZZ
                        ((.CST_ADDR [.CUOFF + OF_BEG1, D BEG1] eqlu             !
!ZZZ
                          .CST_ADDR [.CUOFF + OF END1, D END1]) and             !
!ZZZ
                        (.CST_ADDR [.CUOFF + OF_BEG, D BEG0] gtru               !
!ZZZ
                         .CST_ADDR [.CUOFF + OF_END, D_END0] ))                 !
!ZZZ
                                                                                !
!ZZZ
                     then                                                       !
!ZZZ
                        begin                                                   !
 ZZZ
                        CST_ADDR [.CUOFF + OF_BEG1, D_BEG1] = 0;                !IF IT IS, THEN
 ZZZ
                        CST_ADDR [.CUOFF + OF_BEG, D BEG0] = 0;                 ! change beginning lbn to 0
 ZZZ
                        end;                                                    !
 ZZZ
!ZZZ

                        if (((.ENTRY_REASON eql RESTART) or              ! if restart or
                            (.ENTRY_REASON eql START)) and               ! if continue

                            (.CRN_LOW leq 8) and                         ! and
                            (.CRN_HIGH eql 0))                           ! first initialization
                        THEN                                             ! intialize block numbers
                            begin
                            BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0];     ! LOAD sequential LBN table
                            BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1];     !
                            TRK_SGN [.L$LUN] = 1;                                      ! POSITIVE TRACKING DIRECTIO
N
                            end;



!ZZZ                             selectoneu .RP_ADDR [R_MODEL] of        ! THIS SECTION LOADS TYPE INTO  CST TABLE
                                                                              ! MODEL BYTE TELLS WHAT TYPE OF UNIT
IN
!ZZZ                                                                          ! INDENTIFICATION BLOCK
!ZZZ                                 set
!ZZZ
!ZZZ                                 [%O'6'] :   CST_ADDR [.CUOFF, D_TYPE] = RD_51;       ! RD 51
!ZZZ                                 [%O'7'] :   CST_ADDR [.CUOFF, D_TYPE] = RX_50;       ! RX 50
!ZZZ                                 [%O'10'] :  CST_ADDR [.CUOFF, D_TYPE] = RD_52;       ! RD 52
!ZZZ
!ZZZ                                 [otherwise] : BEGIN
!ZZZ                                               ERRDF (25 ,EGD_24 ,EMS_30);           ! ERROR UNKNOWN DEVICE
!ZZZ                                               END;
!ZZZ                                 tes;



                     if ((.RP_ADDR [U_FLGS] and UF_WPH) eql UF_WPH) and          ! STATUS CODE IS O.K.
                        (.CST_ADDR [.CUOFF, D_PROT] eql UNPROTECTED)
```

```
;    3215  5                        then
;    3216  6                            begin
;    3217  6                            T_ADDR [ERR_HRD DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    3218  6
;    3219  6                            if .APT_MODE                        !ZZZ
;    3220  6                            then
;    3221  7                                begin
;    3222  7                                .MAIL_BOX_TESTNUM = 1;
;    3223  7                                .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    3224  6                                end;
;    3225  6
;    3226  6                            CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    3227  6                            ERROF (16, EGD_16, EMS_30);                       ! WRITE-PROTECT CONFLICT
;    3228  6                            DUR [.L$LUN] = DU_PROTECT;                        ! SET REASON TO DROP UNIT
;    3229  6                            DODU (.L$LUN);                                    ! DROP UNIT
;    3230  6                            end
;    3231  5                        else
;    3232  6                            begin                                             ! WRITE PROTECT SWITCH IS O.K.
;    3233  6                            CST_ADDR [.CUOFF, D_STAT] = ONLINE;               ! SET ONLINE FLAG
;    3234  6                            CST [.CCTLR, U_CNT] = .CST [.CCTLR, U_CNT] + 1;   ! ADD UNIT TO CTRL TABLE
;    3235  5                            end;
;    3236  4                        end;
;    3237  3                    end;                    ! IF RETPKT HAS CORRECT ENDCODE
;    3238  3
;    3239  3        PUT_RETPKT (.RP_INDX);
;    3240  2        end;                                ! IF SEND WAS SUCCESSFUL
;    3241  2
;    3242  1    end;                                    ! ROUTINE UNIT-INIT

                                                  .SBTTL  UNIT.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G                   UNIT.INIT:
                                                  JSR     R1,$SAVE5                       ;                    3003
000004  005746                                    TST     -(SP)
000006  013746  000000G                           MOV     CCTLR,-(SP)                     ;                    3024
000012  004737  000000G                           JSR     PC,GET.PKT
000016  010037  000000G                           MOV     R0,P.INDEX
000022  010016                                    MOV     R0,(SP)                         ; P.INDEX,*          3025
000024  012746  000106                            MOV     #106,-(SP)
000030  004737  000000G                           JSR     PC,BL$MUL
000034  012760  000044   000006G                  MOV     #44,MSCP.PKT+6(R0)
000042  013760  000000G  000016G                  MOV     CDISK,MSCP.PKT+16(R0)           ;                    3026
000050  112760  000011   000022G                  MOVB    #11,MSCP.PKT+22(R0)             ;                    3027
000056  112760  000001   000004G                  MOVB    #1,MSCP.PKT+4(R0)               ;                    3029
000064  013716  000000G                           MOV     P.INDEX,(SP)                    ;                    3031
000070  004737  000000G                           JSR     PC,SEND
000074  005700                                    TST     R0
000076  001054                                    BNE     2$
000100  013700  000000G                           MOV     T.ADDR,R0                       ;                    3034
000104  105260  000051                            INCB    51(R0)
000110  032737  000001   001254                   BIT     #1,APT.MODE                     ;                    3036
000116  001415                                    BEQ     1$
000120  012777  000001   001256                   MOV     #1,@MAIL BOX TESTNUM            ;                    3039
```

```
000126  013700  000000G                  MOV     CUOFF,R0                 ;                3040
000132  006300                           ASL     R0
000134  063700  000000G                  ADD     CST.ADDR,R0
000140  111077  001260'                  MOVB    (R0),@MAIL.BOX.SUBTST
000144  042777  177760  001260'          BIC     #177760,@MAIL.BOX.SUBTST
000152  013700  000000G          1$:     MOV     CUOFF,R0                 ;                3043
000156  006300                           ASL     R0
000160  063700  000000G                  ADD     CST.ADDR,R0
000164  052710  010000                   BIS     #10000,(R0)
000170  104455                           TRAP    55                       ;                3044
000172  000026                           .WORD   26
000174  000000G                          .WORD   EGD.22
000176  000000                           .WORD   0
000200  013700  000000G                  MOV     L$LUN,R0                 ;                3045
000204  112760  000007  000000G          MOVB    #7,DUR(R0)
000212  104451                           TRAP    51
000214  013716  000000G                  MOV     P.INDEX,(SP)             ;                3046
000220  004737  000000G                  JSR     PC,PUT.PKT               ;                3047
000224  000137  C07164'                  JMP     28$
000230  004737  000000G          2$:     JSR     PC,WAIT                  ;                3031
000234  004737  000000G                  JSR     PC,OUT.IODQ              ;                3054
000240  010037  000000G                  MOV     R0,RP.INDX               ;                3055
000244  010016                           MOV     R0,(SP)                  ; RP.INDX,*
000246  012746  000054                   MOV     #54, (SP)                                 3056
000252  004737  000000G                  JSR     PC,BL$MUL
000256  062700  000000G                  ADD     #RETPKT,R0
000262  010037  000000G                  MOV     R0,RP.ADDR
000266  132760  000360  000002           BITB    #360,2(R0)               ;                3058
000274  001404                           BEQ     3$
000276  013716  000000G                  MOV     RP.INDX,(SP)             ;                3060
000302  004737  000000G                  JSR     PC,PUT.RETPKT
000306  005726                   3$:     TST     (SP)+                    ;                3053
000310  013702  000000G                  MOV     RP.ADDR,R2               ;                3063
000314  005000                           CLR     R0
000316  126227  000003  000003           CMPB    3(R2),#3
000324  001002                           BNE     4$
000326  005200                           INC     R0
000330  000407                           BR      5$
000332  132762  000360  000002   4$:     BITB    #360,2(R2)               ;                3064
000340  001333                           BNE     2$
000342  105762  000014                   TSTB    14(R2)                   ;                3065
000346  100330                           BPL     2$
000350  006000                   5$:     ROR     R0                       ;                3067
000352  103012                           BCC     6$
000354  012716  000000G                  MOV     #DBM26,(SP)              ;                3070
000360  012746  000001                   MOV     #1,-(SP)
000364  010600                           MOV     SP,R0                    ; SP,*
000366  104417                           TRAP    17
000370  004737  000000V                  JSR     PC,DR.ERR                ;                3071
000374  005726                           TST     (SP)+                    ;                3069
0003'6  000456                           BR      8$                       ;                306'
0004(0  013766  000000G 000004   6$:     MOV     CUOFF,4(SP)              ;                308'
000406  006366  000004                   ASL     4(SP)
```

```
000412  063766  000000G 000004          ADD     CST.ADDR,4(SP)
000420  126227  000014  000211          CMPB    14(R2),#211            ;        3075
000426  001444                          BEQ     9$
000430  013700  000000G                 MOV     T.ADDR,R0              ;        3078
000434  105260  000050                  INCB    50(R0)
000440  032737  000001  001254'         BIT     #1,APT.MODE            ;        3080
000446  001415                          BEQ     7$
000450  012777  000001  001256          MOV     #1,@MAIL.BOX.TESTNUM   ;        3083
000456  013700  000000G                 MOV     CUOFF,R0               ;        3084
000462  006300                          ASL     R0
000464  063700  000000G                 ADD     CST.ADDR,R0
000470  111077  001260'                 MOVB    (R0),@MAIL.BOX.SUBTST
000474  042777  177760  001260'         BIC     #177760,@MAIL.BOX.SUBTST
000502  052776  010000  000004   7$:    BIS     #10000,@4(SP)          ;        3087
000510  104455                          TRAP    55                     ;        3088
000512  000027                          .WORD   27
000514  000000G                         .WORD   EGD.23
000516  000000G                         .WORD   EMS.21
000520  013700  000000G                 MOV     L$LUN,R0               ;        3089
000524  112760  000007  000000G         MOVB    #7,DUR(R0)
000532  104451                          TRAP    51                     ;        3090
000534  000137  007154'          8$:    JMP     27$                    ;        3075
000540  116237  000016  000000G  9$:    MOVB    16(R2),ST.CODE         ;        3094
000546  042737  177740  000000G         BIC     #177740,ST.CODE
000554  016200  000016                  MOV     16(R2),R0              ;        3095
000560  006200                          ASR     R0
000562  006200                          ASR     R0
000564  006200                          ASR     R0
000566  006200                          ASR     R0
000570  006200                          ASR     R0
000572  042700  174000                  BIC     #174000,R0
000576  010037  000000G                 MOV     R0,SB.CODE
000602  013701  000000G                 MOV     CUOFF,R1               ;        3097
000606  006301                          ASL     R1
000610  063701  000000G                 ADD     CST.ADDR,R1
000614  012703  000012                  MOV     #12,R3
000620  060103                          ADD     R1,R3
000622  116200  000036                  MOVB    36(R2),R0
000626  006200                          ASR     R0
000630  042700  177740                  BIC     #177740,R0
000634  062700  000100                  ADD     #100,R0
000640  110013                          MOVB    R0,(R3)
000642  116200  000036                  MOVB    36(R2),R0              ;        3098
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                     ;        3099
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
000722  062701  000100                  ADD     #100,R1                 ;                       3100
000726  110163  000001                  MOVB    R1,1(R3)
000732  013701  000000G                 MOV     CUOFF,R1                ;                       3101
000736  006301                          ASL     R1
000740  063701  000000G                 ADD     CST.ADDR,R1
000744  116216  000034                  MOVB    34(R2),(SP)
000750  042716  177700                  BIC     #177700,(SP)
000754  012746  000012                  MOV     #12,-(SP)
000760  004737  000000G                 JSR     PC,BL$DIV
000764  010004                          MOV     R0,R4
000766  062704  000060                  ADD     #60,R4
000772  110461  000014                  MOVB    R4,14(R1)
000776  116216  000034                  MOVB    34(R2),(SP)             ;                       3102
001002  042716  177700                  BIC     #177700,(SP)
001006  012746  000012                  MOV     #12,-(SP)
001012  004737  000000G                 JSR     PC,BL$MOD
001016  010004                          MOV     R0,R4
001020  062704  000060                  ADD     #60,R4
001024  110461  000015                  MOVB    R4,15(R1)
001030  126327  000001  000104          CMPB    1(R3),#104              ;                       3106
001036  001004                          BNE     10$
001040  152776  000020  000010          BISB    #20,@10(SP)             ;                       3108
001046  000403                          BR      11$                     ;                       3106
001050  142776  000020  000010  10$:    BICB    #20,@10(SP)             ;                       3110
001056  005737  000000G         11$:    TST     ST.CODE                 ;                       3114
001062  001440                          BEQ     13$
001064  013700  000000G                 MOV     T.ADDR,R0               ;                       3117
001070  105260  000050                  INCB    50(R0)
001074  032737  000001  001254'         BIT     #1,APT.MODE             ;                       3119
001102  001411                          BEQ     12$
001104  012777  000001  001256'         MOV     #1,@MAIL.BOX.TESTNUM    ;                       3122
001112  117677  000010  001260'         MOVB    @10(SP),@MAIL.BOX.SUBTST ;                      3123
001120  042777  177760  001260'         BIC     #177760,@MAIL.BOX.SUBTST
001126  052776  010000  000010  12$:    BIS     #10000,@10(SP)          ;                       3126
001134  104455                          TRAP    55                      ;                       3127
001136  000017                          .WORD   17
001140  000000G                         .WORD   EGD.15
001142  000000G                         .WORD   EMS.30
001144  013700  000000G                 MOV     L$LUN,R0                ;                       3128
001150  112760  000007  000000G         MOVB    #7,DUR(R0)
001156  104451                          TRAP    51                      ;                       3129
001160  000137  007152'                 JMP     26$                     ;                       3114
001164  016203  000044          13$:    MOV     44(R2),R3               ; *,MAX0.LBNS           3134
001170  016204  000046                  MOV     46(R2),R4               ; *,MAX1.LBNS           3135
001174  005703                          TST     R3                      ; MAX0.LBNS             3137
001176  001004                          BNE     14$
```

```
001200  012703  177777                MOV   # 1,R3               ; *,MAX0.LBNS        3140
001204  005304                        DEC   R4                   ; MAX1.LBNS          3141
001206  000401                        BR    15$                  ;                    3137
001210  005303                14$:    DEC   R3                   ; MAX0.LBNS          3144
001212  013701  000000G       15$:    MOV   CUOFF,R1             ;                    3146
001216  006301                        ASL   R1
001220  063701  000000G               ADD   CST.ADDR,R1
001224  012705  000004                MOV   #4,R5
001230  060105                        ADD   R1,R5
001232  021504                        CMP   (R5),R4              ; *,MAX1.LBNS
001234  101013                        BHI   16$
001236  001022                        BNE   17$                  ;                    3148
001240  013701  000000G               MOV   CUOFF,R1             ;                    3149
001244  006301                        ASL   R1
001246  063701  000000G               ADD   CST.ADDR,R1
001252  010300                        MOV   R3,R0                ; MAX0.LBNS,*
001254  005300                        DEC   R0
001256  026100  000002                CMP   2(R1),R0
001262  101410                        BLOS  17$
001264  005015                16$:    CLR   (R5)                 ;                    3153
001266  013701  000000G               MOV   CUOFF,R1             ;                    3154
001272  006301                        ASL   R1
001274  063701  000000G               ADD   CST.ADDR,R1
001300  005061  000002                CLR   2(R1)
001304  013701  000000G       17$:    MOV   CUOFF,R1             ;                    3158
001310  006301                        ASL   R1
001312  063701  000000G               ADD   CST.ADDR,R1
001316  012700  000010                MOV   #10,R0
001322  060100                        ADD   R1,R0
001324  021004                        CMP   (R0),R4              ; *,MAX1.LBNS
001326  101011                        BHI   18$
001330  001020                        BNE   19$                  ;                    3160
001332  013701  000000G               MOV   CUOFF,R1             ;                    3161
001336  006301                        ASL   R1
001340  063701  000000G               ADD   CST.ADDR,R1
001344  026103  000006                CMP   6(R1),R3             ; *,MAX0.LBNS
001350  101410                        BLOS  19$
001352  010410                18$:    MOV   R4,(R0)              ; MAX1.LBNS,*        3164
001354  013701  000000G               MOV   CUOFF,R1             ;                    3165
001360  006301                        ASL   R1
001362  063701  000000G               ADD   CST.ADDR,R1
001366  010361  000006                MOV   R3,6(R1)             ; MAX0.LBNS,*
001372  021510                19$:    CMP   (R5),(R0)            ;                    3168
001374  101017                        BHI   20$
001376  001026                        BNE   21$                  ;                    3171
001400  013700  000000G               MOV   CUOFF,R0             ;                    3173
001404  006300                        ASL   R0
001406  063700  000000G               ADD   CST.ADDR,R0
001412  013701  000000G               MOV   CUOFF,R1             ;                    3174
001416  006301                        ASL   R1
001420  063701  000000G               ADD   CST.ADDR,R1
001424  026061  000002  000006        CMP   2(R0),6(R1)          ;                    3173
001432  101410                        BLOS  21$
```

```
001434  005015                          20$:    CLR     (R5)                                    ;       3178
001436  013701  000000G                         MOV     CUOFF,R1                                ;       3179
001442  006301                                  ASL     R1
001444  063701  000000G                         ADD     CST.ADDR,R1
001450  005061  000002                          CLR     2(R1)
001454  123727  000000G 000002          21$:    CMPB    ENTRY.REASON,#2                         ;       3183
001462  001404                                  BEQ     22$
001464  123727  000000G 000001                  CMPB    ENTRY.REASON,#1                         ;       3184
001472  001031                                  BNE     23$
001474  023727  000000G 000010          22$:    CMP     CRN.LOW,#10                             ;       3186
001502  003025                                  BGT     23$
001504  005737  000000G                         TST     CRN.HIGH                                ;       3187
001510  001022                                  BNE     23$
001512  013700  000000G                         MOV     L$LUN,R0                                ;       3191
001516  010004                                  MOV     R0,R4
001520  006304                                  ASL     R4
001522  006304                                  ASL     R4
001524  013701  000000G                         MOV     CUOFF,R1
001530  006301                                  ASL     R1
001532  063701  000000G                         ADD     CST.ADDR,R1
001536  016164  000002  000000G                 MOV     2(R1),BST(R4)
001544  011564  000002G                         MOV     (R5),BST+2(R4)
001550  112760  000001  000000G                 MOVB    #1,TRK.SGN(R0)                          ;       3192
001556  032762  020000  000022          23$:    BIT     #20000,22(R2)                           ;       3193
001564  001442                                  BEQ     25$                                     ;       3213
001566  005776  000010                          TST     @10(SP)
001572  100037                                  BPL     25$                                     ;       3214
001574  013700  000000G                         MOV     T.ADDR,R0
001600  105260  000050                          INCB    50(R0)                                  ;       3217
001604  032737  000001  001254'                 BIT     #1,APT.MODE
001612  001411                                  BEQ     24$                                     ;       3219
001614  012777  000001  001256'                 MOV     #1,@MAIL.BOX.TESTNUM
001622  117677  000010  001260'                 MOVB    @10(SP),@MAIL.BOX.SUBTST                ;       3222
001630  042777  177760  001260'                 BIC     #177760,@MAIL.BOX.SUBTST                ;       3223
001636  052776  010000  000010          24$:    BIS     #10000,@10(SP)
001644  104455                                  TRAP    55                                      ;       3226
001646  000020                                  .WORD   20                                      ;       3227
001650  000000G                                 .WORD   EGD.16
001652  000000G                                 .WORD   EMS.30
001654  013700  000000G                         MOV     L$LUN,R0
001660  112760  000011  000000G                 MOVB    #11,DUR(R0)                             ;       3228
001666  104451                                  TRAP    51
001670  000414                                  BR      26$                                     ;       3229
001672  052776  020000  000010          25$:    BIS     #20000,@10(SP)                          ;       3213
001700  013716  000000G                         MOV     CCTLR,(SP)                              ;       3233
001704  012746  000126                          MOV     #126,-(SP)                              ;       3234
001710  004737  000000G                         JSR     PC,BL$MUL
001714  105260  000005G                         INCB    CST+5(R0)
001720  005726                                  TST     (SP)+
001722  022626                          26$:    CMP     (SP)+,(SP)+                             ;       3232
001724  013716  000000G                 27$:    MOV     RP.INDX,(SP)                            ;       3093
001730  004737  000000G                         JSR     PC,PUT.RETPKT                           ;       3239
001734  062706  000006                  28$:    ADD     #6,SP                                   ;       3003
```

```
001740  000207                                   RTS     PC

; Routine Size:  497 words,     Routine Base:  $CODE$ + 5230
; Maximum stack depth per invocation:  13 words
```

```
;   3243  1     GLOBAL routine DR ERR : novalue =
;   3244  1
;   3245  1     !+
;   3246  1     !       THIS ROUTINE IS DESIGNED TO PROCESS RETURN PACKETS THAT ORIGINATE AT
;   3247  1     !       THE "DRIVER" RATHER THAN THE DEVICE. DRIVER-ORIGINATED PACKETS INDICATE
;   3248  1     !       EITHER A FATAL DEVICE ERROR OR A COMMAND TIMEOUT. SINCE THIS ROUTINE IS
;   3249  1     !       ONLY CALLED DURING THE INITIALIZATION TEST, IT TREATS A COMMAND TIMEOUT
;   3250  1     !       AS AN INITIALIZATION ERROR.
;   3251  1     !
;   3252  1     !       IMPLICIT INPUTS:
;   3253  1     !               RP_ADDR - ADDRESS OF A RETPKT THAT ORIGINATED AT THE "DRIVER"
;   3254  1     !                       (I.E., CONNECTION ID = CID_DRIVER)
;   3255  1     !-
;   3256  1
;   3257  2         begin
;   3258  2
;   3259  2         local
;   3260  2             REASON : word initial (DU_TIME);         ! ASSUME COMMAND TIMEOUT
;   3261  2
;   3262  2         if .RP_ADDR [MESTYP] eql MT_FATAL            ! IF FATAL DEVICE ERROR
;   3263  2         then
;   3264  2
;   3265  2         DROP_CTLR (.CCTLR, .REASON);                 ! DROP ALL UNITS ON CONTROLLER
;   3266  1         end;


                                        .SBTTL  DR.ERR INITIALIZATION TEST ROUTINES
000000  010146                  DR.ERR::MOV     R1,-(SP)                                        3243
000002  012701  000012                  MOV     #12,R1                  ;                       3257
000006  013700  000000G                 MOV     RP.ADDR,R0              ; *,REASON              3262
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)                                     3265
000034  010146                          MOV     R1,-(SP)                ;
000036  004737  000000G                 JSR     PC,DROP.CTLR            ; REASON,*
000042  022626                          CMP     (SP)+,(SP)+
000044  012601                  1$:     MOV     (SP)+,R1                ;                       3243
000046  000207                          RTS     PC

; Routine Size:  20 words,      Routine Base:  $CODE$ + 7172
; Maximum stack depth per invocation:  4 words
```

```
routine ACCESS : novalue =

!+
!       THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT DISK
!       CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
!       SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE DISK, AND CHECKING
!       THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
!
!       IMPLICIT INPUTS:
!               CCTLR - CURRENT CONTROLLER NUMBER
!               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
!               L$LUN - CURRENT (DRS) UNIT NUMBER
!-

    begin

local
        RESULT : word initial (FAILURE),        ! GUILTY UNTIL PROVEN INNOCENT
        LBN : word,
        PASS : word initial (1);                ! LOOP PASS COUNT

    ST_CODE = SB_CODE = 0;                      ! STATUS CODE AND SUB-CODE
!ZZZ    LBN = (((.MAX_LBN [.L$LUN] + 1) ^ 1) and %o'77777') - 1;
        LBN = 0;                                !TRY LBN 0 FIRST                !ZZZ

    do
        begin                                   ! LOOP STARTS HERE
        P_INDEX = GET_PKT (.CCTLR);             ! GET AN MSCP PACKET
        MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;   ! SET DISK ADDR (RD/RX DISK NUMBER)
        MSCP_PKT [.P_INDEX, OPCODE] = OP_ACC;   ! ACCESS OPCODE
        MSCP_PKT [.P_INDEX, BC_LO] = 512;       ! BYTE COUNT (1 BLOCK)
        MSCP_PKT [.P_INDEX, LBN_L] = .LBN;      ! LOGICAL BLOCK NUMBER
        MSCP_PKT [.P_INDEX, CMD_TYPE] = NON_SEQ_CMD;    ! NON-SEQUENTIAL COMMAND

        if SEND (.P_INDEX) eql FAILURE          ! ATTEMPT TO SEND; IF CTLR NOT ONLINE
        then
            begin
            PUT_PKT (.P INDEX);                 ! RETURN PACKET TO POOL
            PASS = 2;                           ! NO MORE TRIES
            end
        else
            begin                               ! IF SEND WAS SUCCESSFUL

            do
                begin
                WAIT ();                        ! WAIT FOR RESPONSE
                RP_INDX = OUT_IODQ ();          ! GET RETPKT (RESPONSE) INDEX
                RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS

                if .RP_ADDR [MESTYP] neq MT_SEQ ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
                then
                    PUT_RETPKT (.RP_INDX);
```

```
;  5                  end
;  4              until (.RP_ADDR [CONID] eql CID_DRIVER) or
;  5                    ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;  4                     ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;  4
;  4              if .RP_ADDR [CONID] eql CID_DRIVER   ! IF RETPKT CAME FROM "DRIVER"
;  4              then
;  4                  PASS = 2                         ! NO MORE TRIES
;  4              else
;  4
;  5                  if .RP_ADDR [ENDCOD] neq (OP_ACC or OP_END)
;  4                  then
;  5                      begin
;  5                      PRINTF (DBM29);              ! "RETPKT HAS BAD ENDCODE"
;  5  !ZZZ                EMSCMD ();
;  5                      end
;  4                  else
;  5                      begin                        ! RETPKT HAS CORRECT ENDCODE
;  5                      ST_CODE = .RP_ADDR [STSCOD];! GET STATUS CODE FROM PACKET
;  5                      SB_CODE = .RP_ADDR [SUBCOD];! GET SUB-CODE FROM PACKET
;  5
;  5                      if .ST_CODE eql ST_SUC       ! IF STATUS CODE INDICATES SUCCESS
;  5                      then
;  6                          begin
;  6                          RESULT = SUCCESS;
;  6                          PASS = 2;                ! NO NEED TO TRY AGAIN
;  5                          end;
;  5
;  4                      end;                         ! IF RETPKT HAS CORRECT ENDCODE
;  4
;  4              PUT_RETPKT (.RP_INDX);
;  3              end;                                 ! IF SEND WAS SUCCESSFUL
;  3
;  3          LBN = .LBN + 100;                        !TRY ANOTHER ONE                    !ZZZ
;  3          PASS = .PASS + 1;                        ! SECOND PASS
;  3          end                                      ! END OF PASS LOOP
;  2      until .PASS gequ 3;
;  2
;  2      if .RESULT eql FAILURE
;  2      then
;  3          begin
;  3          T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  3          CST_ADDR [.CUOFF, D_FATAL] = TRUE,       ! FATAL ERROR
;  3          ERRDF (17, EGD_17, EMS_30);                        ! ACCESS FAILED
;  3          DUR [.L$LUN] = DU_ACCESS;                ! SET REASON TO DROP UNIT
;  3          DODU (.L$LUN);                           ! DROP UNIT
;  2          end;                                     ! IF ACCESS FAILED
;  2
;  1      end;                                         ! ROUTINE ACCESS

                                     .SBTTL  ACCESS INITIALIZATION TEST ROUTINES
000000  004137  000000G     ACCESS: JSR     R1,$SAVE4                      ;           3267
```

```
000004  005003                                  CLR     R3                          ; RESULT            3281
000006  012702  000001                          MOV     #1,R2                       ; *.PASS
000012  005037  000000G                         CLR     SB.CODE                     ;                   3288
000016  005037  000000G                         CLR     ST.CODE
000022  005004                                  CLR     R4                          ; LBN               3290
000024  013746  000000G                 1$:     MOV     CCTLR,-(SP)                 ;                   3294
000030  004737  000000G                         JSR     PC,GET.PKT
000034  010037  000000G                         MOV     R0,P.INDEX
000040  010016                                  MOV     R0,(SP)                     ; P.INDEX,*         3295
000042  012746  000106                          MOV     #106,-(SP)
000046  004737  000000G                         JSR     PC,BL$MUL
000052  013760  000000G 000016G                 MOV     CDISK,MSCP.PKT+16(R0)
000060  112760  000020  000022G                 MOVB    #20,MSCP.PKT+22(R0)         ;                   3296
000066  012760  001000  000026G                 MOV     #1000,MSCP.PKT+26(R0)       ;                   3297
000074  010460  000046G                         MOV     R4,MSCP.PKT+46(R0)          ; LBN,*             3298
000100  112760  000002  000004G                 MOVB    #2,MSCP.PKT+4(R0)           ;                   3299
000106  013716  000000G                         MOV     P.INDEX,(SP)                ;                   3301
000112  004737  000000G                         JSR     PC,SEND
000116  005700                                  TST     R0
000120  001007                                  BNE     2$
000122  013716  000000G                         MOV     P.INDEX,(SP)                ;                   3304
000126  004737  000000G                         JSR     PC,PUT.PKT
000132  012702  000002                          MOV     #2,R2                       ; *.PASS            3305
000136  000522                                  BR      9$                          ;                   3301
000140  004737  000000G                 2$:     JSR     PC,WAIT                     ;                   3312
000144  004737  000000G                         JSR     PC,OUT.IODQ                 ;                   3313
000150  010037  000000G                         MOV     R0,RP.INDX
000154  010016                                  MOV     R0,(SP)                     ; RP.INDX,*         3314
000156  012746  000054                          MOV     #54,-(SP)
000162  004737  000000G                         JSR     PC,BL$MUL
000166  062700  000000G                         ADD     #RETPKT,R0
000172  010037  000000G                         MOV     R0,RP.ADDR
000176  132760  000360  000002                  BITB    #360,2(R0)                  ;                   3316
000204  001404                                  BEQ     3$
000206  013716  000000G                         MOV     RP.INDX,(SP)                ;                   3318
000212  004737  000000G                         JSR     PC,PUT.RETPKT
000216  005726                          3$:     TST     (SP)+                       ;                   3311
000220  013701  000000G                         MOV     RP.ADDR,R1                  ;                   3321
000224  005000                                  CLR     R0
000226  126127  000003  000003                  CMPB    3(R1),#3
000234  001002                                  BNE     4$
000236  005200                                  INC     R0
000240  000407                                  BR      5$
000242  132761  000360  000002          4$:     BITB    #360,2(R1)                  ;                   3322
000250  001333                                  BNE     2$
000252  105761  000014                          TSTB    14(R1)                      ;                   3323
000256  100330                                  BPL     2$
000260  006000                          5$:     ROR     R0                          ;                   3325
000262  103442                                  BLO     7$                          ;                   3327
000264  126127  000014  000220                  CMPB    14(R1),#220                 ;                   3330
000272  001410                                  BEQ     6$
000274  012716  000000G                         MOV     #DBM29,(SP)                 ;                   3333
000300  012746  000001                          MOV     #1,-(SP)
```

```
000304  010600                              MOV     SP,R0                   ; SP,*
000306  104417                              TRAP    17
000310  005726                              TST     (SP)+
000312  000430                              BR      8$                      ;                       3332
000314  116137  000016  000000G     6$:     MOVB    16(R1),ST.CODE          ;                       3330
000322  042737  177740  000000G             BIC     #177740,ST.CODE         ;                       3338
000330  016100  000016                      MOV     16(R1),R0               ;                       3339
000334  006200                              ASR     R0
000336  006200                              ASR     R0
000340  006200                              ASR     R0
000342  006200                              ASR     R0
000344  006200                              ASR     R0
000346  042700  174000                      BIC     #174000,R0
000352  010037  000000G                     MOV     R0,SB.CODE
000356  005737  000000G                     TST     ST.CODE                 ;                       3341
000362  001004                              BNE     8$
000364  012703  000001                      MOV     #1,R3                   ; *,RESULT              3344
000370  012702  000002              7$:     MOV     #2,R2                   ; *,PASS                3345
000374  013716  000000G             8$:     MOV     RP.INDX,(SP)            ;                       3350
000400  004737  000000G                     JSR     PC,PUT.RETPKT
000404  062704  000144              9$:     ADD     #144,R4                 ; *,LBN                 3353
000410  005202                              INC     R2                      ; PASS                  3354
000412  022626                              CMP     (SP)+,(SP)+             ;                       3293
000414  020227  000003                      CMP     R2,#3                   ; PASS,*                3356
000420  103601                              BLO     1$
000422  005703                              TST     R3                      ; RESULT                3358
000424  001025                              BNE     10$
000426  013700  000000G                     MOV     T.ADDR,R0               ;                       3361
000432  105260  000050                      INCB    50(R0)
000436  013700  000000G                     MOV     CUOFF,R0                ;                       3362
000442  006300                              ASL     R0
000444  063700  000000G                     ADD     CST.ADDR,R0
000450  052710  010000                      BIS     #10000,(R0)
000454  104455                              TRAP    55                      ;                       3363
000456  000021                              .WORD   21
000460  000000G                             .WORD   EGD.17
000462  000000G                             .WORD   EMS.30
000464  013700  000000G                     MOV     L$LUN,R0                ;                       3364
000470  112760  000010  000000G             MOVB    #10,DUR(R0)
000476  104451                              TRAP    51                      ;                       3365
000500  000207                      10$:    RTS     PC                      ;                       3267

; Routine Size:  161 words,     Routine Base:  $CODE$ + 7242
; Maximum stack depth per invocation:  10 words
```

```
;  1      %sbttl 'MULTI-DRIVE TEST ROUTINES'
;  1
;  1
;  1      GLOBAL routine MULTI DRIVE : novalue =
;  1
;  1
;  1      !+
;  1      !     THIS SUBTEST IS THE MOST SIGNIFICANT PART OF THE ENTIRE PROGRAM. THE
;  1      !     MULTI-DRIVE TEST IS A HOST-CONTROLLED EXERCISER DESIGNED TO GIVE THE
;  1      !     USER AN INDICATION OF HOW ONE OR SEVERAL RDRX DRIVES WOULD PERFORM IN
;  1      !     AN OPERATING SYSTEM ENVIRONMENT.
;  1      !
;  1      !     THIS ROUTINE ACTS AS AN "EXECUTIVE" TO THE WHOLE  ‹OCESS. AFTER
;  1      !     INVOKIN  MD_INIT TO INITIALIZE MULTI DRIVE TEST DATA, THIS ROUTINE
;  1      !     ENTERS   LOOP WHICH ISSUES QIOs TO ALL ACTIVE CONTROLLERS AND PROCESSES
;  1      !     ANY RESPONSES. IN ADDITION, ALL OUTSTANDING COMMANDS ARE TIMED IN
;  1      !     DRV_TIMCHK WHICH IS INVOKED EVERY SECOND. NORMAL TERMINATION OF THIS
;  1      !     LOOP OCCURS WHEN QIOs ARE NO LONGER BEING ISSUED, AND ALL OUTSTANDING
;  1      !     QIOS HAVE COMPLETED.
;  1      !-
;  1
;  1
;  2          begin
;  2
;  2          local
;  2              CUR_PRIORITY : word;
;  2
;  2          label
;  2              SEND_COMMANDS;
;  2
;  2          MD_INIT ();                                       ! INIT MULTI-DRIVE TEST DATA
;  2          INIT_OCCURED = TRUE;                              !
;  2
;  2
;  3          do  begin                                         ! START OF EXECUTIVE LOOP
;  3
;  3              incr CTLR from 0 to (MAX_CTLR - 1) do          ! FOR EACH CONTROLLER
;  4                  begin
;  4                  SET_CPAR (.CTLR);                         ! SET UP CURRENT CONTROLLER PARAMETERS
;  4                  GETPRI (CUR_PRIORITY);                    !
;  4      !ZZZ        SETPRI (PRIO4);                           ! NO INTERRUPTS WHEN EXAMINING SA
;  4                  SETPRI (.BRLEVEL);                        ! NO INTERRUPTS WHEN EXAMINING SA        ZZ
Z
;  4                  ICTLR = .CCTLR;                           ! FAKE INTERRUPTING CONTROLLER'S NUMBER
;  4                  ICST_ADDR = .CST_ADDR;                    ! FAKE INTERRUPTING CONTROLLER'S CST ADDR
;  4                  IDCT_ADDR = .DCT_ADDR;                    ! FAKE INTERRUPTING CONTROLLER'S DCT ADDR
;  4                  IRDRX_ADDR = .ICST_ADDR [IP_ADDR];        ! FAKE INTERRUPTING CONTROLLER'S ADDRESS
;  4                  IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];  ! CONTENTS OF THE SA REGISTER
;  4
;  5                  if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)  ! IF SA SHOWS AN ERROR
;  4                  then
;  5                      begin
;  5                      FATAL_ERROR ();                       ! DECLARE FATAL ERROR
;  5                      SETPRI (.CUR_PRIORITY);               ! LOWER PRIORITY
```

```
;   5                       exitloop;                                   ! QUIT
;   5                       end
;   5
;   4                   else
;   4                       SETPRI (.CUR_PRIORITY);                     ! IF NO ERROR, CONTINUE
;   4
;   4                   if QIO_OK ()                                    ! IF O.K. TO ISSUE QIO(S) TO CONTROLLER
;   4                   then
;   4                       SEND_COMMANDS:
;   5                           begin
;   5                           QIO_GEN ();                             ! GENERATE 1 OR 2 QIOs
;   5
;   5                           if (.MX1 geq 0) and                     ! IF SUCCESS ON FIRST QIO
;   6                              (not .EOP_FLAG)
;   5                           then
;   5
;   5                               if SEND (.MX1) eql SUCCESS          ! ATTEMPT TO SEND IT. IF SUCCESS
;   5                               then
;   6                               BEGIN                               !ZZZ
;   6                                   QIO [.CTLR] = .QIO [.CTLR] + 1; ! INCR OUTSTANDING QIO COUNT ZZZ
;   6                               END                                 !ZZZ
;   6
;   5                               else
;   6                                   begin
;   6                                   PUT_PKT (.MX1);                 ! RETURN PACKET TO POOL
;   6                                   leave SEND_COMMANDS;
;   5                                   end;
;   5
;   5
;   5                           if (.MX2 geq 0) and                     ! IF SUCCESS ON SECOND QIO
;   6                              (not .EOP_FLAG)
;   5                           then
;   6                               begin
;   6
;   6                               do
;   6                                   BREAK                           ! WAIT TILL 1 MORE SLOT AVAILABLE IN CRING
;   6                               until (.DCT_ADDR [CRING_CNT] lssu CRING_LEN); !
;   6
;   6                               if SEND (.MX2) eql SUCCESS          ! ATTEMPT TO SEND IT.
;   6                               then
;   7                               BEGIN                               !ZZZ
;   7                                   QIO [.CTLR] = .QIO [.CTLR] + 1; ! IF SUCCESS, INCR OUTSTANDING QIO COUNT
;   7                               END                                 !ZZZ
;   6
;   7                               else
;   7                                   begin
;   7                                   PRINTF (DBM121, .CRN_HIGH, .CRN_LOW
;   7                                   COMPARE_DATA = FALSE;           ! NO SENSE IN COMPARING WRITE DATA
;   6                                   PUT_PKT (.MX2);                 ! RETURN PACKET TO POOL
;   6                                   end;
;   5
;   5                               end;
```

```
;     3475  4                          end;                               ! O.K. TO ISSUE QIO(S)
;     3476  3                  end;                                       ! CONTROLLER LOOP
;     3477  3
;     3478  3
;     3479  3            BREAK;                                           ! LET SUPERVISOR CATCH USER REQUESTS
;     3480  3            PROC_RETPKT ();                                  ! PROCESS ANY RETURN PACKETS
;     3481  3
;     3482  3            end                                              ! EXECUTIVE PROCESSING LOOP
;     3483  3        until ((not QIO_OUT ()) or
;     3484  4            ((.DCT_ADDR [CRING_CNT] eql 0) and
;     3485  2             (.EOP_FLAG)));
;     3486  2
;     3487  2
;     3488  2        DCT_ADDR [IG_INT] = TRUE;                            ! NO FURTHER INTERRUPTS ON THIS CONTROLLER
;     3489  2
;     3490  2
;     3491  1        end;                                                 ! EXERCISER


                                                .SBTTL  MULTI.DRIVE MULTI-DRIVE TEST ROUTINES
                                        MULTI.DRIVE::
000000  004137  000000G                         JSR     R1,$SAVE3                       ;               3372
000004  005746                                  TST     -(SP)
000006  004737  000000V                         JSR     PC,MD.INIT                      ;               3399
000012  112737  000001  000000G                 MOVB    #1,INIT.OCCURED                 ;               3400
000020  005001                          1$:     CLR     R1                              ; CTLR          3405
000022  010146                          2$:     MOV     R1,-(SP)                        ; CTLR,*        3407
000024  004737  000000G                         JSR     PC,SET.CPAR
000030  104440                                  TRAP    40
000032  010003                                  MOV     R0,R3                           ;               3408
000034  013700  000000G                         MOV     BRLEVEL,R0                      ; *,CUR.PRIORITY
000040  104441                                  TRAP    41                              ;               3410
000042  013737  000000G 000104'                 MOV     CCTLR,ICTLR
000050  013737  000000G 000076'                 MOV     CST.ADDR,ICST.ADDR              ;               3411
000056  013737  000000G 000100'                 MOV     DCT.ADDR,IDCT.ADDR              ;               3412
000064  013737  000076' 000000G                 MOV     @ICST.ADDR,IRDRX.ADDR           ;               3413
000072  013700  000100                          MOV     IDCT.ADDR,R0                    ;               3414
000076  013702  000000G                         MOV     IRDRX.ADDR,R2                   ;               3415
000102  016266  000002  000002                  MOV     2(R2),2(SP)                     ; *,RC.REG
000110  016660  000002  000002                  MOV     2(SP),2(R0)                     ; RC.REG,*
000116  016600  000002                          MOV     2(SP),R0                        ;               3417
000122  042700  077777                          BIC     #77777,R0
000126  020027  100000                          CMP     R0,# 100000
000132  001006                                  BNE     3$
000134  004737  000000V                         JSR     PC,FATAL.ERROR
000140  010300                                  MOV     R3,R0                           ;               3420
000142  104441                                  TRAP    41                              ; CUR.PRIORITY,* 3421
000144  005726                                  TST     (SP)+
000146  000511                                  BR      9$                              ;               3419
000150  010300                          3$:     MOV     R3,R0                           ; CUR.PRIORITY,* 3426
000152  104441                                  TRAP    41
000154  004737  000000V                         JSR     PC,QIO.OK                       ;               3428
000160  006000                                  ROR     R0
```

```
000162  103077                              BCC     8$
000164  004737  000000V                     JSR     PC,QIO.GEN
000170  013700  000110'                     MOV     MX1,R0                  ;                       3432
000174  002422                              BLT     5$                      ;                       3434
000176  132737  000001  000000G             BITB    #1,EOP.FLAG
000204  001016                              BNE     5$                      ;                       3435
000206  010016                              MOV     R0,(SP)
000210  004737  000000G                     JSR     PC,SEND                 ;                       3438
000214  020027  000001                      CMP     R0,#1
000220  001003                              BNE     4$
000222  105261  000000G                     INCB    QIO(R1)
000226  000405                              BR      5$                      ; *(CTLR)               3441
000230  013716  000110'         4$:         MOV     MX1,(SP)                ;                       3438
000234  004737  000000G                     JSR     PC,PUT.PKT              ;                       3446
000240  000450                              BR      8$
000242  005737  000112'         5$:         TST     MX2                     ;                       3445
000246  002445                              BLT     8$                      ;                       3451
000250  132737  000001  000000G             BITB    #1,EOP.FLAG
000256  001041                              BNE     8$                      ;                       3452
000260  104422                  6$:         TRAP    22
000262  127727  000000G 000004              CMPB    @DCT.ADDR,#4            ;                       3456
000270  103373                              BHIS    6$                      ;                       3458
000272  013716  000112'                     MOV     MX2,(SP)
000276  004737  000000G                     JSR     PC,SEND                 ;                       3460
000302  020027  000001                      CMP     R0,#1
000306  001003                              BNE     7$
000310  105261  000000G                     INCB    QIO(R1)
000314  000422                              BR      8$                      ; *(CTLR)               3463
000316  013716  000000G         7$:         MOV     CRN.LOW,(SP)            ;                       3460
000322  013746  000000G                     MOV     CRN.HIGH,-(SP)          ;                       3468
000326  012746  000000G                     MOV     #DBM121,-(SP)
000332  012746  000003                      MOV     #3,-(SP)
000336  010600                              MOV     SP,R0
000340  104417                              TRAP    17                      ; SP,*
000342  105037  001262'                     CLRB    COMPARE.DATA
000346  013716  000112'                     MOV     MX2,(SP)                ;                       3469
000352  004737  000000G                     JSR     PC,PUT.PKT              ;                       3470
000356  062706  000006                      ADD     #6,SP
000362  005726                  8$:         TST     (SP)+                   ;                       3467
000364  005201                              INC     R1                      ;                       3406
000366  000243                              .WORD   CLV!CLC                 ; CTLR                  3405

000370  003614                              BLE     2$
000372  104422                  9$:         TRAP    22
000374  004737  000000V                     JSR     PC,PROC.RETPKT          ;                       3476
000400  004737  000000V                     JSR     PC,QIO.OUT              ;                       3480
000404  006000                              ROR     R0                      ;                       3483
000406  103011                              BCC     10$
000410  105777  000000G                     TSTB    @DCT.ADDR
000414  001201                              BNE     1$                      ;                       3484
000416  132737  000001  000000G             BITB    #1,EOP.FLAG
000424  001002                              BNE     10$                     ;                       3485
000426  000137  007764'                     JMP     1$
```

```
000432  052777  040000  000000G         10$:    BIS     #40000,@OCT.ADDR                        ;                         3488
000440  005726                                  TST     (SP)+                                   ;                         3372
000442  000207                                  RTS     PC

; Routine Size:  146 words,     Routine Base:  $CODE$ + 7744
; Maximum stack depth per invocation:  11 words


;    3492  1
```

```
;     3493  1      GLOBAL routine MD INIT : novalue =
;     3494  1
;     3495  1      !+
;     3496  1      !       THIS ROUTINE IS CALLED BY ROUTINE MULTI DRIVE TO INITIALIZE DATA ITEMS
;     3497  1      !       USED BY THE MULTI DRIVE TEST.
;     3498  1      !
;     3499  1      !-
;     3500  2          begin
;     3501  2
;     3502  2      !!ZZZ     local
;     3503  2      !!ZZZ   AVG_XFER_SIZE : word,                              ! SIZE (BYTES) OF AN AVERAGE I/O XFER
;     3504  2      !!ZZZ   QUICK_PASS_CNT : word;                             ! AVG NO. OF I/O OPERATIONS IN A QUICK PASS
;     3505  2
;     3506  2          if not .INIT_OCCURED                                   ! IF THIS IS A START
;     3507  2          then
;     3508  2              INIT_IO_BUFF ();                                   ! PARTITION FREE MEMORY INTO I/O BUFFERS
;     3509  2
;     3510  2          if (.ENTRY_REASON neq CONT) and                        ! IF START, RESTART, OR PWR FAIL
;     3511  3             (.ENTRY_REASON neq NEW_PASS)
;     3512  2          then
;     3513  2
;     3514  2              incr CTLR from 0 to (MAX_CTLR   1) do
;     3515  3                  begin
;     3516  3                  SET_CPAR (.CTLR);
;     3517  3
;     3518  4                  INCR DISK FROM (0 + OF_UN) TO (3 * UNIT_SIZE              !ZZZ
;     3519  3                      + OF_UN) BY UNIT_SIZE DO                          !ZZZ
;     3520  4                      BEGIN                                             !ZZZ
;     3521  4                      SET_UPAR (.DISK);                                 !ZZZ
;     3522  4                      DPST [.L$LUN] = DP_CNT;       !INIT DATA PTRN SEQ TABLEZZZ
;     3523  3                      END;                                              !ZZZ
;     3524  3
;     3525  2                  END;                                                  !ZZZ
;     3526  2              INCR COUNT FROM 0 TO (QIO_PER_CTLR * MAX_CTLR - 1) DO   !INIT   !ZZZ
;     3527  2                  BUFF_OWN [.COUNT] =  1;              !I/O BUFF ALLOC TABLE   !ZZZ
:     3528  1              END;                                 !END MD INIT            !ZZZ

                                                   .SBTTL  MD.INIT MULTI DRIVE TEST ROUTINES
 000000  004137  000000G                     MD.INIT::
                                                   JSR     R1,$SAVE2
 000004  132737  000001  000000G                   BITB    #1,INIT.OCCURED          ;                    3493
 000012  001002                                    BNE     1$                       ;                    3506
 000014  004737  000000V                           JSR     PC,INIT.IO.BUFF
 000020  123727  000000G 000003           1$:      CMPB    ENTRY.REASON,#3          ;                    3508
 000026  001433                                    BEQ     4$                       ;                    3510
 000030  123727  000000G 000005                    CMPB    ENTRY.REASON,#5
 000036  001427                                    BEQ     4$                       ;                    3511
 000040  005002                                    CLR     R2                       ; CTLR
 000042  010246                           2$:      MOV     R2,-(SP)                 ; CTLR,*             3514
 000044  004737  000000G                           JSR     PC,SET.CPAR                                   3516
 000050  012701  000003                            MOV     #3,R1                    ; *,DISK
 000054  010116                           3$:      MOV     R1,(SP)                  ; DISK,*             3518
                                                                                                         3521
```

```
000056  004737  000000G                  JSR     PC,SET.UPAR
000062  013700  000000G                  MOV     L$LUN,R0
000066  112760  000025  000050'          MOVB    #25,DPST(R0)             ;                     3522
000074  062701  000012                   ADD     #12,R1
000100  020127  000041                   CMP     R1,#41                   ; *,DISK              3518
000104  003763                           BLE     3$                       ; DISK,*
000106  005726                           TST     (SP)+
000110  005202                           INC     R2                       ;                     3515
000112  000243                           .WORD CLV!CLC                    ; CTLR                3514

000114  003752                           BLE     2$
000116  005000                   4$:     CLR     R0                       ; COUNT               3526
000120  112760  000377  000000G  5$:     MOVB    #377,BUFF.OWN(R0)        ; *,*(COUNT)          3527
000126  005200                           INC     R0                       ; COUNT               3526
000130  020027  000007                   CMP     R0,#7                    ; COUNT,*
000134  003771                           BLE     5$
000136  000207                           RTS     PC                       ;                     3493

; Routine Size:  48 words,      Routine Base:  $CODE$ + 10410
; Maximum stack depth per invocation:  5 words

;    3529  1
```

```
;  1   GLOBAL routine INIT_IO_BUFF : novalue =
;  1
;  1   !+
;  1   !        THIS ROUTINE IS CALLED BY MD_INIT WHEN THE MULTI DRIVE TEST IS FIRST
;  1   !        STARTED. IT IS RESPONSIBLE FOR PARTITIONING FREE MEMORY INTO A
;  1   !        COLLECTION OF I/O BUFFERS. THE SIZE OF EACH I/O BUFFER IS DETERMINED
;  1   !        BY A NUMBER OF FACTORS, INCLUDING THE NUMBER OF UNITS, THE NUMBER OF
;  1   !        CONTROLLERS, AND THE SIZE OF FREE MEMORY.
;  1   !
;  1   !        ONCE THE BUFFER SIZE IS DETERMINED, THE NUMBER OF I/O BUFFERS IS
;  1   !        CALCULATED. FINALLY, THE BUFFER ADDRESS (BUFF_ADDR) TABLE IS LOADED
;  1   !        WITH FIXED BUFFER DESCRIPTORS THAT ARE USED IN THE ALLOCATION AND
;  1   !        DEALLOCATION PROCESS.
;  1   !
;  1   !        IMPLICIT INPUTS:
;  1   !                CTLR_CNT - THE NUMBER OF CONTROLLERS CONFIGURED
;  1   !                L$UNIT - THE NUMBER OF UNITS AVAILABLE FOR TESTING
;  1   !                FREE MEM_ADDR - START OF FREE MEMORY
;  1   !-
;  1
;  2       begin
;  2       BUFF_ADDR [0] = (.FREE_MEM_ADDR + 2 + 1) and %o'177776';                     ! START OF READ/WRITE BUFFERS
;  2
;  2       while (.BUFF_ADDR [0] and %o'37') neq 0 do                                   ! FORCE FIRST I/O BUFFER TO START
;  2           BUFF_ADDR [0] = .BUFF_ADDR [0] + 2;                                      ! ON EVEN BOUNDARY
;  2
;  2       BYTS_PER_QIO = ((.DRS_START - .BUFF_ADDR [0]) / (QIO_PER_CTLR * MAX_CTLR)) and %o'177740';
;  2                                                                                    ! MAX TRANSFER SIZE
;  2
;  2       if .BYTS_PER_QIO gtru MAX_XFER_SIZE
;  2       then
;  2           BYTS_PER_QIO = MAX_XFER_SIZE;                                            ! ADJUST TRANSFER SIZE LOWER
;  2
;  2       if .BYTS_PER_QIO lssu 32
;  2       then
;  3           begin
;  3           ERRSF (2, EGS_02, 0);                                                    ! ERROR IF NOT ENOUGH MEMORY
;  3           DOCLN;
;  2           end;
;  2
;  2       if (QIO_PER_CTLR * MAX_CTLR) gtru 1
;  2       then
;  2
;  2           incr index from 1 to (QIO_PER_CTLR * MAX_CTLR - 1) do                    ! INIT REMAINING TABLE ENTRIES
;  2               BUFF_ADDR [.index] = .BUFF_ADDR [.index - 1] + .BYTS PER_QIO;        ! FIXED BUFFER ADDRESS
;  2
;  1       end;                                                                         ! ROUTINE INIT IO BUFF

                                          .SBTTL  INIT.IO.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G         INIT.IO.BUFF::
                                          JSR     R1,$SAVE3                 ;       3530
000004  013700  000000G                   MOV     FREE.MEM.ADDR,R0          ;       3551
```

```
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$:   BIT   #37,BUFF.ADDR                 ;                 3553
000034  001404                            BEQ   2$
000036  062737  000002  000000G           ADD   #2,BUFF.ADDR                  ;                 3554
000044  000770                            BR    1$                            ;                 3553
000046  013746  001250'             2$:   MOV   DRS.START, (SP)               ;                 3556
000052  163716  000000G                   SUB   BUFF.ADDR,(SP)
000056  012746  000010                    MOV   #10, (SP)
000062  004737  000000G                   JSR   PC,BL$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            ;                 3559
000106  101403                            BLOS  3$
000110  012737  001400  000000G           MOV   #1400,BYTS.PER.QIO            ;                 3561
000116  023727  000000G 000040      3$:   CMP   BYTS.PER.QIO,#40              ;                 3563
000124  103005                            BHIS  4$
000126  104454                            TRAP  54                            ;                 3566
000130  000002                            .WORD 2
000132  000000G                           .WORD EGS.02
000134  000000                            .WORD 0
000136  104444                            TRAP  44
000140  012702  000001              4$:   MOV   #1,R2                         ; *,INDEX         3570
000144  010200                      5$:   MOV   R2,R0                         ; INDEX,*         3574
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           3570
000172  020227  000007                    CMP   R2,#7                         ; INDEX,*
000176  003762                            BLE   5$
000200  022626                            CMP   (SP)+,(SP)+                   ;                 3550
000202  000207                            RTS   PC                            ;                 3530

; Routine Size:  66 words,      Routine Base:  $CODE$ + 10550
; Maximum stack depth per invocation:  8 words
```

```
GLOBAL routine QIO_OK =


!+
!       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE "EXECUTIVE" IN ORDER TO
!       DETERMINE WHETHER OR NOT A QIO REQUEST (OR QIO PAIR) SHOULD BE
!       GENERATED TO THE CURRENT CONTROLLER. A VALUE OF "TRUE" IS RETURNED IF
!       THE CONTROLLER MEETS 3 REQUIREMENTS:
!
!               A. THE CONTROLLER IS ONLINE;
!               B. THE NUMBER OF OUTSTANDING QIOs IS AT LEAST 2 LESS THAN THE
!                       MAXIMUM ALLOWED FOR ANY ONE CONTROLLER;
!               C. THERE IS AT LEAST ONE DISK ONLINE TO THE CONTROLLER.
!
!       IF ANY OF THESE TEST FAIL, THEN A VALUE OF "FALSE' IS RETURNED.
!
!       IMPLICIT INPUTS:
!               CCTLR - CURRENT CONTROLLER NUMBER
!               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
!-


    begin

    local                                                   !MMM

        MILLISECOND : WORD,                                 !MMM
        DUTY_CYCLE : WORD;                                  !MMM

    MILLISECOND = .CLK_TICKS mod 60;                        !MMM

    if (.CST_ADDR [STATE] eql ONLINE) and                   ! IF CONTROLLER IS ONLINE
       (not .EOP_FLAG) and
!MMM   ((.QIO [.CCTLR] + 2) lequ QIO_PER_CTLR) and          ! IF OUTSTANDING QIO COUNT IS O.K.
       (.QIO [.CCTLR] lequ (CRING_LEN-2)) and               ! IF OUTSTANDING QIO COUNT IS O.K.    MMM
       (.CST_ADDR [U_CNT] neq 0) and                        ! IF THERE IS VALID UNIT
       ((not .EL_FLUSH [.CCTLR]) or (.QIO [.CCTLR] eqlu 0))    ! MMM

    then
        begin                                               ! MMM
!         if .CSR_MEM and .TST_PAR                          ! MMM
!        then                                               ! MMM
!           if not .PAR_TSD                                 ! MMM
!           then                                            ! MMM
!               if .QIO [.CCTLR] geq 1                      ! MMM PROCESS ONE COMMAND AT A TIME UNTIL
!               then                                        ! PARITY TEST IS COMPLETE
!                   return FALSE;
        EL_FLUSH [.CCTLR] = 0;                              ! MMM
!         if (((not .TST_PAR) or (.TST_PAR and .PAR_TSD)) and (.SWP XFER NEQ 0))          ! MMM
!       then
        if .MAN_TST                                         ! MMM
        then                                                ! MMM
```

```
;    3630  3              if (DUTY_CYCLE = .MILLISECOND mod 60) gtru 15      !LOOK AT 'CURRENT SECOND' MMM
;    3631  3              then
;    3632  3                  return FALSE;                                   !FIRST 15 OF 60 TICKS (=1 SECOND) IS 25% DUTY CYCLE MMM
;    3633  3              return TRUE;                                         ! "TRUE" EXIT POINT
;    3634  3              end
;    3635  2          else
;    3636  2              return FALSE;                                        ! "FALSE" EXIT POINT
;    3637  2
;    3638  1          end;                                                     !MMM


                                                    .SBTTL  QIO.OK MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                     QIO.OK::JSR     R1,$SAVE2                              3577
000004  013746  000000G                             MOV     CLK.TICKS,-(SP)             ;          3607
000010  012746  000074                              MOV     #74,-(SP)
000014  004737  000000G                             JSR     PC,BL$MOD
000020  013701  000000G                             MOV     CST.ADDR,R1                 ;          3609
000024  005761  000002                              TST     2(R1)
000030  100061                                      BPL     3$
000032  132737  000001  000000G                     BITB    #1,EOP.FLAG                 ;          3610
000040  001055                                      BNE     3$
000042  013701  000000G                             MOV     CCTLR,R1                    ;          3612
000046  126127  000000G 000002                      CMPB    QIO(R1),#2
000054  101047                                      BHI     3$
000056  013701  000000G                             MOV     CST.ADDR,R1                 ;          3613
000062  105761  000005                              TSTB    5(R1)
000066  001442                                      BEQ     3$
000070  013701  000000G                             MOV     CCTLR,R1                    ;          3614
000074  010102                                      MOV     R1,R2
000076  006302                                      ASL     R2
000100  032762  000001  000106'                     BIT     #1,EL.FLUSH(R2)
000106  001403                                      BEQ     1$
000110  105761  000000G                             TSTB    QIO(R1)
000114  001027                                      BNE     3$
000116  013701  000000G                 1$:         MOV     CCTLR,R1                    ;          3625
000122  006301                                      ASL     R1
000124  005061  000106'                             CLR     EL.FLUSH(R1)
000130  032737  000001  000000G                     BIT     #1,MAN.TST                  ;          3628
000136  001413                                      BEQ     2$
000140  010016                                      MOV     R0,(SP)                     ; MILLISECOND,*   3630
000142  012746  000074                              MOV     #74,-(SP)
000146  004737  000000G                             JSR     PC,BL$MOD
000152  005726                                      TST     (SP)+
000154  020027  000017                              CMP     R0,#17                      ; DUTY.CYCLE,*
000160  101402                                      BLOS    2$
000162  022626                                      CMP     (SP)+,(SP)+                 ;          3632
000164  000406                                      BR      5$
000166  012700  000001                  2$:         MOV     #1,R0                       ;          3636
000172  000401                                      BR      4$
000174  005000                          3$:         CLR     R0
000176  022626                          4$:         CMP     (SP)+,(SP)+                 ;          3609
000200  000207                                      RTS     PC                          ;          3599
000202  005000                          5$:         CLR     R0                          ;          3577
```

```
000204  000207                                          RTS     PC

; Routine Size:  67 words.      Routine Base:  $CODE$ + 10754
; Maximum stack depth per invocation:  7 words

;    3639  1
```

```
;    3640  1      GLOBAL routine QIO OUT =
;    3641  1
;    3642  1      !+
;    3643  1      !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR DETERMINING THE
;    3644  1      !       END OF THE MULTI DRIVE TEST. ITS PURPOSE IS TO EXAMINE THE QIO VECTOR
;    3645  1      !       FOR ANY OUTSTANDING QIOs ON ANY CONTROLLER. A VALUE OF "TRUE" IS
;    3646  1      !       RETURNED IF THERE IS AT LEAST ONE QIO OUTSTANDING ON ANY CONTROLLER.
;    3647  1      !       OTHERWISE, "FALSE" IS RETURNED INDICATING NO OUTSTANDING QIOs.
;    3648  1      !-
;    3649  1
;    3650  2          begin
;    3651  2
;    3652  2          incr CTLR from 0 to (MAX_CTLR - 1) do
;    3653  3              begin
;    3654  3              SET_CPAR (.CTLR);                     ! SET UP CURRENT CONTROLLER PARAMETERS
;    3655  3
;    3656  3              if .CST_ADDR [STATE] eql ONLINE       ! IF CONTROLLER IS ONLINE
;    3657  3              then
;    3658  3                  return TRUE;
;    3659  3
;    3660  2              end;
;    3661  2
;    3662  2          return FALSE;                             ! EXIT - NO CONTROLLERS ONLINE
;    3663  1          end;
```

```
                                          .SBTTL  QIO.OUT MULTI-DRIVE TEST ROUTINES
000000  010146                    QIO.OUT::
                                          MOV     R1,-(SP)              ;                    3640
000002  005001                            CLR     R1                    ; CTLR               3652
000004  010146                    1$:     MOV     R1,-(SP)              ; CTLR,*             3654
000006  004737  000000G                   JSR     PC,SET.CPAR
000012  013700  000000G                   MOV     CST.ADDR,R0           ;                    3656
000016  005760  000002                    TST     2(R0)
000022  100004                            BPL     2$
000024  005726                            TST     (SP)+                 ;                    3658
000026  012700  000001                    MOV     #1,R0
000032  000405                            BR      3$
000034  005726                    2$.     TST     (SP)+                 ;                    3653
000036  005201                            INC     R1                    ; CTLR               3652
000040  000243                            .WORD CLV!CLC

000042  003760                            BLE     1$
000044  005000                            CLR     R0
000046  012601                    3$:     MOV     (SP)+,R1              ;                    3650
000050  000207                            RTS     PC                    ;                    3640

; Routine Size:  21 words,      Routine Base:  $CODE$ + 11162
; Maximum stack depth per invocation:  3 words
```

```
GLOBAL routine QIO GEN : novalue =

!+
!       THIS ROUTINE IS CALLED BY THE MULTI DRIVE EXECUTIVE FOR AN ONLINE
!       CONTROLLER ELIGIBLE TO RECEIVE I/O TRANSFER REQUESTS. IT IS
!       RESPONSIBLE FOR SECURING ONE OR TWO MSCP PACKETS AND LOADING THEM
!       WITH VARIOUS PARAMETERS COMPRISING THE I/O REQUEST. THE I/O REQUEST
!       GENERATED HERE IS DESTINED TO A PARTICULAR UNIT SELECTED AT RANDOM FROM
!       THOSE CONFIGURED UNDER THE CURRENT CONTROLLER.
!
!       EACH FIELD OF THE PACKET(S) IS LOADED WITHIN INDIVIDUAL ROUTINES
!       (QIO_FUNC, QIO_LBN, QIO_SIZE, ETC.). MOST OF THE VALUES SELECTED FOR
!       EACH FIELD ARE BASED ON A SET OF RANDOM NUMBER GENERATED AT THE START.
!
!       UNDER NORMAL CIRCUMSTANCES, ONLY ONE I/O REQUEST IS GENERATED. HOWEVER,
!       IF THIS I/O REQUEST IS A "WRITE", AND IF THE OPERATOR SELECTED THE
!       OPTION FOR HOST WRITE-COMPARES, THEN A SECOND "READ" REQUEST WILL BE
!       GENERATED WITH THE SAME LBN AND BYTE COUNT.
!
!       AFTER THE PACKET(S) HAVE BEEN LOADED, THIS ROUTINE REGAINS CONTROL
!       AND ATTEMPTS TO GET ONE OR TWO I/O BUFFERS FOR THE ACTUAL DATA
!       TRANSFERS. THE SUCCESS / FAIL STATUS OF THIS ENTIRE OPERATION IS
!       PASSED BACK TO THE CALLER THROUGH THE GLOBALS "MX1" AND "MX2"; THEY
!       CONTAIN VALID MSCP PACKET INDECES, OR -1.
!
!       IMPLICIT INPUTS:
!               CCTLR - CURRENT CONTROLLER NUMBER
!-

    begin
    MX2 = -1;                                   ! ASSUME FAILURE IN SECURING 2ND PACKET

    if (MX1 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 1ST PACKET. IF FAILURE
    then
        return;                                 ! NO POINT IN CONTINUING

    if (MX2 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 2ND PACKET. IF FAILURE
    then
        begin
        PUT_PKT (.MX1);                         ! RETURN 1ST PACKET TO POOL
        MX1 = -1;                               ! INDICATE FAILURE
        return;                                 ! DONE
        end;

    MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2);     ! CALCULATE STARTING ADDRESSES
    MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2);     !     OF BOTH PACKETS
    GET_RANDOM ();                              ! GENERATE A SET OF RANDOM NUMBERS
    QIO_UNIT ();                                ! LOAD RANDOM UNIT NUMBER INTO PACKETS

    if .EOP_FLAG                                ! RETURN IF NO UNIT ONLINE
    then
        return;
```

```
;  2        QIO_FUNC ();                              ! LOAD RANDOM FUNCTION CODE (OPCODE)
;  2        QIO_LBN ();                               ! LOAD LBN (RANDOM OR SEQUENTIAL)
;  2        QIO_SIZE ();                              ! LOAD RANDOM BYTE COUNT
;  2        GET_IO_BUFF (MAD1 [BUF_0]);               ! TRY TO GET AN I/O BUFFER
;  2
;  2        if .MX2 geq 0                             ! IF TWO QIOs ARE TO BE ISSUED
;  2        then
;  3            begin
;  3            GET_IO_BUFF (MAD2 [BUF_0]);           ! TRY TO GET 2ND I/O BUFFER
;  3
;  3            if .MAD2 [BUF_0] eqla 0               ! IF 2ND BUFFER ALLOCATION FAILED
;  3            then
;  4                begin
;  4
;  4                if .MAD1 [BUF_0] neqa 0           ! IF 1ST I/O BUFFER WAS ALLOCATED
;  4                then
;  5                    begin
;  5                    PUT_IO_BUFF (MAD1 [BUF_0]);   ! RETURN 1ST I/O BUFFER TO POOL
;  5                    MAD1 [BUF_0] = 0;             ! MARK IT AS FAILED
;  4                    end;
;  4
;  4                PUT_PKT (.MX2);                   ! RETURN 2ND PACKET TO POOL
;  4                MX2 = -1;                         ! INDICATE FAILURE
;  3                end;                              ! IF 2ND I/O BUFFER ALLOCATION FAILED
;  3
;  2            end;                                  ! IF TWO QIOs ARE TO BE ISSUED
;  2
;  2        if .MAD1 [BUF_0] eqla 0                   ! IF 1ST I/O BUFFER ALLOCATION FAILED
;  2        then
;  3            begin
;  3            PUT_PKT (.MX1);                       ! RETURN 1ST PACKET TO POOL
;  3            MX1 = -1;                             ! INDICATE FAILURE
;  3            end
;  2        else
;  2
;  2            if .MAD1 [OPCODE] eql OP_WRT          ! OTHERWISE, IF 1ST OPCODE IS A WRITE  (ALL IS O.K.)
;  2            then
;  2                FILL_BUFF ();                     ! FILL 1ST I/O BUFFER WITH APPROPRIATE DATA PATTERN
;  2
;  1        end;                                      ! ROUTINE QIO_GEN

                                          .SBTTL  QIO.GEN MULTI-DRIVE TEST ROUTINES
000000  012737  177777  000112'   QIO.GEN::
                                          MOV     #-1,MX2
000006  013746  000000G                   MOV     CCTLR,-(SP)            ;                3694
000012  004737  000000G                   JSR     PC,GET.PKT             ;                3696
000016  010037  000110'                   MOV     R0,MX1
000022  005726                            TST     (SP)+
000024  005700                            TST     R0                     ; MX1
000026  002563                            BLT     6$
000030  013746  000000G                   MOV     CCTLR,-(SP)            ;                3698
000034  004737  000000G                   JSR     PC,GET.PKT             ;                3700
```

```
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)               ;                       3703
000056  004737  000000G                 JSR     PC,PUT.PKT
000062  012737  177777  000110'         MOV     #-1,MX1                 ;                       3704
000070  005726                          TST     (SP)+                   ;                       3705
000072  000207                          RTS     PC                      ;                       3702
000074  013746  000110'         1$:     MOV     MX1,-(SP)               ;                       3708
000100  012746  000106                  MOV     #106,-(SP)
000104  004737  000000G                 JSR     PC,BL$MUL
000110  062700  000000G                 ADD     #MSCP.PKT,R0
000114  010037  000114'                 MOV     R0,MAD1
000120  013716  000112'                 MOV     MX2,(SP)                ;                       3709
000124  012746  000106                  MOV     #106,-(SP)
000130  004737  000000G                 JSR     PC,BL$MUL
000134  062700  000000G                 ADD     #MSCP.PKT,R0
000140  010037  000116'                 MOV     R0,MAD2
000144  004737  000000V                 JSR     PC,GET.RANDOM           ;                       3710
000150  004737  000000V                 JSR     PC,QIO.UNIT             ;                       3711
000154  132737  000001  000000G         BITB    #1,EOP.FLAG             ;                       3713
000162  001103                          BNE     5$                      ;                       3664
000164  004737  000000V                 JSR     PC,QIO.FUNC             ;                       3717
000170  004737  000000V                 JSR     PC,QIO.LBN              ;                       3718
000174  004737  000000V                 JSR     PC,QIO.SIZE             ;                       3719
000200  013716  000114'                 MOV     MAD1,(SP)               ;                       3720
000204  062716  000032                  ADD     #32,(SP)
000210  004737  000000G                 JSR     PC,GET.IO.BUFF
000214  005737  000112'                 TST     MX2                     ;                       3722
000220  002437                          BLT     3$
000222  013716  000116'                 MOV     MAD2,(SP)               ;                       3725
000226  062716  000032                  ADD     #32,(SP)
000232  004737  000000G                 JSR     PC,GET.IO.BUFF
000236  013700  000116'                 MOV     MAD2,R0                 ;                       3727
000242  005760  000032                  TST     32(R0)
000246  001024                          BNE     3$
000250  013700  000114'                 MOV     MAD1,R0                 ;                       3731
000254  062700  000032                  ADD     #32,R0
000260  005710                          TST     (R0)
000262  001407                          BEQ     2$
000264  010016                          MOV     R0,(SP)                 ;                       3734
000266  004737  000000G                 JSR     PC,PUT.IO.BUFF
000272  013700  000114'                 MOV     MAD1,R0                 ;                       3735
000276  005060  000032                  CLR     32(R0)
000302  013716  000112'         2$:     MOV     MX2,(SP)                ;                       3738
000306  004737  000000G                 JSR     PC,PUT.PKT
000312  012737  177777  000112'         MOV     #-1,MX2                 ;                       3739
000320  013700  000114'         3$:     MOV     MAD1,R0                 ;                       3744
000324  005760  000032                  TST     32(R0)
000330  001010                          BNE     4$
000332  013716  000110'                 MOV     MX1,(SP)                ;                       3747
000336  004737  000000G                 JSR     PC,PUT.PKT
```

```
000342  012737  177777  000110'          MOV     # 1,MX1                ;            3748
000350  000410                           BR      5$                     ;            3744
000352  013700  000114'          4$:     MOV     MAD1,R0                ;            3752
000356  126027  000022  000042           CMPB    22(R0),#42
000364  001002                           BNE     5$
000366  004737  000000V                  JSR     PC,FILL.BUFF
000372  062706  000006           5$:     ADD     #6,SP                  ;            3754
000376  000207                   6$:     RTS     PC                     ;            3693
                                                                        ;            3664

; Routine Size:  128 words,     Routine Base:  $CODE$ + 11234
; Maximum stack depth per invocation:  4 words
```

```
;    3757  1     GLOBAL routine GET_RANDOM : novalue =
;    3758  1
;    3759  1     !+
;    3760  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO GENERATE A SET OF RANDOM NUMBERS,
;    3761  1     !       AND TO STORE THEM INTO THE RANDOM NUMBER TABLE (RANDOM). THE RANDOM
;    3762  1     !       NUMBERS ARE USED TO SELECT I/O REQUEST PARAMETERS FOR THE CURRENT QIO
;    3763  1     !       OR QIO PAIR. IN ADDITION, IF DATA PATTERN #1 IS BEING USED, THESE
;    3764  1     !       RANDOM NUMBERS WILL BE USED IN THE WRITE OPERATION.
;    3765  1     !-
;    3766  1
;    3767  2         begin
;    3768  2
;    3769  2         own
;    3770  2             SEED : word initial (173),
;    3771  2             NEXT_RANDNUM : word initial (245);
;    3772  2
;    3773  2         incr COUNT from 0 to (RDM_LEN - 1) do
;    3774  3             begin
;    3775  3             SEED = (.SEED + .NEXT_RANDNUM + 1) * 4;
;    3776  3             NEXT_RANDNUM = (.NEXT_RANDNUM / 4) + .SEED;
;    3777  3             RANDOM [.COUNT] = .NEXT_RANDNUM;
;    3778  2             end;
;    3779  2
;    3780  1         end;


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


                                                 .SBTTL  GET.RANDOM MULTI DRIVE TEST ROUTINES
011634                                           .PSECT  $CODE$,  RO

000000  004137  000000G                  GET.RANDOM::
                                                 JSR     R1,$SAVE3
000004  013703  001272'                          MOV     SEED,R3                 ;                3757
000010  013702  001274'                          MOV     NEXT.RANDNUM,R2         ;                3775
000014  005001                                   CLR     R1                      ; COUNT
000016  010200                           1$:     MOV     R2,R0                   ;                3773
000020  060300                                   ADD     R3,R0                   ;                3775
000022  006300                                   ASL     R0
000024  006300                                   ASL     R0
000026  010037  001272'                          MOV     R0,SEED
000032  062737  000004  001272'                  ADD     #4,SEED
000040  010246                                   MOV     R2,-(SP)
000042  012746  000004                           MOV     #4,-(SP)                ;                3776
000046  004737  000000G                          JSR     PC,BL$DIV
000052  013703  001272'                          MOV     SEED,R3
000056  060300                                   ADD     R3,R0
000060  010037  001274'                          MOV     R0,NEXT.RANDNUM
```

```
000064  010002                  MOV     R0,R2                   ; NEXT.RANDNUM.+            3777
000066  010261  000000G         MOV     R2,RANDOM(R1)           ; +,+(COUNT)
000072  022626                  CMP     (SP)+,(SP)+             ;                           3774
000074  062701  000002          ADD     #2,R1                   ; +,COUNT                   3773
000100  020127  000036          CMP     R1,#36                  ; COUNT,+
000104  003744                  BLE     1$
000106  000207                  RTS     PC                      ;                           3757

; Routine Size:  36 words,      Routine Base:  $CODE$ + 11634
; Maximum stack depth per invocation:  7 words
```

```
GLOBAL routine RANDY : novalue =

!+
!       THIS ROUTINE GENERATES A 32-BIT RANDOM NUMBER. THE LOW 16 BITS
!       ARE OUTPUT IN "RNDYO". THE HIGH 16 BITS ARE OUTPUT IN "RNDY1".
!
!       THE LOW 3 BITS OF CLK_TICKS SELECTS A WORD FROM 'RNDYIN'. THIS
!       IS 'R_STRING'. FRAME_CNT (0-9) SELECTS A 7-BIT FRAME OF THIS
!       WORD. BITS OF THIS FRAME ARE USED AS FOLLOWS:
!
!             BITS 0-2 ... SELECT A PATTERN FOR LOW WORD.
!             BITS 1-3 ... SELECT A PATTERN FOR HIGH WORD.
!             BIT   4  ... IF 1, SHIFT PATTERN LEFT.
!             BITS 4-6 ... SELECTS MASKS FOR FINAL OUTPUT.
!
!-

    begin

    local
      PAT_LO: WORD,                                         !LO WORD OF PATTERN
      PAT_HI: WORD,                                         !HI WORD OF PATTERN
      SHIFT : WORD,                                         !LEFT-SHIFT BIT
      MSKNO : WORD;                                         !WHICH MASK TO USE

   IF .FRAME_CNT EQLU 0                                     !IF IT'S TIME TO SAMPLE CLOCK AGAIN
   THEN
      BEGIN
      R_STRING = .RNDYIN [.CLK_TICKS AND 7]                 !CLOCK BITS SELECT 16 BIT STRING
      END;

   PAT_LO = .RNDYIN [(.R_STRING ↑ .FRAME_CNT) AND 7];       !BITS 0-2 OF FRAME SELECT LO WD OF PATTERN
   PAT_HI = .RNDYIN [(.R_STRING ↑( 1 -.FRAME_CNT)) AND 7];  !BITS 1-3 OF FRAME SELECT HI WD OF PATTERN

   SHIFT =  (.R_STRING ↑ (-4 -.FRAME_CNT)) AND 1;           !BIT 4 OF FRAME IS SHIFTER.
   PAT_LO =  .PAT_LO ↑ .SHIFT;                              !SHIFT PATTERN IF SHIFTER = 1
   PAT_HI = (.PAT_HI ↑ .SHIFT) + .SHIFT;                    !SHIFT PATTERN AND ADD 1 IF SHIFTER = 1

   MSKNO = (.R_STRING ↑ (-4 -.FRAME_CNT)) AND 7;            !GET MASK INDEX
   RNDYO = .PAT_LO AND (.RNDMSO [.MSKNO]);                  !MASK LO WORD
   RNDY1 = .PAT_HI AND (.RNDMS1 [.MSKNO]);                  !MASK HI WORD

   FRAME_CNT = .FRAME_CNT + 1;                              !SHIFT FRAME LEFT ONE BIT
   IF .FRAME_CNT GTRU 9                                     !IF DONE TEN RANDOM 32-BIT NUMBERS
   THEN
      FRAME_CNT = 0;                                        !ZERO IT, SO WE'LL READ CLOCK NEXT TIME
```

```
;   3834  1     END;

                                                  .SBTTL  RANDY MULTI DRIVE TEST ROUTINES
000000  004137  000000G                  RANDY::  JSR     R1,$SAVE4
000004  013702  000132'                           MOV     FRAME.CNT,R2                 ;                      3782
000010  001010                                    BNE     1$                           ;                      3808
000012  013700  000000G                           MOV     CLK.TICKS,R0
000016  042700  177770                            BIC     #177770,R0                   ;                      3811
000022  006300                                    ASL     R0
000024  016037  000136' 000134'                   MOV     RNDYIN(R0),R.STRING
000032  013701  000134'                  1$:      MOV     R.STRING,R1                  ;                      3810
000036  010146                                    MOV     R1,-(SP)                     ;                      3815
000040  010246                                    MOV     R2,-(SP)
000042  005416                                    NEG     (SP)
000044  004737  000000G                           JSR     PC,BL$SHF
000050  042700  177770                            BIC     #177770,R0
000054  006300                                    ASL     R0
000056  016004  000136'                           MOV     RNDYIN(R0),R4                ; *,PAT.LO
000062  010116                                    MOV     R1,(SP)                      ;                      3816
000064  012746  177777                            MOV     #-1,-(SP)
000070  160216                                    SUB     R2,(SP)
000072  004737  000000G                           JSR     PC,BL$SHF
000076  042700  177770                            BIC     #177770,R0
000102  006300                                    ASL     R0
000104  016003  000136'                           MOV     RNDYIN(R0),R3                ; *,PAT.HI
000110  010116                                    MOV     R1,(SP)                      ;                      3819
000112  012746  177774                            MOV     #-4,-(SP)
000116  160216                                    SUB     R2,(SP)
000120  004737  000000G                           JSR     PC,BL$SHF
000124  010001                                    MOV     R0,R1
000126  010102                                    MOV     R1,R2                        ; *,SHIFT
000130  042702  177776                            BIC     #177776,R2                   ; *,SHIFT
000134  010416                                    MOV     R4,(SP)                      ; PAT.LO,*             3820
000136  010246                                    MOV     R2,-(SP)                     ; SHIFT,*
000140  004737  000000G                           JSR     PC,BL$SHF
000144  010004                                    MOV     R0,R4                        ; *,PAT.LO
000146  010316                                    MOV     R3,(SP)                      ; PAT.HI,*             3821
000150  010246                                    MOV     R2,-(SP)                     ; SHIFT,*
000152  004737  000000G                           JSR     PC,BL$SHF
000156  060200                                    ADD     R2,R0                        ; SHIFT,*
000160  010003                                    MOV     R0,R3                        ; *,PAT.HI
000162  010102                                    MOV     R1,R2                        ; *,MSKNO              3824
000164  042702  177770                            BIC     #177770,R2                   ; *,MSKNO
000170  010200                                    MOV     R2,R0                        ; MSKNO,*              3825
000172  006300                                    ASL     R0
000174  016037  000160' 000126'                   MOV     RNDMS0(R0),RNDY0
000202  005104                                    COM     R4
000204  040437  000126'                           BIC     R4,RNDY0
000210  010200                                    MOV     R2,R0                        ; MSKNO,*              3826
000212  006300                                    ASL     R0
000214  016037  000200' 000130'                   MOV     RNDMS1(R0),RNDY1
000222  005103                                    COM     R3
```

```
000224  040337  000130'                     BIC     R3,RNDY1
000230  005237  000132'                     INC     FRAME.CNT                  ;                    3829
000234  023727  000132'  000011             CMP     FRAME.CNT,#11              ;                    3830
000242  101402                              BLOS    2$
000244  005037  000132'                     CLR     FRAME.CNT                  ;                    3832
000250  062706  000014           2$:        ADD     #14,SP                     ;                    3799
000254  000207                              RTS     PC                         ;                    3782

; Routine Size:  87 words,     Routine Base:  $CODE$ + 11744
; Maximum stack depth per invocation:  12 words
```

```
;  1   GLOBAL routine QIO_UNIT : novalue =
;  1
;  1   !+
;  1   !       THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
;  1   !       CONFIGURED UNDER THE CURRENT CONTROLLER (CCTLR) TO BE USED FOR THE
;  1   !       CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
;  1   !       UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
;  1   !       RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
;  1   !
;  1   !       IMPLICIT INPUTS:
;  1   !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  1   !
;  1   !       IMPLICIT OUTPUTS:
;  1   !               THE RD/RX DISK NUMBER (DISK ADDRESS) IS LOADED INTO THE
;  1   !                       APPROPRIATE FIELD OF BOTH MSCP PACKETS.
;  1   !-
;  1
;  2       begin
;  2
;  2       local
;  2           MOD_COUNT : byte,
;  2           TBL_COUNT : byte,
;  2           SELECT_RD : byte initial (byte (TRUE)),
;  2   !ZZZ    RD_COUNT : word initial (0),
;  2           RX_COUNT : word initial (0);
;  2
;  2   !
;  2   ! THE UNITS WILL BE SELECTED ON AN ADJUSTABLE RATIO, RD51/52 TO RX50,
;  2   ! SELECTED VIA THE SOFTWARE PARAMETERS
;  2   !
;  2   ! THIS MODE IS FOR SELECTING DEVICES ON THE FOLLOWING SCHEME:
;  2   ! CHOOSE A DEVICE AND KEEP IT SELECTED FOR A CONSTANT TIME, THEN
;  2   ! MOVE TO THE NEXT. THIS IS NON-RANDOM, FIXED SEQUENTIAL OPERATIONAL
;  2   ! MODE
;  2   !
;  2
;  2               RD_COUNT = 0;                                                          !ZZZ
;  2               RX_COUNT = 0;                                                          !ZZZ
;  2
;  2               incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT SIZE do
;  2
;  2                   if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  2                      (.CST ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  3                      (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;  2                   then
;  2
;  3                       if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;  2                       then
;  2                           RD_COUNT = .RD_COUNT + 1        ! NUMBER OF RD51/52s UNDER TEST
;  2                       else
;  2                           RX COUNT = .RX_COUNT + 1;       ! NUMBER OF RX50s UNDER TEST
;  2
```

```
;  2
;  2        if (not BIT_TST (SWP_FLAGS, SWF_RDM)) and              ! NOT RANDOM MODE
;  3           (not BIT_TST (SWP_FLAGS, SWF SEQ))                  ! NOT RANDOM SEQUEUNTIAL MODE
;  2        then
;  2
;  2            if (.BST_CNT r q 0) and
;  2               (.CST_ADDR , .BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;  2               (.CST_ADDR [.BST_DEV + OF DATA, D_STAT] eql ONLINE) and
;  3               (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;  2            then
;  3                begin                                            ! ALREADY WITHIN DEVICE
;  3                BST_CNT = .BST_CNT   1;
;  3                SET_UPAR (.BST_DEV);
;  3                MAD1 [DK_NUM] = .CDISK;
;  3                MAD2 [DK_NUM] = .CDISK;
;  3                return;
;  3                end
;  2            else
;  3                begin                                            ! GET NEW DEVICE
;  3
;  3  !ZZZ           incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  3  !ZZZ
;  3  !ZZZ               if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  3  !ZZZ                  (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  3  !ZZZ                  (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;  3  !ZZZ               then
;  3  !ZZZ
;  3  !ZZZ                   if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;  3  !ZZZ                   then
;  3  !ZZZ                       RD_COUNT = .RD_COUNT + 1         ! NUMBER OF RD51/52s UNDER TEST
;  3  !ZZZ                   else
;  3  !ZZZ                       RX COUNT = .RX_COUNT + 1;        ! NUMBER OF RX50s UNDER TEST
;  3
;  3             incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF UN) by UNIT SIZE do
;  4                 begin
;  4
;  4                 if (.BST_DEV eql 0) or
;  5                    (.BST_DEV eql ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN))
;  4                 then
;  4                     BST_DEV = OF_UN
;  4                 else
;  4                     BST_DEV = .BST_DEV + UNIT_SIZE;
;  4
;  4                 if (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;  4                    (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;  5                    (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;  4                 then
;  5                     begin
;  5
;  5                     if .CST ADDR [.BST_DEV + OF_DATA, D_TYPF] eql REMOVABLE
;  5                     then
;  5                         BST_CNT = .RX_MAX_SEQ_CNT / .RX_COUNT
;  5                     else
```

```
;  5                      BST_CNT = .RD_MAX_SEQ_CNT / .RD_COUNT;
;  5
;  5                  if .BST_CNT eql 0
;  5                  then
;  5                      BST_CNT = 1;
;  5
;  5                  SET_UPAR (.BST_DEV);
;  5                  MAD1 [DK_NUM] = .CDISK;
;  5                  MAD2 [DK_NUM] = .CDISK;
;  5                  return;
;  4                  end;
;  4
;  3              end;
;  3
;  2          end;
;  2
;  2  !
;  2  ! RANDOM SELECTION OF DRIVES
;  2  !
;  2  !
;  2  ! DETERMINE IF RD51/52s ARE TO BE SELECTED
;  2  !
;  2
;  2      if ((.RANDOM [RDM_LEN - 1] and %o'077777') mod 100) gequ .SWP_RAT
;  2      then
;  2          SELECT_RD = FALSE;
;  2
;  2  !
;  2  ! IF RD51/52s SELECTED
;  2  !
;  2  ! COUNT NUMBER OF RD51/52s AVAILABLE
;  2  !
;  2
;  2      if .SELECT_RD
;  2      then
;  3          begin
;  3          MOD_COUNT = 0;                              ! COUNT THE NUMBER OF RDs UNDER TEST
;  3
;  3          incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  3
;  3              if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  3                 (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  3                 (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED) and
;  4                 (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;  3              then
;  4                  begin
;  4                  STORAGE [.MOD_COUNT] = .OFFSET;
;  4                  MOD_COUNT = .MOD_COUNT + 1;
;  3                  end;
;  3
;  3  !
;  3  ! SELECT ON OF THE RD51/52s
;  3  !
```

```
            if .MOD_COUNT neq 0                                     ! IF AT LEAST ONE RD51/52 PRESENT
            then
                begin
                TBL_COUNT = 0;

                do
                    begin
                    SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
                    TBL_COUNT = .TBL_COUNT + 1;
                    end
                until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
                       (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
                       (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
                      (.TBL_COUNT eql RDM_LEN);

                MAD1 [DK_NUM] = .CDISK;
                MAD2 [DK_NUM] = .CDISK;
                return;
                end;

            end;

    !
    ! IF NO RD51/52 SELECTED, SELECT AN RX50
    !
    ! COUNT THE NUMBER OF RX50s
    !

        MOD_COUNT = 0;

        incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do

            if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
               (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
               (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql REMOVABLE) and
               (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
            then
                begin
                STORAGE [.MOD_COUNT] = .OFFSET;
                MOD_COUNT = .MOD_COUNT + 1;
                end;

    !
    ! AND CHOOSE ONE!
    !

        if .MOD_COUNT neq 0
        then
            begin
            TBL_COUNT = 0;

            do
```

```
;  4                     begin
;  4                     SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;  4                     TBL_COUNT = .TBL_COUNT + 1;
;  4                     end
;  4                until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;  4                       (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;  3                       (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;  3                     (.TBL_COUNT eql RDM_LEN);
;  3
;  3                MAD1 [DK_NUM] = .CDISK;
;  3                MAD2 [DK_NUM] = .CDISK;
;  3                return;
;  2                end;
;  2
;  2        !
;  2        ! IF NO UNIT SELECTED SO FAR BY ABOVE METHOD, SELECT ANY ONE AT RANDOM
;  2        !
;  2        ! COUNT ALL UNITS AVAILABLE
;  2        !
;  2
;  2           MOD_COUNT = 0;
;  2
;  2           incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF UN) by UNIT_SIZE do
;  2
;  2                if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  2                   (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  3                   (not .CST_ADDR [.OFFSET + OF_DATA, D FATAL])
;  2                then
;  3                    begin
;  3                    STORAGE [.MOD_COUNT] = .OFFSET;
;  3                    MOD_COUNT = .MOD_COUNT + 1;
;  2                    end;
;  2
;  2        !
;  2        ! SELECT ANY ONE ONE UNIT AT RANDOM
;  2        !
;  2           if .MOD_COUNT neq 0
;  2           then
;  3                begin
;  3                TBL_COUNT = 0;
;  3
;  3                do
;  4                    begin
;  4                    SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;  4                    TBL_COUNT = .TBL_COUNT + 1;
;  4                    end
;  4                until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;  4                       (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;  3                       (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;  3                     (.TBL_COUNT eql RDM_LEN);
;  3
;  3                MAD1 [DK_NUM] = .CDISK;
;  3                MAD2 [DK_NUM] = .CDISK;
```

```
;    4100  3              return
;    4101  3              end
;    4102  3
;    4103  3      !
;    4104  3      ! DECLARE END-OF-PASS IF NO UNIT ONLINE
;    4105  3      !
;    4106  3
;    4107  2          else
;    4108  2              EOP_FLAG = TRUE;
;    4109  2
;    4110  1          end;                                      ! ROUTINE QIO_UNIT


                                            .SBTTL  QIO.UNIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G             QIO.UNIT::
                                            JSR     R1,$SAVE4                  ;                      3835
000004  112704  000001                      MOVB    #1,R4                      ; *,SELECT.RD          3852
000010  005003                              CLR     R3                         ; RX.COUNT
000012  005037  000000G                     CLR     RD.COUNT                   ;                      3871
000016  013702  000000G                     MOV     CST.ADDR,R2                ;                      3876
000022  012701  000006                      MOV     #6,R1                      ; *,OFFSET             3874
000026  010100                      1$:     MOV     R1,R0                      ; OFFSET,*             3876
000030  060200                              ADD     R2,R0
000032  032710  040000                      BIT     #40000,(R0)
000036  001415                              BEQ     3$
000040  032710  020000                      BIT     #20000,(R0)                ;                      3877
000044  001412                              BEQ     3$
000046  032710  010000                      BIT     #10000,(R0)                ;                      3878
000052  001007                              BNE     3$
000054  132710  000020                      BITB    #20,(R0)                   ;                      3881
000060  001403                              BEQ     2$
000062  005237  000000G                     INC     RD.COUNT                   ;                      3883
000066  000401                              BR      3$                         ;                      3881
000070  005203                      2$:     INC     R3                         ; RX.COUNT             3885
000072  062701  000024              3$:     ADD     #24,R1                     ; *,OFFSET             3874
000076  020127  000102                      CMP     R1,#102                    ; OFFSET,*
000102  003751                              BLE     1$
000104  032737  000002  000000G             BIT     #2,SWP.FLAGS               ;                      3889
000112  001163                              BNE     13$
000114  032737  001000  000000G             BIT     #1000,SWP.FLAGS            ;                      3890
000122  001157                              BNE     13$
000124  005737  001242'                     TST     BST.CNT                    ;                      3893
000130  001447                              BEQ     4$
000132  013700  001244                      MOV     BST.DEV,R0                 ;                      3894
000136  006300                              ASL     R0
000140  060200                              ADD     R2,R0
000142  032710  040000                      BIT     #40000,(R0)
000146  001440                              BEQ     4$
000150  013700  001244                      MOV     BST.DEV,R0                 ;                      3895
000154  006300                              ASL     R0
000156  060200                              ADD     R2,R0
000160  032710  020000                      BIT     #20000,(R0)
000164  001431                              BEQ     4$
```

```
000166 013700 001244'            MOV  BST.DEV,R0          ;                 3896
000172 006300                    ASL  R0
000174 060200                    ADD  R2,R0
000176 032710 010000             BIT  #10000,(R0)
000202 001022                    BNE  4$
000204 005337 001242             DEC  BST.CNT
000210 013746 001244'            MOV  BST.DEV,-(SP)       ;                 3899
000214 004737 000000G            JSR  PC,SET.UPAR         ;                 3900
000220 013700 000114'            MOV  MAD1,R0
000224 013760 000000G 000016     MOV  CDISK,16(R0)        ;                 3901
000232 013700 000116'            MOV  MAD2,R0
000236 013760 000000G 000016     MOV  CDISK,16(R0)        ;                 3902
000244 005726                    TST  (SP)+
000246 000207                    RTS  PC                  ;                 3903
000250 012702 000003         4$: MOV  #3,R2               ;                 3898
000254 013700 001244'        5$: MOV  BST.DEV,R0          ; *,OFFSET        3921
000260 001403                    BEQ  6$                  ;                 3924
000262 020027 000041             CMP  R0,#41
000266 001004                    BNE  7$                  ;                 3925
000270 012737 000003 001244' 6$: MOV  #3,BST.DEV
000276 000403                    BR   8$                  ;                 3927
000300 062737 000012 001244' 7$: ADD  #12,BST.DEV         ;                 3924
000306 013700 001244'        8$: MOV  BST.DEV,R0          ;                 3929
000312 006300                    ASL  R0                  ;                 3931
000314 063700 000000G            ADD  CST.ADDR,R0
000320 032710 040000             BIT  #40000,(R0)
000324 001451                    BEQ  12$
000326 032710 020000             BIT  #20000,(R0)         ;                 3932
000332 001446                    BEQ  12$
000334 032710 010000             BIT  #10000,(R0)         ;                 3933
000340 001043                    BNE  12$
000342 132710 000020             BITB #20,(R0)            ;                 3937
000346 001004                    BNE  9$
000350 013746 001270'            MOV  RX.MAX.SEQ.CNT,-(SP)
000354 010346                    MOV  R3,-(SP)            ;                 3939
                                                          ; RX.COUNT,*
000356 000404                    BR   10$
000360 013746 001266         9$: MOV  RD.MAX.SEQ.CNT,-(SP)
000364 013746 000000G            MOV  RD.COUNT,-(SP)      ;                 3941
000370 004737 000000G       10$: JSR  PC,BL$DIV
000374 010037 001242'            MOV  R0,BST.CNT
000400 001003                    BNE  11$
000402 012737 000001 001242      MOV  #1,BST.CNT          ;                 3943
000410 013716 001244'       11$: MOV  BST.DEV,(SP)        ;                 3945
000414 004737 000000G            JSR  PC,SET.UPAR         ;                 3947
000420 013700 000114'            MOV  MAD1,R0
000424 013760 000000G 000016     MOV  CDISK,16(R0)        ;                 3948
000432 013700 000116'            MOV  MAD2,R0
000436 013760 000000G 000016     MOV  CDISK,16(R0)        ;                 3949
000444 022626                    CMP  (SP)+,(SP)+
000446 000207                    RTS  PC                  ;                 3950
000450 062702 000012        12$: ADD  #12,R2              ;                 3935
000454 020227 000041             CMP  R2,#41              ; *,OFFSET        3921
                                                          ; OFFSET,*
000460 003675                    BLE  5$
```

```
000462 013746 000036G    13$:  MOV   RANDOM+36,-(SP)      ;                 3964
000466 042716 100000           BIC   #100000,(SP)
000472 012746 000144           MOV   #144,-(SP)
000476 004737 000000G          JSR   PC,BL$MOD
000502 022626                  CMP   (SP)+,(SP)+
000504 020037 000000G          CMP   R0,SWP.RAT
000510 103401                  BLO   14$
000512 105004                  CLRB  R4                   ; SELECT.RD       3966
000514 006004            14$:  ROR   R4                   ; SELECT.RD       3974
000516 103105                  BCC   19$
000520 105003                  CLRB  R3                   ; MOD.COUNT       3977
000522 012701 000003           MOV   #3,R1                ; *,OFFSET        3979
000526 010100            15$:  MOV   R1,R0                ; OFFSET,*        3981
000530 006300                  ASL   R0
000532 063700 000000G          ADD   CST.ADDR,R0
000536 032710 040000           BIT   #40000,(R0)
000542 001417                  BEQ   16$
000544 032710 020000           BIT   #20000,(R0)          ;                 3982
000550 001414                  BEQ   16$
000552 132710 000020           BITB  #20,(R0)             ;                 3983
000556 001411                  BEQ   16$
000560 032710 010000           BIT   #10000,(R0)          ;                 3984
000564 001006                  BNE   16$
000566 005000                  CLR   R0                   ;                 3987
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       3988
000602 062701 000012     16$:  ADD   #12,R1               ; *,OFFSET        3979
000606 020127 000041           CMP   R1,#41               ; OFFSET,*
000612 003745                  BLE   15$
000614 105703                  TSTB  R3                   ; MOD.COUNT       3995
000616 001445                  BEQ   19$
000620 105002                  CLRB  R2                   ; TBL.COUNT       3998
000622 005000            17$:  CLR   R0                   ;                 4002
000624 150200                  BISB  R2,R0                ; TBL.COUNT,*
000626 006300                  ASL   R0
000630 016046 000000G          MOV   RANDOM(R0),-(SP)
000634 042716 100000           BIC   #100000,(SP)
000640 005046                  CLR   -(SP)
000642 110316                  MOVB  R3,(SP)              ; MOD.COUNT,*
000644 004737 000000G          JSR   PC,BL$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       4003
000664 022626                  CMP   (SP)+,(SP)+          ;                 4001
000666 013700 000000G          MOV   CUOFF,R0             ;                 4005
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)          ;                 4006
```

```
000712  001403                      BEQ     18$
000714  032710  010000              BIT     #10000,(R0)         ;                   4007
000720  001510                      BEQ     24$
000722  120227  000020      18$:    CMPB    R2,#20              ; TBL.COUNT,*       4008
000726  001335                      BNE     17$
000730  000504                      BR      24$                 ;                   4010
000732  105003              19$:    CLRB    R3                  ; MOD.COUNT         4023
000734  012701  000003              MOV     #3,R1               ; *,OFFSET          4025
000740  010100              20$:    MOV     R1,R0               ; OFFSET,*          4027
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)         ;                   4028
000762  001414                      BEQ     21$
000764  132710  000020              BITB    #20,(R0)            ;                   4029
000770  001011                      BNE     21$
000772  032710  010000              BIT     #10000,(R0)         ;                   4030
000776  001006                      BNE     21$
001000  005000                      CLR     R0                  ;                   4033
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         4034
001014  062701  000012      21$:    ADD     #12,R1              ; *,OFFSET          4025
001020  020127  000041              CMP     R1,#41              ; OFFSET,*
001024  003745                      BLE     20$
001026  105703                      TSTB    R3                  ; MOD.COUNT         4041
001030  001445                      BEQ     25$
001032  105002                      CLRB    R2                  ; TBL.COUNT         4044
001034  005000              22$:    CLR     R0                  ;                   4048
001036  150200                      BISB    R2,R0               ; TBL.COUNT,*
001040  006300                      ASL     R0
001042  016046  000000G             MOV     RANDOM(R0),-(SP)
001046  042716  100000              BIC     #100000,(SP)
001052  005046                      CLR     -(SP)
001054  110316                      MOVB    R3,(SP)             ; MOD.COUNT,*
001056  004737  000000G             JSR     PC,BL$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         4049
001076  022626                      CMP     (SP)+,(SP)+         ;                   4047
001100  013700  000000G             MOV     CUOFF,R0            ;                   4051
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)         ;                   4052
001124  001403                      BEQ     23$
001126  032710  010000              BIT     #10000,(R0)         ;                   4053
001132  001505                      BEQ     30$
001134  120227  000020      23$:    CMPB    R2,#20              ; TBL.COUNT,*       4054
```

```
001140  001335                          BNE     22$
001142  000501                  24$:    BR      30$
001144  105003                  25$:    CLRB    R3                      ;                               4056
001146  012701  000003                  MOV     #3,R1                   ; MOD.COUNT                     4067
001152  010100                  26$:    MOV     R1,R0                   ; *,OFFSET                      4069
001154  006300                          ASL     R0                      ; OFFSET,*                      4071
001156  063700  000000G                 ADD     CST.ADDR,R0
001162  032710  040000                  BIT     #40000,(R0)
001166  001414                          BEQ     27$
001170  032710  020000                  BIT     #20000,(R0)
001174  001411                          BEQ     27$                     ;                               4072
001176  032710  010000                  BIT     #10000,(R0)
001202  001006                          BNE     27$                     ;                               4073
001204  005000                          CLR     R0
001206  150300                          BISB    R3,R0                   ;                               4076
001210  006300                          ASL     R0                      ; MOD.COUNT,*
001212  010160  000064'                 MOV     R1,STORAGE(R0)
001216  105203                          INCB    R3                      ; OFFSET,*
001220  062701  000012          27$:    ADD     #12,R1                  ; MOD.COUNT                     4077
001224  020127  000041                  CMP     R1,#41                  ; *,OFFSET                      4069
001230  003750                          BLE     26$                     ; OFFSET,*
001232  105703                          TSTB    R3
001234  001457                          BEQ     31$                     ; MOD.COUNT                     4083
001236  105002                          CLRB    R2
001240  005000                  28$:    CLR     R0                      ; TBL.COUNT                     4086
001242  150200                          BISB    R2,R0                   ;                               4090
001244  006300                          ASL     R0                      ; TBL.COUNT *
001246  016046  000000G                 MOV     RANDOM(R0),-(SP)
001252  042716  100000                  BIC     #100000,(SP)
001256  005046                          CLR     -(SP)
001260  110316                          MOVB    R3,(SP)
001262  004737  000000G                 JSR     PC,BL$MOD               ; MOD.COUNT,*
001266  006300                          ASL     R0
001270  016016  000064'                 MOV     STORAGE(R0),(SP)
001274  004737  000000G                 JSR     PC,SET.UPAR
001300  105202                          INCB    R2
001302  022626                          CMP     (SP)+,(SP)+             ; TBL.COUNT                     4091
001304  013700  000000G                 MOV     CUOFF,R0                ;                               4089
001310  006300                          ASL     R0                      ;                               4093
001312  063700  000000G                 ADD     CST.ADDR,R0
001316  032710  040000                  BIT     #40000,(R0)
001322  001406                          BEQ     29$
001324  032710  020000                  BIT     #20000,(R0)
001330  001403                          BEQ     29$                     ;                               4094
001332  032710  010000                  BIT     #10000,(R0)
001336  001403                          BEQ     30$                     ;                               4095
001340  120227  000020          29$:    CMPB    R2,#20
001344  001335                          BNE     28$                     ; TBL.COUNT,*                   4096
001346  013700  000114'         30$:    MOV     MAD1,R0
001352  013760  000000G 000016          MOV     CDISK,16(R0)            ;                               4098
001360  013700  000116'                 MOV     MAD2,R0
001364  013760  000000G 000016          MOV     CDISK,16(R0)            ;                               4099
001372  000207                          RTS     PC                      ;                               4085
```

```
001374  112737  000001  000000G          31$:     MOVB    #1,EOP.FLAG                          ;          4108
001402  000207                                    RTS     PC                                   ;          3835

; Routine Size:  386 words,     Routine Base:  $CODE$ + 12222
; Maximum stack depth per invocation:  8 words
```

```
;  1   GLOBAL routine QIO_FUNC : novalue =
;  1
;  1   !+
;  1   !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
;  1   !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
;  1   !       BY THE FOLLWING ALGORITHM:
;  1   !
;  1   !               IF THE CHOSEN UNIT IS PROTECTED
;  1   !               THEN
;  1   !                       FUNCTION = READ
;  1   !               ELSE (UNPROTECTED)
;  1   !                       FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
;  1   !                       NUMBER
;  1   !
;  1   !       IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE-
;  1   !       COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
;  1   !       THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
;  1   !       PACKET. OTHERWISE, THE SECOND MSCP PACKET IS RETURNED TO THE POOL.
;  1   !
;  1   !
;  1   !MMM!       PERIODIACLLY, THIS ROUTINE WILL CALL THE DUP ROUTINE BEFORE IT        ZZZ
;  1   !MMM!   BEGINS ITS OWN TASK. IF THE OPERATOR HAS SELECTED, "ALSO RUN     ZZZ
;  1   !MMM!   DUP EXERCISER," THEN DUP TESTING OF DBNS WILL BE INTERLEAVED     ZZZ
;  1   !MMM!   WITH THE REGULAR MSCP TESTING OF THE LBNS.                       ZZZ
;  1   !MMM!
;  1   !MMM!   TO AVOID LONG, CUMULATIVE INIT TIMES, THE DUP CODE IS ONLY       ZZZ
;  1   !MMM!   EXECUTED AFTER (25 TIMES 'DUPROUND') MSCP I/O's HAVE BEEN DONE.  ZZZ
;  1   !MMM!   THE DUMBER OF DUP I/O's IS 'DUPROUND'. THIS GIVES US A 25 TO 1   ZZZ
;  1   !MMM!   INTERLEAVE.                                                      ZZZ
;  1   !MMM!
;  1   !MMM!   THE DUP TESTING IS DONE BY EXECUTING CONTROLLER LOCAL PROGRAMS   ZZZ
;  1   !MMM!   TO READ OR WRITE/READ DBNS. AFTER THE DUP TESTING, THE CON-      ZZZ
;  1   !MMM!   TROLLER IS REINITIALIZED, AND QIO_FUNC ROUTINE CONTINUES FROM    ZZZ
;  1   !MMM!   WHERE IT LEFT OFF.                                               ZZZ
;  1   !MMM!
;  1   !
;  1   !       IMPLICIT INPUTS:
;  1   !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  1   !               CUOFF - CURRENT UNIT CST OFFSET
;  1   !
;  1   !       IMPLICIT OUTPUTS:
;  1   !               THE OPCODE FIELD OF ONE OR BOTH MSCP PACKETS IS LOADED.
;  1   !-
;  1
;  2       begin
;  2
;  2       local
;  2           FUNC : word,                                    ! OPCODE (READ OR WRITE)
;  2           INDEX : word;                                   ! UNIT NO.          ZZZ
;  2
;  2   !MMM DUOFF = .CUOFF;                          !SAVE IN CASE OTHER CMDS ZZZ
;  2   !MMM                                         !LEFT IN QUEUE          ZZZ
;  2   !MMM IF ((.CST_ADDR [.DUOFF + OF COUNT, D_COUNT] LEQ 0) AND !MSCP CNT=0        ZZZ
```

```
2   !MMM    (.CST_ADDR [.DUOFF, D_TYPE] NEQ RX_50) AND        !FIXED DISK          ZZZ
2   !MMM    (.CST_ADDR [.DUOFF + OF DUPFLAGS, NODUPMEDIA] NEQ 1)) !MEDIA IN        ZZZ
2   !MMM                                                                          !ZZZ
2   !MMM THEN                                                                     !ZZZ
2   !MMM     BEGIN                                                                !ZZZ
2   !MMM     PUT_PKT (.MX2);                           !RETURN 2ND ENVLOPE        !ZZZ
2   !MMM     MX2 = -1;                                 !INDICATE FAILURE           ZZZ
2   !MMM     DUP ();                                   !DO DUP TEST                ZZZ
2   !MMM     CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] =  !REINIT MSCP FUN-      ZZZ
2   !MMM             (25 * .DUPROUND);                 !CTION COUNTER              ZZZ
2   !MMM                                                                    !ZZZ
2   !MMM !  THE FOLLOWING REINITS 2 ENVELOPES, SO THAT THE MSCP EXERCISER    ZZZ
2   !MMM !  CAN PROCEED AS BEFORE THE DUP EXERCISER WAS CALLED.              ZZZ
2   !MMM !                                                                   ZZZ
2   !MMM    DUP_FLAGS = .DUP_FLAGS OR SWP_DINT;        !SET DUP INIT FLAG          ZZZ
2   !MMM    INIT_TEST ();                      !REINIT CONTROLLER             ZZZ
2   !MMM    DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT); !CLR DUP INIT DLAG          ZZZ
2   !MMM                                                                          !ZZZ
2   !MMM    MX2 = -1;                                  !ASSUME NO 2ND ENVELOPE     ZZZ
2   !MMM    IF (MX1 = GET_PKT (.CCTLR)) LSS 0          !TRY FOR 1ST ENVELOPE       ZZZ
2   !MMM              OR (.EOP_FLAG)                   !IF FAILURE                 ZZZ
2   !MMM    THEN      RETURN;                          !NO POINT TO GO ON          ZZZ
2   !MMM    IF (MX2 = GET_PKT (.CCTLR)) LSS 0          !TRY FOR 2ND ENVELOPE       ZZZ
2   !MMM              OR (.EOP_FLAG)                   !IF FAILURE                 ZZZ
2   !MMM    THEN      BEGIN                                                 !ZZZ
2   !MMM          PUT_PKT (.MX1);              !PUT 1ST BACK IN POOL         ZZZ
2   !MMM              MX1 = -1;                        !INDICATE FAILURE           ZZZ
2   !MMM              RETURN;                          !DONE                       ZZZ
2   !MMM              END;                                                  !ZZZ
2   !MMM                                                                    !ZZZ
2   !MMM              MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2); !CALC START ADDR   ZZZ
2   !MMM              MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2); !OF BOTH ENVELOPES ZZZ
2   !MMM              GET_RANDOM ();                   !GET SET OF RANDOM NOS      ZZZ
2   !MMM              QIO_UNIT ();             !PUT RAND UNIT NO IN          ZZZ
2   !MMM              END;                     !ENVELOPES                    ZZZ
2   !MMM                                                                    !ZZZ
2
2   !MMM!   MSCP CODE STARTS HERE                                            ZZZ
2   !MMM!                                                                    ZZZ
2
2   INDEX = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];              !POINT TO TYPER + TYPEW FOR THIS UNIT   ZZZ
2
2   CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] =          !                   ZZZ
2       .CST_ADDR [.CUOFF + OF_COUNT,
2   D_COUNT] -1;    !DECR MSCP FUNCTION CNTR ZZZ
2
2       MAD2 [OPCODE] = 0;                                         ! ASSUME 2ND PACKET NOT NEEDED
2
2       if (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED) and      ! IF "FORCED ERROR" SET IN LAST READ,
2          (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and            !       REWRITE SAME BLOCK
3          (.FORCED_ERROR)                                                  !
2       then
2           FUNC = OP_WRT
```

```
2            else
2
2                if .CST_ADDR [.CUOFF + OF DATA, D_PROT] eql PROTECTED      ! IF UNIT IS PROTECTED
2                then
2                    FUNC = OP_RD                                           ! SET FUNCTION TO READ
2                else
2
2   !MMM             if (.RANDOM [1] and 1)                                 ! USE 2ND RANDOM NUMBER TO SELECT
2   !MMM             then
2   !MMM                 FUNC = OP_RD                                       !     READ
2   !MMM             else
2   !MMM                 FUNC = OP_WRT;                                     !     WRITE
2
2                    IF NOT .BAL_IN_PROGRESS [.INDEX]                       ! MMM
2                    THEN                                                   ! MMM
3                       IF (.RANDOM [1] AND 1)                              ! MMM
2                       THEN                                                ! MMM
2                          FUNC = OP_RD                                     ! MMM
2                       ELSE                                                ! MMM
2                          FUNC = OP_WRT                                    ! MMM
2                     ELSE                                                  ! MMM
2                        IF .FORCE_WR [.INDEX]                              ! MMM
2                        THEN                                               ! MMM
2                           FUNC = OP_WRT                                   ! MMM
2                        ELSE                                               ! MMM
2                           FUNC = OP_RD;                                   ! MMM
2
2
2        if (MAD1 [OPCODE] = .FUNC) eql OP_WRT                              ! LOAD CHOSEN OPCODE. IF WRITE
2        then
3            begin
3            MAD1 [CMD_TYPE] = NON_SEQ_CMD;                                 ! NON-SEQUENTIAL COMMAND
3
4            if BIT_TST (SWP_FLAGS, SWF_CWC)                                ! IF CONTROLLER DOES WRITE COMPARES
3            then
3                MAD1 [MODIFY] = MD CMP                                     ! ADD COMPA.= MODIFIER
3            else
3
4                if BIT_TST (SWP_FLAGS, SWF_HWC)                            ! IF HOST DOES WRITE COMPARES
3                then
4                    begin
4                    MAD1 [MODIFY] = MD_EXP;                                ! SET WRITE AS AN EXPRESS REQUEST
4                    MAD2 [OPCODE] = OP_RD;                                 ! SET READ OPCODE INTO 2ND MSCP PACKET
4                    MAD2 [MODIFY] = MD_EXP;                                ! SET READ AS AN EXPRESS REQUEST TOO
4                    MAD2 [CMD TYPE] = NON_SEQ_CMD;                         ! NON-SEQUENTIAL COMMAND
3                    end;
3            end
2        else
3            begin
3            MAD1 [CMD_TYPE] = NON_SEQ_CMD;                                 ! NON-SEQUENTIAL COMMAND
3
4            if BIT_TST (SWP_FLAGS, SWF_CRC)                                ! IF READ COMPARES   FUNCTION IS READ
3            then
```

```
;      4270  3                    MAD1 [MODIFY] = MD_CMP;                           ! ADD COMPARE MODIFIER
;      4271  3
;      4272  2                end;
;      4273  2
;      4274  2            if .MAD2 [OPCODE] eql 0                                       ! IF NO OPCODE IN 2ND PACKET
;      4275  2            then
;      4276  3                begin
;      4277  3                PUT_PKT (.MX2);                                           ! RETURN 2ND PACKET TO POOL
;      4278  3                MX2 = -1;                                                 ! MARK IT UNUSED
;      4279  2                end;
;      4280  2
;      4281  1            end;                                                          ! ROUTINE QIO FUNC

                                                  .SBTTL  QIO.FUNC MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                  QIO.FUNC::
                                                  JSR     R1,$SAVE4
000004  013702  000000G                           MOV     CUOFF,R2                  ;                    4111
000010  006302                                    ASL     R2                        ;                    4204
000012  063702  000000G                           ADD     CST.ADDR,R2
000016  111200                                    MOVB    (R2),R0
000020  042700  177760                            BIC     #177760,R0                ; *.INDEX
000024  013701  000000G                           MOV     CUOFF,R1                  ; *.INDEX
000030  006301                                    ASL     R1                        ;                    4206
000032  063701  000000G                           ADD     CST.ADDR,R1
000036  005361  000022                            DEC     22(R1)
000042  013701  000116'                           MOV     MAD2,R1                   ;                    4208
000046  012704  000022                            MOV     #22,R4                    ;                    4210
000052  060104                                    ADD     R1,R4
000054  105014                                    CLRB    (R4)
000056  005712                                    TST     (R2)
000060  100007                                    BPL     1$                        ;                    4212
000062  132712  000020                            BITB    #20,(R2)
000066  001404                                    BEQ     1$                        ;                    4213
000070  132737  000001  000000G                   BITB    #1,FORCED.ERROR
000076  001021                                    BNE     3$                        ;                    4214
000100  032712  100000           1$:              BIT     #100000,(R2)              ;                    4216
000104  001421                                    BEQ     4$                        ;                    4219
000106  006300                                    ASL     R0                        ;                    4221
000110  032760  000001  000000G                   BIT     #1,BAL.IN.PROGRESS(R0)    ;                    4230
000116  001005                                    BNE     2$
000120  032737  000001  000002G                   BIT     #1,RANDOM+2
000126  001405                                    BEQ     3$                        ;                    4232
000130  000407                                    BR      4$
000132  032760  000001  000000G  2$:              BIT     #1,FORCE.WR(R0)           ;                    4234
000140  001403                                    BEQ     4$                        ;                    4238
000142  012702  000042           3$:              MOV     #42,R2
000146  000402                                    BR      5$                        ; *.FUNC             4240
000150  012702  000041           4$:              MOV     #41,R2                    ;                    4238
000154  013700  000114'          5$:              MOV     MAD1,R0                   ; *.FUNC             4242
000160  013703  000000G                           MOV     SWP.FLAGS,R3              ;                    4245
000164  110260  000022                            MOVB    R2,22(R0)                 ;                    4250
000170  020227  000042                            CMP     R2,#42                    ; FUNC.*             4245
                                                                                    ; FUNC.*
```

```
000174  001025                               BNE     6$
000176  112760  000002  000004               MOVB    #2,4(R0)                  ;   4248
000204  032703  000020                       BIT     #20,R3                    ;   4250
000210  001025                               BNE     7$                        ;   4252
000212  032703  000040                       BIT     #40,R3                    ;   4255
000216  001425                               BEQ     8$
000220  012760  100000  000024               MOV     #-100000,24(R0)           ;   4258
000226  112714  000041                       MOVB    #1,(R4)                   ;   4259
000232  012761  100000  000024               MOV     #-100000,24(R1)           ;   4260
000240  112761  000002  000004               MOVB    #2,4(R1)                  ;   4261
000246  000411                               BR      8$                        ;   4245
000250  112760  000002  000004       6$:     MOVB    #2,4(R0)                  ;   4266
000256  032703  000004                       BIT     #4,R3                     ;   4268
000262  001403                               BEQ     8$
000264  012760  040000  000024       7$:     MOV     #40000,24(R0)             ;   4270
000272  105714                       8$:     TSTB    (R4)                      ;   4274
000274  001010                               BNE     9$
000276  013746  000112'                      MOV     MX2,-(SP)                 ;   4277
000302  004737  000000G                      JSR     PC,PUT.PKT
000306  012737  177777  000112'              MOV     #-1,MX2                   ;   4278
000314  005726                               TST     (SP)+                     ;   4276
000316  000207                       9$:     RTS     PC                        ;   4111

; Routine Size:  104 words,     Routine Base:  $CODE$ + 13626
; Maximum stack depth per invocation:  7 words
```

```
!MMMGLOBAL ROUTINE DUP : NOVALUE =                                          !ZZZ
!MMM!+                                                          !ZZZ
!MMM!   THIS ROUTINE IS CALLED BY QIO_FUNC AFTER 25 * 'DUPROUND' RD/WTS. ZZZ
!MMM!   THIS EXERCISER WAS PLACED IN THE MIDDLE OF THE MSCP EXERCISER,  ZZZ
!MMM!   SO COMMON INIT AND OTHER ROUTINES COULD BE USED.                ZZZ
!MMM!                                                                   ZZZ
!MMM!   THE DUP EXERCISER WILL PERFORM EITHER READ-ONLY, OR WRITE-READ- ZZZ
!MMM!   COMPARE OPERATIONS ON THE DIAGNOSTIC BLOCKS (DBNS). IT WILL     ZZZ
!MMM!   RECORD THE STATISTICS IN THE TALLY TABLES.                      ZZZ
!MMM!                                                                   ZZZ
!MMM!   THE PROGRAM USES CONTROLLER LOCAL PROGRAMS TO WRITE AND READ    ZZZ
!MMM!   DBNS. WHEN WRITING TO THE DBNS, A ONE WORD PATTERN WILL BE      ZZZ
!MMM!   SELECTED, AND REPLICATED THROUGH A 256 WORD BLOCK FOR DATA.     ZZZ
!MMM!   THE ROUTINE WILL WRITE 'DUPROUND' NUMBER OF SEQUENTIAL DBN      ZZZ
!MMM!   BLOCKS. IF THE CONTROLLER LOCAL PROGRAMS EXIST, AND THE OPERATOR ZZZ
!MMM!   SELECTS 'WRITE TO DIAGNOSTIC AREA', WRITE-READ-COMPARES WILL BE ZZZ
!MMM!   PERFORMED ON THE DBNS. OTHERWISE, READS WITH NO COMPARES WILL BE ZZZ
!MMM!   DONE. BAD BLOCKS FOUND IN THE COMPARISON TESTS WILL NOT BE LIST- ZZZ
!MMM!   ED IN THE RCT TABLES.                                           ZZZ
!MMM!                                                                   ZZZ
!MMM!   AFTER 'DUPROUND' NUMBER OF DBNS HAVE BEEN TESTED, THE ENVELOPES ZZZ
!MMM!   WILL BE REINITIATED, SO THAT THE MSCP EXERCISER CAN CONTINUE    ZZZ
!MMM!   AS BEFORE.                                                      ZZZ
!MMM!                                                                   ZZZ
!MMM!   IMPLICIT INPUTS:                                                ZZZ
!MMM!                   CCTLR - CURRENT CONTROLLER NUMBER               ZZZ
!MMM!                   CST_ADDRS   CONTAINS THE CURRENT CONTROLLER     ZZZ
!MMM!                               STATUS TABLE                        ZZZ
!MMM!                   CUOFF - CURRENT OFFSET IN CST TABLE FOR         ZZZ
!MMM!                           PARTICULAR DRIVE                        ZZZ
!MMM!                                                                   ZZZ
!MMM!   IMPLICIT OUTPUTS:                                               ZZZ
!MMM!                   S_PATTERN - PATTERN BEING WRITTEN TO DBNS       ZZZ
!MMM!                                                                   ZZZ
!MMM!                                                                   !ZZZ
!MMM!                                                                   !ZZZ
```

```
!MMM


!MMM BEGIN
!MMM OWN
!MMM     TEMP : WORD;

!PRINTX (DBM110);
!PRINTX (DER10);

!until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL PETURN MESSAGES HAVE BEEN PROCESSED
!      (.EOP_FLAC eql true) do                   ! Make sure all MSCP commands are completed
!    begin
!    BREAK;                                      ! BREAK FOR ACT
!    PROC_RETPKT();                              ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
!    RP_INDX = .RP_INDX + 1;                     ! INCREMENT RP_INDX
!    if .RP_INDX geq RP_CNT then (RP_INDX = 0;); ! MAKE SURE THE COUNTER DOES NOT GET TO BIG
!    RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);        ! CALCULATE RETPKT ADDRESS
!    end;


!S_PATTERN = .RANDOM [1];                                    !OTHER UNIT VARIABLES

!IF (.CST_ADDR [.DUOFF + OF_DBN, D_DBN] + .duporound) GEQ 144     ! TEST TO SEE IF NEXT DBN'S TO LARGE
!THEN (CST_ADDR [.DUOFF + OF_DBN, D_DBN] = 0);               ! CIRCLE ARROUND IF DBN TO LARGE

!DUPIDLE ();                                                 ! DO A GET DUST STATUS TO FIND IF LOCAL DUP MEDIA
!IF .CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] EQL 1 THEN RETURN; ! IF DUP LOCAL MEDIA NOT THERE THEN RETURN

!TEMP = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];
!INCR DBNCNT FROM (.TEMP + 1) TO (.TEMP + .duporound) DO                      ! INCREMENT FROM RELATIVF DBN TO DBN
 + duporound
!       BEGIN
!       IF .CST_ADDR [.DUOFF + OF DBN, DUPWRITE]                            ! IF WRITE FLAG SET IN CST TABLE THE
N WRITE DBN'S
!       THEN
!               BEGIN
!               DUPIDLE ();                                          ! MAKE SURE THE CONTROLLER IS IN AN IDLE STA
TE
!               DUPWRTDBN ();                                        ! CALL ROUTINE TO HANDLE WRITTING ROUTINES
!               END;

!       DUPIDLE ();                                                  ! MAKE SURE CONTROLLER IN IDLE STATE
!       DUPREDDBN ();                                                ! CALL ROUTINE TO HANDLE READING DBN'S

!       CST_ADDR [.DUOFF + OF_DBN, D_DBN] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN] + 1; ! INCREMENT RELATIVE DBN COUNTER
!        IF .CST_ADDR [.DUOFF + OF_DBN, D_DBN]  GTRU  MAX_DBN                  !BUT NOT MORE THAN MAX NUMBER
ZZZ
!        THEN                                                                  !IF BIGGER THAN MAX
ZZZ
!          CST_ADDR [.DUOFF + OF_DBN, D_DBN]  =  0                             !MAKE IT ZERO
ZZZ



!       IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1               ! ERROR IN DUP REINITIALIZE
!       THEN RETURN;                                                 ! AND RETURN
!       END;
!END;
```

. 4373 1

```
!MMMGLOBAL ROUTINE DUPWRTDBN : NOVALUE =
!MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
!+
! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
! "WRTDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM WRITES TO A DIAGONOSTIC BLOCK (DBN)
! THE WORD IN "S_PATTERN" IS WRITTEN TO THE 256 WORDS IN THE DBN.  IF AN ERROR OCCURS
! WHILE RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
!
!        IMPLICIT INPUTS:
!                 CST_ADDRS    CONTAINS THE CURRENT CONTROLLER STATUS TABLE
!                 DUOFF      - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
!                 S_PATTERN    CONTAINS PATTERN WORD!-
!BEGIN
!LOCAL                                                !                                    ZZZ
!       TRYNUM : WORD,                                                                     ZZZ
!       MAX_TRY_COUNT : word initial (9);             ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR ZZZ
!                                                                                          ZZZ
!LABEL                                                !                                    ZZZ
!      DUP_WLOOP;                                     !START OF DUP WRITE RETRY LOOP       ZZZ

!PRINTX (DER11);
!T_ADDR [T_DBN_WT] = .T ADDR [T DBN WT] + 1;        ! INCREMENT # OF WRITES GIVEN

!TRYNUM = 0;                                          !ZERO TRY COUNTER                    ZZZ
!DUP_WLOOP:                                           !LABEL FOR LOOP EXCAPE ON GOOD WRITE ZZZ
!BEGIN                                                !BEGIN DUP_WLOOP                     ZZZ
!INCR TRIES FROM 1 TO 10 DO                           !START TRYING DUP WRITES             ZZZ
!BEGIN                                                !BEGIN LARGE DO LOOP                 ZZZ
                                                      !                                    ZZZ

!MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                    ! PACKET SIZE            EXECUTE LOCAL PROGRAM WRT DB
N
!MSCP_PKT [.MX1, OPCODE] = OP_ELP;                    ! OPCODE = EXECUTE LOCAL PROGRAM
!MSCP_PKT [.MX1, L1] = $ascii'W';                     ! FILL IN PROGRAM NAME WITH ASCII LETTERS
!MSCP_PKT [.MX1, L2] = $ascii'R';
!MSCP_PKT [.MX1, L3] = $ascii'T';
!MSCP_PKT [.MX1, L4] = $ascii'D';
!MSCP_PKT [.MX1, L5] = $ascii'B';
!MSCP_PKT [.MX1, L6] = $ascii'N';
!MSCP_PKT [.MX1, MODIFY] = 1;                         ! STANDALONE MODIFIER
!ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;               ! CALL IT IMMEDIATE
!MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                    ! CALL ALL DUP CMDS SEQUENTIAL.              ZZZ
!DUPCOMMAND ();                                       ! SENDS AND RECEIVES THE COMMAND

!IF .CST_ADDR [ DUOFF + OF_DBN, DUPERROR] EQL 1            !status error
!THEN RETURN;                                         ! AND RETURN

!DO (MX1 = GET_PKT (.CCTLR))
!UNTIL (.MX1 GEQ 0);                      ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR

!        MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                ! PACKET SIZE               RECIEVE DATA
!        MSCP PKT [.MX1, OPCODE] = OP RCD;                ! OPCODE = RECEIVE DATA
```

```
!       MSCP_PKT [.MX1, BC LO] = 80;                     ! BYTE COUNT TO BE TRANSFERE. EQUALS 2 ***see pg 26 of DUP s
pec
!       MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
!       MSCP_PKT [.MX1, MODIFY] = 0;                     !
!ZZZ    MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequent al
!       MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.          ZZZ
!       DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND

!       IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
!          (.DUPPKT [DUPTYPE] NEQU 1) OR                 !dup type error
!          (.DUPPKT [DUPMESG] NEQU 6)
!       THEN
!               (D_FAIL = 1;                             !TELL HARD_ERROR IT WAS A DUP PROBLEM    ZZZ
!                HARD_ERROR ();                          !
!                D_FAIL = 0;                             !                                        ZZZ
!               CST_ADDR [.DUOFF + OF DBN, DUPERROR] = 1;       ! SET FLAG
!               RETURN;);                                ! NO POINT IN CONTINUING

!       DO (MX1 = GET_PKT (.CCTLR))
!       UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE  IF FAILURE LOOP PRG ERROR

!       MSCP_PKT [.MX1, MSGLEN] = SZ SEN;                ! PACKET SIZE                    SEND DATA
!       MSCP_PKT [.MX1, OPCODE] = OP_SDD;                ! OPCODE = SEND DATA
!       MSCP_PKT [.MX1, BC_LO] = 6;                      ! BYTE COUNT TO BE TRANSFERED EQUALS 6
!       MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
!       DUPPKT [DUPBF0] = .CST ADDR [.DUOFF, D_DISK_NUM];  !LOAD UNIT NUMBER (RDRX)
!       DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
!       DUPPKT [DUPBF2] = .S_PATTERN;                    ! LOAD PATTERN
!       MSCP_PKT [.MX1, MODIFY] = 0;                     !
!ZZZ    MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
!       MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.          ZZZ
!       DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND

!        IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1         ! status error
!        THEN RETURN;
!
!       DO (MX1 = GET_PKT (.CCTLR))
!       UNTIL (.MX1 GEQ 0);                     ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR

!       MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                ! PACKET SIZE                    RECEIVE DATA
!       MSCP_PKT [.MX1, OPCODE] = OP_RCD;                ! OPCODE = RECEIVE DATA
!       MSCP_PKT [.MX1, BC_LO] = 4;                      ! BYTE COUNT TO BE TRANSFERED EQUALS 4
!       MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
!       MSCP_PKT [.MX1, MODIFY] = 0;                     !
!ZZZ    MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
!       MSCP_PKT [.MX1, MSGTYP] = MT SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.          ZZZ
!       DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND


!       IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND     ZZZ
!          (.DUPPKT [DUPTYPE] EQL 3) AND                 !NO dup type error                       ZZZ
!          (.DUPMESG] EQL 3) AND                         !                                        ZZZ
!          (.DUPPKT [DUPBF1]  EQL 0)                     !AND A successful wr te code             ZZZ
                                                         !                                        ZZZ
```

```
;  1   !       THEN                                          !THEN                                   ZZZ
;  1   !           LEAVE DUP_WLOOP                            !I/O OK. EXIT RETRY LOOP.               ZZZ
;  1
;  1   !ZZZ ;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  1   !        ELSE                                                                                !ZZZ
;  1   !        IF (.DUPPKT [DUPBF0] EQL 50003) AND          !IF DBN  IS OUT OF RANGE              ZZZ
;  1   !            (.DUPPKT [DUPBF1]  EQL  2)                !WD 0 = 2 FOR OUT OF RANGE STATUS.    ZZZ
;  1   !       THEN                                          !THEN                                 ZZZ
;  1   !           LEAVE DUP_WLOOP                            !EXIT RETRY LOOP.                     ZZZ
;  1   !ZZZ ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  1
;  1   !        ELSE                                          !                                       ZZZ
;  1   !            BEGIN                                     !                                       ZZZ
;  1   !            TRYNUM = .TRYNUM + 1;                     !INCR ATTEMPT COUNT                     ZZZ
;  1   !            IF .TRYNUM  EQL .MAX TRY_COUNT            !IF IT FAILED ALL RETRIES, THEN         ZZZ
;  1   !            THEN                                      !REPORT THE ERROR.                      ZZZ
;  1   !                    (D_FAIL = 1;                     !TELL HARD_ERROR IT WAS A DUP PROBLEM  ZZZ
;  1   !                      HARD_ERROR ();                  !                                       ZZZ
;  1   !                      D_FAIL = 0;                     !                                       ZZZ
;  1   !                     CST ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;    ! SET FLAG                ZZZ
;  1   !                     RETURN;);                       ! NO POINT IN CONTINUING               ZZZ
;  1   !            END;                                      !                                       ZZZ
;  1   ! END;                                                 !END LARGE DO LOOP                      ZZZ
;  1   !                                                      !                                       ZZZ
;  1   !END;                                                  !END DUP WLOOP                          ZZZ
;  1
;  1
;  1   !DO (MX1 = GET_PKT (.CCTLR))
;  1   !UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE.
;  1   !
;  1   !T_ADDR [T BLK_WT] = .T_ADDR [T BLK WT] + 1;    !INCREMENT COUNTER IF A SUCCESS
;  1   !
;  1   !END;
```

```
!GLOBAL ROUTINE DUPREDDBN : NOVALUE =

!+
! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
! "REDDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB PROTOCAL IS USED TO
! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM READS A DIAGONOSTIC BLOCK (DBN)
! AND PLACES IT IN THE DUP BUFFER CALLED "DUPPKT".  IF AN ERROR OCCURS WHILE
! RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
!
!
!       IMPLICIT INPUTS:
!               CST_ADDR        CONTAINS THE CURRENT CONTROLLER STATUS TABLE
!               DUOFF           CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
!
!BEGIN
!LOCAL                                          !                                    ZZZ
!       TRYNUM : WORD,                          !                                    ZZZ
!       MAX_TRY_COUNT : word initial (9);       ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR ZZZ
                                                !                                    ZZZ
!LABEL                                          !                                    ZZZ
!       DUP_RLOOP;                              !START OR DUP READ RETRY LOOP        ZZZ


!PRINTX (DER12);
!T_ADDR [T_DBN RD] = .T_ADDR [T_DBN_RD] + 1;    ! INCREMENT # OF READS GIVEN

!TRYNUM = 0;                                    !ZERO TRY COUNTER                    ZZZ
!DUP_RLOOP:                                     !LABEL FOR LOOP EXCAPE ON GOOD READ  ZZZ
!BEGIN                                          !BEGIN DUP_RLOOP                     ZZZ
!INCR TRIES FROM 1 TO 10 DO                     !START TRYING DUP READS              ZZZ
!BEGIN                                          !BEGIN LARGE DO LOOP                 ZZZ
                                                !                                    ZZZ

! MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;             ! PACKET SIZE               EXECUTE REDDBN PROGRAM
! MSCP_PKT [.MX1, OPCODE] = OP_ELP;             ! OPCODE = EXECUTE LOCAL PROGRAM
! MSCP_PKT [.MX1, L1] = %ascii'R';              ! FILL IN PROGRAM NAME WITH ASCII LETTERS
! MSCP_PKT [.MX1, L2] = %ascii'E';
! MSCP_PKT [.MX1, L3] = %ascii'D';
! MSCP_PKT [.MX1, L4] = %ascii'D';
! MSCP_PKT [.MX1, L5] = %ascii'B';
! MSCP_PKT [.MX1, L6] = %ascii'N';
! MSCP_PKT [.MX1, MODIFY] = 1;                  ! STANDALONE MODIFIER
!ZZZ  MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                ! CALL IT IMMEDIATE
! MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;             ! CALL ALL DUP CMDS SEQUENTIAL.                ZZZ
! DUPCOMMAND ();                                        ! SENDS AND RECEIVES THE COMMAND
!
! IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1        !status error
! THEN RETURN;

! DO (MX1 = GET_PKT (.CCTLR))
! UNTIL (.MX1 GEQ 0);                   ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
```

```
:  1    !       MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                    ! PACKET SIZE                    RECIEVE DATA
;  1    !       MSCP_PKT [.MX1, OPCODE] = OP_RCD;                    ! OPCODE = RECEIVE DATA
;  1    !       MSCP_PKT [.MX1, BC_LO] = 80;                         ! BYTE COUNT TO BE TRANSFERED EQUALS 2 *****see pg 26 DUP sp
ec
;  1    !       MSCP_PKT [.MX1, BUF_0] = DUPPKT;                     ! LOAD DESCRIBTOR BUFFER
;  1    !       MSCP_PKT [.MX1, MODIFY] = 0;                         !
;  1    !ZZZ    MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                   ! CALL IT sequential
;  1    !       MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                    ! CALL ALL DUP CMDS SEQUENTIAL.              ZZZ
;  1    !       DUPCOMMAND ();                                       ! SENDS AND RECEIVES THE COMMAND
;  1
;  1    !       IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;  1    !          (.DUPPKT [DUPTYPE] NEQU 1) OR                     !dup type error
;  1    !          (.DUPPKT [DUPMESG] NEQU 5)
;  1    !       THEN
;  1    !               (D_FAIL = 1;                                 !TELL HARD ERROR IT WAS A DUP PROBLEM   ZZZ
;  1    !                 HARD_ERROR ();
;  1    !                 D_FAIL = 0;                                !                                       ZZZ
;  1    !                CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;        ! SET FLAG
;  1    !                RETURN;);                                   ! NO POINT IN CONTINUING
;  1
;  1    !       DO (MX1 = GET_PKT (.CCTLR))
;  1    !       UNTIL (.MX1 GEQ 0);                                  ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  1
;  1    !       MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;                    ! PACKET SIZE                    SEND DATA
;  1    !       MSCP_PKT [.MX1, OPCODE] = OP_SDD;                    ! OPCODE = SEND DATA
;  1    !       MSCP_PKT [.MX1, BC_LO] = 4;                          ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;  1    !       MSCP_PKT [.MX1, BUF_0] = DUPPKT;                     ! LOAD DESCRIBTOR BUFFER
;  1    !       DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM];  ! LOAD UNIT NUMBER (RDRX)
;  1    !       DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF DBN, D_DBN];  ! LOAD DBN NUMBER
;  1    !       MSCP PKT [.MX1, MODIFY] = 0;                         !
;  1    !!ZZZ   MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                   ! CALL IT sequential
;  1    !       MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                    ! CALL ALL DUP CMDS SEQUENTIAL.              ZZZ
;  1    !       DUPCOMMAND ();                                       ! SENDS AND RECEIVES THE COMMAND
;  1
;  1    !         IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1        !status error
;  1    !         THEN RETURN;
;  1
;  1    !       DO (MX1 = GET_PKT (.CCTLR))
;  1    !       UNTIL (.MX1 GEQ 0);                         ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  1
;  1    !       MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                    ! PACKET SIZE                    RECEIVE DATA
;  1    !       MSCP_PKT [.MX1, OPCODE] = OP_RCD;                    ! OPCODE = GET DUST STATUS
;  1    !       MSCP_PKT [.MX1, BC_LO] = 514;                        ! BYTE COUNT TO BE TRANSFERED EQUALS 512
;  1    !       MSCP_PKT [.MX1, BUF_0] = DUPPKT;                     ! LOAD DESCRIBTOR BUFFER
;  1    !       MSCP_PKT [.MX1, MODIFY] = 0;                         !
;  1    !ZZZ    MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                   ! CALL IT sequential
;  1    !       MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                    ! CALL ALL DUP CMDS SEQUENTIAL.              ZZZ
;  1    !       DUPCOMMAND ();                                       ! SENDS AND RECEIVES THE COMMAND
;  1
;  1    !       IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND        ZZZ
;  1    !          (.DUPPKT [DUPTYPE] EQL 6) AND                     !NO dup type error         ZZZ
;  1    !          (.DUPPKT [DUPMESG] EQL 2)                         !                          ZZZ
;  1    !       THEN                                                 !THEN                      ZZZ
;  1    !           LEAVE DUP RLOOP                                  !I/O OK. EXIT RETRY LOOP.  ZZZ
```

```

!ZZZ ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
!          ELSE                                                            !ZZZ
!          IF (.DUPPKT [DUPBF0] EQL 50003) AND              !IF DBN  IS OUT OF RANGE          ZZZ
!!             (.DUPPKT [DUPBF1]  EQL  2)                    !WD 0 = 2 FOR OUT OF RANGE STATUS.  ZZZ
!         THEN                                              !THEN                             ZZZ
!             LEAVE DUP_RLOOP                                !EXIT RETRY LOOP.                  ZZZ
!ZZZ ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

!          ELSE                                              !                                 ZZZ
!             BEGIN                                          !                                 ZZZ
!             TRYNUM = .TRYNUM + 1;                          !INCR ATTEMPT COUNT               ZZZ
!             IF .TRYNUM  EQL .MAX_TRY COUNT                 !IF IT FAILED ALL RETRIES, THEN   ZZZ
!             THEN                                           !REPORT THE ERROR.                ZZZ
!                    (D_FAIL = 1;                           !TELL HARD_ERROR IT WAS A DUP PROBLEM  ZZZ
!                      HARD_ERROR ();                        !                                 ZZZ
!                      D_FAIL = 0;                           !                                 ZZZ
!                     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;    ! SET FLAG                 ZZZ
!                     RETURN;);                             ! NO POINT IN CONTINUING          ZZZ
!             END;                                           !                                 ZZZ
! END;                                                       !END LARGE DO LOOP                ZZZ
                                                             !                                 ZZZ
!END;                                                        !END DUP RLOOP                    ZZZ

!DO (MX1 = GET_PKT (.CCTLR))
!UNTIL (.MX1 GEQ 0);                      ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR

!T_ADDR [T_BLK_RD] = .T_ADDR [T BLK_RD] + 1;           !IF DUP NO ERROR THEN INCREMENT COUNTER

!END;

```

```
!
!GLOBAL ROUTINE DUPCOMMAND : NOVALUE =

!!+
! THIS ROUTINE IS CALLED BY DUP TO PROCESS COMMANDS.
! THE COMMAND ENVOLOPES ARE FILLED IN DUP ROUTINES IN THE "MX1" INDEX.
! WITH THE INDEX THIS ROUTINE SENDS THE COMMAND, WAITS FOR A
! RESPONSES AND THEN PROCESSES THE RETURN PACKET.
!-
!BEGIN
!PRINTX (DER13);

!MSCP_PKT [.MX1, CREDITS] = 0;              ! DUP DOES NOT USE THE CREDIT SYSTEM
!MSCP_PKT [.MX1, CONNID] = CID_DUP;         ! MAKE PACKAGE EQUAL A DUP COMMAND
!MSCP_PKT [.MX1, DK_NUM] = 0;               ! DISK NUMBER (NOT APPLICABLE)
!
!IF SEND (.MX1) EQLU FAILURE                ! ATTEMPT SEND; IF CTLR IS OFFLINE
!THEN
!       BEGIN
!       PUT_PKT (.MX1);
!       MX1 = -1;                           ! RETURN ENVELOPE TO POOL
!       CST_ADDR [.DUOFF + OF DBN, DUPERROR] = 1;
!!      PRINTF (DBM112);                    ! "DUP: PKT NOT AVAILABLE"  ZZZ
!       END

!ELSE
!       do
!               begin
!               BREAK;                              ! BREAK FOR ACT
!               PROC_RETPKT ();                             ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
!               end
!       until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or  ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
!       (.EOP_FLAG eql true);                       ! or end of pass caused by error
!END;
```

```
!GLOBAL ROUTINE DUPIDLE : NOVALUE =
!+
! THIS ROUTINE IS CALLED BY DUP ROUTINE TO INSURE THAT THE CONTROLLER
! IS NOT IN A ACTIVE STATE. IF CALLED AND THE CONTROLLER IS IN AN ACTIVE
! STATE THE CONTROLLER WILL GIVE AN ABORT COMMAND WHICH SHOULD KILL THE
! CURRENT JOB OR LOCAL PROGRAM.
!-
!BEGIN
!CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 0;                  !CLEAR DUP ERROR FLAG;
!
!MSCP_PKT [.MX1, MSGLEN] = SZ_GDS;            ! PACKET SIZE                         GET DUST STATUS
!MSCP_PKT [.MX1, OPCODE] = OP_GDS;            ! OPCODE = GET DUST STATUS
!MSCP_PKT [.MX1, MODIFY] = 0;                 !
!!ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                 ! CALL IT IMMEDIATE
!MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;            ! CALL ALL DUP CMDS SEQUENTIAL.              ZZZ
!DUPCOMMAND ();                               ! SENDS AND RECEIVES THE COMMAND
!                                             ! GDS ONLY RETURNS SUCCESS or it don't return

!DO (MX1 = GET_PKT (.CCTLR))
!UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
!
!if .CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] neq IDLE         ! if not in idle state then abort the program
!then
!     begin
!     MSCP_PKT [.MX1, MSGLEN] = SZ_ABT;                    ! PACKET SIZE                    ABORT CMD
!     MSCP_PKT [.MX1, OPCODE] = OP_ABT;                    ! OPCODE = ABORT PROGRAM
!     MSCP_PKT [.MX1, MODIFY] = 0;                         !
!!ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                             ! CALL IT IMMEDIATE
!     MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                    ! CALL ALL DUP CMDS SEQUENTIAL.          ZZZ
!     DUPCOMMAND ();                                       ! SENDS AND RECEIVES THE COMMAND
!                                                          !ONLY ERROR IS already in idle state
!     DO (MX1 = GET_PKT (.CCTLR))
!     UNTIL (.MX1 GEQ 0);                                  ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERRCR
!     end;
!end;
```

```
GLOBAL routine QIO_LBN : novalue =

!+
!       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE LOGICAL BLOCK NUMBER TO
!       BE USED FOR THE CURRENT QIO OR QIO PAIR.
!
!       IF THE OPERATOR CHOSE THE RANDOM SEEK MODE OPTION, THEN THE LBN IS
!       RANDOMLY CHOSEN WITHIN THE SPECIFIED LIMITS FOR THE LBN.
!       OTHERWISE, THE NEXT SEQUENTIAL LBN IS DERIVED FROM THE BLOCK SEQUENCE
!       TABLE (BST).
!
!       IMPLICIT INPUTS:
!               L$LUN   CURRENT (DIAGNOSTIC SUPERVIOR) UNIT NUMBER
!
!       IMPLICIT OUTPUTS:
!               THE LBN IS LOADED INTO ONE OR BOTH MSCP PACKETS.
!-

    begin

    own
        LBN0_SAVE : word initial (0),                   !LO LBN SELECTED IN PREVIOUS PASS

        LBN1_SAVE : word initial (0);                   !HI LBN SELECTED IN PREVIOUS PASS

    local
        S0_TEMP : word,                                 ! TEMPORARY STORAGE FOR START LBN LO

        S1_TEMP : word,                                 ! TEMPORARY STORAGE FOR START LBN HI

        E0_TEMP : word,                                 ! TEMPORARY STORAGE FOR END LBN LO

        E1_TEMP : word,                                 ! TEMPORARY STORAGE FOR END LBN HI

        ADD0_LBN : word,                                ! TEMPORARY STORAGE USED FOR COMPUTING DESIRED LBN L

        ADD1_LBN : word,                                ! TEMPORARY STORAGE USED FOR COMPUTING DESIRED LBN H

        LBN0 : word,                                    ! LOGICAL BLOCK NUMBER LO

        LBN1 : word,                                    ! LOGICAL BLOCK NUMBER HI

        WINCHESTER : byte initial (byte (TRUE));        ! FLAG TO INDICATE WINCHESTER DISK SELECTED

    label
        FIND_LBN;

    S0_TEMP = .CST_ADDR [.CUOFF + OF_BEG, D_BEG0];      ! STARTING LBN LO

    S1_TEMP = .CST_ADDR [.CUOFF + OF_BEG1, D_BEG1];     ! STARTING LBN HI

    E0_TEMP = .CST_ADDR [.CUOFF + OF_END, D_END0];      ! ENDING LBN LO

    E1_TEMP = .CST_ADDR [.CUOFF + OF_END1, D_END1];     ! ENDING LBN HI

    FIND_LBN:
        begin                                           !BEGIN A.

        if (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and
           (BIT_TST (SWP_FLAGS, SWF_FER)) and
           (.MAD1 [OPCODE] eql OP_WRT) and
           (.FORCED_ERROR)
        then
            begin
            LBN0= .FER0_LBN;                            ! IF "FORCED ERROR" DETECTED, REWRITE ERROR LBN LO
```

```
                LBN1= .FER1_LBN;                                ! IF "FORCED ERROR" DETECTED, REWRITE ERROR LBN HI
                leave FIND_LBN;
                end;

            if .CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql REMOVABLE
            then
                WINCHESTER = FALSE;

            if BIT_TST (SWP_FLAGS, SWF_RDM)                     ! IF RANDOM SEEK MODE
            then

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                                                                                        !
!     NOTE:  UNOMMENT THE 2 LINES AT "IF WINCHESTER", AND DELETE "IF 1 EQLU 0" TO        !
!            REDUCE SEEKS ON THE WINCHESTERS BY HALF.                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

                begin
                IF 1 EQLU 0                                     !COMMENT OUT FOR REDUCED SEEKS

!ZZZ            if (.WINCHESTER) and
!ZZZ               (((.RANDOM [0] and %o'077777') mod (100)) lequ 49)
                then
                    BEGIN                                       !
                    LBN0 = .LBN0_SAVE;                          ! REDUCE SEEKS ON RDs by 50%
                    LBN1 = .LBN1_SAVE;                          ! REDUCE SEEKS ON RDs by 50%
                    END                                         !
                else

                    begin
                    RANDY ();                                   !GET A 32 BIT RANDOM NUMBER
                    IF (.RNDY1 GTRU .E1_TEMP) OR                !IF NUMBER GREATER THAN MAX
                       ((.RNDY1 EQLU .E1_TEMP) AND              !
                       (.RNDY0 GTRU .E0_TEMP))                  !
                    THEN                                        !
                       BEGIN                                    !THEN MASK IT WITH HI LIMIT
                       RNDY1 = .RNDY1 AND .E1_TEMP;             !
                       R[illegible]    = .RNDY0 AND .E0_TEMP;   !
                       END;                                     !
                                                                !
                    IF (.RNDY1 LSSU .S1_TEMP) OR                !IF NUMBER LESS THAN MIN
                       ((.RNDY1 EQLU .S1_TEMP) AND              !
                       (.RNDY0 LSSU .S0_TEMP))                  !
                    THEN                                        !
                       BEGIN                                    !THEN MASK IT WITH LO LIMIT
                       RNDY1 = .RNDY1 AND .S1_TEMP;             !
                       RNDY0 = .RNDY0 AND .S0_TEMP;             !
```

```
;    4824  5            LBN0 = .RNDY0;                                  !LO HALF
     ZZZ
;    4825  5            LBN1 = .RNDY1;                                  !HI HALF
     ZZZ
```

```
;      4                    end;
;      4                 end
;      4
;      3           ELSE                                                      !
  ZZZ
;      4           begin                                                     ! ELSE - SEQUENTIAL LBN MODE  (BEGIN A)
  ZZZ
;      4              LBNO = .BST [.L$LUN, LO_WRD];                          ! GET LBN FROM BST (LO WORD)
  ZZZ
;      4              LBN1 = .BST [.L$LUN, HI_WRD];                          ! GET LBN FROM BST (HI WORD)
  ZZZ
;      4
;      4              IF .TRK_SGN [.L$LUN] EQLU 1                            !IF WE WANT SERIAL INCREMENT
  ZZZ
;      4              THEN                                                   !
  ZZZ
;      5              BEGIN                                                  !(BEGIN B)
  ZZZ
;      5                 IF .BST [.L$LUN, LO_WRD] EQLU %o'177777'            !IF OVERFLOW FROM LO WD TO HI WD
  ZZZ
;      5                 THEN                                                !
  ZZZ
;      6                 BEGIN                                               !
  ZZZ
;      6                    BST [.L$LUN, LO_WRD] = 0;                        'ZERO LO WORD
  ZZZ
;      6                    BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] + 1,        !INCREMENT HI WORD
  ZZZ
;      6                 END                                                 !
  ZZZ
;      5                 ELSE                                                        !OTHERWISE JUST INCR LO WORD
  ZZZ
;      5                    BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] + 1;        !
  ZZZ
;      5
;      5                                                          !NOW TAKE CARE OF OVERFLOW WHILE INCREMENTIN
G.  ZZZ
;      6                 IF (.BST [.L$LUN, HI_WRD] GTRU .E1_TEMP)         !IF LBN1 OVER HI LIMIT
  ZZZ
;      7                   OR ((.BST [.L$LUN, HI_WRD] EQLU .E1_TEMP)      !OR LBN1 EQUALS HI LIMIT AND LBNO IS OVER LI
MIT ZZZ
;      6                   AND (.BST [.L$LUN, LO_WRD] GTRU .E0_TEMP))     !
  ZZZ
;      5                 THEN                                             !
  ZZZ
;      6                    BEGIN                                         !
  ZZZ
;      6                    BST [.L$LUN, LO_WRD] = .E0_TEMP;              !THEN SET HI LIMITS
  ZZZ
;      6                    BST [.L$LUN, HI_WRD] = .E1_TEMP;              !INTO BST FOR NEXT TIME
  ZZZ
;      6                    TRK_SGN [.L$LUN] = - 1;                       !AND REVERSE D'.ECTION
  ZZZ
;      5                    END;                                          !
  ZZZ
;      5              END                                                 !(END B)
  ZZZ
;      5
;      5
;      5
;      4              ELSE                                                  !IF WE WANT SERIAL DECREMENT
  ZZZ
;      5              BEGIN                                                 !(BEGIN C)
  ZZZ
;      5                 IF .BST [.L$LUN, LO_WRD] EQLU 0                    !IF NEED TO BORROW FROM HI WD
  ZZZ
;      5                 THEN                                               !
  ZZZ
;      6                 BEGIN                                              !
  ZZZ
;      6                    BST [.L$LUN, LO_WRD] = %o'177777';              !LO WORD
  ZZZ
```

```
ELSE                                                        !OTHERWISE JUST DECR LO WORD
    BST [.L$LUN, LO_WRD] = .BST [ L$LUN, LO WRD]   1;       !

                                                    !NOW TAKE CARE OF UNDERFLOW WHILE INCREMENTING.
IF (.BST [.L$LUN, HI_WRD] LSS  .S1_TEMP)            !IF LBN1 IS BELOW LIMIT
  OR ((.BST [.L$LUN, HI WRD] EQLU .S1_TEMP)         !OR LBN1 EQUALS LO LIMIT AND LBN0 IS BELOW
  AND (.BST [.L$LUN, LO_WRD] LSSU .S0_TEMP))        !
THEN                                                !
   BEGIN                                            !
   BST [.L$LUN, LO_WRD] = .S0_TEMP;                 !THEN SET LO LIMITS
   BST [.L$LUN, HI_WRD] = .S1_TEMP;                 !INTO BST FOR NEXT TIME
```

```
;      4879  6                   TRK_SGN [.L$LUN] = + 1;                          !AND REVERSE DIRECTION
       ZZZ
;      4880  5                   END;                                             !
       ZZZ
;      4881  4              END;                                             !
       ZZZ
;      4882  3          END;                                             !END C.
       ZZZ
;      4883  3                                                           !
       ZZZ
;      4884  2        END;                                             !END A.
       ZZZ
;      4885  2
;      4886  2
;      4887  3        IF ((.S1_TEMP EQLU .E1_TEMP) AND (.S0_TEMP EQLU .E0_TEMP))       !IF START ADDR SAME AS END ADDR
       ZZZ
;      4888  2        THEN                                             !JUST USE THE START ADDRESS.
       ZZZ
;      4889  3          BEGIN                                          !
       ZZZ
;      4890  3          LBN0 = .S0 TEMP;                               !
       ZZZ
;      4891  3          LBN1 = .S1_TEMP;                               !
       ZZZ
;      4892  2          END;                                           !
       ZZZ
;      4893  2
;      4894  2        MAD1 [LBN L] = .LBN0;                    ! LOAD LBN INTO 1ST PACKET
       ZZZ
;      4895  2        MAD1 [LBN H] = .LBN1;                    ! LOAD LBN INTO 1ST PACKET
       ZZZ
;      4896  2
;      4897  2        if .MX2 geq 0                            ! IF 2 QIOs
       ZZZ
;      4898  2        then
;      4899  2            MAD2 [LBN_L] = .LBN0;                ! LOAD LBN INTO 2ND PACKET
       ZZZ
;      4900  2            MAD2 [LBN H] = .LBN1;                ! LOAD LBN INTO 2ND PACKET
       ZZZ
;      4901  2
;      4902  2        LBN0_SAVE = .LBN0;                       ! SAVE FOR USE NEXT CYCLE IF NEEDED
       ZZZ
;      4903  2        LBN1_SAVE = .LBN1;                       ! SAVE FOR USE NEXT CYCLE IF NEEDED
       ZZZ
;      4904  2
;      4905  1        end;                                     ! ROUTINE QIO LBN


001276                                        .PSECT  $GGG$,  R0
001276  000000                    LBN0.SAVE:
                                              .WORD   0
001300  000000                    LBN1.SAVE:
                                              .WORD   0


                                              .SBTTL  QIO.LBN MULTI DRIVE TEST ROUTINES
014146                                        .PSECT  $CODE$,  R0

000000  004137  000000G           QIO.LBN::
                                              JSR     R1,$SAVE5                        ;                         4720
000004  024646                                CMP     (SP),(SP)
000006  112746  000001                        MOVB    #1,-(SP)                         ; *.WINCHESTER            4738
000012  013701  000000G                       MOV     CST.ADDR,R1                      ;                         4758
000016  013702  000000G                       MOV     CUOFF,R2
~~000022  010200                                MOV     R2,R0~~
000024  006300                                ASL     R0
000026  060100                                ADD     R1,R0
000030  016005  000002                        MOV     2(R0),R5                         ; *.S0.TEMP
000034  010200                                MOV     R2,R0                            ;                         4759
```

```
000042  016003  000004                  MOV     4(R0),R3            ; *,S1.TEMP
000046  010200                          MOV     R2,R0               ;                       4760
000050  006300                          ASL     R0
000052  060100                          ADD     R1,R0
000054  016046  000006                  MOV     6(R0),-(SP)         ; *,E0.TEMP
000060  010200                          MOV     R2,R0               ;                       4761
000062  006300                          ASL     R0
000064  060100                          ADD     R1,R0
000066  016004  000010                  MOV     10(R0),R4           ; *,E1.TEMP
000072  006302                          ASL     R2                  ;                       4766
000074  060102                          ADD     R1,R2
000076  132712  000020                  BITB    #20,(R2)
000102  001430                          BEQ     2$
000104  032737  004000  000000G         BIT     #4000,SWP.FLAGS     ;                       4767
000112  001421                          BEQ     1$
000114  013700  000114'                 MOV     MAD1,R0             ;                       4768
000120  126027  000022  000042          CMPB    22(R0),#42
000126  001013                          BNE     1$
000130  132737  000001  000000G         BITB    #1,FORCED.ERROR     ;                       4769
000136  001407                          BEQ     1$
000140  013766  000000G 000004          MOV     FER0.LBN,4(SP)      ; *,LBN0                4772
000146  013766  000000G 000006          MOV     FER1.LBN,6(SP)      ; *,LBN1                4773
000154  000544                          BR      16$                 ;                       4771
000156  132712  000020          1$:     BITB    #20,(R2)            ;                       4777
000162  001002                          BNE     3$
000164  105066  000002          2$:     CLRB    2(SP)               ; WINCHESTER            4779
000170  032737  000002  000000G 3$:     BIT     #2,SWP.FLAGS        ;                       4781
000176  001447                          BEQ     8$
000200  004737  011744                  JSR     PC,RANDY            ;                       4805
000204  023704  000130'                 CMP     RNDY1,R4            ; *,E1.TEMP             4806
000210  101004                          BHI     4$
000212  001013                          BNE     5$                  ;                       4807
000214  023716  000126'                 CMP     RNDY0,(SP)          ; *,E0.TEMP             4808
000220  101410                          BLOS    5$
000222  010400                  4$:     MOV     R4,R0               ; E1.TEMP,*             4811
000224  005100                          COM     R0
000226  040037  000130                  BIC     R0,RNDY1
000232  011600                          MOV     (SP),R0             ; E0.TEMP,*             4812
000234  005100                          COM     R0
000236  040037  000126'                 BIC     R0,RNDY0
000242  023703  000130'         5$:     CMP     RNDY1,R3            ; *,S1.TEMP             4815
000246  103404                          BLO     6$
000250  001013                          BNE     7$                  ;                       4816
000252  023705  000126'                 CMP     RNDY0,R5            ; *,S0.TEMP             4817
000256  103010                          BHIS    7$
000260  010300                  6$:     MOV     R3,R0               ; S1.TEMP,*             4820
000262  005100                          COM     R0
000264  040037  000130'                 BIC     R0,RNDY1
000270  010500                          MOV     R5,R0               ; S0.TEMP,*             4821
000272  005100                          COM     R0
000274  040037  000126                  BIC     R0,RNDY0
000300  013766  000126' 000004  7$:     MOV     RNDY0,4(SP)         ; *,LBN0                4824
000306  013766  000130  000006          MOV     RNDY1,6(SP)         ; *,LBN1                4825
```

```
000314  000464                             BR      16$
000316  013702  000000G          8$:       MOV     L$LUN,R2                 ;                          4781
000322  010201                             MOV     R2,R1                    ;                          4831
000324  006301                             ASL     R1
000326  006301                             ASL     R1
000330  012700  000000G                    MOV     #BST,R0
000334  060100                             ADD     R1,R0
000336  011066  000004                     MOV     (R0),4(SP)               ; *,LBN0
000342  062701  000002G                    ADD     #BST+2,R1                ;                          4832
000346  011166  000006                     MOV     (R1),6(SP)               ; *,LBN1
000352  062702  000000G                    ADD     #TRK.SGN,R2              ;                          4834
000356  121227  000001                     CMPB    (R2),#1
000362  001021                             BNE     12$
000364  021027  177777                     CMP     (R0),#-1                 ;                          4837
000370  001003                             BNE     9$
000372  005010                             CLR     (R0)                     ;                          4840
000374  005211                             INC     (R1)                     ;                          4841
000376  000401                             BR      10$                      ;                          4837
000400  005210                   9$:       INC     (R0)                     ;                          4844
000402  021104                   10$:      CMP     (R1),R4                  ; *,E1.TEMP                4847
000404  101003                             BHI     11$
000406  001027                             BNE     16$                      ;                          4848
000410  021016                             CMP     (R0),(SP)                ; *,E0.TEMP                4849
000412  101425                             BLOS    16$
000414  011610                   11$:      MOV     (SP),(R0)                ; E0.TEMP,*                4852
000416  010411                             MOV     R4,(R1)                  ; E1.TEMP,*                4853
000420  112712  000377                     MOVB    #377,(R2)                ;                          4854
000424  000420                             BR      16$                      ;                          4834
000426  005710                   12$:      TST     (R0)                     ;                          4862
000430  001004                             BNE     13$
000432  012710  177777                     MOV     #-1,(R0)                 ;                          4865
000436  005311                             DEC     (R1)                     ;                          4866
000440  000401                             BR      14$                      ;                          4862
000442  005310                   13$:      DEC     (R0)                     ;                          4869
000444  021103                   14$:      CMP     (R1),R3                  ; *,S1.TEMP                4872
000446  002403                             BLT     15$
000450  001006                             BNE     16$                      ;                          4873
000452  021005                             CMP     (R0),R5                  ; *,S0.TEMP                4874
000454  103004                             BHIS    16$
000456  010510                   15$:      MOV     R5,(R0)                  ; S0.TEMP,*                4877
000460  010311                             MOV     R3,(R1)                  ; S1.TEMP,*                4878
000462  112712  000001                     MOVB    #1,(R2)                  ;                          4879
000466  020304                   16$:      CMP     R3,R4                    ; S1.TEMP,E1.TEMP          4887
000470  001006                             BNE     17$
000472  020516                             CMP     R5,(SP)                  ; S0.TEMP,E0.TEMP
000474  001004                             BNE     17$
000476  010566  000004                     MOV     R5,4(SP)                 ; S0.TEMP,LBN0             4890
000502  010366  000006                     MOV     R3,6(SP)                 ; S1.TEMP,LBN1             4891
000506  013700  000114'          17$:      MOV     MAD1,R0                  ;                          4894
000512  016660  000004  000046             MOV     4(SP),46(R0)             ; LBN0,*
000520  016660  000006  000050             MOV     6(SP),50(R0)             ; LBN1,*                   4895
000526  005737  000112'                    TST     MX2                      ;                          4897
000532  002405                             BLT     18$
```

```
000534  013700  000116'                      MOV     MAD2,R0              ;                 4899
000540  016660  000004  000046               MOV     4(SP),46(R0)         ; LBN0,*
000546  013700  000116'             18$:     MOV     MAD2,R0              ;                 4900
000552  016660  000006  000050               MOV     6(SP),50(R0)         ; LBN1,*
000560  016637  000004  001276'              MOV     4(SP),LBN0.SAVE      ; LBN0,*          4902
000566  016637  000006  001300'              MOV     6(SP),LBN1.SAVE      ; LBN1,*          4903
000574  062706  000010                       ADD     #10,SP               ;                 4720
000600  000207                               RTS     PC

; Routine Size:  193 words,     Routine Base:  $CODE$ + 14146
; Maximum stack depth per invocation:  11 words


;    4906  1
```

```
;  1    !!ZZZ routine QIO_SIZE : novalue =
;  1    GLOBAL ROUTINE QIO_SIZE : NOVALUE =
;  1
;  1    !+
;  1    !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O TRANSFER BYTE COUNT
;  1    !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE BYTE COUNT IS
;  1    !       DETERMINED BY A RANDOM NUMBER, AND WILL ALWAYS FALL BETWEEN 1 AND THE
;  1    !       I/O BUFFER SIZE (BYTS_PER_QIO). It is assumed that BYTS_PER_QIO will
;  1    !       never be larger than one binary word or  65000 bytes.
;  1    !
;  1    !       IMPLICIT OUTPUTS:
;  1    !               THE BYTE COUNT IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;  1    !-
;  1
;  2        begin
;  2
;  2        local
;  2            SIZE : word,                                                    ! BYTE COUNT
;  2            BLOCKS_LEFT : word;                                             ! REMAINING BLOCKS LEFT
;  2
;  2        SIZE = ((.RANDOM [4] and %o'077777') mod (.BYTS_PER_QIO + 1)) and %o'177760';!GET BYTE COUNT FROM RANDOM NUMBER
;  2
;  2        if .SIZE eql 0
;  2        then
;  2            SIZE = 16;
;  2
;  2        if .CST_ADDR [.CUOFF + 4, D_END1] gtru .MAD1 [LBN_H]
;  2        then BLOCKS_LEFT = %o'177777'                                               ! find
;  2        else BLOCKS_LEFT = .CST_ADDR [.CUOFF + 3, D_END0] - .MAD1 [LBN_L] + 1;      ! REMAINING BLOCK COUNT
;  2
;  2        if ((.SIZE + BYTES_PER_SECT - 1) / BYTES_PER_SECT) gtru .BLOCKS_LEFT   ! IF BLOCK COUNT NOT ENOUGH
;  2        then
;  2            SIZE = .BLOCKS_LEFT * BYTES_PER_SECT;                               ! ADJUST BYTE COUNT DOWN
;  2
;  2        MAD1 [BC_LO] = .SIZE;                                                   ! LOAD SIZE INTO 1ST MSCP PACKET
;  2
;  2        if .MX2 geq 0                                                           ! IF 2 QIOS
;  2        then
;  2            MAD2 [BC_LO] = .SIZE;                                               ! LOAD SIZE INTO 2ND MSCP PACKET
;  2
;  1        end;                                                                    ! ROUTINE QIO_SIZE


                                            .SBTTL  QIO.SIZE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G             QIO.SIZE::
                                            JSR     R1,$SAVE3
000004  013746  000010G                     MOV     RANDOM+10,-(SP)             ;                   4908
000010  042716  100000                      BIC     #100000,(SP)                ;                   4927
000014  013746  000000G                     MOV     BYTS.PER.QIO,-(SP)
000020  005216                              INC     (SP)
000022  004737  000000G                     JSR     PC,BL$MOD
000026  010003                              MOV     R0,R3                       ; *.SIZE
000030  042703  000017                      BIC     #17,R3                      ; *.SIZE
```

```
000034  001002                            BNE     1$
000036  012703  000020                    MOV     #20,R3                    ;                       4929
000042  013700  000000G           1$:     MOV     CUOFF,R0                  ; *.SIZE                4931
000046  006300                            ASL     R0                        ;                       4933
000050  063700  000000G                   ADD     CST.ADDR,R0
000054  013701  000114'                   MOV     MAD1,R1
000060  026061  000010  000050            CMP     10(R0),50(R1)
000066  101403                            BLOS    2$
000070  012702  177777                    MOV     #-1,R2
000074  000413                            BR      3$                        ; *.BLOCKS.LEFT         4934
000076  013700  000000G           2$:     MOV     CUOFF,R0                  ;                       4933
000102  006300                            ASL     R0                        ;                       4935
000104  063700  000000G                   ADD     CST.ADDR,R0
000110  016000  000006                    MOV     6(R0),R0
000114  166100  000046                    SUB     46(R1),R0
000120  010002                            MOV     R0,R2
000122  005202                            INC     R2                        ; *.BLOCKS.LEFT
000124  010316                    3$:     MOV     R3,(SP)                   ; BLOCKS.LEFT
000126  062716  000777                    ADD     #777,(SP)                 ; SIZE,*                4937
000132  012746  001000                    MOV     #1000,-(SP)
000136  004737  000000G                   JSR     PC,BL$DIV
000142  005726                            TST     (SP)+
000144  020002                            CMP     R0,R2
000146  101405                            BLOS    4$                        ; *.BLOCKS.LEFT
000150  010200                            MOV     R2,R0
000152  000300                            SWAB    R0                        ; BLOCKS.LEFT,*         4939
000154  105000                            CLRB    R0
000156  006300                            ASL     R0
000160  010003                            MOV     R0,R3
000162  010361  000026            4$:     MOV     R3,26(R1)                 ; *.SIZE
000166  005737  000112'                   TST     MX2                       ; SIZE,*                4941
000172  002404                            BLT     5$                        ;                       4943
000174  013700  000116'                   MOV     MAD2,R0
000200  010360  000026                    MOV     R3,26(R0)                 ;                       4945
000204  022626                    5$:     CMP     (SP)+,(SP)+               ; SIZE,*
000206  000207                            RTS     PC                        ;                       4921
                                                                            ;                       4908

; Routine Size:  68 words,      Routine Base:  $CODE$ + 14750
; Maximum stack depth per invocation:  8 words
```

```
1	GLOBAL routine FILL_BUFF : novalue =
1
1	!+
1	!	THIS ROUTINE IS CALLED BY QIO_GEN TO LOAD THE I/O BUFFER DESCRIBED IN
1	!	THE FIRST MSCP PACKET WITH THE APPROPRIATE DATA PATTERN.
1	!
1	!	THE DATA PATTERN TO BE SELECTED IS BASED ON THE FOLLOWING ALGORITHM:
1	!
1	!		IF THE OPERATOR DEFINED A DATA PATTERN
1	!		THEN
1	!			SELECT IT
1	!		ELSE
1	!			GET DATA PATTERN NUMBER FROM SW P-TABLE
1	!			IF DATA PATTERN NUMBER = 0
1	!			THEN
1	!				GET DATA PATTERN NUMBER FROM THE UNIT'S ENTRY
1	!				IN THE DATA PATTERN SEQUENCE TABLE (DPST)
1	!
1	!	NOTE THAT PATTERN # 1 CONSISTS OF RANDOM NUMBERS, AND PATTERNS # 17 -
1	!	21 USE THE ACTUAL LBN OF THE WRITE REQUEST
1	!
1	!	IMPLICIT INPUTS:
1	!		L$LUN - CURRENT (DRS) UNIT NUMBER
1	!-
1
2	    begin
2
2	    local
2		DP_NUM : word,				! DATA PATTERN NUMBER SELECTED
2		DP_ADDR,				! ADDR OF DATA PATTERN (LENGTH)
2		IOB_ADDR,				! I/O BUFFER ADDRESS (DESTINATION)
2		SRC_ADDR,				! WORKING SOURCE ADDRESS
2		COUNT : word,				! NO. OF WORDS IN DATA PATTERN
2		CUR_PRIORITY : word;			! MMM
2
3	    if BIT_TST (SWP_FLAGS, SWF UDP)		! IF USER DEFINED A DATA PATTERN
2	    then
2		DP_ADDR = SWP_UCNT			! SELECT IT
2	    else
3		begin
3
3		if .SWP_DPAT neq 0			! IF USER SELECTED A PRE-DEFINED DATA PATTERN
3		then
3		    DP_NUM = .SWP_DPAT			! SELECT IT
3		else
4		    begin
4		    DP_NUM = .DPST [.L$LUN];		! GET PATTERN NUMBER FROM SEQUENCE TABLE
4		    DPST [.L$LUN] = .DPST [.L$LUN] + 1;	! ADVANCE TO NEXT PATTERN NUMBER
4
4		    if .DPST [.L$LUN] gtru DP_CNT	! CHECK FOR HIGH LIMIT
4		    then
4			DPST [.L$LUN] = 1;
4
```

```
;  3                 end;
;  3
;  3             DP_ADDR = .DPA_TBL [.DP_NUM - 1];                      ! ADDRESS OF DATA PATTERN (COUNT)
;  3
;  3             if .DP NUM gequ 17
;  3             then
;  3
;  3                 if .DP_NUM                                         ! CHECK MACRO (IF PATTERN 17, 19, OR 21)
;  3                 then
;  3                     (.DP ADDR + 2) = .MAD1 [LBN_L]                 ! LOAD LBN INTO FIRST WORD OF PATTERN
;  3                 else
;  3                     (.DP_ADDR + 4) = .MAD1 [LBN_L];                ! LOAD LBN INTO SECOND WORD OF PATTERN
;  2             end;
;  2
;  2         IOB_ADDR = .MAD1 [BUF_0];                                  ! I/O BUFFER ADDRESS
;  2         COUNT = ..DP_ADDR;                                         ! NO. OF WORDS IN DATA PATTERN
;  2         SRC_ADDR = .DP_ADDR + 2;                                   ! START OF THE ACTUAL DATA PATTERN
;  2
;  2         incr N from 1 to ((.MAD1 [BC_LO] + 1) / 2) do              ! FOR EACH WORD IN THIS WRITE REQUEST
;  3             begin
;  3             .IOB_ADDR = ..SRC_ADDR;                                ! MOVE 1 WORD
;  3             IOB_ADDR = .IOB_ADDR + 2;                              ! ADVANCE DESTINATION ADDRESS
;  3             SRC_ADDR = .SRC_ADDR + 2;                              ! ADVANCE SOURCE ADDRESS
;  3             COUNT = .COUNT - 1;                                    ! DECREMENT COUNT
;  3
;  3             if .COUNT eql 0                                        ! IF END OF DATA PATTERN
;  3             then
;  4                 begin
;  4                 COUNT = ..DP_ADDR;                                 ! REPEAT DATA PATTERN
;  4                 SRC_ADDR = .DP_ADDR + 2;
;  3                 end;
;  3
;  2             end;                                                   ! WORD TRANSFER LOOP
;  2
;  2     !    if (.CSR_MEM and .TST_PAR)
;  2     !    then                                                                            ! MMM
;  2     !        if not .PAR_TSD                                                             ! MMM
;  2     !        then                                                                        ! MMM
;  2     !            begin                                                                   ! MMM
;  2     !            GETPRI(CUR_PRIORITY);                                                  ! MMM
;  2     !            SETPRI(PRIO7);                                                         ! MMM
;  2     !            IOB_ADDR = .MAD1 [BUF_0];                                          ! I/O BUFFER ADDRESS MMM
;  2     !            incr COUNT from 1 to 2 do                                              ! MMM
;  2     !                begin                                                              ! MMM
;  2     !                %o'172100' = %o'5';                                                ! MMM
;  2     !                end;                                                               ! MMM
;  2     !            .IOB_ADDR = %o'2';                                                 ! MOVE 1 WORD MMM
;  2     !            %o'172100' = %o'1';                                                    ! MMM
;  2     !            SETPRI(.CUR_PRIORITY);                                                 ! MMM
;  2     !            end;                                                                   ! MMM
;  2
;  1         end;                                                       ! ROUTINE FILL_BUFF
```

```
                                                  .SBTTL  FILL.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          FILL.BUFF::
                                                  JSR     R1,$SAVE5
000004  005746                                    TST     -(SP)                       ;                          4948
000006  032737  000100  000000G                   BIT     #100,SWP.FLAGS              ;                          4983
000014  001403                                    BEQ     1$
000016  012701  000000G                           MOV     #SWP.UCNT,R1                ; *,DP.ADDR                4985
000022  000443                                    BR      5$                          ;                          4983
000024  013700  000000G          1$:              MOV     SWP.DPAT,R0                 ;                          4989
000030  001402                                    BEQ     2$
000032  010002                                    MOV     R0,R2                       ; *,DP.NUM                 4991
000034  000414                                    BR      3$                          ;                          4989
000036  013700  000000G          2$:              MOV     L$LUN,R0                    ;                          4994
000042  062700  000050'                           ADD     #DPST,R0
000046  005002                                    CLR     R2                          ; DP.NUM
000050  151002                                    BISB    (R0),R2                     ; *,DP.NUM
000052  105210                                    INCB    (R0)                        ;                          4995
000054  121027  000025                            CMPB    (R0),#25                    ;                          4997
000060  101402                                    BLOS    3$
000062  112710  000001                            MOVB    #1,(R0)                     ;                          4999
000066  010200                   3$:              MOV     R2,R0                       ; DP.NUM,*                 5003
000070  006300                                    ASL     R0
000072  016001  001166'                           MOV     DPA.TBL-2(R0),R1            ; *,DP.ADDR
000076  020227  000021                            CMP     R2,#21                      ; DP.NUM,*                 5005
000102  103413                                    BLO     5$
000104  013700  000114'                           MOV     MAD1,R0                     ;                          5010
000110  006002                                    ROR     R2                          ; DP.NUM                   5008
000112  103004                                    BCC     4$
000114  016061  000046  000002                    MOV     46(R0),2(R1)                ; *,*(DP.ADDR)             5010
000122  000403                                    BR      5$                          ;                          5008
000124  016061  000046  000004   4$:              MOV     46(R0),4(R1)                ; *,*(DP.ADDR)             5012
000132  013700  000114'          5$:              MOV     MAD1,R0                     ;                          5016
000136  016004  000032                            MOV     32(R0),R4                   ; *,IOB.ADDR
000142  011103                                    MOV     (R1),R3                     ; DP.ADDR,COUNT            5017
000144  012705  000002                            MOV     #2,R5                       ;                          5018
000150  060105                                    ADD     R1,R5                       ; DP.ADDR,*
000152  010502                                    MOV     R5,R2                       ; *,SRC.ADDR
000154  016046  000026                            MOV     26(R0),-(SP)                ;                          5020
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$:              MOV     (R2)+,(R4)+                 ; SRC.ADDR,IOB.ADDR        5022
000204  005303                                    DEC     R3                          ; COUNT                    5025
000206  001002                                    BNE     7$                          ;                          5027
000210  011103                                    MOV     (R1),R3                     ; DP.ADDR,COUNT            5030
000212  010502                                    MOV     R5,R2                       ; *,SRC.ADDR               5031
000214  005200                   7$:              INC     R0                          ; N                        5020
000216  020066  000004                            CMP     R0,4(SP)                    ; N,*
```

```
000222  003767                          BLE     6$
000224  062706  000006                  ADD     #6,SP
000230  000207                          RTS     PC                                      ;                                       4948

; Routine Size:  77 words,      Routine Base:  $CODE$ + 15160
; Maximum stack depth per invocation:  10 words


;   5054  1
;   5055  1
```

```
;     5056  1    GLOBAL ROUTINE PROC_RETPKT : NOVALUE =
;     5057  1
;     5058  1    !+
;     5059  1    !         THIS ROUTINE IS CALLED FROM THE MULTI_DRIVE "EXECUTIVE" AND DUP COMMAND TO CHECK FOR
;     5060  1    !         AND PROCESS ANY RETURN PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER"
;     5061  1    !         PORTION OF THE PROGRAM. THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK
;     5062  1    !         BETWEEN THE TWO PROGRAM PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH
;     5063  1    !         REQUIRE PROCESSING.
;     5064  1    !
;     5065  1    !         UNDER THE MULTI-DRIVE TEST, RETURN PACKETS ORIGINATE FROM THREE SOURCES:
;     5066  1    !                   1. MSCP           THE MORE COMMON, DESCRIBING A COMPLETED I/O
;     5067  1    !                                     OPERATION.
;     5068  1    !MMM!               2. DUP            - THE LESS COMMON, DESCRIBING A PORTION OF I/O
;     5069  1    !                                     COMMUNICATIONS WITH THE CONTROLLER PROGRAM.
;     5070  1    !                   3. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
;     5071  1    !                                     COMMAND TIMEOUT.
;     5072  1    !-
;     5073  1
;     5074  1        while .IODQ_IN neq .IODQ_OUT do                          ! DO UNTIL I/O DONE QUEUE IS EMPTY
;     5075  2             begin
;     5076  2             RP_INDX = OUT_IODQ ();                              ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
;     5077  2             RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);         ! CALCULATE RETPKT ADDRESS
;     5078  3             if NOT (.RP_ADDR [CONID] eql CID_DUP)               ! if not DUP than
;     5079  2             then (SET_CPAR (.RP_ADDR [CTLR]););                 ! SET UP CURRENT CONTROLLER PARAMETERS
;     5080  2
;     5081  2             selectoneu .RP_ADDR [CONID] of                      ! CONNECTION ID INDICATES PACKET SOURCE
;     5082  2                  set
;     5083  2
;     5084  2                  [CID_MSCP] :         IO_RETPKT ();            ! DISK MSCP (I/O TRANSFER DONE)
;     5085  2    !MMM                  [CID_DUP] :          DIO_RETPKT ();            ! DUP (I/O TRANSFER DONE)
      5086  2                  [CID_DRIVER] :       DR_RETPKT ();            ! MESSAGE FROM "DRIVER"
;     5087  2
;     5088  2                  [otherwise] :        PRINTF (DBM12, .RP_ADDR [CONID]);!"CONN ID = XXXXX RECEIVED"
;     5089  2                  tes;
;     5090  2
:     5091  1             end;                                                ! UNITL I/O DONE QUEUE IS EMPTY


                                                 .SBTTL  PROC.RETPKT MUL'I-DRIVE TEST ROUTINES
000000  010146                         PROC.RETPKT::
                                                 MOV     R1,-(SP)
000002  023737  000000G 000000G        1$:       CMP     IODQ.IN,IODQ.OUT            ;                        5056
000010  001461                                   BEQ     6$                          ;                        5074
000012  004737  000000G                          JSR     PC,OUT.IODQ
000016  010037  000000G                          MOV     R0,RP.INDX                  ;                        5076
000022  C10046                                   MOV     R0,-(SP)
000024  012746  000054                           MOV     #54,-(SP)                   ; RP.INDX,*              5077
000030  004737  000000G                          JSR     PC,BL$MUL
000034  062700  000000G                          ADD     #RETPKT,R0
000040  010037  000000G                          MOV     R0,RP.ADDR
000044  126027  000003  000002                   CMPB    3(R0),#2
000052  001406                                   BEQ     2$                          ;                        5078
000054  116016  000002                           MOVB    2(R0),(SP)                  ;                        5079
```

```
000060  042716  177760                      BIC     #177760,(SP)
000064  004737  000000G                     JSR     PC,SET.CPAR
000070  013700  000000G         2$:         MOV     RP.ADDR,R0
000074  005001                              CLR     R1                      ;                       5081
000076  156001  000003                      BISB    3(R0),R1
000102  005701                              TST     R1
000104  001003                              BNE     3$                      ;                       5084
000106  004737  000000V                     JSR     PC,IO.RETPKT
000112  000416                              BR      5$
000114  020127  000003          3$:         CMP     R1,#3                   ;                       5081
000120  001003                              BNE     4$                      ;                       5086
000122  004737  000000V                     JSR     PC,DR.RETPKT
000126  000410                              BR      5$
000130  010116                  4$:         MOV     R1,(SP)                 ;                       5081
000132  012746  000000G                     MOV     #DBM12,-(SP)            ;                       5088
000136  012746  000002                      MOV     #2,-(SP)
000142  010600                              MOV     SP,R0                   ; SP,*
000144  104417                              TRAP    17
000146  022626                              CMP     (SP)+,(SP)+
000150  022626                  5$:         CMP     (SP)+,(SP)+
000152  000713                              BR      1$                      ;                       5075
000154  012601                  6$:         MOV     (SP)+,R1                ;                       5074
000156  000207                              RTS     PC                      ;                       5056

; Routine Size:  56 words.      Routine Base:  $CODE$ + 15412
; Maximum stack depth per invocation:  7 words
```

```
.MMM
!GLOBAL ROUTINE DIO_RETPKT : NOVALUE =

!+
!	THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL DUP I/O TRANSFER
!	RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
!	HARD ERRORS THAT MAY HAVE OCCURRED. UPDATING THE STATISTICS.
!
!	IMPLICIT INPUTS:
!		RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
!		T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
!		CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
!		DUOFF - CST OFFSET FOR THE CURRENT UNIT
!		L$LUN - CURRENT UNIT NUMBER
!		CCTLR - CURRENT CONTROLLER NUMBER
!
!	IMPLICIT OUTPUTS
!		CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] - IF THIS BIT SET NO DUP EXERCISER
!
!-
!BEGIN

!LOCAL	FLAG : BYTE INITIAL(BYTE(TRUE)),
!	 SUM2 : WORD,
!	 SUM : WORD;				! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
!PRINTX (DER18);

!IF .RP_ADDR [STATUS] NEQU ST_SUC			! IF STATUS CODE INDICATES ERROR
!THEN
!	BEGIN
!	CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;		! SET DUP ERROR FLAG
!	HARD_ERROR ();
!	IF .RP_ADDR [ENDCOD] EQLU (OP_ELP + OP_END) OR	! IF ENDCODE IS EXECUTE LOCAL PROGRAM
!	   .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END)	!		OR GET DUST STATUS
!	THEN	BEGIN
!		CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;		! TURN OFF DUP EXERCISR
!		END;
!	END
!ELSE						! ELSE - I/O WAS SUCCESSFUL
!    BEGIN
!
!    IF .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END)	! IF ENDCODE IS GET DUST STATUS
!    THEN
!	BEGIN
!	IF .RP_ADDR [9,11,1,0] EQL 1
!		THEN CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = ACTIVE		! CONTROLLER IN AN ACTIVE STAE
!		ELSE CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = IDLE;		! CONTROLLER IN AN IDLE STATE
!	IF .RP_ADDR [9,9,1,0] NEQ 1 THEN	! TEST TO SEE IF CONTROLLER LOCAL PROGRAMS(PG 18 OF DUP DOC)
!		BEGIN
!		HARD_ERROR ();
!		CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;		! TURN OFF DUP EXERCISR
!		END;
!	END;
```

```
!
!
!IF (.RP_ADDR [ENDCOD] EQL (OP_RCD + OP END)) AND
!    (.DUPPKT [DUPTYPE] EQL 6) AND
!    (.DUPPKT [DUPMESG] EQL 2) AND                   !IF IT IS A RECEIVE DBN COMMAND WITH TYPE 6 AND MESSAGE 2 THEN
!    (.CST_ADDR [.DUOFF + OF DBN, DUPWRITE] EQLU 1)      ! IF WRITE FLAG SET IN CST TABLE THEN COMPARE BLOCKS
!        THEN DUP_COMPARE ();
!
!END;                                               ! COMPARE THE FOLLOWING 512 BYTES
!
!PUT_RETPKT (.RP INDX);
!END;                                               ! ROUTINE DIO_RETPKT
```

```
!MMMGLOBAL ROUTINE DUP COMPARE : NOVALUE =


!+
!       THIS ROUTINE IS CALLED BY DIO_RETPKT WHEN THE RECEIVE DATA COMMAND
!       IS BEING PROCESSED. THIS COMMAND COMPARES THE WRITTEN BUFFER WITH
!       THE PATERN WORD GIVEN IN SEND DATA COMMAND.  FOR EVERY WORD COMPARED
!       THE ROUTINE INCREMENTS THE TALLY TABLE.  IF THE COMPARE SHOWS AN
!       ERROR, THE DBN HARD ERROR COUNTER WILL BE INCREMENTED AND THE
!       THE DBN NUMBER AND BYTE COUNT WILL BE PRINTED.
!
!       IMPLICIT INPUTS:
!               S_PATTERN       ! THE SAVED PATTERN WRITTEN TO THE DBN'S
!               S_DUPPKT        ! THE POINTER FOR DUP BUFFER
!               T_ADDR          ! THE ADDRESS OF THE TALLY TABLE FOR THIS UNIT
!               CST_ADDR        ! THE ADDRESS OF PRESENT CONTROLLER STATUS TABLE
!-
!BEGIN

!OWN
!       COUNT : WORD;
!
!!PRINTX (DER19);
!S_DUPPKT = 0;
!INCR COUNT FROM 1 TO 256 DO    !INDEX PIONTER FOR DATA STORED IN MSCP ENV PACKET
!       BEGIN
!       S_DUPPKT = .S_DUPPKT + 2;                               ! INITIALLY THIS SKIPS THE FIRST WORD OF DUPPKT
!       IF .(DUPPKT + .S_DUPPKT) NEQ .S_PATTERN THEN            !IF THE CONTENTS OF DBN DOESN T EQUAL PATTERN
!               BEGIN
!               CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;               ! SET DUP ERROR FLAG
!               ERRHRD (46, EH_10, EMS 22);                     !LIST ERROR
!               EXITLOOP;
!               END;
!       END;                                            !GO THROUGH ALL DBN WORDS
!END;                                                   !END ROUTINE DUP-COMPARE
```

```
GLOBAL routine IO_RETPKT : novalue =

!+
!       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
!       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
!       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
!       PERFORMING HOST WRITE-COMPARES IF REQUIRED.
!
!       IMPLICIT INPUTS:
!               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
!               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
!               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
!               CCTLR - CURRENT CONTROLLER NUMBER
!               L$LUN   CURRENT UNIT NUMBER
!-

    begin

    local
        FLAG : byte initial (byte (TRUE));

    FSET_UPAR ();                                   ! FIND UNIT'S ENTRY IS CST AND SET UP UNIT RELATED DATA
    ST_CODE = .RP_ADDR [STSCOD];                    ! GET STATUS CODE FROM RETPKT
    SB_CODE = .RP_ADDR [SUBCOD];                    ! GET SUB-CODE, IF ANY

    if (.ST_CODE neq ST_SUC)                        ! IF STATUS CODE INDICATES ERROR
    then
        begin
        HARD_ERROR ();                              ! UPDATE ERROR COUNT
        COMPARE_DATA = FALSE;                       ! NO POINT IN DOING HOST COMPARES ON ERRORS

        if (.ST_CODE neq ST_OFL) and                ! DROP UNIT IF ERROR COUNTS EXCEEDS LIMIT
           (.ST_CODE neq ST_AVL) and
           (.T_ADDR [ERR_HARD] gequ .SWP_ERROR)
        then
            begin
            DUR [.L$LUN] = DU_HERR;                 ! LOAD REASON FOR DROPPING UNIT
            DODU (.L$LUN);                          ! DROP UNIT
            end;

        end
    else                                            ! IF I/O WAS SUCCESSFUL
        begin
        UPD_IO_TALLY ();                            ! UPDATE I/O TALLY (STATISTICS)

        if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)
        then
            COMPARE_DATA = TRUE;                    ! HOST COMPARES MAY BE ALLOWED IF NO FURTHER ERRORS

        if (BIT_TST (SWP_FLAGS, SWF_HWC)) and       ! IF HOST IS DOING WRITE-COMPARES
           (.COMPARE_DATA)
        then
            FLAG = HOST_WRT_CHK ();                 ! SAVE I/O PACKET OR DO WRITE CHECK
```

```
 :    5249  3
 :    5250  2          end;
 :    5251  2
 :    5252  2      if .FLAG                                ! IF FLAG IS STILL TRUE
 :    5253  2      then
 :    5254  2          SWEEP ();                           ! DEALLOCATE BUFFER(S) AND RETPKT(S)
 :    5255  2
 :    5256  2      QIO [.CCTLR] = .QIO [.CCTLR] - 1;      ! DECREMENT NO. OF OUTSTANDING QIOs
 :    5257  1      end;                                    ! ROUTINE IO RETPKT

                                        .SBTTL  IO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G         IO.RETPKT::
                                        JSR     R1,$SAVE2           ;                       5196
000004  112701  000001                  MOVB    #1,R1               ; *,FLAG                5212
000010  004737  000000V                 JSR     PC,FSET.UPAR        ;                       5217
000014  013700  000000G                 MOV     RP.ADDR,R0          ;                       5218
000020  116037  000016  000000G         MOVB    16(R0),ST.CODE
000026  042737  177740  000000G         BIC     #177740,ST.CODE
000034  016002  000016                  MOV     16(R0),R2           ;                       5219
000040  006202                          ASR     R2
000042  006202                          ASR     R2
000044  006202                          ASR     R2
000046  006202                          ASR     R2
000050  006202                          ASR     R2
000052  042702  174000                  BIC     #174000,R2
000056  010237  000000G                 MOV     R2,SB.CODE
000062  005737  000000G                 TST     ST.CODE             ;                       5221
000066  001431                          BEQ     1$
000070  004737  000000V                 JSR     PC,HARD.ERROR       ;                       5224
000074  105037  001262'                 CLRB    COMPARE.DATA        ;                       5225
000100  023727  000000G 000003          CMP     ST.CODE,#3          ;                       5227
000106  001447                          BEQ     3$
000110  023727  000000G 000004          CMP     ST.CODE,#4          ;                       5228
000116  001443                          BEQ     3$
000120  013700  000000G                 MOV     T.ADDR,R0           ;                       5229
000124  026037  000014  000000G         CMP     14(R0),SWP.ERROR
000132  103435                          BLO     3$
000134  013700  000000G                 MOV     L$LUN,R0            ;                       5232
000140  112760  000004  000000G         MOVB    #4,DUR(R0)
000146  104451                          TRAP    51                  ;                       5233
000150  000426                          BR      3$                  ;                       5221
000152  004737  000000V         1$:     JSR     PC,UPD.IO.TALLY     ;                       5239
000156  013700  000000G                 MOV     RP.ADDR,R0          ;                       5241
000162  126027  000014  000242          CMPB    14(R0),#242
000170  001003                          BNE     2$
000172  112737  000001  001262'         MOVB    #1,COMPARE.DATA     ;                       5243
000200  032737  000040  000000G 2$:     BIT     #40,SWP.FLAGS       ;                       5245
000206  001407                          BEQ     3$
000210  032737  000001  001262'         BIT     #1,COMPARE.DATA     ;                       5246
000216  001403                          BEQ     3$
000220  004737  000000V                 JSR     PC,HOST.WRT.CHK     ;                       5248
000224  110001                          MOVB    R0,R1               ; *,FLAG
```

```
000226  006001                  3$:     ROR     R1                      ; FLAG                  5252
000230  103002                          BCC     4$
000232  004737  000000V                 JSR     PC,SWEEP                ;                       5254
000236  013700  000000G         4$:     MOV     CCTLR,R0                ;                       5256
000242  105360  000000G                 DECB    QIO(R0)
000246  000207                          RTS     PC                      ;                       5196

; Routine Size:  84 words,      Routine Base:  $CODE$ + 15572
; Maximum stack depth per invocation:  5 words
```

```
;   5258  1    GLOBAL routine FSET_UPAR : novalue =
;   5259  1
;   5260  1    !+
;   5261  1    !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
;   5262  1    !       CONTROLLER STATUS TABLE (CST) FOR THE DISK ADDRESS WHICH IS
;   5263  1    !       CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
;   5264  1    !       CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT-RELATED
;   5265  1    !       DATA PARAMETERS.
;   5266  1    !
;   5267  1    !       IMPLICIT INPUTS:
;   5268  1    !               RP_ADDR  - ADDRESS OF CURRENT RETURN PACKET
;   5269  1    !               CST_ADDR   ADDRESS OF CURRENT CONTROLLER'S CST
;   5270  1    !-
;   5271  1
;   5272  2        begin
;   5273  2
;   5274  2        incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR   1) * UNIT_SIZE + OF UN) by UNIT_SIZE do  ! FOR EACH UNIT
;   5275  2
;   5276  2            if .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .RP_ADDR [DISK]        ! IF RETPKT UNIT MATCHES CST ENTRY
;   5277  2            then
;   5278  3                begin
;   5279  3                SET_UPAR (.OFFSET);                                                 ! SET UP UNIT RELATED DATA
;   5280  3                return;                                                             ! DONE
;   5281  2                end;
;   5282  2
;   5283  2        PRINTF (DBM19, .RP_ADDR [DISK], .CCTLR);                                    ! "CAN'T FIND DISK XXX IN CST X"
;   5284  1        end;                                                                        ! ROUTINE FSET_UPAR

                                                .SBTTL  FSET.UPAR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G             FSET.UPAR::
                                                JSR     R1,$SAVE4
000004  012702  000003                          MOV     #3,R2                   ;                       5258
000010  010201                      1$:         MOV     R2,R1                   ; *,OFFSET              5274
000012  006301                                  ASL     R1                      ; OFFSET,*              5276
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,*              5279
000044  004737  000000G                         JSR     PC,SET.UPAR
000050  005726                                  TST     (SP)+
000052  000207                                  RTS     PC                      ;                       5280
000054  062702  000012              2$:         ADD     #12,R2                  ;                       5278
000060  020227  000041                          CMP     R2,#41                  ; *,OFFSET              5274
000064  003751                                  BLE     1$                      ; OFFSET,*
000066  013746  000000G                         MOV     CCTLR,-(SP)             ;                       5283
000072  013700  000000G                         MOV     RP.ADDR,R0
000076  016046  000010                          MOV     10(R0),-(SP)
000102  012746  000000G                         MOV     #DBM19,-(SP)
```

```
000106  012746  000003          MOV     #3,-(SP)
000112  010600                  MOV     SP,R0                   ; SP,*
000114  104417                  TRAP    17
000116  062706  000010          ADD     #10,SP                  ;                        5272
000122  000207                  RTS     PC                      ;                        5258

; Routine Size:  42 words,      Routine Base:  $CODE$ + 16042
; Maximum stack depth per invocation:  11 words
```

```
;  1   GLOBAL routine HARD_ERROR : novalue =
;  1
;  1   !+
;  1   !      THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO INCREMENT THE HARD
;  1   !      ERROR STATISTIC FIELD FOR THE CURRENT UNIT. IF THE HARD ERROR COUNT
;  1   !      HAS EXCEEDED THE OPERATOR-SPECIFIED LIMIT, THEN THE UNIT IS DROPPED
;  1   !      FROM TESTING.
;  1   !
;  1   !      IMPLICIT INPUTS:
;  1   !             L$LUN - CURRENT UNIT NUMBER
;  1   !             CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  1   !             CUOFF - CST OFFSET FOR CURRENT UNIT
;  1   !             T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  1   !
;  1
;  2      begin
;  2      T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;                       ! INCREMENT UNIT'S HARD ERROR COUNT
;  2   if .RP_ADDR [CONID] EQL CID_MSCP                                     !FOR MSCP ERRORS   ?ZZ
;  2   THEN                                                                 !                  ZZZ
;  2
;  2      selectoneu .ST_CODE of
;  2          set
;  2
;  2          [ST_SUC]:        if .SB_CODE neq 0                           ! SUCCESS WITH NON-ZERO SUB CODE
;  2                           then
;  3                               begin
;  3
;  3                               if .SB_CODE eql 4
;  3                               then
;  3                                   T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1
;  3                               else
;  3                                   T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  3
;  3                               if .APT_MODE
;  3                               then
;  3                                   ERR_HRD_RTNE_APT (44)
;  3                               else
;  3                                   ERR_HRD_RTNE (44);
;  3
;  2                               end;
;  2
;  3          [ST_CMD]:        begin                                        ! INVALID COMMAND
;  3                           T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  3
;  3                           if .APT_MODE
;  3                           then
;  3                               ERR_HRD_RTNE_APT (31)
;  3                           else
;  3                               ERR_HRD_RTNE (31);
;  3
;  2                           end;
;  2
;  3          [ST_ABO]:        begin                                        ! COMMAND ABORTED
```

```
                        T ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;

                        if .APT_MODE
                        then
                            ERR_HRD RTNE_APT (32)
                        else
                            ERR_HRD_RTNE (32);

                        end;

        [ST_OFL] :      begin                                               ! OFFLINE
                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;

                        if .APT_MODE
                        then
                            begin
                            .MAIL_BOX_TESTNUM = 1;
                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D DISK_NUM];
                            end;

                        ERRDF (18, EGD_18, EMS_18);                          !
                        DUR [.L$LUN] = DU_DFATAL;                            ! DEVICE FATAL ERROR
                        DODU (.L$LUN);                                       ! DROP UNIT
                        end;

        [ST_AVL]:       begin                                               ! WENT TO AVAILABLE STATE
                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;

                        if .APT_MODE
                        then
                            begin
                            .MAIL_BOX_TESTNUM = 1;
                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
                            end;

                        ERRDF (24, EGD_18, EMS_24);                          !
                        DUR [.L$LUN] = DU_DFATAL;                            ! DEVICE FATAL ERROR
                        DODU (.L$LUN);                                       ! DROP UNIT
                        end;

        [ST_MFE]:       begin                                               ! MEDIA FORMAT ERROR
                        T_ADDR [ERR_HRD SEK] = .T_ADDR [ERR_HRD_SEK] + 1;

                        if .APT_MODE
                        then
                            ERR_HRD_RTNE_APT (35)
                        else
                            ERR_HRD_RTNE (35);

                        end;

        [ST_WPT]:       begin                                               ! DEVICE WRITE PROTECTED
```

```
;  3                                  if .SB_CODE eql 128
;  3                                  then
;  3                                      T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1
;  3                                  else
;  3                                      T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  3
;  3                                  if .APT_MODE
;  3                                  then
;  3                                      ERR_HRD_RTNE_APT (36)
;  3                                  else
;  3                                      ERR_HRD_RTNE (36);
;  3
;  2                                  end;
;  2
;  3              [ST_CMP]:           begin                                          ! COMPARE ERROR
;  3                                  T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  3
;  3                                  if .APT_MODE
;  3                                  then
;  3                                      ERR_HRD_RTNE_APT (37)
;  3                                  else
;  3                                      ERR_HRD_RTNE (37);
;  3
;  2                                  end;
;  2
;  3              [ST_DAT]:           begin                                          ! DATA ERROR
;  3
;  3                                  if .SB_CODE eql 2
;  3                                  then
;  3                                      T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;  3                                  else
;  3                                      T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  3
;  3                                  if (.SB_CODE eql 0) and
;  3                                     (not .FORCED_ERROR) and
;  4                                     (BIT_TST (SWP_FLAGS, SWF_FER))
;  3                                  then
;  4                                      begin
;  4                                      FORCED_ERROR = TRUE;
;  4                                      FER0_LBN = .RP_ADDR [LBN_LO];              ! BLOCK WITH "FORCED ERROR" FOUND
ZZZ                                                                                  !
;  4                                      FER1_LBN = .RP_ADDR [LBN_HI];              !
ZZZ
;  4                                      FER_BC = .RP_ADDR [CBCNT_LO];              !
;  3                                      end;
;  3
;  3
;  3                                  if .APT_MODE
;  3                                  then
;  3                                      ERR_HRD_RTNE_APT (38)
;  3                                  else
;  3                                      ERR_HRD_RTNE (38);
;  3
;  2                                  end;
;  2
```

```
        [ST_HST]:       begin                                                       ! HOST ACCESS ERROR
                        T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;

                        if .APT_MODE
                        then
                            ERR_HRD_RTNE_APT (39)
                        else
                            ERR_HRD_RTNE (39);

                        end;

        [ST_CNT]:       begin                                                       ! CONTROLLER ERROR
                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;

                        if .APT_MODE
                        then
                            ERR_HRD_RTNE_APT (40)
                        else
                            ERR_HRD_RTNE (40);

                        end;

        [ST_DRV]:       begin                                                       ! DRIVE ERROR

                        if .SB_CODE eql 3
                        then
                            T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
                        else
                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;

                        if .APT_MODE
                        then
                            ERR_HRD_RTNE_APT (41)
                        else
                            ERR_HRD_RTNE (41);

                        end;

        [ST_DIA]:       begin                                                       ! MESSAGE FROM INTERNAL DIAGNOSTICS
                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;

                        if .APT_MODE
                        then
                            ERR_HRD_RTNE_APT (43)
                        else
                            ERR_HRD_RTNE (43);

                        end;

        [otherwise]:    begin                                                       ! PRINT STATUS CODE IF NO MATCH
                        C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;

                        if .APT_MODE
```

```
                                                then
                                                    ERR_HRD RTNE_APT (45)
                                                else
                                                    ERR_HRD_RTNE (45);

                                                end;

                            tes;

        !MMMif .RP_ADDR [CONID] EQL CID_DUP                                              !FOR DUP ERRORS                 ZZZ
        !MMMOR .D FAIL EQL 1                                                             !EVEN IF UNRECOGNIZABLE AS SUCH ZZZ
        !MMMTHEN                                                                         !                               ZZZ

        !       selectoneu .RP_ADDR [STSCOD] of
        !               SET
        !               [%o'0']                 : begin                 ! if status code succesful
        !                                        if .RP_ADDR [ENDCOD] EQLU (OP GDS + OP_END) and ! IF ENDCODE IS GET DUST STATUS
        !                                           .RP_ADDR [9,9,1,0] NEQ 1                     ! TEST TO SEE IF CONTROLLER LOCAL PR
OGRAMS
        !                                        then                                            ! (PG 18 OF DUP DOC)
        !                                                       BEGIN
        !                                                        ERR_HRD_RTNE (60);              !UNABLE TO LOAD LOCAL CONTROLLER DUP
 MEDIA !ZZZ
        !                                                       T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
        !                                                       END
        !                                        else
        !                                               begin
        !                                               if (.DUPPKT [DUPTYPE] eql 5)             ! if fatal error
        !                                               then
        !!ZZZ                                                   begin                            !DON'T DROP DEVICE ON DUP ERROR
   ZZZ
        !!ZZZ                                                   DUR [.L$LUN] = DU_DFATAL;        !GIVE F.E. A CHANCE TO SEE ERRORS
   ZZZ
        !!ZZZ                                                   DODU (.L$LUN);                   ! FATAL DEVICE ERROR DROP UNIT);
   ZZZ
        !!ZZZ                                                   end;                             ! SET REASON FOR DROPPING UNIT
   ZZZ
        !                                               selectoneu .DUPPKT [DUPMESG] of
        !                                                       SET
        !                                                       [%o'1'] : begin
        !                                                               ERR_HRD_RTNE (62);          ! illegal unit number     !ZZZ
        !                                                               T_ADDR                   - ADDR [ERR_HRD_HST] + 1;
        !                                                               end;
        !                                                       [%o'2'] : begin
        !                                                               ERR_HRD_RTNE (63);          ! illegal relative or physical b
lock # !ZZZ
        !                                                               T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
        !                                                               end;
        !                                                       [%o'3'] : begin
        !                                                               ERR_HRD_RTNE (64);          ! device error      !ZZZ
        !                                                               T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
        !                                                               end;
        !                                                       [%o'4'] : begin
        !                                                               ERR_HRD_RTNE (65);          ! zero lenght message    !ZZZ
        !                                                               T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
        !                                                               end;
        !                                                       [%o'5'] : begin                                                !MMM
        !                                                               ERR_HRD_RTNE (64);          ! device error            !MMM
        !                                                               T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;     !MMM
        !                                                               end;                                                  !MMM
```

```
;     5550  2     !                                     [OTHERWISE] : begin
;     5551  2     !                                             ERR_HRD_RTNE (66);      ! DUP UNKNOWN STATUS CODE    !ZZZ
;     5552  2     !                                             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HR
D] + 1;
;     5553  2     !                                             end;
;     5554  2     !                                     tes;
;     5555  2     !                               end;
;     5556  2     !
;     5557  2     !                       end;
;     5558  2     !           [%o'1']          : begin
;     5559  2     !                       ERR_HRD_RTNE (67);      ! INVALID COMMAND           !ZZZ
;     5560  2     !                       T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;     5561  2     !                       end;
;     5562  2     !           [%o'2']          : begin
;     5563  2     !                       ERR_HRD_RTNE (68);      ! NO REGION AVAILABLE       !ZZZ
;     5564  2     !                       T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;     5565  2     !                       end;
;     5566  2     !           [%o'3']          : begin
;     5567  2     !                       ERR_HRD_RTNE (69);              ! NO REGION SUITABLE        !ZZZ
;     5568  2     !                       T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;     5569  2     !                       end;
;     5570  2     !           [%o'4']          : begin
;     5571  2     !                       ERR_HRD_RTNE (70);      ! PROGRAM NOT KNOWN         !ZZZ
;     5572  2     !                       T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;     5573  2     !                       end;
;     5574  2     !           [%o'5']          : begin
;     5575  2     !                       ERR_HRD_RTNE (71);              ! LOAD FAILURE              !ZZZ
;     5576  2     !                       T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;     5577  2     !                       end;
;     5578  2     !           [%o'6']          : begin
;     5579  2     !                       ERR_HRD_RTNE (72);              ! STANDALONE                !ZZZ
;     5580  2     !                       T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;     5581  2     !                       end;
;     5582  2     !           [OTHERWISE] : begin
;     5583  2     !                       ERR_HRD_RTNE (73);              ! DUP UNKNOWN STATUS CODE   !ZZZ
;     5584  2     !                       C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;     5585  2     !                       end;
;     5586  2     !MMM        TES;
;     5587  2
;     5588  1         end;                                                          ! ROUTINE HARD_ERROR


                                                .SBTTL  HARD.ERROR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                 HARD.ERROR::
                                                JSR     R1,$SAVE4                       ;                       5285
000004  013701  000000G                         MOV     T.ADDR,R1                       ;                       5301
000010  005261  000014                          INC     14(R1)
000014  013703  000000G                         MOV     RP.ADDR,R3                      ;                       5302
000020  105763  000003                          TSTB    3(R3)
000024  001401                                  BEQ     2$
000026  000207                          1$:     RTS     PC
000030  013702  000000G                 2$:     MOV     ST.CODE,R2                      ;                       5305
000034  001027                                  BNE     6$                              ;                       5308
000036  013704  000000G                         MOV     SB.CODE,R4
```

```
000042  001771                          BEQ     1$
000044  012700  000050                  MOV     #50,R0                          ;       5314
000050  060100                          ADD     R1,R0
000052  020427  000004                  CMP     R4,#4                           ;       5312
000056  001002                          BNE     3$
000060  105210                          INCB    (R0)                            ;       5314
000062  000402                          BR      4$                              ;       5312
000064  105260  000001          3$:     INCB    1(R0)                           ;       5316
000070  032737  000001  001254' 4$:     BIT     #1,APT.MODE                     ;       5318
000076  001403                          BEQ     5$
000100  012746  000054                  MOV     #54,-(SP)                       ;       5320
000104  000556                          BR      15$
000106  012746  000054          5$:     MOV     #54,-(SP)                       ;       5322
000112  000556                          BR      17$
000114  020227  000001          6$:     CMP     R2,#1                           ;       5326
000120  001014                          BNE     8$
000122  105261  000051                  INCB    51(R1)                          ;       5327
000126  032737  000001  001254'         BIT     #1,APT.MODE                     ;       5329
000134  001403                          BEQ     7$
000136  012746  000037                  MOV     #37,-(SP)                       ;       5331
000142  000567                          BR      21$
000144  012746  000037          7$:     MOV     #37,-(SP)                       ;       5333
000150  000567                          BR      23$
000152  020227  000002          8$:     CMP     R2,#2                           ;       5337
000156  001014                          BNE     10$
000160  105261  000050                  INCB    50(R1)                          ;       5338
000164  032737  000001  001254'         BIT     #1,APT.MODE                     ;       5340
000172  001403                          BEQ     9$
000174  012746  000040                  MOV     #40,-(SP)                       ;       5342
000200  000567                          BR      25$
000202  012746  000040          9$:     MOV     #40,-(SP)                       ;       5344
000206  000567                          BR      27$
000210  020227  000003          10$:    CMP     R2,#3                           ;       5348
000214  001036                          BNE     12$
000216  105261  000050                  INCB    50(R1)                          ;       5349
000222  032737  000001  001254'         BIT     #1,APT.MODE                     ;       5351
000230  001415                          BEQ     11$
000232  012777  000001  001256'         MOV     #1,@MAIL.BOX.TESTNUM            ;       5354
000240  013700  000000G                 MOV     CUOFF,R0                        ;       5355
000244  006300                          ASL     R0
000246  063700  000000G                 ADD     CST.ADDR,R0
000252  111077  001260'                 MOVB    (R0),@MAIL.BOX.SUBTST
000256  042777  177760  001260'         BIC     #177760,@MAIL.BOX.SUBTST
000264  104455                  11$:    TRAP    55                              ;       5358
000266  000022                          .WORD   22
000270  000000G                         .WORD   EGD.18
000272  000000G                         .WORD   EMS.18
000274  013700  000000G                 MOV     L$LUN,R0                        ;       5359
000300  112760  000005  000000G         MOVB    #5,DUR(R0)
000306  104451                          TRAP    51                              ;       5360
000310  000207                          RTS     PC                              ;       5305
000312  020227  000004          12$:    CMP     R2,#4                           ;       5363
000316  001036                          BNE     14$
```

```
000320  105261  000050                      INCB    50(R1)                         ;     5364
000324  032737  000001  001254'             BIT     #1,APT.MODE                    ;     5366
000332  001415                              BEQ     13$
000334  012777  000001  001256'             MOV     #1,@MAIL.BOX.TESTNUM           ;     5369
000342  013700  000000G                     MOV     CUOFF,R0                       ;     5370
000346  006300                              ASL     R0
000350  063700  000000G                     ADD     CST.ADDR,R0
000354  111077  001260'                     MOVB    (R0),@MAIL.BOX.SUBTST
000360  042777  177760  001260'             BIC     #177760,@MAIL.BOX.SUBTST
000366  104455                      13$:    TRAP    55                             ;     5373
000370  000030                              .WORD   30
000372  000000G                             .WORD   EGD.18
000374  000000G                             .WORD   EMS.24
000376  013700  000000G                     MOV     L$LUN,R0                       ;     5374
000402  112760  000005  000000G             MOVB    #5,DUR(R0)
000410  104451                              TRAP    51                             ;     5375
000412  000207                              RTS     PC                             ;     5305
000414  020227  000005              14$:    CMP     R2,#5                          ;     5378
000420  001014                              BNE     18$
000422  105261  000046                      INCB    46(R1)                         ;     5379
000426  032737  000001  001254'             BIT     #1,APT.MODE                    ;     5381
000434  001403                              BEQ     16$
000436  012746  000043                      MOV     #43,-(SP)                      ;     5383
000442  000563                      15$:    BR      36$
000444  012746  000043              16$:    MOV     #43,-(SP)                      ;     5385
000450  000563                      17$:    BR      38$
000452  020227  000006              18$:    CMP     R2,#6                          ;     5389
000456  001025                              BNE     24$
000460  012700  000050                      MOV     #50,R0                         ;     5393
000464  060100                              ADD     R1,R0
000466  023727  000000G 000200              CMP     SB.CODE,#200                   ;     5391
000474  001003                              BNE     19$
000476  105260  000001                      INCB    1(R0)                          ;     5393
000502  000401                              BR      20$                            ;     5391
000504  105210                      19$:    INCB    (R0)                           ;     5395
000506  032737  000001  001254'     20$:    BIT     #1,APT.MODE                    ;     5397
000514  001403                              BEQ     22$
000516  012746  000044                      MOV     #44,-(SP)                      ;     5399
000522  000561                      21$:    BR      42$
000524  012746  000044              22$:    MOV     #44,-(SP)                      ;     5401
000530  000561                      23$:    BR      44$
000532  020227  000007              24$:    CMP     R2,#7                          ;     5405
000536  001014                              BNE     28$
000540  105261  000047                      INCB    47(R1)                         ;     5406
000544  032737  000001  001254'             BIT     #1,APT.MODE                    ;     5408
000552  001403                              BEQ     26$
000554  012746  000045                      MOV     #45,-(SP)                      ;     5410
000560  000561                      25$:    BR      46$
000562  012746  000045              26$:    MOV     #45,-(SP)                      ;     5412
000566  000561                      27$:    BR      48$
000570  020227  000010              28$:    CMP     R2,#10                         ;     5416
000574  001054                              BNE     33$
000576  012700  000046                      MOV     #46,R0                         ;     5420
```

```
000602  060100                              ADD     R1,R0
000604  023727  000000G 000002              CMP     SB.CODE,#2              ;                       5418
000612  001002                              BNE     29$
000614  105210                              INCB    (R0)                    ;                       5420
000616  000402                              BR      30$                     ;                       5418
000620  105260  000001             29$:     INCB    1(R0)                   ;                       5422
000624  005737  000000G            30$:     TST     SB.CODE                 ;                       5424
000630  001024                              BNE     31$
000632  132737  000001  000000G             BITB    #1,FORCED.ERROR         ;                       5425
000640  001020                              BNE     31$
000642  032737  004000  000000G             BIT     #4000,SWP.FLAGS         ;                       5426
000650  001414                              BEQ     31$
000652  112737  000001  000000G             MOVB    #1,FORCED.ERROR         ;                       5429
000660  016337  000050  000000G             MOV     50(R3),FER0.LBN         ;                       5430
000666  016337  000052  000000G             MOV     52(R3),FER1.LBN         ;                       5431
000674  016337  000044  000000G             MOV     44(R3),FER.BC           ;                       5432
000702  032737  000001  001254'    31$:     BIT     #1,APT.MODE             ;                       5436
000710  001403                              BEQ     32$
000712  012746  000046                      MOV     #46,-(SP)               ;                       5438
000716  000521                              BR      50$
000720  012746  000046             32$:     MOV     #46,-(SP)               ;                       5440
000724  000523                              BR      52$
000726  020227  000011             33$:     CMP     R2,#11                  ;                       5444
000732  001014                              BNE     35$
000734  105261  000051                      INCB    51(R1)                  ;                       5445
000740  032737  000001  001254'             BIT     #1,APT.MODE             ;                       5447
000746  001403                              BEQ     34$
000750  012746  000047                      MOV     #47,-(SP)               ;                       5449
000754  000502                              BR      50$
000756  012746  000047             34$:     MOV     #47,-(SP)               ;                       5451
000762  000504                              BR      52$
000764  020227  000012             35$:     CMP     R2,#12                  ;                       5455
000770  001014                              BNE     39$
000772  105261  000050                      INCB    50(R1)                  ;                       5456
000776  032737  000001  001254'             BIT     #1,APT.MODE             ;                       5458
001004  001403                              BEQ     37$
001006  012746  000050                      MOV     #50,-(SP)               ;                       5460
001012  000453                     36$:     BR      50$
001014  012746  000050             37$:     MOV     #50,-(SP)               ;                       5462
001020  000465                     38$:     BR      52$
001022  020227  000013             39$:     CMP     R2,#13                  ;                       5466
001026  001023                              BNE     45$
001030  023727  000000G 000003              CMP     SB.CODE,#3              ;                       5468
001036  001003                              BNE     40$
001040  105261  000046                      INCB    46(R1)                  ;                       5470
001044  000402                              BR      41$                     ;                       5468
001046  105261  000050             40$:     INCB    50(R1)                  ;                       5472
001052  032737  000001  001254'    41$:     BIT     #1,APT.MODE             ;                       5474
001060  001403                              BEQ     43$
001062  012746  000051                      MOV     #51,-(SP)               ;                       5476
001066  000435                     42$:     BR      50$
001070  012746  000051             43$:     MOV     #51,-(SP)               ;                       5478
001074  000437                     44$:     BR      52$
```

```
001076  020227  000037              45$:   CMP     R2,#37                          ;       5482
001102  001014                             BNE     49$
001104  105261  000050                     INCB    50(R1)                          ;       5483
001110  032737  000001  001254'            BIT     #1,APT.MODE                     ;       5485
001116  001403                             BEQ     47$
001120  012746  000053                     MOV     #53,-(SP)                       ;       5487
001124  000416                      46$:   BR      50$
001126  012746  000053              47$:   MOV     #53,-(SP)                       ;       5489
001132  000420                      48$:   BR      52$
001134  013700  000000G             49$:   MOV     CCTLR,R0                        ;       5494
001140  006300                             ASL     R0
001142  105260  000000G                    INCB    C.ERR.TBL(R0)
001146  032737  000001  001254'            BIT     #1,APT.MODE                     ;       5496
001154  001405                             BEQ     51$
001156  012746  000055                     MOV     #55,-(SP)                       ;       5498
001162  004737  000000V             50$:   JSR     PC,ERR.HRD.RTNE.APT
001166  000404                             BR      53$                             ;       5496
001170  012746  000055              51$:   MOV     #55,-(SP)                       ;       5500
001174  004737  000000V             52$:   JSR     PC,ERR.HRD.RTNE
001200  005726                      53$:   TST     (SP)+                           ;       5493
001202  000207                             RTS     PC                              ;       5285

; Routine Size:  322 words,     Routine Base:  $CODE$ + 16166
; Maximum stack depth per invocation:  7 words


;    5589  1
```

```
;  1   GLOBAL routine UFD_IO_TALLY : novalue =
;  1
;  1   !+
;  1   !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;  1   !       PACKETS WITH "SUCCESS" STATUS CODES. ITS PURPOSE IS TO UPDATE ALL THE
;  1   !       APPROPRIATE STATISTICAL FIELDS FOR THE CURRENT UNIT. A CHECK IS ALSO
;  1   !       MADE ON THE TOTAL NUMBER OF BYTES TRANSFERRED THUS FAR; IF THE
;  1   !       OPERATOR-SPECIFIED LIMIT HAS BEEN REACHED, THEN THE UNIT IS DROPPED.
;  1   !
;  1   !       IMPLICIT INPUTS:
;  1   !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  1   !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  1   !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;  1   !               CUOFF - CST OFFSET FOR THE CURRENT UNIT
;  1   !               L$LUN - CURRENT UNIT NUMBER
;  1   !-
;  1
;  2       begin
;  2
;  2       local
;  2           THOUSANDS : word,
;  2           MILLIONS : word,                                   ! TOTAL NO. OF BYTES XFERRED TO/FROM A UNIT
;  2           FILL_CNT : word,                                   ! PARTIAL SECTOR FILL COUNT             MMM
;  2           PARTIAL_CNT : word,                                ! PARTIAL SECTOR TRANSFER COUNT          MMM
;  2           BYTES_PER_SECTOR : word initial (512),             ! RD AND RX BYTES PER SECTOR            MMM
;  2           INDEX : word;                                      ! UNIT NO.                              MMM
;  2
;  2       INDEX = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];      !POINT TO TYPER + TYPEW FOR THIS UNIT   MMM
;  2
;  2       PARTIAL_CNT = .RP_ADDR [BCNT_LO] MOD .BYTES_PER_SECTOR; ! PARTIAL SECTOR COUNT TRANSFERED      MMM
;  2
;  2       if .PARTIAL_CNT eql 0                                  ! CALCULATE ZERO FILL LENGHT ON         MMM
;  2       then FILL_CNT = 0                                      ! PARTIAL SECTOR TRANSFER AND
;  2       else FILL_CNT = .BYTES_PER_SECTOR - .PARTIAL_CNT;      ! ADD TO TALLY COUNT
;  2
;  3       if .RP_ADDR [ENDCOD] eql (OP_RD or OP_END)             ! IF ENDCODE IS READ
;  2       then
;  3           begin
;  3           TYPER [.INDEX] = .TYPER [.INDEX] + 1;              !INCREMENT READ COUNT                   ZZZ
MMM
;  3           T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1; ! INCREMENT NO. OF READS AND ADD BYTE COUNT
;  3           T_ADDR [BYTES_READ_LO] = .T_ADDR [BYTES_READ_LO] + .RP_ADDR [BCNT_LO] + .FILL_CNT ;        ! MMM

;  3           T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + .RP_ADDR [BCNT_LO] + .FILL_CNT ;      ! MMM
;  3           OVF_CHK (T_ADDR [TOT_READS_LO]);                   ! CHECK FOR FIELD OVERFLOW
;  3           OVF_CHK (T_ADDR [BYTES_READ_LO]);                  !
;  3           OVF_CHK (T_ADDR [TOT_BYT_RED_LO]);                 !
;  3           end
;  2       else
;  2
;  3           if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)        ! IF ENDCODE IS WRITE
;  2           then
;  3               begin
;  3               TYPEW [.INDEX] = .TYPEW [.INDEX] + 1;          !INCREMENT WRITE COUNT                  ZZZ
MMM
;  3               T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1;   ! INCREMENT NO. OF WRITES, ADD BYTE COUNT
```

```
;  3                T_ADDR [BYTES_WRIT_LO] = .T_ADDR [BYTES_WRIT_LO] + .RP_ADDR [BCNT_LO] + .FILL_CNT ;          ! MMM
;  3                T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + .RP_ADDR [BCNT_LO] + .FILL_CNT ;        ! MMM
;  3                OVF_CHK (T_ADDR [TOT_WRITES_LO]);                              ! CHECK FOR FIELD OVERFLOW
;  3                OVF_CHK (T_ADDR [BYTES_WRIT_LO]);                              !
;  3                OVF_CHK (T_ADDR [TOT_BYT_WRT_LO]);                             !
;  2                end;
;  2
;  2        if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) or
;  3           (.RP_ADDR [ENDCOD] eql (OP_WRT or OP_END))
;  2        then
;  3            begin
;  3            MILLIONS = .T_ADDR [MBYTES_READ] + .T_ADDR [MBYTES_WRIT];          ! TOTAL BYTES TRANSFERRED
;  3            THOUSANDS = .T_ADDR [BYTES_READ_HI] + .T_ADDR [BYTES_WRIT_HI];
;  3
;  3            if .THOUSANDS gequ 1000
;  3            then
;  4                begin
;  4                MILLIONS = .MILLIONS + 1;                                      ! COUNT THE LOWER OVERFLOW TOO!
;  4                THOUSANDS = .THOUSANDS - 1000;
;  3                end;
;  3
;  3
;  4    IF  (.TYPER [.INDEX] GEQU %o'100000' ) OR (.TYPEW [.INDEX] GEQU %O'100000')   ! RESET COUNTERS AT SOME      MMM
;  3    THEN                                                                          ! CONVENIENT POINT TO AVOID   MMM
;  4       BEGIN                                                                      ! OVERFLOWING                 MMM
;  4       TYPER [.INDEX] = 0;                                                        !                             MMM
;  4       TYPEW [.INDEX] = 0;                                                        !                             MMM
;  4       BAL_IN_PROGRESS [.INDEX] = 0;                                              !                             MMM
;  4       FORCE_WR [.INDEX] = 0;                                                     !                             MMM
;  3       END;                                                                       !                             MMM
;  3
;  4    IF .TYPER [.INDEX] GTRU ((2 * .TYPEW [.INDEX]) + 20)                          ! TOO MANY READS ?            MMM
;  3    THEN                                                                          !                             MMM
;  4       BEGIN                                                                      ! FORCE WRITE OPERATIONS      MMM
;  4       BAL_IN_PROGRESS [.INDEX] = TRUE;                                           !                             MMM
;  4       FORCE_WR [.INDEX] = TRUE;                                                  !                             MMM
;  3       END;                                                                       !                             MMM
;  3
;  3    IF ((2 * .TYPEW [.INDEX]) + 20)  GTRU .TYPER [.INDEX]                         ! TOO MANY WRITES ?           MMM
;  3    THEN                                                                          !                             MMM
;  4       BEGIN                                                                      ! FORCE READ OPERATIONS       MMM
;  4       BAL_IN_PROGRESS [.INDEX] = TRUE;                                           !                             MMM
;  4       FORCE_WR [.INDEX] = FALSE;                                                 !                             MMM
;  3       END;                                                                       !                             MMM
;  3
;  3    IF .TYPER [.INDEX] GTRU ((2 * .TYPEW [.INDEX]) + 5)  AND                      ! ENOUGH WRITES TO STOP       MMM
;  3       BAL_IN_PROGRESS [.INDEX]                                                   ! BALANCE PROCESS ?           MMM
;  3    THEN                                                                          !                             MMM
;  4       BEGIN                                                                      !                             MMM
;  4       BAL_IN_PROGRESS [.INDEX] = FALSE;                                          !                             MMM
;  4       FORCE_WR [.INDEX] = FALSE;                                                 !                             MMM
;  3       END;                                                                       !                             MMM
;  3
```

```
3   IF (  ((2 * .TYPEW [.INDEX]) + 5) GTRU .TYPER [.INDEX]  )  AND        ! ENOUGH READS TO STOP        MMM
3      BAL_IN_PROGRESS [.INDEX]                                           ! BALANCE PROCESS ?           MMM
3   THEN                                                                  !                             MMM
4      BEGIN                                                              !                             MMM
4      BAL_IN_PROGRESS [.INDEX] = FALSE;                                  !                             MMM
4      FORCE_WR [.INDEX] = FALSE;                                         !                             MMM
3      END;                                                               !                             MMM
3
3
3
3
3
3   !
3   ! THIS ADDED BECAUSE IT WILL TAKE FOREVER TO TRANSFER ON THE ORDER OF A MEGABYTE TO A FLOPPY
3   ! BUT IT IS A MUCH MORE REASONABLE MEASURE FOR THE RD51/52 WINCHESTER.  THE QUESTION NOW REFERS TO
3   ! THE TOTAL DATA TRANSFER TO THE CONTROLLER AND THIS IS PRETTY CLOSE SINCE THE FLOPPIES GET
3   ! ABOUT 1/1000 THE DATA THE HARD DISK(S) GET.
3   !
3
3           if .SWP_XFER eql 0                                        ! IF THERE IS A TRANSFER LIMIT
3           then
4               begin
4
4               if .THOUSANDS gtru 50                                 !ZZZ
4               then
4                   EOP_FLAG = TRUE;                                  ! SET END-OF-PASS FLAG
4
4               end
3           else
3
3               if .MILLIONS gequ .SWP_XFER                           ! IF TRANSFER LIMIT IS REACHED
3               then
3                   EOP_FLAG = TRUE;                                  ! SET END-OF-PASS FLAG
3
2           end;                                                      ! IF UNIT IS STILL ALIVE
2
2      !
2      !.....................................................................
2      !
2      ! THE FOLLOWING IS ADDED TO MAKE THE RUN TIME ABOUT 1.5 MINUTES FOR A  .
2      ! QUICK PASS IF ALL UNITS UNDER TEST ARE FLOPPIES.                     .
2      !.....................................................................
2
2   !!ZZZ   IF .RD_COUNT EQL 0                                            !IF THERE ARE NO WINCHESTERS   ZZZ
2   !!ZZZ      THEN                                                       !                              ZZZ
2   !!ZZZ        BEGIN                                                    !                              ZZZ
2   !!ZZZ        IF .THOUSANDS GTRU 44                                    !IF ABOUT 1.5 MINUTES GONE BY  ZZZ
2   !!ZZZ        THEN                                                     !                              ZZZ
2   !!ZZZ           EOP_FLAG = TRUE;                                      !SET THE END OF PASS FLAG      ZZZ
2   !!ZZZ        END;                                                     !                              ZZZ
2
2
2       ROUND_OUTPUT ();                                              ! ROUND TOTALS TO FIT PRINT POSITIONS
1       end;                                                          ! ROUTINE UPD_IO_TALLY
```

```
                                                  .SBTTL  UPD.IO.TALLY MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                 UPD.IO.TALLY::
                                                  JSR     R1,$SAVE5               ;                            5590
000004  012701  001000                            MOV     #1000,R1                ; *,BYTES.PER.SECTO          5607
000010  013700  000000G                           MOV     CUOFF,R0                ;                            5617
000014  006300                                    ASL     R0
000016  063700  000000G                           ADD     CST.ADDR,R0
000022  111003                                    MOVB    (R0),R3                 ; *,INDEX
000024  042703  177760                            BIC     #177760,R3              ; *,INDEX
000030  013700  000000G                           MOV     RP.ADDR,R0              ;                            5619
000034  016004  000020                            MOV     20(R0),R4
000040  010446                                    MOV     R4,-(SP)
000042  010146                                    MOV     R1,-(SP)                ; BYTES.PER.SECTO,*
000044  004737  000000G                           JSR     PC,BL$MOD
000050  005700                                    TST     R0                      ; PARTIAL.CNT                5621
000052  001002                                    BNE     1$
000054  005002                                    CLR     R2                      ; FILL.CNT                   5622
000056  000402                                    BR      2$                      ;                            5621
000060  010102                          1$:       MOV     R1,R2                   ; BYTES.PER.SECTO,FILL.CNT   5623
000062  160002                                    SUB     R0,R2                   ; PARTIAL.CNT,FILL.CNT
000064  013700  000000G                 2$:       MOV     RP.ADDR,R0              ;                            5625
000070  126027  000014  000241                    CMPB    14(R0),#241
000076  001040                                    BNE     3$
000100  010300                                    MOV     R3,R0                   ; INDEX,*                    5628
000102  006300                                    ASL     R0
000104  005260  000000G                           INC     TYPER(R0)
000110  013700  000000G                           MOV     T.ADDR,R0               ;                            5629
000114  005260  000016                            INC     16(R0)
000120  010401                                    MOV     R4,R1                   ;                            5630
000122  061001                                    ADD     (R0),R1
000124  060201                                    ADD     R2,R1                   ; FILL.CNT,*
000126  010110                                    MOV     R1,(R0)
000130  010401                                    MOV     R4,R1                   ;                            5631
000132  066001  000032                            ADD     32(R0),R1
000136  060201                                    ADD     R2,R1                   ; FILL.CNT,*
000140  010160  000032                            MOV     R1,32(R0)
000144  012716  000016                            MOV     #16,(SP)                ;                            5632
000150  060016                                    ADD     R0,(SP)
000152  004737  000000V                           JSR     PC,OVF.CHK
000156  013716  000000G                           MOV     T.ADDR,(SP)             ;                            5633
000162  004737  000000V                           JSR     PC,OVF.CHK
000166  013716  000000G                           MOV     T.ADDR,(SP)             ;                            5634
000172  062716  000032                            ADD     #32,(SP)
000176  000447                                    BR      4$
000200  126027  000014  000242          3$:       CMPB    14(R0),#242             ;                            5638
000206  001045                                    BNE     5$
000210  010300                                    MOV     R3,R0                   ; INDEX,*                    5641
000212  006300                                    ASL     R0
000214  005260  000000G                           INC     TYPEW(R0)
000220  013700  000000G                           MOV     T.ADDR,R0               ;                            5642
000224  005260  000024                            INC     24(R0)
```

```
000230  010401                                MOV     R4,R1                       ;                        5643
000232  066001  000006                        ADD     6(R0),R1
000236  060201                                ADD     R2,R1                       ; FILL.CNT.*
000240  010160  000006                        MOV     R1,6(R0)
000244  010401                                MOV     R4,R1                       ;                        5644
000246  066001  000040                        ADD     40(R0),R1
000252  060201                                ADD     R2,R1                       ; FILL.CNT.*
000254  010160  000040                        MOV     R1,40(R0)
000260  012716  000024                        MOV     #24,(SP)                    ;                        5645
000264  060016                                ADD     R0,(SP)
000266  004737  000000V                       JSR     PC,OVF.CHK
000272  013716  000000G                       MOV     T.ADDR,(SP)                 ;                        5646
000276  062716  000006                        ADD     #6,(SP)
000302  004737  000000V                       JSR     PC,OVF.CHK
000306  013716  000000G                       MOV     T.ADDR,(SP)                 ;                        5647
000312  062716  000040                        ADD     #40,(SP)
000316  004737  000000V              4$:      JSR     PC,OVF.CHK
000322  013700  000000G              5$:      MOV     RP.ADDR,R0                  ;                        5650
000326  126027  000014  000241                CMPB    14(R0),#241
000334  001404                                BEQ     6$
000336  126027  000014  000242                CMPB    14(R0),#242                 ;                        5651
000344  001134                                BNE     16$
000346  013700  000000G              6$:      MOV     T.ADDR,R0                   ;                        5654
000352  016005  000004                        MOV     4(R0),R5                    ; *.MILLIONS
000356  066005  000012                        ADD     12(R0),R5                   ; *.MILLIONS
000362  016004  000002                        MOV     2(R0),R4                    ; *.THOUSANDS            5655
000366  066004  000010                        ADD     10(R0),R4                   ; *.THOUSANDS
000372  020427  001750                        CMP     R4,#  50                    ; THOUSANDS.*            5657
000376  103403                                BLO     7$
000400  005205                                INC     R5                          ; MILLIONS               5660
000402  162704  001750                        SUB     #1750,R4                    ; *.THOUSANDS            5661
000406  010300                       7$:      MOV     R3,R0                       ; INDEX.*                5665
000410  006300                                ASL     R0
000412  012702  000000G                       MOV     #TYPER,R2
000416  060002                                ADD     R0,R2
000420  021227  100000                        CMP     (R2),#100000
000424  103004                                BHIS    8$
000426  026027  000000G 100000                CMP     TYPEW(R0),#100000
000434  103407                                BLO     9$
000436  005012                       8$:      CLR     (R2)                        ;                        5668
000440  005060  000000G                       CLR     TYPEW(R0)                   ;                        5669
000444  005060  000000G                       CLR     BAL.IN.PROGRESS(R0)         ;                        5670
000450  005060  000000G                       CLR     FORCE.WR(R0)                ;                        5671
000454  016003  000000G              9$:      MOV     TYPEW(R0),R3                ;                        5674
000460  006303                                ASL     R3
000462  010301                                MOV     R3,R1
000464  062701  000024                        ADD     #24,R1
000470  021201                                CMP     (R2),R1
000472  101406                                BLOS    10$
000474  012760  000001  000000G               MOV     #1,BAL.IN.PROGRESS(R0)      ;                        5677
000502  012760  000001  000000G               MOV     #1,FORCE.WR(R0)             ;                        5678
000510  020112                       10$:     CMP     R1,(R2)                     ;                        5681
000512  101405                                BLOS    11$
```

```
000514  012760  000001  000000G          MOV     #1,BAL.IN.PROGRESS(R0)     ;                    5684
000522  005060  000000G                  CLR     FORCE.WR(R0)               ;                    5685
000526  062703  000005          11$:     ADD     #5,R3                      ;                    5685
000532  021203                           CMP     (R2),R3
000534  101411                           BLOS    12$
000536  010001                           MOV     R0,R1                      ;                    5689
000540  062701  000000G                  ADD     #BAL.IN.PROGRESS,R1
000544  006001                           ROR     R1
000546  103004                           BCC     12$
000550  005060  000000G                  CLR     BAL.IN.PROGRESS(R0)        ;                    5692
000554  005060  000000G                  CLR     FORCE.WR(R0)               ;                    5693
000560  020312                  12$:     CMP     R3,(R2)                    ;                    5696
000562  101411                           BLOS    13$
000564  010001                           MOV     R0,R1                      ;                    5697
000566  062701  000000G                  ADD     #BAL.IN.PROGRESS,R1
000572  006001                           ROR     R1
000574  103004                           BCC     13$
000576  005060  000000G                  CLR     BAL.IN.PROGRESS(R0)        ;                    5700
000602  005060  000000G                  CLR     FORCE.WR(R0)               ;                    5701
000606  013700  000000G         13$:     MOV     SWP.XFER,R0                ;                    5715
000612  001004                           BNE     14$
000614  020427  000062                   CMP     R4,#62                     ; THOUSANDS,*        5719
000620  101406                           BLOS    16$
000622  000402                           BR      15$                        ;                    5721
000624  020500                  14$:     CMP     R5,R0                      ; MILLIONS,*         5726
000626  103403                           BLO     16$
000630  112737  000001  000000G 15$:     MOVB    #1,EOP.FLAG                ;                    5728
000636  004737  000000V         16$:     JSR     PC,ROUND.OUTPUT            ;                    5747
000642  022626                           CMP     (SP)+,(SP)+                ;                    5607
000644  000207                           RTS     PC                         ;                    5590

; Routine Size:  211 words,     Routine Base:  $CODE$ + 17372
; Maximum stack depth per invocation:  9 words
```

```
;   5749  1     GLOBAL routine OVF_CHK (ADDR) : novalue =
;   5750  1     !+
;   5751  1     !       THIS ROUTINE IS CALLED FROM UPD_IO_TALLY TO CHECK FOR OVERFLOW IN
;   5752  1     !       CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
;   5753  1     !       LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
;   5754  1     !       EXCEEDING 1000. IF TRUE, THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
;   5755  1     !       THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IN INCREMENTED.
;   5756  1     !
;   5757  1     !       INPUTS:
;   5758  1     !               ADDR - ADDRESS OF THE BYTES_READ_LO OR BYTES_WRIT_LO FIELD FOR
;   5759  1     !                       THE CURRENT UNIT (SEE STATISTIC TABLE (TALLY) LAYOUT)
;   5760  1     !-
;   5761  2         begin
;   5762  2
;   5763  2         while ..ADDR gequ 1000 do                     ! IF LO-ORDER OVERFLOW
;   5764  3             begin
;   5765  3             .ADDR = ..ADDR - 1000;                    ! SUBTRACT 1000
;   5766  3             (.ADDR + 2) = .(.ADDR + 2) + 1;           ! INCR HI-ORDER
;   5767  2             end;
;   5768  2
;   5769  2         if .(.ADDR + 2) gequ 1000                     ! IF HI-ORDER OVERFLOW
;   5770  2         then
;   5771  3             begin
;   5772  3             (.ADDR + 2) = .(.ADDR + 2) - 1000;        ! SUBTRACT 1000
;   5773  3             (.ADDR + 4) = .(.ADDR + 4) + 1;           ! INCREMENT MBYTES
;   5774  2             end;
;   5775  2
;   5776  1         end;                                          ! ROUTINE OVF_CHK

                                          .SBTTL  OVF.CHK MULTI-DRIVE TEST ROUTINES
000000  010146                  OVF.CHK::
                                          MOV     R1,-(SP)               ;                 5749
000002  016600  000004                    MOV     4(SP),R0               ; ADDR,*          5763
000006  012701  000002                    MOV     #2,R1                  ;                 5766
000012  060001                            ADD     R0,R1
000014  021027  001750          1$:       CMP     (R0),#1750             ;                 5763
000020  103404                            BLO     2$
000022  162710  001750                    SUB     #1750,(R0)             ;                 5765
000026  005211                            INC     (R1)                   ;                 5766
000030  000771                            BR      1$                     ;                 5763
000032  021127  001750          2$:       CMP     (R1),#1750             ;                 5769
000036  103404                            BLO     3$
000040  162711  001750                    SUB     #1750,(R1)             ;                 5772
000044  005260  000004                    INC     4(R0)                  ;                 5773
000050  012601                  3$:       MOV     (SP)+,R1               ;                 5749
000052  000207                            RTS     PC

; Routine Size:  22 words,      Routine Base:  $CODE$ + 20240
; Maximum stack depth per invocation:  2 words
```

```
GLOBAL routine ROUND_OUTPUT : novalue =

!+
!       THIS ROUTINE ROUNDS THE TOTALS TO FIT PRINT POSITIONS.
!-

    begin

    if .T_ADDR [TOT_READS_HI] gtru 9999
    then
        begin

        if .T_ADDR [TOT_READS_LO] lssu 999
        then
            begin
            T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 1;
            T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1000;
            end;

        T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] - 999;
        T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 9999;
        end;

    if .T_ADDR [TOT_WRITES_HI] gtru 9999
    then
        begin

        if .T_ADDR [TOT_WRITES_LO] lssu 999
        then
            begin
            T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 1;
            T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1000;
            end;

        T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] - 999;
        T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 9999;
        end;

    if .T_ADDR [MTOT_BYT_RED] gtru 999
    then
        begin

        if .T_ADDR [TOT_BYT_RED_HI] lssu 999
        then
            begin
            T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
            T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
            end;

        if .T_ADDR [TOT_BYT_RED_LO] lssu 999
        then
            begin
            T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 1;
```

```
;   4                   T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + 1000;
;   4
;   4                   if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;   4                   then
;   5                       begin
;   5                       T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;   5                       T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;   4                       end;
;   3                   end;
;   3
;   3               T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] - 999;
;   3               T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 999;
;   3               T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 999;
;   2               end;
;   2
;   2           if .T_ADDR [MTOT_BYT_WRT] gtru 999
;   2           then
;   3               begin
;   3
;   3               if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;   3               then
;   4                   begin
;   4                   T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;   4                   T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;   3                   end;
;   3
;   3               if .T_ADDR [TOT_BYT_WRT_LO] lssu 999
;   3               then
;   4                   begin
;   4                   T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 1;
;   4                   T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + 1000;
;   4
;   4                   if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;   4                   then
;   5                       begin
;   5                       T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;   5                       T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;   4                       end;
;   3                   end;
;   3
;   3               T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] - 999;
;   3               T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 999;
;   3               T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 999;
;   2               end;
;   2
;   1       end;

                                         .SBTTL  ROUND.OUTPUT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G         ROUND.OUTPUT::
                                         JSR     R1,$SAVE3                       ;
000004  013700  000000G                  MOV     T.ADDR,R0                       ;
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                  ;      5789
000030  060001                       ADD     R0,R1
000032  021127  001747               CMP     (R1),#1747
000036  103003                       BHIS    1$
000040  005312                       DEC     (R2)                    ;      5792
000042  062711  001750               ADD     #1750,(R1)              ;      5793
000046  162711  001747       1$:     SUB     #1747,(R1)              ;      5796
000052  162712  023417               SUB     #23417,(R2)             ;      5797
000056  012702  000026       2$:     MOV     #26,R2                  ;      5800
000062  060002                       ADD     R0,R2
000064  021227  023417               CMP     (R2),#23417
000070  101415                       BLOS    4$
000072  012701  000024               MOV     #24,R1                  ;      5804
000076  060001                       ADD     R0,R1
000100  021127  001747               CMP     (R1),#1747
000104  103003                       BHIS    3$
000106  005312                       DEC     (R2)                    ;      5807
000110  062711  001750               ADD     #1750,(R1)              ;      5808
000114  162711  001747       3$:     SUB     #1747,(R1)              ;      5811
000120  162712  023417               SUB     #23417,(R2)             ;      5812
000124  012703  000036       4$:     MOV     #36,R3                  ;      5815
000130  060003                       ADD     R0,R3
000132  021327  001747               CMP     (R3),#1747
000136  101436                       BLOS    7$
000140  012701  000034               MOV     #34,R1                  ;      5819
000144  060001                       ADD     R0,R1
000146  021127  001747               CMP     (R1),#1747
000152  103003                       BHIS    5$
000154  005313                       DEC     (R3)                    ;      5822
000156  062711  001750               ADD     #1750,(R1)              ;      5823
000162  012702  000032       5$:     MOV     #32,R2                  ;      5826
000166  060002                       ADD     R0,R2
000170  021227  001747               CMP     (R2),#1747
000174  103011                       BHIS    6$
000176  005311                       DEC     (R1)                    ;      5829
000200  062712  001750               ADD     #1750,(R2)              ;      5830
000204  021127  001747               CMP     (R1),#1747              ;      5832
000210  103003                       BHIS    6$
000212  005313                       DEC     (R3)                    ;      5835
000214  062711  001750               ADD     #1750,(R1)              ;      5836
000220  162712  001747       6$:     SUB     #1747,(R2)              ;      5840
000224  162711  001747               SUB     #1747,(R1)              ;      5841
000230  162713  001747               SUB     #1747,(R3)              ;      5842
000234  012702  000044       7$:     MOV     #44,R2                  ;      5845
000240  060002                       ADD     R0,R2
000242  021227  001747               CMP     (R2),#1747
000246  101435                       BLOS    10$
000250  012701  000042               MOV     #42,R1                  ;      5849
000254  060001                       ADD     R0,R1
000256  021127  001747               CMP     (R1),#1747
```

```
000262  103003                    BHIS    8$
000264  005312                    DEC     (R2)                    ;       5852
000266  062711  001750            ADD     #1750,(R1)              ;       5853
000272  062700  000040     8$:    ADD     #40,R0                  ;       5856
000276  021027  001747            CMP     (R0),#1747
000302  103011                    BHIS    9$
000304  005311                    DEC     (R1)                    ;       5859
000306  062710  001750            ADD     #1750,(R0)              ;       5860
000312  021127  001747            CMP     (R1),#1747              ;       5862
000316  103003                    BHIS    9$
000320  005312                    DEC     (R2)                    ;       5865
000322  062711  001750            ADD     #1750,(R1)              ;       5866
000326  162710  001747     9$:    SUB     #1747,(R0)              ;       5870
000332  162711  001747            SUB     #1747,(R1)              ;       5871
000336  162712  001747            SUB     #1747,(R2)              ;       5872
000342  000207            10$:    RTS     PC                      ;       5777

; Routine Size:  114 words,     Routine Base:  $CODE$ + 20314
; Maximum stack depth per invocation:  5 words
```

```
GLOBAL routine HOST_WRT_CHK =

!+
!       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
!       PACKETS WITH "SUCCESS" STATUS CODES, BUT ONLY IF THE HOST WRITE-COMPARE
!       OPTION WAS SELECTED BY THE OPERATOR.
!
!       IF THE CURRENT RETPKT BEING PROCESSED IS A WRITE FUNCTION, THEN THE
!       PACKET INDEX (RP_INDX) IS SAVED IN THE CONTROLLER'S RETURN PACKET SAVE
!       AREA (RP_SAVE). OTHERWISE, THE PACKET IS A READ, SO ITS ASSOCIATED
!       WRITE PACKET IS REMOVED FROM THE SAVE AREA, AND A BYTE-BY-BYTE
!       COMPARISON IS PERFORMED ON THE TWO I/O BUFFERS. ANY DIFFERENCES
!       ENCOUNTERED RESULTS IN THE DECLARATION OF A HARD ERROR.
!
!       IMPLICIT INPUTS:
!               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
!               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
!-

    begin

    local
        BUFF1 : ref block [MAX_XFER_SIZE, byte],            ! I/O BUFFER ADDRESS
        BUFF2 : ref block [MAX_XFER_SIZE, byte],            ! I/O BUFFER ADDRESS
        BUFFW,                                              ! I/O BUFFER ADDRESS
        COUNT : word,                                       ! BYTE COUNT
        FLAG : byte initial (byte (TRUE)),
        index : signed word;

    if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)             ! IF WRITE OPERATION
    then
        FLAG = FALSE                                        ! DON'T CALL SWEEP FROM IO_RETPKT
    else

        if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) and
          ((index = RPS_REM ()) geq 0)                      ! IF ASSOCIATED WRITE PACKET IS FOUND
        then
            begin
            BUFFW = RETPKT [.index, BUFF_0];                ! ADDR OF ADDR OF WRITE I/O BUFFER
            BUFF1 = ..BUFFW;                                ! ADDR OF WRITE I/O BUFFER
            BUFF2 = .RP_ADDR [BUFF_0];                      ! ADDR OF READ I/O BUFFER
            COUNT = .RP_ADDR [BCNT_LO];                     ! BYTE COUNT

            incr I from 1 to .COUNT do                      ! FOR EACH BYTE IN BUFFERS

                if .(.BUFF1)<0, 8, 0> eql .(.BUFF2)<0, 8, 0>    ! IF BYTES COMPARE O.K.
                then
                    begin
                    BUFF1 = .BUFF1 + 1;                     ! ADVANCE WRITE BUFFER ADDR
                    BUFF2 = .BUFF2 + 1;                     ! ADVANCE READ BUFFER ADDR
                    end
                else
                    begin                                   ! ELSE - COMPARE ERROR
```

```
;     5929  4                    T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;
;     5930  4                    T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;     5931  4
;     5932  4                    EMS_CMP (RETPKT + (.index * RP_LEN * 2));    !MMM PRINT ERROR INFORMATION BEFORE
;     5933  4                                                                 ! HARD ERROR CALL
;     5934  4                                                                 ! IF FLAG:HOE IS SET, THEN ERROR INFORMATION
;     5935  4                                                                 ! IS PRINTED BEFORE HALT !
;     5936  4                    if .APT_MODE
;     5937  4                    then
;     5938  4                        ERR_HRD_RTNE_APT (42)                    ! I/O REQUEST FAILED
;     5939  4                    else
;     5940  4                        ERR_HRD_RTNE (42);
;     5941  4
;     5942  4       !MMM         EMS_CMP (RETPKT + (.index * RP_LEN * 2));
;     5943  4
;     5944  4                    if .T_ADDR [ERR_HARD] gequ .SWP_ERROR
;     5945  4                    then
;     5946  5                        begin
;     5947  5                        DUR [.L$LUN] = DU_HERR;                  ! IF ERROR COUNT EXCEEDED
;     5948  5                        DODU (.L$LUN);                           ! DROP UNIT
;     5949  4                        end;
;     5950  4
;     5951  4                    exitloop;                                    ! NO NEED 10 CONTINUE
;     5952  3                    end;                                         ! IF COMPARE ERROR
;     5953  3
;     5954  2                end;                                             ! IF ASSOCIATED WRITE RETPKT WAS FOUND
;     5955  2
;     5956  2       return (.FLAG);
;     5957  1       end;                                                     ! RCUTINE HOST_WRT_CHK


                                                .SBTTL  HOST.WRT.CHK MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                 HOST.WRT.CHK::
                                                JSR     R1,$SAVES                       ;                               5876
000004  005746                                  TST     -(SP)
000006  112705  000001                          MOVB    #1,R5                           ; *,FLAG                        5895
000012  013700  000000G                         MOV     RP.ADDR,R0                      ;                               5905
000016  126027  000014  000242                  CMPB    14(R0),#242
000024  001002                                  BNE     1$
000026  105005                                  CLRB    R5                              ; FLAG                          5907
000030  000511                                  BR      8$                              ;                               5905
000032  126027  000014  000241          1$:     CMPB    14(R0),#241                     ;                               5910
000040  001105                                  BNE     8$
000042  004737  000000V                         JSR     PC,RPS.REM                      ;                               5911
000046  005700                                  TST     R0                              ; INDEX
000050  002501                                  BLT     8$
000052  010046                                  MOV     R0,-(SP)                        ; INDEX,*                       5914
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                   5915
000076  013700  000000G                         MOV     RP.ADDR,R0                      ;                               5916
```

```
000102  016002  000024                  MOV     24(R0),R2                       ; *,BUFF2
000106  016004  000020                  MOV     20(R0),R4                       ; *,COUNT                       5917
000112  005003                          CLR     R3                              ; I                             5919
000114  000453                          BR      6$
000116  121112                  2$:     CMPB    (R1),(R2)                       ; BUFF1,BUFF2                   5921
000120  001003                          BNE     3$
000122  005201                          INC     R1                              ; BUFF1                         5924
000124  005202                          INC     R2                              ; BUFF2                         5925
000126  000446                          BR      6$                              ;                               5921
000130  013700  000000G         3$:     MOV     T.ADDR,R0                       ;                               5929
000134  005260  000014                  INC     14(R0)
000140  105260  000051                  INCB    51(R0)                          ;                               5930
000144  016616  000004                  MOV     4(SP),(SP)                      ;                               5932
000150  062716  000000G                 ADD     #RETPKT,(SP)
000154  004737  000000G                 JSR     PC,EMS.CMP
000160  032737  000001  001254'         BIT     #1,APT.MODE                     ;                               5936
000166  001405                          BEQ     4$
000170  012716  000052                  MOV     #52,(SP)                        ;                               5938
000174  004737  000000V                 JSR     PC,ERR.HRD.RTNE.APT
000200  000404                          BR      5$                              ;                               5936
000202  012716  000052          4$:     MOV     #52,(SP)                        ;                               5940
000206  004737  000000V                 JSR     PC,ERR.HRD.RTNE
000212  013700  000000G         5$:     MOV     T.ADDR,R0                       ;                               5944
000216  026037  000014  000000G         CMP     14(R0),SWP.ERROR
000224  103412                          BLO     7$
000226  013700  000000G                 MOV     L$LUN,R0                        ;                               5947
000232  112760  000004  000000G         MOVB    #4,DUR(R0)
000240  104451                          TRAP    51                              ;                               5948
000242  000403                          BR      7$                              ;                               5928
000244  005203                  6$:     INC     R3                              ; I                             5919
000246  020304                          CMP     R3,R4                           ; I,COUNT
000250  003722                          BLE     2$
000252  022626                  7$:     CMP     (SP)+,(SP)+                     ;                               5913
000254  005000                  8$:     CLR     R0                              ;                               5956
000256  150500                          BISB    R5,R0                           ; FLAG,*
000260  005726                          TST     (SP)+                           ;                               5876
000262  000207                          RTS     PC

; Routine Size:  90 words,      Routine Base:  $CODE$ + 20660
; Maximum stack depth per invocation:  11 words
```

```
;    5958  1   GLOBAL routine SWEEP : novalue =
;    5959  1
;    5960  1   !+
;    5961  1   !       THIS ROUTINE IS CALLED FROM IO_RETPKT AND OTHERS TO DEALLOCATE THE
;    5962  1   !       RESOURCES ASSOCIATED WITH THE CURRENT RETURN PACKET. THIS INCLUDES THE
;    5963  1   !       PACKET ITSELF AND THE I/O BUFFER. IN ADDITION, IF THE HOST IS
;    5964  1   !       PERFORMING WRITE-COMPARES, AND IF THE CURRENT RETURN PACKET IS A READ
;    5965  1   !       FUNCTION, THEN THE CURRENT CONTROLLER'S RP_SAVE AREA IS SEARCHED FOR
;    5966  1   !       THE ASSOCIATED WRITE RETPKT SO THAT ITS RESOURCES CAN ALSO BE
;    5967  1   !       DEALLOCATED.
;    5968  1   !
;    5969  1   !       IMPLICIT INPUTS:
;    5970  1   !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    5971  1   !               RP_INDX - INDEX OF CURRENT RETURN PACKET
;    5972  1   !-
;    5973  1
;    5974  2       begin
;    5975  2
;    5976  2       local
;    5977  2           index : signed word;
;    5978  2
;    5979  2       if (.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD         ! IF READ OPCODE OR ENDCODE
;    5980  2       then
;    5981  2
;    5982  3           if BIT_TST (SWP_FLAGS, SWF_HWC)                 ! IF HOST IS DOING WRITE-COMPARES
;    5983  2           then
;    5984  2
;    5985  2               if (index = RPS_REM ()) geq 0               ! IF ASSOCIATED WRITE RETPKT IS FOUND
;    5986  2               then
;    5987  3                   begin
;    5988  3                   PUT_IO_BUFF (RETPKT [.index, BUFF_0]);  ! RETURN WRITE I/O BUFFER TO POOL
;    5989  3                   PUT_RETPKT (.index);                    ! RETURN WRITE PACKET TO POOL
;    5990  2                   end;
;    5991  2
;    5992  2       PUT_IO_BUFF (RP_ADDR [BUFF_0]);                     ! RETURN CURRENT I/O BUFFER TO POOL
;    5993  2       PUT_RETPKT (.RP_INDX);                              ! RETURN CURRENT RETPKT TO POOL
;    5994  1       end;                                                ! ROUTINE SWEEP

                                          .SBTTL  SWEEP MULTI-DRIVE TEST ROUTINES
000000  010146                     SWEEP:: MOV     R1,-(SP)                ;                   5958
000002  013700  000000G                    MOV     RP.ADDR,R0              ;                   5979
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           ;                   5982
000032  001422                             BEQ     1$
000034  004737  000000V                    JSR     PC,RPS.REM              ;                   5985
000040  010001                             MOV     R0,R1                   ; *,INDEX
000042  002416                             BLT     1$
000044  010146                             MOV     R1,-(SP)                ; INDEX,*           5988
000046  012746  000054                     MOV     #54,-(SP)
```

```
000052  004737  000000G          JSR     PC,BL$MUL
000056  062700  000024G          ADD     #RETPKT+24,R0
000062  010016                   MOV     R0,(SP)
000064  004737  000000G          JSR     PC,PUT.IO.BUFF
000070  010116                   MOV     R1,(SP)              ; INDEX,*             5989
000072  004737  000000G          JSR     PC,PUT.RETPKT
000076  022626                   CMP     (SP)+,(SP)+          ;                     5987
000100  013746  000000G    1$:   MOV     RP.ADDR,-(SP)        ;                     5992
000104  062716  000024           ADD     #24,(SP)
000110  004737  000000G          JSR     PC,PUT.IO.BUFF
000114  013716  000000G          MOV     RP.INDX,(SP)         ;                     5993
000120  004737  000000G          JSR     PC,PUT.RETPKT
000124  005726                   TST     (SP)+                ;                     5974
000126  012601                   MOV     (SP)+,R1             ;                     5958
000130  000207                   RTS     PC

; Routine Size:  45 words,     Routine Base:  $CODE$ + 21144
; Maximum stack depth per invocation:  4 words
```

```
;   5995  1     GLOBAL routine RPS_REM =
;   5996  1
;   5997  1     !+
;   5998  1     !       THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP_SAVE AREA FOR A
;   5999  1     !       RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
;   6000  1     !       CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
;   6001  1     !       OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
;   6002  1     !       RP_SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
;   6003  1     !       OPERATION IS RETURNED TO THE CALLER.
;   6004  1     !
;   6005  1     !       IMPLICIT INPUTS:
;   6006  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;   6007  1     !
;   6008  1     !       OUTPUTS:
;   6009  1     !               INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
;   6010  1     !                               A CRN WHICH IS ONE LESS THAN THE CURRENT
;   6011  1     !-
;   6012  1
;   6013  2         begin
;   6014  2
;   6015  2         local
;   6016  2             index : signed word initial (-1);                               ! ASSUME NOT FOUND
;   6017  2
;   6018  2         incr COUNT from 0 to RP_CNT - 1 do                                  ! FOR EACH ENTRY IN RP_SAVE
;   6019  2
;   6020  2             if (.RP_USE [.COUNT] eql .CCTLR) and                            ! IF THIS IS A VALID RETPKT INDEX
;   6021  3                (.RETPKT [.COUNT, ENDCOD] eql (OP_WRT or OP_END))
;   6022  2             then
;   6023  2
;   6024  3                 if ((.RETPKT [.COUNT, CRF_LO] eql (.RP_ADDR [CRF_LO] - 1)) and      ! IF CORRECT CRN
;   6025  2                     (.RETPKT [.COUNT, CRF_HI] eql .RP_ADDR [CRF_HI])) or
;   6026  3                    ((.RETPKT [.COUNT, CRF_HI] eql (.RP_ADDR [CRF_HI] - 1)) and
;   6027  3                     (.RETPKT [.COUNT, CRF_LO] eql %o'177777') and
;   6028  3                     (.RP_ADDR [CRF_LO] eql 0))
;   6029  2                 then
;   6030  3                     begin
;   6031  3                     index = .COUNT;                                         ! INDEX TO BE RETURNED
;   6032  3                     exitloop;                                               ! DONE
;   6033  2                     end;
;   6034  2
;   6035  2         return .index;
;   6036  1         end;                                                                ! ROUTINE RPS REM

                                                .SBTTL  RPS.REM MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                 RPS.REM::
                                                JSR     R1,$SAVE4                       ;                       5995
000004  012704  177777                          MOV     #-1,R4                          ; *,INDEX               6013
000010  005003                                  CLR     R3                              ; COUNT                 6018
000012  116300  000000G                 1$:     MOVB    RP.USE(R3),R0                   ; *(COUNT),*            6020
000016  020037  000000G                         CMP     R0,CCTLR
000022  001053                                  BNE     4$
000024  010346                                  MOV     R3,-(SP)                        ; COUNT,*               6021
```

```
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,*                6024
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)       ;                        6025
000112  001415                         BEQ     3$
000114  016102  000006          2$:    MOV     6(R1),R2                 ;                        6026
000120  005302                         DEC     R2
000122  026002  000006G                CMP     RETPKT+6(R0),R2
000126  001011                         BNE     4$
000130  026027  000004G 177777         CMP     RETPKT+4(R0),#-1         ;                        6027
000136  001005                         BNE     4$
000140  005761  000004                 TST     (R1)                     ;                        6028
000144  001002                         BNE     4$
000146  010304                  3$:    MOV     R3,R4                    ; COUNT,INDEX            6031
000150  000404                         BR      5$                       ;                        6030
000152  005203                  4$:    INC     R3                       ; COUNT                  6018
000154  020327  000007                 CMP     R3,#7                    ; COUNT,*
000160  003714                         BLE     1$
000162  010400                  5$:    MOV     R4,R0                    ; INDEX,*                6013
000164  000207                         RTS     PC                       ;                        5995

; Routine Size:  59 words,      Routine Base:  $CODE$ + 21276
; Maximum stack depth per invocation:  8 words
```

```
;  1     GLOBAL routine DR_RETPKT : novalue =
;  1
;  1     !+
;  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT FOR ALL PACKETS ORIGINATING AT
;  1     !       THE "DRIVER" PORTION OF THE PROGRAM. THIS INCLUDES PACKETS DESCRIBING
;  1     !       FATAL DEVICE ERRORS.
;  1     !
;  1     !       FOR FATAL DEVICE ERRORS, THIS ROUTINE RELEASES ALL RESOURCES HELD BY
;  1     !       THE CONTROLLER. THE CONTROLLER IS MARKED OFFLINE IN ITS CST, AND ALL
;  1     !       UNITS ATTACHED TO THE CONTROLLER ARE DROPPED.
;  1     !
;  1     !       IMPLICIT INPUTS:
;  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  1     !-
;  1
;  2         begin
;  2
;  2
;  2         PUTA_BUFF ();                          ! RELEASE ALL I/O BUFFERS HELD BY CONTROLLER
;  2
;  2         incr index from 0 to RP_CNT - 1 do     ! FOR EACH ENTRY IN CONTROLLER'S RP_SAVE
;  2
;  2             if .RP_USE [.index] eql .CCTLR     ! IF VALID RETPKT INDEX
;  2             then
;  2                 PUT_RETPKT (.index);           ! RETURN RETPKT TO POOL
;  2
;  2         QIO [.CCTLR] = 0;                      ! CLEAR NO. OF OUTSTANDING QIOs
;  2         CST_ADDR [STATE] = OFFLINE;            ! MARK CST OFFLINE
;  2         DROP_CTLR (.CCTLR, DU_CFATAL);         ! DROP CONTROLLER'S UNITS
;  2         PUT_RETPKT (.RP_INDX);                 ! PUT BACK RETPKT
;  1         end;                                   ! ROUTINE DR_RETPKT

                                   .SBTTL  DR.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                DR.RETPKT::
                                   MOV     R1,-(SP)                 ;                     6037
000002  004737  000000G            JSR     PC,PUTA.BUFF             ;                     6058
000006  005001                     CLR     R1                       ; INDEX               6060
000010  116100  000000G       1$:  MOVB    RP.USE(R1),R0            ; *(INDEX),*          6062
000014  020037  000000G            CMP     R0,CCTLR
000020  001004                     BNE     2$
000022  010146                     MOV     R1,-(SP)                 ; INDEX,*             6064
000024  004737  000000G            JSR     PC,PUT.RETPKT
000030  005726                     TST     (SP)+
000032  005201                2$:  INC     R1                       ; INDEX               6060
000034  020127  000007             CMP     R1,#7                    ; INDEX,*
000040  003763                     BLE     1$
000042  013701  000000G            MOV     CCTLR,R1                 ;                     6066
000046  105061  000000G            CLRB    QIO(R1)
000052  013700  000000G            MOV     CST.ADDR,R0              ;                     6067
```

```
000056  042760  100000  000002          BIC     #100000,2(R0)
000064  010146                          MOV     R1,-(SP)                      ;          6068
000066  012746  000006                  MOV     #6,-(SP)
000072  004737  000000G                 JSR     PC,DROP.CTLR
000076  013716  000000G                 MOV     RP.INDX,(SP)                  ;          6069
000102  004737  000000G                 JSR     PC,PUT.RETPKT
000106  022626                          CMP     (SP)+,(SP)+                   ;          6055
000110  012601                          MOV     (SP)+,R1                      ;          6037
000112  000207                          RTS     PC

; Routine Size:  38 words,      Routine Base:  $CODE$ + 21464
; Maximum stack depth per invocation:  4 words
```

```
;     6071  1     %sbttl 'RDRX INTERRUPT SERVICE ROUTINES'
;     6072  1
;     6073  1     !+
;     6074  1     !        THERE EXISTS AN RDRX INTERRUPT SERVICE ROUTINE FOR EACH DEVICE
;     6075  1     !        CONTROLLER. EACH SERVICE ROUTINE BEGINS BY SIMPLY SETTING THE
;     6076  1     !        APPROPRIATE CONTROLLER NUMBER INTO "ICTLR"  ALL SERVICE ROUTINES THEN
;     6077  1     !        BRANCH TO A COMMON INTERRUPT PROCESSING ROUTINE.
;     6078  1     !-
;     6079  1
;     6080  2     BGNSRV (AZINTO);
;     6081  2     ICTLR = 0;
;     6082  2     AZINT ();
;     6083  1     ENDSRV;


                                                .SBTTL  AZINTO RDRX INTERRUPT SERVICE ROUTINES
000000  010046                         AZINTO::MOV      RO,-(SP)                         ;          6080
000002  005037  000104'                        CLR      ICTLR                            ;          6081
000006  004737  C00000V                        JSR      PC,AZINT                         ;          6082
000012  012600                                 MOV      (SP)+,RO                         ;          6080
000014  000002                                 RTI

; Routine Size:  7 words,        Routine Base:  $CODE$ + 21600
; Maximum stack depth per invocation:  2 words
```

```
;   6084  1     GLOBAL routine AZINT : novalue =
;   6085  1
;   6086  1     !+
;   6087  1     !       THIS IS THE COMMON INTERRUPT SERVICE ROUTINE FOR ALL RDRX CONTROLLERS.
;   6088  1     !       AFTER CALCULATING THE DCT ADDRESS FOR THE INTERRUPTING DEVICE, THIS
;   6089  1     !       ROUTINE WILL SAVE THE CURRENT CONTENTS OF THE SA REGISTER IN THE DCT.
;   6090  1     !       THEN, IF THE "IGNORE INTERRUPT" BIT IS SET, NO FURTHER ACTION IS TAKEN.
;   6091  1     !       OTHERWISE, THE SA VALUE IS CHECKED FOR A FATAL ERROR, AND THE COMMAND
;   6092  1     !       AND RESPONSE RINGS ARE POLLED.
;   6093  1     !-
;   6094  1
;   6095  2         begin
;   6096  2         IDCT_ADDR = DCT + (.ICTLR * DCT_LEN * 2);              ! GET DCT ADDRESS
;   6097  2         ICST_ADDR = CST + (.ICTLR * CST_LEN * 2);              ! GET CST ADDRESS
;   6098  2         IRDRX_ADDR = .ICST_ADDR [IP_ADDR];                      ! GET RDRX ADDRESS
;   6099  2         ICOM_ADDR = COMM_AREA + (.ICTLR * COMM_LEN * 2);       ! GET COMM_AREA ADDR
;   6100  2         IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];      ! SAVE SA REGISTER
;   6101  2
;   6102  2         if .IDCT_ADDR [IG_INT]                                  ! IGNORE INTERRUPT?
;   6103  2         then
;   6104  2             return;                                             ! RETURN IF INTERRUPTS IGNORED
;   6105  2
;   6106  3         if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)               ! IF FATAL ERROR
;   6107  2         then
;   6108  2             FATAL_ERROR ()
;   6109  2         else
;   6110  3             begin
;   6111  3             POLL_CRING ();                                      ! POLL COMMAND RING
;   6112  3             POLL_RRING ();                                      ! POLL RESPONSE RING
;   6113  2             end;
;   6114  2
;   6115  1         end;


                                        .SBTTL  AZINT RDRX INTERRUPT SERVICE ROUTINES
000000  010146                  AZINT:: MOV     R1,-(SP)                        ;                       6084
000002  005746                          TST     -(SP)
000004  013701  000104'                 MOV     ICTLR,R1                        ;                       6096
000010  010146                          MOV     R1,-(SP)
000012  012746  000022                  MOV     #22,-(SP)
000016  004737  000000G                 JSR     PC,BL$MUL
000022  062700  000000G                 ADD     #DCT,R0
000026  010037  000100'                 MOV     R0,IDCT.ADDR
000032  010116                          MOV     R1,(SP)                         ;                       6097
000034  012746  000126                  MOV     #126,-(SP)
000040  004737  000000G                 JSR     PC,BL$MUL
000044  062700  000000G                 ADD     #CST,R0
000050  010037  000076'                 MOV     R0,ICST.ADDR
000054  011037  000000G                 MOV     (R0),IRDRX.ADDR                 ; ICST.ADDR,*           6098
000060  010116                          MOV     R1,(SP)                         ;                       6099
000062  012746  000050                  MOV     #50,-(SP)
000066  004737  000000G                 JSR     PC,BL$MUL
000072  062700  000000'                 ADD     #COMM.AREA,R0
```

```
000076  010037  000074'                MOV     R0,ICOM.ADDR
000102  013701  000100'                MOV     IDCT.ADDR,R1                          ;                              6100
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     #4000 ,(R1)                           ; *,IDCT.ADDR                  6102
000132  001016                         BNE     2$                                    ;                              6084
000134  016601  000010                 MOV     10(SP),R1                             ;                              6106
000140  042701  077777                 BIC     #77777,R1
000144  020127  100000                 CMP     R1,#-100000
000150  001003                         BNE     1$
000152  004737  000000V                JSR     PC,FATAL.ERROR                        ;                              6108
000156  000404                         BR      2$                                    ;                              6106
000160  004737  000000V        1$:     JSR     PC,POLL.CRING                         ;                              6111
000164  004737  000000V                JSR     PC,POLL.RRING                         ;                              6112
000170  062706  000012         2$:     ADD     #12,SP                                ;                              6084
000174  012601                         MOV     (SP)+,R1
000176  000207                         RTS     PC

; Routine Size:  64 words,      Routine Base:  $CODE$ + 21516
; Maximum stack depth per invocation:  7 words


;   6116  1
```

```
;  1
;  1    !MMMGLOBAL ROUTINE DUP_RSP : NOVALUE =                              !ZZZ
;  1    !MMM
;  1    !+
;  1    !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH DUP RESPONSE
;  1    !       ITS GENERAL PURPOSE IS TO ACT ON A DATAGRAM OR SEQUENTIAL MESSAGE.
;  1    !       IF THE MESSAGE TYPE IS SEQUENTIAL, THE ROUTINE COPIES THE
;  1    !       CONTENTS OF THE MESSAGE ENVELOPE INTO A RETURN PACKET SO THAT THE
;  1    !       ENVELOPE CAN BE RETURNED TO THE CONTROLLER.
;  1    !
;  1    !       IMPLICIT INPUTS:
;  1    !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;  1    !               IPKT_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING RESPONSE
;  1    !-
;  1    !    begin
;  1    !
;  1    !    local
;  1    !       R_INDEX : signed word,
;  1    !        DEBUG,                                         !ZZZ
;  1    !       SRC_ADDR,
;  1    !       DST_ADDR,
;  1    !       R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;  1    !    !PRINTX (DER34);
;  1
;  1    !    incr COUNT from 0 to PKT_CNT - 1 do
;  1    !
;  1    !       if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and     ! IF THIS IS THE ASSOC CMD
;  1    !          (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  1    !          (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;  1    !          ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;  1    !          (.MSCP_PKT [.COUNT, CONNID] eql CID_DUP) and
;  1    !          ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END)
;  1    !       then
;  1    !           begin
;  1    !           P_INDEX = .COUNT;                    ! SET PKT NUMBER
;  1    !           exitloop;
;  1    !           end;
;  1    !
;  1    !    if .P_INDEX lss 0                           ! IF COMMAND NOT FOUND
;  1    !    then
;  1    !       begin
;  1    !       PRINTF (DBM108, .IPKT_ADDR [CRN_LO]);   ! UNKNOWN COMMAND REF. NUMBER
;  1    !       return;
;  1    !       end;
;  1    !
;  1    !    if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0   ! IF RETPKT IS NOT AVAILABLE
;  1    !    then
;  1    !        DEBUG = TRUE                           !TO SEE IF THIS PATH TAKEN        ZZZ
;  1    !!      PRINTF (DBM112)                         ! "DUP-RSP: RETPKT NOT AVAILABLE"  ZZZ
;  1    !    else
;  1    !       begin
;  1    !       SRC_ADDR = .IPKT_ADDR + 6;              ! SET UP COPY (SKIP OVER PKT DESC)
;  1    !       R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);   ! START OF ALLOCATED RETPKT
```

```
;     1     !
;     1     !          incr COUNT from 1 to RP_LEN do
;     1     !              begin
;     1     !              .DST_ADDR = ..SRC_ADDR;                   ! COPY 1 WORD
;     1     !              DST_ADDR = .DST_ADDR + 2;                 ! ADVANCE DESTINATION ADDR
;     1     !              SRC_ADDR = .SRC_ADDR + 2;                 ! ADVANCE SOURCE ADDR
;     1     !              end;                                      ! COPY LOOP
;     1     !
;     1     !          IN_IODQ (.R_INDEX);                           ! PUT RETPKT INDEX INTO IODQ
;     1     !          end;                                          ! IF RETPKT WAS ALLOCATED
;     1     !
;     1     !
;     1     !      if .P_INDEX geq 0                                 ! IF ASSOC CMD PKT WAS FOUND
;     1     !      then
;     1     !          PUT_PKT (.P_INDEX);                           ! RETURN COMMAND PACKET TO POOL
;     1     !
;     1     !      end;                                              ! ROUTINE DUP RSP
;     1
```

```
GLOBAL routine FATAL_ERROR : novalue =

!+
!       THIS ROUTINE IS CALLED BY THE INTERRUPT SERVICE ROUTINE (AZINT) UPON
!       DETECTING AN UNRECOVERABLE ERROR THROUGH THE DEVICE'S SA REGISTER.
!       ITS PURPOSE IS TO CLEAN UP DEVICE DATA IN THE "DRIVER" PORTION OF
!       THE EXERCISER, AND TO INFORM THE "PROGRAM" PORTION OF THE EVENT VIA
!       RETURN PACKET.
!
!       IMPLICIT INPUTS:
!               ICTLR - INTERRUPTING CONTROLLER NUMBER
!               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
!               ICST_ADDR - ADDRESS OF INTERRUPTING CONTROLLER S CST
!               IRDRX_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S IP REGISTER
!-

    begin

    local
        index : signed word,
        U_SAVE : word;

    SA_REG = .IDCT_ADDR [SA_SAVE];
    U_SAVE = .L$LUN;                                    ! SAVE PRE-INTERRUPT CURRENT UNIT NUMBER
    C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C ERR_HRD] + 1;

    if .APT_MODE
    then
        begin
        .MAIL_BOX_TESTNUM = 1;
        .MAIL_BOX_SUBTST = 0;
        end;

    L$LUN = .ICST_ADDR [OF_UN + OF_DATA, D_UNIT];       ! SET CURRENT UNIT TO FIRST IN CONTROLLER
    ERRDF (14, EGD_14, EMS_14);                         ! FATAL CONTROLLER ERROR
    L$LUN = .U_SAVE;                                    ! RESTORE PRE-INTERRUPT CURRENT UNIT
    DRV_CTLERR (.ICTLR);                                ! CLEAN UP DRIVER DATA FOR CONTROLLER

    if (index = GET_RETPKT (.ICTLR)) lss 0              ! TRY TO GET A RETPKT; IF FAILURE
    then
        PRINTF (DBM18)                                  ! "FATAL_ERROR: RETPKT NOT AVAILABLE"
    else
        begin                                           ! IF RETPKT WAS ALLOCATED
        RETPKT [.index, CONID] = CID_DRIVER;            ! SET CONNECTION ID TO "DRIVER"
        RETPKT [.index, MESTYP] = MT_FATAL;             ! FATAL ERROR
        RETPKT [.index, CTLR] = .ICTLR;                 ! CONTROLLER NUMBER
        IN_IODQ (.index);                               ! LOAD RETPKT INDEX INTO IODQ
        end;                                            ! IF RETPKT WAS ALLOCATED

    end;                                                ! ROUTINE FATAL_ERR

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

```
000000  004137  000000G                  FATAL.ERROR::
                                                 JSR     R1,$SAVE2                      ;                    6188
000004  013700  000100'                          MOV     IDCT.ADDR,R0                   ;                    6210
000010  016037  000002  000000G                  MOV     2(R0),SA.REG
000016  013701  000000G                          MOV     L$LUN,R1                       ; *,U.SAVE           6211
000022  013700  000104'                          MOV     ICTLR,R0                       ;                    6212
000026  006300                                   ASL     R0
000030  105260  000000G                          INCB    C.ERR.TBL(R0)
000034  032737  000001  001254'                  BIT     #1,APT.MODE                    ;                    6214
000042  001405                                   BEQ     1$
000044  012777  000001  001256'                  MOV     #1,@MAIL.BOX.TESTNUM           ;                    6217
000052  005077  001260'                          CLR     @MAIL.BOX.SUBTST               ;                    6218
000056  013700  000076'                  1$:     MOV     ICST.ADDR,R0                   ;                    6221
000062  016002  000006                           MOV     6(R0),R2
000066  000302                                   SWAB    R2
000070  042702  177760                           BIC     #177760,R2
000074  010237  000000G                          MOV     R2,L$LUN
000100  104455                                   TRAP    55                             ;                    6222
000102  000016                                   .WORD   16
000104  000000G                                  .WORD   EGD.14
000106  000000G                                  .WORD   EMS.14
000110  010137  000000G                          MOV     R1,L$LUN                       ; U.SAVE,*           6223
000114  013746  000104'                          MOV     ICTLR,-(SP)                    ;                    6224
000120  004737  000000G                          JSR     PC,DRV.CTLERR
000124  013716  000104'                          MOV     ICTLR,(SP)                     ;                    6226
000130  004737  000000G                          JSR     PC,GET.RETPKT
000134  010001                                   MOV     R0,R1                          ; *,INDEX
000136  002007                                   BGE     2$
000140  012716  000000G                          MOV     #DBM18,(SP)                    ;                    6228
000144  012746  000001                           MOV     #1,-(SP)
000150  010600                                   MOV     SP,R0                          ; SP,*
000152  104417                                   TRAP    17
000154  000424                                   BR      3$                             ;                    6226
000156  010116                           2$:     MOV     R1,(SP)                        ; INDEX,*            6231
000160  012746  000054                           MOV     #54,-(SP)
000164  004737  000000G                          JSR     PC,BL$MUL
000170  062700  000002G                          ADD     #RETPKT+2,R0
000174  112760  000003  000001                   MOVB    #3,1(R0)
000202  013702  000104'                          MOV     ICTLR,R2                       ;                    6233
000206  042702  177760                           BIC     #177760,R2
000212  112710  000060                           MOVB    #60,(R0)
000216  150210                                   BISB    R2,(R0)
000220  010116                                   MOV     R1,(SP)                        ; INDEX,*            6234
000222  004737  000000G                          JSR     PC,IN.IODQ
000226  022626                           3$:     CMP     (SP)+,(SP)+                    ;                    6204
000230  000207                                   RTS     PC                             ;                    6188

; Routine Size:  77 words,      Routine Base:  $CODE$ + 22016
; Maximum stack depth per invocation:  7 words
```

```
;    6238  1     GLOBAL routine POLL_CRING : novalue =
;    6239  1
;    6240  1     !+
;    6241  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    6242  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    6243  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S COMMAND RING AND CHECK FOR ANY
;    6244  1     !       COMMAND SLOTS THAT HAVE BEEN "TAKEN" BY THE CONTROLLER. SUCH SLOTS
;    6245  1     !       HAVE BEEN RETURNED TO THE HOST, INDICATED BY A ZERO OWNERSHIP BIT. FOR
;    6246  1     !       EACH SLOT THAT HAS BEEN RETURNED TO THE HOST, THE CRING COUNT IS
;    6247  1     !       DECREMENTED, AND THE CR_POLL ADDRESS IS ADVANCED TO THE NEXT SLOT IN
;    6248  1     !       THE COMMAND RING.
;    6249  1     !
;    6250  1     !       IMPLICIT INPUTS:
;    6251  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    6252  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    6253  1     !               ICOM_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S COMM_AREA
;    6254  1     !-
;    6255  1
;    6256  2         begin
;    6257  2         ICOM_ADDR [CMD_INT] = 0;                                    ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER    ZZZ
;    6258  2
;    6259  3         while ((.IDCT_ADDR [CRING_CNT] gtru 0) and                  ! WHILE # OF COMMANDS IN CRING > 0 AND
;    6260  2             not (BIT_TST ((.IDCT_ADDR [CR_POLL] + 2), ED_OWN))) do  ! CURRENT SLOT IS HOST-OWNED
;    6261  3             begin
;    6262  3             IDCT_ADDR [CRING_CNT] = .IDCT_ADDR [CRING_CNT] - 1;     ! DECREMENT # CMDs IN CRING
;    6263  3             IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_POLL] + 4;         ! ADVANCE TO NEXT SLOT TO POLL
;    6264  3
;    6265  3             if .IDCT_ADDR [CR_POLL] gtra .IDCT_ADDR [CR_END]        ! IF BEYOND END OF RING
;    6266  3             then
;    6267  3                 IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_BEG];          ! SET POINTER TO TOP OF CRING
;    6268  3
;    6269  2             end;
;    6270  2
;    6271  2     !ZZZ    ICOM_ADDR [CMD_INT] = 0;                                        ! CLEAR COMMAND INTERRUPT WORD IN RING HEADE
R
;    6272  1         end;


                                                .SBTTL  POLL.CRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G                 POLL.CRING::
                                                JSR     R1,$SAVE2                       ;                       6238
000004  013700  000074'                         MOV     ICOM.ADDR,R0                    ;                       6257
000010  005060  000004                          CLR     4(R0)
000014  013701  000100'                         MOV     IDCT.ADDR,R1                    ;                       6259
000020  012702  000016                          MOV     #16,R2                          ;                       6263
000024  060102                                  ADD     R1,R2
000026  105711                          1$:     TSTB    (R1)                            ;                       6259
000030  001422                                  BEQ     2$
000032  016100  000016                          MOV     16(R1),R0                       ;                       6260
000036  016000  000002                          MOV     2(R0),R0
000042  042700  077777                          BIC     #77777,R0
000046  020027  100000                          CMP     R0,#-100000
000052  001411                                  BEQ     2$
000054  105311                                  DECB    (R1)                            ;                       6262
```

```
000056  062712  000004              ADD     #4,(R2)                         ;       6263
000062  021261  000012              CMP     (R2),12(R1)                     ;       6265
000066  101757                      BLOS    1$
000070  016112  000010              MOV     10(R1),(R2)                     ;       6267
000074  000754                      BR      1$                              ;       6259
000076  000207              2$:     RTS     PC                              ;       6238

; Routine Size:  32 words,      Routine Base:  $CODE$ + 22250
; Maximum stack depth per invocation:  4 words
```

```
GLOBAL routine POLL_RRING : novalue =


!+
!       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
!       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
!       ITS PURPOSE IS TO SCAN THE DEVICE'S RESPONSE RING AND CHECK FOR ANY
!       SLOTS WHICH HAVE BEEN RETURNED TO THE HOST (OWNERSHIP BIT = 0). FOR
!       EACH SUCH SLOT, THE ASSOCIATED MESSAGE IS PROCESSED BASED ON ITS
!       CONNECTION ID (DISK OR DUP). AFTER PROCESSING, THE MESSAGE PACKET
!       IS RE-INITIALIZED AND RETURNED TO THE CONTROLLER (OWNERSHIP BIT SET
!       TO 1).
!
!       IMPLICIT INPUTS:
!               ICTLR - NUMBER OF INTERRUPTING CONTROLLER
!               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
!-

    begin

    local
        TEMP : word;                                                    ! MMM

    ICOM_ADDR [RSP_INT] = 0;                                            ! CLR RESPONSE INTERRUPT WRD IN RING HEADER
ZZZ
    TEMP = .PAR_TSD;                                                    ! MMM

    while not (BIT_TST ((.IDCT_ADDR [RR_POLL] + 2), ED_OWN)) do         ! WHILE 0 = 0
        begin
        IPKT_ADDR = ..IDCT_ADDR [RR_POLL] - 10;                         ! ADDRESS OF RESPONSE PACKET


IF NOT (.IPKT_ADDR [CONNID] EQL CID_DUP)                !               ZZZ
        THEN                                            !               ZZZ
        (CREDIT_BAL = .CREDIT_BAL + .IPKT_ADDR [CREDITS];);            !ZZZ
!IT WAS NOTICE THAT DUP WAS SENDIND BACK CREDITS WHICH IT SHOULD NOT.   !ZZZ
        selectoneu .IPKT_ADDR [CONNID] of
            set

            [CID_DISK] :        DISK_RSP ();

!MMM            [CID_DUP]  :        DUP_RSP  ();                               !ZZZ

            [otherwise] :       PRINTF (DBM20, .IPKT_ADDR [CONNID], .IRDRX_ADDR);
                                                                        ! "BAD CONN ID = XXXXX FROM XXXXXX"
            tes;


!         if .PAR_TSD                                                    ! MMM
!         then
!            if  not .TEMP
!            then
!               return;                                                  ! MMM

```

```
;   6326  3          IPKT_ADDR [MSGLEN] = MSG_LEN * 2;                          ! RE-INIT PKT FIELDS; MESSAGE LENGTH
;   6327  3          IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;            ! ADVANCE TO HI ORDER WORD OF RING SLOT
;   6328  3          .IDCT_ADDR [RR_POLL] = .IPKT_ADDR [PKT_HI];                ! RETURN SLOT TO CONTROLLER
;   6329  3          .IDCT_ADDR [RR_POLL] = ..IDCT_ADDR [RR_POLL] or ED_OWN or ED_FLAG; ! OWNERSHIP TOO
;   6330  3          IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;            ! ADVANCE TO NEXT RRING SLOT
;   6331  3
;   6332  3
;   6333  3          if .IDCT_ADDR [RR_POLL] gtra .IDCT_ADDR [RR_END]           ! IF BEYOND END OF RING
;   6334  3
;   6335  3          then
;   6336  3              IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_BEG];             ! CYCLE TO TOP OF RING
;   6337  3
;   6338  2          end;                                                       ! WHILE LOOP
;   6339  2
;   6340  2     !ZZZ ICOM_ADDR [RSP_INT] = 0;                                         ! CLR RESPONSE INTERRUPT WRD IN RING
 HEADER
;   6341  1        end;


                                                  .SBTTL  POLL.RRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G                 POLL.RRING::
                                                  JSR     R1,$SAVE3                                      6273
000004  013700  000074'                           MOV     ICOM.ADDR,R0              ;                    6296
000010  005060  000006                            CLR     6(R0)
000014  013701  000100'                           MOV     IDCT.ADDR,R1              ;                    6299
000020  062701  000014                            ADD     #14,R1
000024  011100                          1$:       MOV     (R1),R0
000026  016000  000002                            MOV     2(R0),R0
000032  042700  077777                            BIC     #77777,R0
000036  020027  100000                            CMP     R0,#-100000
000042  001476                                    BEQ     5$
000044  017137  000000  000000G                   MOV     @0(R1),IPKT.ADDR          ;                    6301
000052  162737  000012  000000G                   SUB     #12,IPKT.ADDR
000060  013700  000000G                           MOV     IPKT.ADDR,R0              ;                    6304
000064  005002                                    CLR     R2
000066  156002  000011                            BISB    11(R0),R2
000072  020227  000002                            CMP     R2,#2
000076  001406                                    BEQ     2$
000100  116003  000010                            MOVB    10(R0),R3                 ;                    6306
000104  042703  177760                            BIC     #177760,R3
000110  060337  000000G                           ADD     R3,CREDIT.BAL
000114  005702                          2$:       TST     R2                        ;                    6311
000116  001003                                    BNE     3$
000120  004737  000000V                           JSR     PC,DISK.RSP
000124  000413                                    BR      4$                        ;                    6308
000126  013746  000000G                 3$:       MOV     IRDRX.ADDR,-(SP)          ;                    6315
000132  010246                                    MOV     R2,-(SP)
000134  012746  000000G                           MOV     #DBM20,-(SP)
000140  012746  000003                            MOV     #3,-(SP)
000144  010600                                    MOV     SP,R0                     ; SP,*
000146  104417                                    TRAP    17
000150  062706  000010                            ADD     #10,SP
000154  013700  000000G                 4$:       MOV     IPKT.ADDR,R0              ;                    6326
000160  012760  000074  000006                    MOV     #74,6(R0)
```

```
000166  013702  000100'                   MOV     IDCT.ADDR,R2                          ;                6327
000172  010201                            MOV     R2,R1
000174  062701  000014                    ADD     #14,R1
000200  062711  000002                    ADD     #2,(R1)
000204  016071  000002  000000            MOV     2(R0),@0(R1)                          ;                6328
000212  052771  140000  000000            BIS     #-40000,@0(R1)                        ;                6329
000220  062711  000002                    ADD     #2,(R1)                               ;                6330
000224  021162  000006                    CMP     (R1),6(R2)                            ;                6333
000230  101675                            BLOS    1$
000232  016211  000004                    MOV     4(R2),(R1)                            ;                6336
000236  000672                            BR      1$                                    ;                6299
000240  000207                    5$:     RTS     PC                                    ;                6273

; Routine Size:  81 words,     Routine Base:  $CODE$ + 22350
; Maximum stack depth per invocation:  10 words


;   6342  1
```

```
;    6343  1
;    6344  1
;    6345  1       GLOBAL routine DISK_RSP : novalue =
;    6346  1
;    6347  1
;    6348  1       !+
;    6349  1       !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH RESPONSE MESSAGE
;    6350  1       !       WHICH HAS A CONNECTION ID INDICATING A DISK MSCP ORIGINATOR
;    6351  1       !       (I.E., ALL EXCEPT DUP RESPONSES). ITS PURPOSE IS TO PASS
;    6352  1       !       CONTROL TO THE APPROPRIATE ROUTINE BASED ON THE MESSAGE TYPE
;    6353  1       !       FIELD (SEQUENTIAL, DATAGRAM, OR CREDIT NOTIFICATION).
;    6354  1       !
;    6355  1       !       IMPLICIT INPUTS:
;    6356  1       !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    6357  1       !                               MESSAGE
;    6358  1       !-
;    6359  1
;    6360  1
;    6361  1
;    6362  1           selectoneu .IPKT_ADDR [MSGTYP] of
;    6363  1
;    6364  1
;    6365  1               set
;    6366  1
;    6367  1               [MT_SEQ] :      SEQUEN ();
;    6368  1
;    6369  1               [MT_DG] :       DATAGM ();
;    6370  1
;    6371  1               [otherwise] :   PRINTF (DBM21, .IPKT_ADDR [MSGTYP]);    ! "MESSAGE TYPE XX RECEIVED"
;    6372  1               tes;

                                        .SBTTL  DISK.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  010146                  DISK.RSP::
                                        MOV     R1,-(SP)                        ;               6345
000002  013700  000000G                 MOV     IPKT.ADDR,R0                    ;               6362
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$                              ;               6367
000030  004737  000000V                 JSR     PC,SEQUEN
000034  000417                          BR      3$                              ;               6362
000036  020127  000001          1$:     CMP     R1,#1                           ;               6369
000042  001003                          BNE     2$
000044  004737  000000V                 JSR     PC,DATAGM
000050  000411                          BR      3$                              ;               6362
000052  010146                  2$:     MOV     R1,-(SP)                        ;               6371
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                 ;                                  6345
000076  000207                              RTS     PC

; Routine Size:  32 words,      Routine Base:  $CODE$ + 22612
; Maximum stack depth per invocation:  6 words


;    6373  1
;    6374  1
;    6375  1
```

```
1   GLOBAL routine SEQUEN : novalue =
1
1   !+
1   !       THIS ROUTINE IS CALLED BY DISK_RSP FOR EACH DISK MSCP RESPONSE MESSAGE
1   !       WITH THE "SEQUENTIAL" MESSAGE TYPE. ITS GENERAL PURPOSE IS TO COPY THE
1   !       CONTENTS OF THE MESSAGE PACKET INTO A RETURN PACKET SO THAT THE
1   !       PACKET CAN BE RETURNED TO THE CONTROLLER.   IN ADDITION,
1   !       IF THE COMMAND WAS AN I/O TRANSFER (READ, WRITE, OR ACCESS), THEN SOME
1   !       FIELDS OF THE COMMAND PACKET ARE COPIED INTO THE RETURN PACKET.
1   !
1   !       IMPLICIT INPUTS:
1   !               ICTLR - INTERRUPTING CONTROLLER NUMBER
1   !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
1   !-
1
2       begin
2
2       local
2           P_INDEX : signed word initial (-1),                          ! ASSUME NO ASSOCIATED COMMAND PKT
2           R_INDEX : signed word,
2           SRC_ADDR,
2           DST_ADDR,
2           SAV_ADDR,
2           R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
2
2       builtin                                                          !MMM
2
2           HALT;                                                        !MMM
2
2       incr COUNT from 0 to PKT_CNT - 1 do
2
2           if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and    ! IF THIS IS THE ASSOC CMD
2              (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
2              (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
2              ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
2              (.MSCP_PKT [.COUNT, MSGTYP] eql MT_SEQ) and
2              ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END) and
3              (.PKT_USE [.COUNT] eql .ICTLR)
2           then
3               begin
3               P_INDEX = .COUNT;                                        ! SET PKT NUMBER
3               exitloop;
2               end;
2
2       if .P_INDEX lss 0                                                ! IF COMMAND NOT FOUND
2       then
3           begin
3           PRINTF (DBM108, .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);   ! UNKNOWN COMMAND REF. NUMBER
3           return;
2           end;
2
3       if .MSCP_PKT [.P_INDEX, OPCODE] neq (.IPKT_ADDR [OPCODE] and (not OP_END))  ! IF OPCODE MISMATCH
2       then
```

```
        PRINTF (DBM111, .MSCP_PKT [.P_INDEX, OPCODE], .IPKT_ADDR [OPCODE], .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO])

    if ((.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or
       (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END))) and
      ((.IPKT_ADDR [STATUS_CODE] eql ST_SUC) and
       (.IPKT_ADDR [STATUS_SUBCODE] eql 0)) and
      ((.MSCP_PKT [.P_INDEX, BC_LO] neq .IPKT_ADDR [BC_LO]) or
       (.MSCP_PKT [.P_INDEX, BC_HI] neq .IPKT_ADDR [BC_HI]))
    then
        PRINTF (DBM112,
                .MSCP_PKT [.P_INDEX, BC_HI], .MSCP_PKT [.P_INDEX, BC_LO], .IPKT_ADDR [BC_HI], .IPKT_ADDR [BC_LO],
                .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);

    if .MSCP_PKT [.P_INDEX, RSP_RECEIVED]
    then
        begin
        PRINTF (DBM120, .MSCP_PKT [.P_INDEX, CRN_HI], .MSCP_PKT [.P_INDEX, CRN_LO]);
        PUT_PKT (.P_INDEX);
        return;
        end
    else
        MSCP_PKT [.P_INDEX, RSP_RECEIVED] = TRUE;                      ! MARK RESPONSE RECEIVED

    if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0                            ! IF RETPKT IS NOT AVAILABLE
    then
        begin
        PRINTF (DBM22);                                                 ! "SEQUEN: RETPKT NOT AVAILABLE"
        PUT_PKT (.P_INDEX);
        return;
        end
    else
        begin
        SRC_ADDR = .IPKT_ADDR + 6;                                      ! SET UP COPY (SKIP OVER PKT DESC)
        R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);           ! START OF ALLOCATED RETPKT

        incr COUNT from 1 to RP_LEN do
            begin
            .DST_ADDR = ..SRC_ADDR;                                     ! COPY 1 WORD
            DST_ADDR = .DST_ADDR + 2;                                   ! ADVANCE DESTINATION ADDR
            SRC_ADDR = .SRC_ADDR + 2;                                   ! ADVANCE SOURCE ADDR

            if .IPKT_ADDR [OPCODE] eql (OP_ONL or OP_END)               ! IF THIS IS THE ONLINE END MESSAGE
            then
                if .COUNT eql 10                                        ! SKIP OVER RESERVED WORDS
                then
                    SRC_ADDR = .SRC_ADDR + 4;                           !      IN ONLINE END   MESSAGE
            end;                                                        ! COPY LOOP

        R_ADDR [CTLR] = .ICTLR;                                         ! LOAD CONTROLLER NUMBER INTO PKT

        if .P_INDEX geq 0                                               ! IF ASSOC. CMD PKT WAS FOUND
        then
```

```
;  3                 if (.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or              ! IF END MESSAGE IS
;  3                    (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END)) or             !      READ, WRITE, OR
;  4                    (.IPKT_ADDR [OPCODE] eql (OP_ACC or OP_END))                !      ACCESS
;  3                 then
;  4                    begin
;  4                    R_ADDR [CMDMOD] = .MSCP_PKT [.P_INDEX, MODIFY];             ! COPY
;  4                    R_ADDR [CBCNT_LO] = .MSCP_PKT [.P_INDEX, BC_LO];            ! RELEVANT
;  4                    R_ADDR [CBCNT_HI] = .MSCP_PKT [.P_INDEX, BC_HI];            ! FIELDS
;  4                    R_ADDR [LBN_LO] = .MSCP_PKT [.P_INDEX, LBN_L];              ! FROM
;  4                    R_ADDR [LBN_HI] = .MSCP_PKT [.P_INDEX, LBN_H];              ! COMMAND
;  4                    R_ADDR [BUFF_0] = .MSCP_PKT [.P_INDEX, BUF_0];              ! PACKET
;  4                    R_ADDR [BUFF_1] = .MSCP_PKT [.P_INDEX, BUF_1];              ! TO RETPKT
;  3                    end;                                                        ! IF ENDCODE WAS READ/WRITE/ACCESS
;  3
;  3              IN_IODQ (.R_INDEX);                                               ! PUT RETPKT INDEX INTO IODQ
;  2              end;                                                              ! IF RETPKT WAS ALLOCATED
;  2
;  2          if (.IPKT_ADDR [STATUS_CODE] neq  ST_SUC) or
;  3             (.IPKT_ADDR [STATUS_SUBCODE] neq 0)
;  2          then
;  2              LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_OCCURED                     ! SAVE ERROR CONDITION
;  2          else
;  2              LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_NOT_OCCURED;                !
;  2
;  2          LAST_PKT [.ICTLR, LAST_CRN_LO] = .IPKT_ADDR [CRN_LO];                 ! SAVE COMMAND REFERENCE NUMBER
;  2          LAST_PKT [.ICTLR, LAST_CRN_HI] = .IPKT_ADDR [CRN_HI];                 !
;  2    !MMM      SCAN_ERRLOG ();                                                   ! PRINT ANY ASSOCIATED ERROR-LOGS
;  2
;  2          if (.R_ADDR [FLAGS] and %o'40') eql %o'40'                            ! MMM IF ERROR LOG(S) GENERATED
;  2          then                                                                  ! MMM PRINT RESPONSE PACKET
;  3             begin
;  3             PRINTB (DBM123);
;  3             EMS_R2 (.R_ADDR);
;  3             EMS_R1 (.R_ADDR);
;  2             end;
;  2
;  2    !MMM      SCAN_ERRLOG ( );                                                      ! PRINT ANY ASSOCIATED ERROR LOG
S MMM
;  2
;  2
;  2          if .CSR_MEM and .TST_PAR                                    !MMM
;  2          then
;  3             if ((..CSR_ADD and %o'100000') eqlu %o'100000')          !MMM
;  2             then
;  3                begin
;  3                PRINTF (DBM125, ..CSR_ADD);
;  3                .CSR_ADD = %o'40000';                                                !MMM PRINT EXTENDED CSR
;  3                PRINTF (DBM126, ..CSR_ADD);
;  3                HALT ();                                                             ! MMM HALT ON HOST MEMORY PARI
TY ERROR
;  2                end;
;  2
;  2          if .P_INDEX geq 0                                                     ! IF ASSOC CMD PKT WAS FOUND
;  2          then
;  2              PUT_PKT (.P_INDEX);                                               ! RETURN COMMAND PACKET TO POOL
```

```
;    6535  2
;    6536  1       end;                                                    ! ROUTINE DISK_RSP

                                        .SBTTL  SEQUEN RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G         SEQUEN::JSR     R1,$SAVE5                       ;                       6376
000004  005746                          TST     -(SP)
000006  012746  177777                  MOV     #-1,(SP)                        ; *,P.INDEX             6391
000012  013701  000000G                 MOV     IPKT.ADDR,R1                    ;                       6407
000016  005002                          CLR     R2                              ; COUNT                 6405
000020  010246                  1$:     MOV     R2,-(SP)                        ; COUNT,*               6407
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)          ;                       6408
000052  001024                          BNE     2$
000054  026011  000000G                 CMP     MSCP.PKT(R0),(R1)               ;                       6409
000060  001421                          BEQ     2$
000062  105760  000022G                 TSTB    MSCP.PKT+22(R0)                 ;                       6410
000066  100416                          BMI     2$
000070  132760  000360  000010G         BITB    #360,MSCP.PKT+10(R0)            ;                       6411
000076  001012                          BNE     2$
000100  105761  000022                  TSTB    22(R1)                          ;                       6412
000104  100007                          BPL     2$
000106  116200  000000G                 MOVB    PKT.USE(R2),R0                  ; *(COUNT),*            6413
000112  020037  000104'                 CMP     R0,ICTLR
000116  001002                          BNE     2$
000120  010216                          MOV     R2,(SP)                         ; COUNT,P.INDEX         6416
000122  000405                          BR      3$                              ;                       6415
000124  005202                  2$:     INC     R2                              ; COUNT                 6405
000126  020227  000013                  CMP     R2,#13                          ; COUNT,*
000132  003732                          BLE     1$
000134  005716                          TST     (SP)                            ; P.INDEX               6420
000136  002013                  3$:     BGE     4$
000140  016146  000012                  MOV     12(R1),(SP)                     ;                       6423
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$                              ;                       6424
000166  011646                  4$:     MOV     (SP),-(SP)                      ; P.INDEX,*             6427
000170  012746  000106                  MOV     #106,-(SP)
000174  004737  000000G                 JSR     PC,BL$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)                 ;                       6429
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$:      MOV     IPKT.ADDR,R0                 ;                       6431
000302  126027  000022  000241            CMPB    22(R0),#241
000310  001404                            BEQ     6$
000312  126027  000022  000242            CMPB    22(R0),#242                  ;                       6432
000320  001045                            BNE     8$
000322  012702  000024           6$:      MOV     #24,R2                       ;                       6433
000326  060002                            ADD     R0,R2
000330  132712  000037                    BITB    #37,(R2)
000334  001037                            BNE     8$
000336  032712  177740                    BIT     #177740,(R2)                 ;                       6434
000342  001034                            BNE     8$
000344  026160  000026G 000026            CMP     MSCP.PKT+26(R1),26(R0)       ;                       6435
000352  001004                            BNE     7$
000354  026160  000030G 000030            CMP     MSCP.PKT+30(R1),30(R0)       ;                       6436
000362  001424                            BEQ     8$
000364  016046  000012           7$:      MOV     12(R0),-(SP)                 ;                       6440
000370  016046  000014                    MOV     14(R0),-(SP)
000374  016046  000026                    MOV     26(R0),-(SP)
000400  016046  000030                    MOV     30(R0),-(SP)
000404  016146  000026G                   MOV     MSCP.PKT+26(R1),-(SP)
000410  016146  000030G                   MOV     MSCP.PKT+30(R1),-(SP)
000414  012746  000000G                   MOV     #DBM112,-(SP)
000420  012746  000007                    MOV     #7,-(SP)
000424  010600                            MOV     SP,R0                        ; SP,*
000426  104417                            TRAP    17
000430  062706  000020                    ADD     #20,SP
000434  132761  000001  000005G  8$:      BITB    #1,MSCP.PKT+5(R1)            ;                       6442
000442  001422                            BEQ     10$
000444  016146  000012G                   MOV     MSCP.PKT+12(R1),-(SP)        ;                       6445
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,*             6446
000474  004737  000000G                   JSR     PC,PUT.PKT
000500  062706  000010           9$:      ADD     #10,SP                       ;                       6447
000504  000137  024230'                   JMP     23$                          ;                       6444
000510  112761  000001  000005G  10$:     MOVB    #1,MSCP.PKT+5(R1)            ;                       6450
000516  013746  000104'                   MOV     ICTLR,-(SP)                  ;                       6452
```

```
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  002010                            BGE    11$
000542  012746  000000G                   MOV    #DBM22,-(SP)               ;                               6455
000546  012746  000001                    MOV    #1,-(SP)
000552  010600                            MOV    SP,R0                      ; SP,*
000554  104417                            TRAP   17
000556  000137  024216'                   JMP    21$                        ;                               6456
000562  013704  000000G          11$:     MOV    IPKT.ADDR,R4               ; *,SRC.ADDR                    6461
000566  062704  000006                    ADD    #6,R4                      ; *,SRC.ADDR
000572  016646  000002                    MOV    2(SP),-(SP)                ; R.INDEX,*                     6462
000576  012746  000054                    MOV    #54,-(SP)
000602  004737  000000G                   JSR    PC,BL$MUL
000606  062700  000000G                   ADD    #RETPKT,R0
000612  010005                            MOV    R0,R5                      ; *,DST.ADDR
000614  010002                            MOV    R0,R2                      ; *,R.ADDR
000616  013700  000000G                   MOV    IPKT.ADDR,R0               ;                               6470
000622  012703  000001                    MOV    #1,R3                      ; *,COUNT                       6464
000626  012425                   12$:     MOV    (R4)+,(R5)+                ; SRC.ADDR,DST.ADDR             6466
000630  126027  000022  000211            CMPB   22(R0),#211                ;                               6470
000636  001005                            BNE    13$
000640  020327  000012                    CMP    R3,#12                     ; COUNT,*                       6472
000644  001002                            BNE    13$
000646  062704  000004                    ADD    #4,R4                      ; *,SRC.ADDR                    6474
000652  005203                   13$:     INC    R3                         ; COUNT                         6464
000654  020327  000026                    CMP    R3,#26                     ; COUNT,*
000660  003762                            BLE    12$
000662  013703  000104'                   MOV    ICTLR,R3                   ;                               6477
000666  042703  177760                    BIC    #177760,R3
000672  142762  000017  000002            BICB   #17,2(R2)                  ; *,*(R.ADDR)
000700  150362  000002                    BISB   R3,2(R2)                   ; *,*(R.ADDR)
000704  005766  000004                    TST    4(SP)                      ; P.INDEX                       6479
000710  002442                            BLT    15$
000712  116000  000022                    MOVB   22(R0),R0                  ;                               6482
000716  042700  177400                    BIC    #177400,R0
000722  020027  000241                    CMP    R0,#241
000726  001406                            BEQ    14$
000730  020027  000242                    CMP    R0,#242                    ;                               6483
000734  001403                            BEQ    14$
000736  020027  000220                    CMP    R0,#220                    ;                               6484
000742  001025                            BNE    15$
000744  016162  000024G 000012   14$:     MOV    MSCP.PKT+24(R1),12(R2)     ; *,*(R.ADDR)                   6487
000752  016162  000026G 000044            MOV    MSCP.PKT+26(R1),44(R2)     ; *,*(R.ADDR)                   6488
000760  016162  000030G 000046            MOV    MSCP.PKT+30(R1),46(R2)     ; *,*(R.ADDR)                   6489
000766  016162  000046G 000050            MOV    MSCP.PKT+46(R1),50(R2)     ; *,*(R.ADDR)                   6490
000774  016162  000050G 000052            MOV    MSCP.PKT+50(R1),52(R2)     ; *,*(R.ADDR)                   6491
001002  016162  000032G 000024            MOV    MSCP.PKT+32(R1),24(R2)     ; *,*(R.ADDR)                   6492
001010  016162  000034G 000026            MOV    MSCP.PKT+34(R1),26(R2)     ; *,*(R.ADDR)                   6493
001016  016616  000006           15$:     MOV    6(SP),(SP)                 ; R.INDEX,*                     6496
001022  004737  000000G                   JSR    PC,IN.IODQ
001026  005726                            TST    (SP)+                      ;                               6460
```

```
001030  013716  000104'                     MOV     ICTLR,(SP)              ;                       6502
001034  012746  000006                      MOV     #6,-(SP)
001040  004737  000000G                     JSR     PC,BL$MUL
001044  013701  000000G                     MOV     IPKT.ADDR,R1            ;                       6499
001050  012703  000024                      MOV     #24,R3
001054  060103                              ADD     R1,R3
001056  132713  000037                      BITB    #37,(R3)
001062  001003                              BNE     16$
001064  032713  177740                      BIT     #177740,(R3)            ;                       6500
001070  001404                              BEQ     17$
001072  012760  000001  000120'     16$:    MOV     #1,LAST.PKT(R0)         ;                       6502
001100  000402                              BR      18$                     ;                       6499
001102  005060  000120'             17$:    CLR     LAST.PKT(R0)            ;                       6504
001106  016160  000012  000122'     18$:    MOV     12(R1),LAST.PKT+2(R0)   ;                       6506
001114  016160  000014  000124'             MOV     14(R1),LAST.PKT+4(R0)   ;                       6507
001122  132762  000040  000015              BITB    #40,15(R2)              ; *,*(R.ADDR)           6510
001130  001415                              BEQ     19$
001132  012716  000000G                     MOV     #DBM123,(SP)            ;                       6513
001136  012746  000001                      MOV     #1,-(SP)
001142  010600                              MOV     SP,R0                   ; SP,*
001144  104414                              TRAP    14
001146  010216                              MOV     R2,(SP)                 ; R.ADDR,*              6514
001150  004737  000000G                     JSR     PC,EMS.R2
001154  010216                              MOV     R2,(SP)                 ; R.ADDR,*              6515
001156  004737  000000G                     JSR     PC,EMS.R1
001162  005726                              TST     (SP)+                   ;                       6512
001164  032737  000001  000000G     19$:    BIT     #1,CSR.MEM              ;                       6521
001172  001441                              BEQ     20$
001174  032737  000001  000000G             BIT     #1,TST.PAR
001202  001435                              BEQ     20$
001204  017700  000000G                     MOV     @CSR.ADD,R0             ;                       6523
001210  042700  077777                      BIC     #77777,R0
001214  020027  100000                      CMP     R0,#-100000
001220  001026                              BNE     20$
001222  017716  000000G                     MOV     @CSR.ADD,(SP)           ;                       6526
001226  012746  000000G                     MOV     #DBM125,-(SP)
001232  012746  000002                      MOV     #2,-(SP)
001236  010600                              MOV     SP,R0                   ; SP,*
001240  104417                              TRAP    17
001242  012777  040000  000000G             MOV     #40000,@CSR.ADD         ;                       6527
001250  012716  040000                      MOV     #40000,(SP)             ;                       6528
001254  012746  000000G                     MOV     #DBM126,-(SP)
001260  012746  000002                      MOV     #2,-(SP)
001264  010600                              MOV     SP,R0                   ; SP,*
001266  104417                              TRAP    17
001270  000000                              HALT                            ;                       6529
001272  062706  000010                      ADD     #10,SP                  ;                       6525
001276  005766  000004              20$:    TST     4(SP)                   ; P.INDEX               6532
001302  002404                              BLT     22$
001304  016616  000004              21$:    MOV     4(SP),(SP)              ; P.INDEX,*             6534
001310  004737  000000G                     JSR     PC,PUT.PKT
001314  022626                      22$:    CMP     (SP)+,(SP)+             ;                       6391
001316  022626                      23$:    CMP     (SP)+,(SP)+             ;                       6376
```

```
001320  000207                          RTS     PC

; Routine Size:  361 words,     Routine Base:  $CODE$ + 22712
; Maximum stack depth per invocation:  18 words
```

```
!MMMGLOBAL routine SCAN_ERRLOG : novalue =

!MMM!+
!MMM! THIS ROUTINE SCANS THE ERROR-LOG SAVE AREA AND PRINTS ANY ERROR-LOGS RECEIVED FOR THE ASSOCIATED RESPONSE
!MMM!-

!MMM    begin

!MMM    local
!MMM    TEMP_UNIT,
!MMM    SFT_ERR_PRINTED : byte initial (byte (FALSE));

!MMM    incr index from 0 to EP_CNT do                                  ! SCAN ERROR-LOG PACKET SAVE AREA
!MMM    begin

!MMM    if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
!MMM       (.ELOG_PKT [.index, EL_CRN_LO] eql .IPKT_ADDR [CRN_LO]) and        ! MMM
!MMM       (.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
!MMM       (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
!MMM    then
!MMM        begin                                                       ! ERROR-LOG PENDING THIS RESPONSE

!MMM        if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED        ! IF SOFT ERROR OCCURED
!MMM        then

!MMM            if .ELOG_PKT [.index, EL_FORMAT] lequ 4
!MMM            then
!MMM                begin
!MMM                SOFT_ERROR (.index);                                ! UPATE SOFT ERROR COUNT
!MMM                TEMP_UNIT = .L$LUN;                                 ! SAVE UNIT NUMBER AS KNOWN TO DRS

!MMM                incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do

!MMM                    if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_PKT [.index, EL_DK_NUM]) and
!MMM                       (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
!MMM                    then
!MMM                        begin
!MMM                        L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];     ! CORECT UNIT NO. FOR ERROR MESSAGE
!MMM                        exitloop;
!MMM                        end;

!MMM                case .ELOG_PKT [.index, EL_FORMAT] from 0 to 4 of
!MMM                    set

!MMM                    [0]:    if .APT_MODE                            ! CONTROLLER ERROR
!MMM                            then
!MMM                                ERR_SOFT_RTNE_APT (50, .index)
!MMM                            else
!MMM                                ERR_SOFT_RTNE (50);

!MMM                    [1]:    if .APT_MODE                            ! HOST MEMORY ACCESS ERROR
!MMM                            then
!MMM                                ERR_SOFT_RTNE_APT (51, .index)
```

```
;  1   !MMM                          else
;  1   !MMM                              ERR_SOFT_RTNE (51);
;  1
;  1   !MMM                  [2]:    if .APT_MODE                                        ! DISK TRANSFER ERROR
;  1   !MMM                          then
;  1   !MMM                              ERR_SOFT_RTNE_APT (52, .index)
;  1   !MMM                          else
;  1   !MMM                              ERR_SOFT_RTNE (52);
;  1
;  1   !MMM                  [3]:    if .APT_MODE                                        ! SDI ERROR
;  1   !MMM                          then
;  1   !MMM                              ERR_SOFT_RTNE_APT (53, .index)
;  1   !MMM                          else
;  1   !MMM                              ERR_SOFT_RTNE (53);
;  1
;  1   !MMM                  [4]:    if .APT_MODE                                        ! SMALL DISK ERROR
;  1   !MMM                          then
;  1   !MMM                              ERR_SOFT_RTNE_APT (54, .index)
;  1   !MMM                          else
;  1   !MMM                              ERR_SOFT_RTNE (54);
;  1   !MMM                  tes;
;  1
;  1   !MMM              L$LUN = .TEMP_UNIT;                                         ! RESTORE UNIT NUMBER
;  1   !MMM              SFT_ERR_PRINTED = TRUE;                                     ! SOFT ERROR PRINTOUT OCCURED
;  1   !MMM              end
;  1   !MMM          else
;  1   !MM               PRINTF (DBM109, .ELOG_PKT [.index, EL_FORMAT]);             ! UNKNOWN ERROR LOG FORMAT
;  1
;  1   !MMM      if not (.SFT_ERR_PRINTED)
;  1   !MMM      then
;  1   !MMM          PRINTB (CRLF);                                                  ! EXTRA CARRIAGE-RETURN/LINE-FEED
;  1
;  1   !MMM          if ((.ELOG_PKT [.index, EL_CRN_LO] neq .IPKT_ADDR [CRN_LO]) or        ! MMM
;  1   !MMM      (.ELOG_PKT [.index, EL_CRN_HI] neq .IPKT_ADDR [CRN_HI])) and
;  1   !MMM          (.QIO [.ICTLR] eql 1)
;  1   !MMM          then
;  1   !MMM             PRINTB (DBM122);                                                ! 'Late or Unexpected Error Log:
 MMM
;  1   !MMM      EMS_EL (.index);                                                    ! PRINT ERROR-LOG CONTENTS
;  1   !MMM      end
;  1   !MMM  else
;  1
;  1   !MMM      if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;  1   !MMM         ((.ELOG_PKT [.index, EL_CRN_HI] lssu .IPKT_ADDR [CRN_HI]) or
;  1   !MMM          ((.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  1   !MMM           (.ELOG_PKT [.index, EL_CRN_LO] lssu .IPKT_ADDR [CRN_LO]))) and
;  1   !MMM         (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;  1
;  1
;  1   !MMM      if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and                     ! MMM
;  1   !MMM         (.ELOG_PKT [.index, EL_CONTENTS] eql FULL) and
;  1   !MMM             (.QIO [.ICTLR] eqlu 1)
;  1
;  1   !MMM      then
```

```
;      1    !MMM            begin
;      1    !MMM            PRINTB (CRLF);                          ! CARRIAGE-RETURN/LINE-FEED
;      1    !MMM            EMS_EL (.index);                        ! PRINT ERROR-LOG CONTENTS
;      1    !MMM            end;
;      1
;      1    !MMM     end;                                           ! ERROR-LOG SAVE AREA SCAN
;      1
;      1    !MMM     end;
```

```
GLOBAL routine DATAGM : novalue =

!+
!       THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
!       THE RDRX
!
!       IMPLICIT INPUTS:
!               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING ERROR LOG
!                               MESSAGE
!               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
!-

    begin

    local
        index : signed word initial (-1),
        SAVE_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
        SRC_ADDR,
        DST_ADDR,
        TEMP_UNIT,
!MMM    SFT_ERR_PRINTED : byte initial (byte (FALSE)),
        PACKET_LEN : word;

!
! FIND AN EMPTY SLOT IN THE ERROR-LOG PACKET SAVE AREA
!

    EL_FLUSH [.ICTLR] = TRUE;                                   ! MMM

    incr COUNT from 0 to EP_CNT - 1 do

        if .ELOG_PKT [.COUNT, EL_CONTENTS] eql EMPTY            ! IF EMPTY SLOT FOUND
        then
            begin
            index = .COUNT;                                     ! SAVE INDEX INTO THE SAVE AREA
            exitloop;
            end;

    if .index lss 0
    then
        index = EP_CNT;                                         ! IF NO SLOT FOUND, USE LAST SPARE SLOT

!
! SAVE THE PACKET CONTENTS
!

    SAVE_ADDR = ELOG_PKT + (.index * EP_LEN * 2);               ! ADDRESS OF THE SAVE AREA
    SAVE_ADDR [EL_CONTENTS ] = FULL;                            ! MARK IT FULL
    SAVE_ADDR [EL_CNTR] = .ICTLR;                               ! OWNERSHIP
    SRC_ADDR = .IPKT_ADDR + 6;                                  ! SETUP COPY ADDRESSES
    DST_ADDR = .SAVE_ADDR + 2;                                  !
    PACKET_LEN = ((.IPKT_ADDR [MSGLEN] + 1) / 2) + 2;           ! LENGTH OF ERROR-LOG INCLUDING ENVELOPE
```

```
;  2        if .PACKET_LEN gtru EP_LEN - 1
;  2        then
;  2            PACKET_LEN = EP_LEN - 1;                                        ! ADJUST LENGTH, IF TOO LONG
;  2
;  2        incr COUNT from 1 to .PACKET_LEN do
;  3            begin
;  3            .DST_ADDR = ..SRC_ADDR;                                         ! COPY A WORD
;  3            SRC_ADDR = .SRC_ADDR + 2;                                       ! UPDATE ADDRESS POINTERS
;  3            DST_ADDR = .DST_ADDR + 2;                                       !
;  2            end;
;  2
;  2    !    if (.CSR_MEM and .TST_PAR) and (not .PAR_TSD) and (.SAVE_ADDR [EL_FORMAT] eql 1)   ! MMM
;  2    !    then                                                                ! MMM
;  2    !       begin                                                            ! MMM
;  2    !       WRT_RDRX (RCIP, RC_ALL, ALL_ONES);                               ! MMM
;  2    !       PAR_TSD = TRUE;                                                  ! MMM
;  2    !       INIT_TEST ();                                                    ! MMM
;  2    !       return;                                                          ! MMM
;  2    !       end;                                                             ! MMM
;  2
;  2
;  2        if .SAVE_ADDR [EL_BRR] eql TRUE                                 ! TEST ERROR LOG MESSAGE FLAGS MMM
;  2        then
;  2           PRINTF (DBM127);                                              ! BAD BLOCK REPLACEMENT REQUEST FLAG SET MMM
;  2
;  2        if .SAVE_ADDR [EL_EDR] eql TRUE                                 !ERROR DURING REPLACEMENT (BAD BLOCK) FLAG SET M
MM
;  2        then
;  2           PRINTF (DBM128);                                             !MMM
;  2
;  2        if .SAVE_ADDR [EL_SUCCESS] eql TRUE                                 !MMM
;  2        then
;  2           if .SAVE_ADDR [EL_FORMAT] lequ 4
;  2           then
;  3               begin
;  3               SOFT_ERROR (.index);                                     ! UPDATE SOFT ERROR COUNT
;  3               TEMP_UNIT = .L$LUN;                                      ! SAVE UNIT NUMBER AS KNOWN TO DRS
;  3
;  3               incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  3
;  3                  if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .SAVE_ADDR [EL_DK_NUM]) and
;  4                     (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;  3                  then
;  4                     begin
;  4                     L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];        ! CORRECT UNIT NUMBER FOR ERROR MESSAGE
;  4                     exitloop;
;  3                     end;
;  3
;  3                  case .SAVE_ADDR [EL_FORMAT] from 0 to 4 of
;  3                     set
;  3
;  3                     [0] :  if .APT_MODE                           ! CONTROLLER ERROR
;  3                            then
;  3                               ERR_SOFT_RTNE_APT (50, .index)
```

```
;  3                          else
;  3                              ERR_SOFT_RTNE (50);
;  3
;  3                  [1] :   if .APT MODE                          ! HOST MEMORY ACCESS ERROR
;  3                          then
;  3                              ERR_SOFT_RTNE_APT (51, .index)
;  3                          else
;  3                              ERR_SOFT_RTNE (51);
;  3
;  3                  [2] :   if .APT_MODE                          ! DISK TRANSFER ERROR
;  3                          then
;  3                              ERR_SOFT_RTNE_APT (52, .index)
;  3                          else
;  3                              ERR_SOFT_RTNE (52);
;  3
;  3                  [3] :   if .APT_MODE                          ! SDI ERROR
;  3                          then
;  3                              ERR_SOFT_RTNE_APT (53, .index)
;  3                          else
;  3                              ERR_SOFT_RTNE (53);
;  3
;  3                  [4] :   if .APT_MODE                          ! SMALL DISK ERROR
;  3                          then
;  3                              ERR_SOFT_RTNE_APT (54, .index)
;  3                          else
;  3                              ERR_SOFT_RTNE (54);
;  3                  tes;
;  3
;  3
;  3          L$LUN = .TEMP_UNIT;                               ! RESTORE UNIT NUMBER
;  2          end;
;  2
;  2  !MMM    else
;  2  !MMM        PRINTF (DBM109, .SAVE_ADDR [EL_FORMAT]);      ! ERROR LOG FORMAT UNKNOWN
;  2
;  2      PRINTB (CRLF);                                        ! EXTRA CARRIEGE-RETURN/LINE-FEED
;  2      EMS_EL (.index);                                      ! PRINT PACKET CONTENTS
;  1      end;                                                  !

                                         .SBTTL  DATAGM RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          DATAGM::JSR     R1,$SAVE5                        ;                    6651
000004  012704  177777                   MOV     #-1,R4                           ; *.INDEX            6663
000010  013700  000104'                  MOV     ICTLR,R0                         ;                    6678
000014  006300                           ASL     R0
000016  012760  000001  000106'          MOV     #1,EL.FLUSH(R0)
000024  005001                           CLR     R1                               ; COUNT              6680
000026  010146                   1$:     MOV     R1,-(SP)                         ; COUNT,*            6682
000030  012746  000102                   MOV     #102,-(SP)
000034  004737  000000G                  JSR     PC,BL$MUL
000040  022626                           CMP     (SP)+,(SP)+
000042  105760  000001G                  TSTB    ELOG.PKT+1(R0)
000046  001002                           BNE     2$
```

```
000050  010104                      MOV     R1,R4                   ; COUNT,INDEX           6685
000052  000405                      BR      3$                      ;                       6684
000054  005201              2$:     INC     R1                      ; COUNT                 6680
000056  020127  000013              CMP     R1,#13                  ; COUNT,*
000062  003761                      BLE     1$
000064  005704                      TST     R4                      ; INDEX                 6689
000066  002002              3$:     BGE     4$
000070  012704  000014              MOV     #14,R4                  ; *,INDEX               6691
000074  010446              4$:     MOV     R4,-(SP)                ; INDEX,*               6697
000076  012746  000102              MOV     #102,-(SP)
000102  004737  000000G             JSR     PC,BL$MUL
000106  062700  000000G             ADD     #ELOG.PKT,R0
000112  010001                      MOV     R0,R1                   ; *,SAVE.ADDR
000114  111761  000001              MOVB    (PC),1(R1)              ; *,*(SAVE.ADDR)        6698
000120  113711  000104'             MOVB    ICTLR,(R1)              ; *,SAVE.ADDR           6699
000124  013700  000000G             MOV     IPKT.ADDR,R0            ;                       6700
000130  012705  000006              MOV     #6,R5                   ; *,SRC.ADDR
000134  060005                      ADD     R0,R5                   ; *,SRC.ADDR
000136  012703  000002              MOV     #2,R3                   ; *,DST.ADDR            6701
000142  060103                      ADD     R1,R3                   ; SAVE.ADDR,DST.ADDR
000144  016016  000006              MOV     6(R0),(SP)              ;                       6702
000150  005216                      INC     (SP)
000152  012746  000002              MOV     #2,-(SP)
000156  004737  000000G             JSR     PC,BL$DIV
000162  062700  000002              ADD     #2,R0
000166  020027  000040              CMP     R0,#40                  ; PACKET.LEN,*          6704
000172  101402                      BLOS    5$
000174  012700  000040              MOV     #40,R0                  ; *,PACKET.LEN          6706
000200  005002              5$:     CLR     R2                      ; COUNT                 6708
000202  000401                      BR      7$
000204  012523              6$:     MOV     (R5)+,(R3)+             ; SRC.ADDR,DST.ADDR     6710
000206  005202              7$:     INC     R2                      ; COUNT                 6708
000210  020200                      CMP     R2,R0                   ; COUNT,PACKET.LEN
000212  003774                      BLE     6$
000214  012702  000016              MOV     #16,R2                  ;                       6725
000220  060102                      ADD     R1,R2                   ; SAVE.ADDR,*
000222  032712  020000              BIT     #20000,(R2)
000226  001407                      BEQ     8$
000230  012716  000000G             MOV     #DBM127,(SP)            ;                       6727
000234  012746  000001              MOV     #1,-(SP)
000240  010600                      MOV     SP,R0                   ; SP,*
000242  104417                      TRAP    17
000244  005726                      TST     (SP)+
000246  032712  010000      8$:     BIT     #10000,(R2)             ;                       6729
000252  001407                      BEQ     9$
000254  012716  000000G             MOV     #DBM128,(SP)            ;                       6731
000260  012746  000001              MOV     #1,-(SP)
000264  010600                      MOV     SP,R0                   ; SP,*
000266  104417                      TRAP    17
000270  005726                      TST     (SP)+
000272  005712              9$:     TST     (R2)                    ;                       6733
000274  100136                      BPL     27$
000276  121227  000004              CMPB    (R2),#4                 ;                       6735
```

```
000302  101133                          BHI     27$
000304  010416                          MOV     R4,(SP)                 ; INDEX,*                       6738
000306  004737  000000V                 JSR     PC,SOFT.ERROR
000312  013705  000000G                 MOV     L$LUN,R5                ; *,TEMP.UNIT                   6739
000316  012703  000006                  MOV     #6,R3                   ; *,OFFSET                      6741
000322  010300                  10$:    MOV     R3,R0                   ; OFFSET,*                      6743
000324  063700  000076'                 ADD     ICST.ADDR,R0
000330  016146  000012                  MOV     12(R1),-(SP)            ; *(SAVE.ADDR),*
000334  111046                          MOVB    (R0),-(SP)
000336  042716  177760                  BIC     #177760,(SP)
000342  022626                          CMP     (SP)+,(SP)+
000344  001012                          BNE     11$
000346  032710  040000                  BIT     #40000,(R0)             ;                               6744
000352  001407                          BEQ     11$
000354  011046                          MOV     (R0),-(SP)              ;                               6747
000356  000316                          SWAB    (SP)
000360  042716  177760                  BIC     #177760,(SP)
000364  012637  000000G                 MOV     (SP)+,L$LUN
000370  000405                          BR      12$                     ;                               6746
000372  062703  000024          11$:    ADD     #24,R3                  ; *,OFFSET                      6741
000376  020327  000102                  CMP     R3,#102                 ; OFFSET,*
000402  003747                          BLE     10$
000404  005000                  12$:    CLR     R0                      ;                               6754
000406  153700  001254'                 BISB    APT.MODE,R0
000412  005001                          CLR     R1                      ;                               6751
000414  151201                          BISB    (R2),R1
000416  006301                          ASL     R1
000420  066107  000000'                 ADD     P.AAA(R1),PC            ; Case dispatch
000424  032700  000001          14$:    BIT     #1,R0                   ;                               6754
000430  001403                          BEQ     15$
000432  012716  000062                  MOV     #62,(SP)                ;                               6756
000436  000442                          BR      23$
000440  012716  000062          15$:    MOV     #62,(SP)                ;                               6758
000444  000446                          BR      25$
000446  032700  000001          16$:    BIT     #1,R0                   ;                               6760
000452  001403                          BEQ     17$
000454  012716  000063                  MOV     #63,(SP)                ;                               6762
000460  000431                          BR      23$
000462  012716  000063          17$:    MOV     #63,(SP)                ;                               6764
000466  000435                          BR      25$
000470  032700  000001          18$:    BIT     #1,R0                   ;                               6766
000474  001403                          BEQ     19$
000476  012716  000064                  MOV     #64,(SP)                ;                               6768
000502  000420                          BR      23$
000504  012716  000064          19$:    MOV     #64,(SP)                ;                               6770
000510  000424                          BR      25$
000512  032700  000001          20$:    BIT     #1,R0                   ;                               6772
000516  001403                          BEQ     21$
000520  012716  000065                  MOV     #65,(SP)                ;                               6774
000524  000407                          BR      23$
000526  012716  000065          21$:    MOV     #65,(SP)                ;                               6776
000532  000413                          BR      25$
000534  006000                  22$:    ROR     R0                      ;                               6778
```

```
000536  103007                          BCC     24$
000540  012716  000066                  MOV     #66,(SP)                        ;                               6780
000544  010446                  23$:    MOV     R4,-(SP)                        ; INDEX,*
000546  004737  000000V                 JSR     PC,ERR.SOFT.RTNE.APT
000552  005726                          TST     (SP)+
000554  000404                          BR      26$                             ;                               6778
000556  012716  000066          24$:    MOV     #66,(SP)                        ;                               6782
000562  004737  000000V         25$:    JSR     PC,ERR.SOFT.RTNE
000566  010537  000000G         26$:    MOV     R5,L$LUN                        ; TEMP.UNIT,*                   6786
000572  012716  000000G         27$:    MOV     #CRLF,(SP)                      ;                               6792
000576  012746  000001                  MOV     #1,-(SP)
000602  010600                          MOV     SP,R0                           ; SP,*
000604  104414                          TRAP    14
000606  010416                          MOV     R4,(SP)                         ; INDEX,*                       6793
000610  004737  000000G                 JSR     PC,EMS.EL
000614  062706  000010                  ADD     #10,SP                          ;                               6663
000620  000207                          RTS     PC                              ;                               6651

; Routine Size:  201 words,     Routine Base:  $CODE$ + 24234
; Maximum stack depth per invocation:  12 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:                                          ; CASE Table for DATAGM+0420    6751
000000  000000                  13$:    .WORD   0                               ; [14$]
000002  000022                          .WORD   22                              ; [16$]
000004  000044                          .WORD   44                              ; [18$]
000006  000066                          .WORD   66                              ; [20$]
000010  000110                          .WORD   110                             ; [22$]


;     6795  1
;     6796  1
;     6797  1     !MMM     if (.SAVE_ADDR [EL_CRN_LO] eql .LAST_PKT [.ICTLR, LAST_CRN_LO]) and
;     6798  1     !MMM        (.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI])
;     6799  1     !MMM     then
;     6800  1     !MMM        begin                                                    ! LOG REFERS TO THE LAST RESPONSE RE
CEIVED
;     6801  1
;     6802  1     !MMM        if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED             ! IF SOFT ERROR HAD OCCURED
;     6803  1     !MMM        then
;     6804  1
;     6805  1     !MMM           if .SAVE_ADDR [EL_FORMAT] lequ 4
;     6806  1     !MMM           then
;     6807  1     !MMM              begin
;     6808  1     !MMM              SOFT_ERROR (.index);                          ! UPDATE SOFT ERROR COUNT
;     6809  1     !MMM              TEMP_UNIT = .L$LUN;                           ! SAVE UNIT NUMBER AS KNOWN TO DRS
;     6810  1
;     6811  1     !MMM              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;     6812  1
;     6813  1     !MMM                 if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .SAVE_ADDR [EL_DK_NUM]) and
;     6814  1     !MMM                    (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
```

```
;   1   !MMM                    then
;   1   !MMM                        begin
;   1   !MMM                        L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT]; ! CORRECT UNIT NUMBER FOR ERROR MESSAGE
;   1   !MMM                        exitloop;
;   1   !MMM                        end;
;   1
;   1   !MMM                case .SAVE_ADDR [EL_FORMAT] from 0 to 4 of
;   1   !MMM                    set
;   1
;   1
;   1   !MMM                    [0] :       if .APT_MODE                        ! CONTROLLER ERROR
;   1   !MMM                                then
;   1   !MMM                                    ERR_SOFT_RTNE_APT (50, .index)
;   1   !MMM                                else
;   1   !MMM                                    ERR_SOFT_RTNE (50);
;   1
;   1   !MMM                    [1] :       if .APT_MODE                        ! HOST MEMORY ACCESS ERROR
;   1   !MMM                                then
;   1   !MMM                                    ERR_SOFT_RTNE_APT (51, .index)
;   1   !MMM                                else
;   1   !MMM                                    ERR_SOFT_RTNE (51);
;   1
;   1   !MMM                    [2] :       if .APT_MODE                        ! DISK TRANSFER ERROR
;   1   !MMM                                then
;   1   !MMM                                    ERR_SOFT_RTNE_APT (52, .index)
;   1   !MMM                                else
;   1   !MMM                                    ERR_SOFT_RTNE (52);
;   1
;   1   !MMM                    [3] :       if .APT_MODE                        ! SDI ERROR
;   1   !MMM                                then
;   1   !MMM                                    ERR_SOFT_RTNE_APT (53, .index)
;   1   !MMM                                else
;   1   !MMM                                    ERR_SOFT_RTNE (53);
;   1
;   1   !MMM                    [4] :       if .APT_MODE                        ! SMALL DISK ERROR
;   1   !MMM                                then
;   1   !MMM                                    ERR_SOFT_RTNE_APT (54, .index)
;   1   !MMM                                else
;   1   !MMM                                    ERR_SOFT_RTNE (54);
;   1   !MMM                    tes;
;   1
;   1
;   1   !MMM                L$LUN = .TEMP_UNIT;                             ! RESTORE UNIT NUMBER
;   1   !MMM                SFT_ERR_PRINTED = TRUE;                         ! SOFT ERROR PRINTOUT OCCURED
;   1   !MMM                end
;   1
;   1   !MMM            else
;   1   !MMM                PRINTF (DBM109, .SAVE_ADDR [EL_FORMAT]);        ! ERROR LOG FORMAT UNKNOWN
;   1
;   1   !MMM        if not (.SFT_ERR_PRINTED)
;   1
;   1   !MMM        then
;   1   !MMM            PRINTB (CRLF);                                  ! EXTRA CARRIEGE-RETURN/LINE-FEED
```

```
;  1
;  1    !MMM    EMS_EL (.index);                                            ! PRINT PACKET CONTENTS
;  1    !MMM    end                                                         ! CORRESPONDING RESPONSE RECEIVED
;  1
;  1    !MMM    else
;  1
;  1    !MMM    if (.SAVE_ADDR [EL_CRN_HI] lssu .LAST_PKT [.ICTLR, LAST_CRN_HI]) or
;  1    !MMM       ((.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI]) and
;  1    !MMM        (.SAVE_ADDR [EL_CRN_LO] lssu .LAST_PKT [.ICTLR, LAST_CRN_LO]))
;  1
;  1    !MMM    if  (.QIO [.ICTLR] eqlu 0)                                  ! MMM
;  1
;  1    !MMM    then
;  1    !MMM        begin                                                   ! LOG REFERS TO SOME PREVIOUS RESPONSE
;  1    !MMM        PRINTB (CRLF);                                          ! CARRIAGE-RETURN/LINE-FEED
;  1    !MMM        EMS_EL (.index);                                        ! PRINT PACKET CONTENTS
;  1    !MMM        end;
;  1
;  1    !MMM    end;
;  1
;  1
;  1
```

```
;  1  GLOBAL routine SOFT_ERROR (index) : novalue =
;  1
;  1  !+
;  1  !      THIS ROUTINE UPDATES THE SOFT ERROR COUNT IN THE TALLY TABLE FOR EACH
;  1  !      ERROR LOG MESSAGE RECEIVED
;  1  !
;  1  !      IMPLICIT INPUTS:
;  1  !              ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;  1  !-
;  1
;  2      begin
;  2
;  2      local
;  2          FOUND: byte initial (byte (FALSE)),
;  2          SOFT_OCCURED : byte initial (byte (FALSE)),
;  2          UNIT: word,
;  2          ERROR_CODE : byte,
;  2          ERROR_SUB : word,
;  2          RETRIES : word,
;  2          TALLY_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;  2          ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;  2
;  2      ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                          ! ADDR OF ERROR PKT
;  2      ERROR_CODE = .ELOG_ADDR [EL_CODE];                                     ! ERROR CODE
;  2      ERROR_SUB = .ELOG_ADDR [EL_SUBCODE];                                   ! ERROR SUBCODE
;  2
;  2      if (BIT_TST (SWP_FLAGS, SWF_TRY)) and
;  3         (.ELOG_ADDR [EL_FORMAT] eql 2)
;  2      then
;  2          RETRIES = .ELOG_ADDR [EL_RETRY]                                    ! COUNT EACH RETRY
;  2      else
;  2          RETRIES = 1;                                                       ! IGNORE RETRIES
;  2
;  2      if .RETRIES eql 0                                                      ! IN CASE OF A BUG
;  2      then
;  2          RETRIES = 1;
;  2
;  2      incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  2
;  2          if (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and        ! DISK TO UNIT NO.
;  3             (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_ADDR [EL_DK_NUM])
;  2          then
;  3              begin
;  3              FOUND = TRUE;
;  3              UNIT = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];                 ! DISK'S UNIT NO.
;  3              exitloop;
;  2              end;
;  2
;  2  !   if (.ELOG_ADDR [EL_SUCCESS]) or
;  2  !      (.ELOG_ADDR [EL_CONTINUE])
;  2  !   then
;  2          SOFT_OCCURED = TRUE;                                               ! SOFT ERROR FLAG
;  2
```

```
        if .FOUND                                                                            ! IF UNIT FOUND
        then
            begin
            TALLY_ADDR = TALLY + (.UNIT * TALLY_LEN * 2);                                    ! ADDR OF TALLY TBL

            if .SOFT_OCCURED                                                                 ! FOR SOFT ERRORRS
            then
                selectoneu .ERROR_CODE of
                    set

                    [ST_MFE]:       TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES;! SOFT-MEDIA FORMAT

                    [ST_DAT]:       if .ERROR_SUB eql 2                                      ! SOFT-DATA
                                    then
                                        TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
                                    else
                                        TALLY_ADDR [ERR_SFT_DAT] = .TALLY_ADDR [ERR_SFT_DAT] + .RETRIES;

                    [ST_HST]:       TALLY_ADDR [ERR_SFT_HST] = .TALLY_ADDR [ERR_SFT_HST] + .RETRIES;! SOFT-HOST ACCESS

                    [ST_CNT]:       C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + .RETRIES;
                                                                                             ! SOFT-CONTROLLER

                    [ST_DRV]:       if .ERROR_SUB eql 3                                      ! SOFT_DRIVE
                                    then
                                        TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
                                    else
                                        TALLY_ADDR [ERR_SFT_DRV] = .TALLY_ADDR [ERR_SFT_DRV] + .RETRIES;
                    tes
            else

                if (.ELOG_ADDR [EL_CRN_LO] eql 0) and
                   (.ELOG_ADDR [EL_CRN_HI] eql 0)
                then
                    selectoneu .ERROR_CODE of
                        set

                        [ST_MFE]:   TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1;        ! HARD-MEDIA FORMAT

                        [ST_DAT]:   if .ERROR_SUB eql 2                                      ! HARD-DATA
                                    then
                                        TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
                                    else
                                        TALLY_ADDR [ERR_HRD_DAT] = .TALLY_ADDR [ERR_HRD_DAT] + 1;

                        [ST_HST]:   TALLY_ADDR [ERR_HRD_HST] = .TALLY_ADDR [ERR_HRD_HST] + 1;        ! HARD-HOST ACCESS

                        [ST_CNT]:   C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
                                                                                             ! HARD-CONTROLLER

                        [ST_DRV]:   if .ERROR_SUB eql 3                                      ! HARD-DRIVE
                                    then
                                        TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
```

```
;   6996  3                                  else
;   6997  3                                     TALLY_ADDR [ERR_HRD_DRV] = .TALLY_ADDR [ERR_HRD_DRV] + 1;
;   6998  3                       tes;
;   6999  3
;   7000  3              end
;   7001  2          else                                                                          ! UNIT NOT FOUND
;   7002  2
;   7003  2              if .SOFT_OCCURED
;   7004  2              then
;   7005  2                 C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + 1
;   7006  2              else
;   7007  2                 C_ERR_TBL [.ICTLP, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;   7008  2
;   7009  1          end;                                                                          ! RTNE SOFT_ERROR


                                                 .SBTTL  SOFT.ERROR RDRX INTERRUPT SERVICE ROUTINES
025056                                           .PSECT  $CODE$,  RO

000000  004137  000000G                 SOFT.ERROR::
                                                 JSR     R1,$SAVE5                    ;                              6890
000004  005746                                   TST     -(SP)
000006  105046                                   CLRB    -(SP)                        ; FOUND                        6900
000010  105046                                   CLRB    -(SP)                        ; SOFT.OCCURED
000012  016646  000024                           MOV     24(SP),-(SP)                 ; INDEX,*                      6912
000016  012746  000102                           MOV     #102,-(SP)
000022  004737  000000G                          JSR     PC,BL$MUL
000026  062700  000000G                          ADD     #ELOG.PKT,R0
000032  010001                                   MOV     R0,R1                        ; *,ELOG.ADDR
000034  116100  000020                           MOVB    20(R1),R0                    ; *(ELOG.ADDR),*               6913
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       6914
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                 ;                              6916
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),*               6917
000116  001004                                   BNE     1$
000120  005004                                   CLR     R4                           ; RETRIES                      6919
000122  156104  000051                           BISB    51(R1),R4                    ; *(ELOG.ADDR),RETRIES
000126  000402                                   BR      2$                           ;                              6916
000130  012704  000001                  1$:      MOV     #1,R4                        ; *,RETRIES                    6921
000134  005704                          2$:      TST     R4                           ; RETRIES                      6923
000136  001002                                   BNE     3$
000140  012704  000001                           MOV     #1,R4                        ; *,RETRIES                    6925
```

```
000144  012702  000006           3$:   MOV     #6,R2                  ; *,OFFSET                  6927
000150  010200                   4$:   MOV     R2,R0                  ; OFFSET,*                  6929
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),*            6930
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                   6933
000210  011046                         MOV     (R0),-(SP)             ;                           6934
000212  000316                         SWAB    (SP)
000214  042716  177760                 BIC     #177760,(SP)
000220  012666  000010                 MOV     (SP)+,10(SP)           ; *,UNIT
000224  000405                         BR      6$                     ;                           6932
000226  062702  000024           5$:   ADD     #24,R2                 ; *,OFFSET                  6927
000232  020227  000102                 CMP     R2,#102                ; OFFSET,*
000236  003744                         BLE     4$
000240  112766  000001  000004   6$:   MOVB    #1,4(SP)               ; *,SOFT.OCCURED            6941
000246  032766  000001  000006         BIT     #1,6(SP)               ; *,FOUND                   6943
000254  001002                         BNE     7$
000256  000137  025744'                JMP     22$
000262  016616  000010           7$:   MOV     10(SP),(SP)            ; UNIT,*                    6946
000266  012746  000066                 MOV     #66,-(SP)
000272  004737  000000G                JSR     PC,BL$MUL
000276  062700  000000G                ADD     #TALLY,R0
000302  032766  000001  000006         BIT     #1,6(SP)               ; *,SOFT.OCCURED            6948
000310  001503                         BEQ     14$
000312  120327  000005                 CMPB    R3,#5                  ; ERROR.CODE,*              6953
000316  001462                         BEQ     12$
000320  120327  000010                 CMPB    R3,#10                 ; ERROR.CODE,*              6955
000324  001022                         BNE     9$
000326  012702  000052                 MOV     #52,R2                 ;                           6957
000332  060002                         ADD     R0,R2                  ; TALLY.ADDR,*
000334  020527  000002                 CMP     R5,#2                  ; ERROR.SUB,*               6955
000340  001005                         BNE     8$
000342  005001                         CLR     R1                     ;                           6957
000344  151201                         BISB    (R2),R1
000346  060401                         ADD     R4,R1                  ; RETRIES,*
000350  110112                         MOVB    R1,(R2)
000352  000543                         BR      21$                    ;                           6955
000354  005001                   8$:   CLR     R1                     ;                           6959
000356  156201  000001                 BISB    1(R2),R1
000362  060401                         ADD     R4,R1                  ; RETRIES,*
000364  110162  000001                 MOVB    R1,1(R2)
000370  000534                         BR      21$                    ;                           6950
000372  120327  000011           9$:   CMPB    R3,#11                 ; ERROR.CODE,*              6961
000376  001007                         BNE     10$
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$                             ;                       6950
000416  120327  000012              10$:    CMPB    R3,#12                          ; ERROR.CODE,*          6963
000422  001012                              BNE     11$
000424  013702  000104'                     MOV     ICTLR,R2
000430  006302                              ASL     R2
000432  005001                              CLR     R1
000434  156201  000001G                     BISB    C.ERR.TBL+1(R2),R1
000440  060401                              ADD     R4,R1                           ; RETRIES,*
000442  110162  000001G                     MOVB    R1,C.ERR.TBL+1(R2)
000446  000505                              BR      21$                             ;                       6950
000450  120327  000013              11$:    CMPB    R3,#13                          ; ERROR.CODE,*          6966
000454  001102                              BNE     21$
000456  020527  000003                      CMP     R5,#3                           ; ERROR.SUB,*
000462  001007                              BNE     13$
000464  005001                      12$:    CLR     R1                              ;                       6968
000466  156001  000052                      BISB    52(R0),R1                       ; *(TALLY.ADDR),*
000472  060401                              ADD     R4,R1                           ; RETRIES,*
000474  110160  000052                      MOVB    R1,52(R0)                       ; *,*(TALLY.ADDR)
000500  000470                              BR      21$                             ;                       6966
000502  005001                      13$:    CLR     R1                              ;                       6970
000504  156001  000054                      BISB    54(R0),R1                       ; *(TALLY.ADDR),*
000510  060401                              ADD     R4,R1                           ; RETRIES,*
000512  110160  000054                      MOVB    R1,54(R0)                       ; *,*(TALLY.ADDR)
000516  000461                              BR      21$                             ;                       6950
000520  005761  000006              14$:    TST     6(R1)                           ; *(ELOG.ADDR)          6974
000524  001056                              BNE     21$
000526  005761  000010                      TST     10(R1)                          ; *(ELOG.ADDR)          6975
000532  001053                              BNE     21$
000534  120327  000005                      CMPB    R3,#5                           ; ERROR.CODE,*          6980
000540  001443                              BEQ     19$
000542  120327  000010                      CMPB    R3,#10                          ; ERROR.CODE,*          6982
000546  001013                              BNE     16$
000550  012704  000046                      MOV     #46,R4                          ;                       6984
000554  060004                              ADD     R0,R4                           ; TALLY.ADDR,*
000556  020527  000002                      CMP     R5,#2                           ; ERROR.SUB,*           6982
000562  001002                              BNE     15$
000564  105214                              INCB    (R4)                            ;                       6984
000566  000435                              BR      21$                             ;                       6982
000570  105264  000001              15$:    INCB    1(R4)                           ;                       6986
000574  000432                              BR      21$                             ;                       6977
000576  120327  000011              16$:    CMPB    R3,#11                          ; ERROR.CODE,*          6988
000602  001003                              BNE     17$
000604  105260  000051                      INCB    51(R0)                          ; *(TALLY.ADDR)
000610  000424                              BR      21$                             ;                       6977
000612  120327  000012              17$:    CMPB    R3,#12                          ; ERROR.CODE,*          6990
000616  001006                              BNE     18$
000620  013702  000104'                     MOV     ICTLR,R2
000624  006302                              ASL     R2
000626  105262  000000G                     INCB    C.ERR.TBL(R2)
000632  000413                              BR      21$                             ;                       6977
000634  120327  000013              18$:    CMPB    R3,#13                          ; ERROR.CODE,*          6993
000640  001010                              BNE     21$
000642  020527  000003                      CMP     R5,#3                           ; ERROR.SUB,*
```

```
000646  001003                         BNE     20$
000650  105260  000046         19$:    INCB    46(R0)                ; *(TALLY.ADDR)        6995
000654  000402                         BR      21$                   ;                      6993
000656  105260  000050         20$:    INCB    50(R0)                ; *(TALLY.ADDR)        6997
000662  005726                 21$:    TST     (SP)+                 ;                      6945
000664  000415                         BR      24$                   ;                      6943
000666  013700  000104'        22$:    MOV     ICTLR,R0              ;                      7005
000672  006300                         ASL     R0
000674  062700  000000G                ADD     #C.ERR.TBL,R0
000700  032766  000001  000004         BIT     #1,4(SP)              ; *.SOFT.OCCURED       7003
000706  001403                         BEQ     23$
000710  105260  000001                 INCB    1(R0)                 ;                      7005
000714  000401                         BR      24$                   ;                      7003
000716  105210                 23$:    INCB    (R0)                  ;                      7007
000720  062706  000012         24$:    ADD     #12,SP                ;                      6890
000724  000207                         RTS     PC

; Routine Size:  235 words,      Routine Base:  $CODE$ + 25056
; Maximum stack depth per invocation:  13 words
```

```
routine ERR_HRD_RTNE (ERRNUM) : novalue =

!+
! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
! THE SAME EFFECT WITHOUT ISSUING THE CALL
!-

    begin

    local
        CUR_PRIORITY : word;

    builtin
        PC;

    GETPRI (CUR_PRIORITY);
!ZZZ    SETPRI (PRI04);                                 ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
    SETPRI (.BRLEVEL);                                  ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW    ZZZ

    if (.ERRNUM lequ 34) or                             ! FOR NON-BAD BLOCK TYPE ERRORS
       (.ERRNUM gtru 38) or
       (.ERRNUM eql 36) or
       (.ERRNUM eql 37)
    then

        if BIT_TST (SWP_FLAGS, SWF_HRD)                 ! IF ERRORS TO BE TREATED NORMALLY
        then

!ZZZ        case .ERRNUM from 31 to 45 of
            case .ERRNUM from 31 to 73 of               !INCLUDE DUP NUMBERS (60-73)         ZZZ
        set

                [31]:   ERRHRD (31, EGH_30, EMS_30);    ! INVALID COMMAND

                [32]:   ERRHRD (32, EGH_30, EMS_30);    ! COMMAND ABORTED

                [33]:   ;                               !

                [34]:   ;                               !

                [35]:   ;                               ! MEDIA FORMAT ERROR

                [36]:   ERRHRD (36, EGH_30, EMS_30);    ! WRITE PROTECTED

                [37]:   ERRHRD (37, EGH_30, EMS_30);    ! COMPARE ERROR

                [38]:   ;                               ! DATA ERROR

                [39]:   ERRHRD (39, EGH_30, EMS_30);    ! HOST BUFFER ACCESS ERROR

                [40]:   ERRHRD (40, EGH_30, EMS_30);    ! CONTROLLER ERROR

                [41]:   ERRHRD (41, EGH_30, EMS_30);    ! DRIVE ERROR
```

```
                    [42]:   ERRHRD (42, EGH_30, 0);         ! HOST WRITE COMPARE ERROR

                    [43]:   ERRHRD (43, EGH_30, EMS_30);    ! MESSAGE FROM INTERNAL DIAGNOSTICS

                    [44]:   ERRHRD (44, EGH_30, EMS_30);    ! DUPLICATE UNIT NUMBER

                    [45]:   ERRHRD (45, EGH_30, EMS_30);    ! INVALID END CODE

                    [46]:   ;                               !LEAVE ROOM FOR SOFT ERROR NUMBERS AND SOME PADDING  ZZZ
                    [47]:   ;                               !
                    [48]:   ;                               !
                    [49]:   ;                               !
                    [50]:   ;                               !
                    [51]:   ;                               !
                    [52]:   ;                               !
                    [53]:   ;                               !
                    [54]:   ;                               !
                    [55]:   ;                               !
                    [56]:   ;                               !
                    [57]:   ;                               !
                    [58]:   ;                               !
                    [59]:   ;                               !

                    [60]:  ERRHRD (60, EH_12, EMS_30);      !NOT USED                                 ZZZ
                    [61]:  ERRHRD (61, EH_13, EMS_30);      !SUCCESSFUL MESSAGE                       ZZZ
                    [62]:  ERRHRD (62, EH_13, EMS_30);      !ILLEGAL UNIT NUMBER                      ZZZ
                    [63]:  ERRHRD (63, EH_13, EMS_30);      !ILLEGAL RELATIVE OR PHYSICAL BLOCK       ZZZ
                    [64]:  ERRHRD (64, EH_13, EMS_30);      !DEVICE ERROR                             ZZZ
                    [65]:  ERRHRD (65, EH_13, EMS_30);      !ZERO LENGTH MESSAGE                      ZZZ
                    [66]:  ERRHRD (66, EH_8,  EMS_30);      !DUP UNKNOWN STATUS CODE                  ZZZ
                    [67]:  ERRHRD (67, EH_7,  EMS_30);      !INVALID COMMAND                          ZZZ
                    [68]:  ERRHRD (68, EH_7,  EMS_30);      !NO REGION AVAILABLE                      ZZZ
                    [69]:  ERRHRD (69, EH_7,  EMS_30);      !NO REGION SUITABLE                       ZZZ
                    [70]:  ERRHRD (70, EH_7,  EMS_30);      !PROGRAM NOT KNOWN                        ZZZ
                    [71]:  ERRHRD (71, EH_7,  EMS_30);      !LOAD FAILURE                             ZZZ
                    [72]:  ERRHRD (72, EH_7,  EMS_30);      !STANDALONE                               ZZZ
                    [73]:  ERRHRD (73, EH_8,  EMS_30);      !DUP UNKNOWN STATUS CODE                  ZZZ

                    tes
            else
                begin
                !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
                PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);     ! PRINT ERROR MESSAGE JUST LIKE DRS

                if .ERRNUM neq 42
                then
                    begin
                    PRINTB (HRD_SUB);                       ! NEXT LINE FOR NON-HOST COMPARE ERRORS
                    EMS_ERR ();                             ! PRINT REST OF THE INFORMATION
                    end;
                end;
```

```
;   7116  2         if (.ERRNUM eql 35) or                       ! FOR BAD-BLOCK TYPE ERRORS
;   7117  3            (.ERRNUM eql 38)
;   7118  2         then
;   7119  2
;   7120  3             if BIT_TST (SWP_FLAGS, SWF_BLK)          ! IF ERRORS TO BE TREATED NORMALLY
;   7121  2             then
;   7122  2
;   7123  2                 selectoneu .ERRNUM of
;   7124  2                     set
;   7125  2
;   7126  2                     [35]:   ERRHRD (35, EGH_30, EMS_30);   ! MEDIA FORMAT ERROR
;   7127  2
;   7128  2                     [38]:   ERRHRD (38, EGH_30, EMS_30);   ! DATA ERROR
;   7129  2                     tes
;   7130  2             else
;   7131  3                 begin
;   7132  3                 !****increment error count           ! INCREMENT TOTAL ERROR COUNT
;   7133  3                 PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);  ! PRINT ERROR LINE JUST LIKE DRS
;   7134  3                 PRINTB (HRD_SUB);                    ! PRINT NEXT LINE TOO
;   7135  3                 EMS_ERR ();                          ! PRINT REST OF THE INFORMATION
;   7136  2                 end;
;   7137  2
;   7138  2         SETPRI (.CUR_PRIORITY);                      ! PRIORITY BACK TO NORMAL
;   7139  2
;   7140  1         end;


                                         .SBTTL  ERR.HRD.RTNE RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          ERR.HRD.RTNE:
                                         JSR     R1,$SAVE2                           ;                      7010
000004  104440                           TRAP    40                                  ;                      7025
000006  010002                           MOV     R0,R2                               ; *,CUR.PRIORITY
000010  013700  000000G                  MOV     BRLEVEL,R0                          ;                      7027
000014  104441                           TRAP    41
000016  016601  000010                   MOV     10(SP),R1                           ; ERRNUM,*             7029
000022  020127  000042                   CMP     R1,#42
000026  101411                           BLOS    1$
000030  020127  000046                   CMP     R1,#46                              ;                      7030
000034  101006                           BHI     1$
000036  020127  000044                   CMP     R1,#44                              ;                      7031
000042  001403                           BEQ     1$
000044  020127  000045                   CMP     R1,#45                              ;                      7032
000050  001176                           BNE     27$
000052  032737  010000  000000G  1$:     BIT     #10000,SWP.FLAGS                    ;                      7035
000060  001002                           BNE     2$
000062  000137  026500'                  JMP     31$
000066  010100                   2$:     MOV     R1,R0                               ;                      7039
000070  162700  000037                   SUB     #37,R0
000074  006300                           ASL     R0
000076  066007  000012'                  ADD     P.AAB(R0),PC                        ; Case dispatch
000102  104456                   4$:     TRAP    56                                  ;                      7042
000104  000037                           .WORD   37
000106  000000G                          .WORD   EGH.30
```

```
000110  000000G                  .WORD  EMS.30
000112  000567                   BR     30$                       ;   7039
000114  104456         5$:       TRAP   56                        ;   7044
000116  000040                   .WORD  40
000120  000000G                  .WORD  EGH.30
000122  000000G                  .WORD  EMS.30
000124  000562                   BR     30$                       ;   7039
000126  104456         6$:       TRAP   56                        ;   7052
000130  000044                   .WORD  44
000132  000000G                  .WORD  EGH.30
000134  000000G                  .WORD  EMS.30
000136  000555                   BR     30$                       ;   7039
000140  104456         7$:       TRAP   56                        ;   7054
000142  000045                   .WORD  45
000144  000000G                  .WORD  EGH.30
000146  000000G                  .WORD  EMS.30
000150  000550                   BR     30$                       ;   7039
000152  104456         8$:       TRAP   56                        ;   7058
000154  000047                   .WORD  47
000156  000000G                  .WORD  EGH.30
000160  000000G                  .WORD  EMS.30
000162  000574                   BR     33$                       ;   7039
000164  104456         9$:       TRAP   56                        ;   7060
000166  000050                   .WORD  50
000170  000000G                  .WORD  EGH.30
000172  000000G                  .WORD  EMS.30
000174  000567                   BR     33$                       ;   7039
000176  104456         10$:      TRAP   56                        ;   7062
000200  000051                   .WORD  51
000202  000000G                  .WORD  EGH.30
000204  000000G                  .WORD  EMS.30
000206  000562                   BR     33$                       ;   7039
000210  104456         11$:      TRAP   56                        ;   7064
000212  000052                   .WORD  52
000214  000000G                  .WORD  EGH.30
000216  000000                   .WORD  0
000220  000555                   BR     33$                       ;   7039
000222  104456         12$:      TRAP   56                        ;   7066
000224  000053                   .WORD  53
000226  000000G                  .WORD  EGH.30
000230  000000G                  .WORD  EMS.30
000232  000550                   BR     33$                       ;   7039
000234  104456         13$:      TRAP   56                        ;   7068
000236  000054                   .WORD  54
000240  000000G                  .WORD  EGH.30
000242  000000G                  .WORD  EMS.30
000244  000543                   BR     33$                       ;   7039
000246  104456         14$:      TRAP   56                        ;   7070
000250  000055                   .WORD  55
000252  000000G                  .WORD  EGH.30
000254  000000G                  .WORD  EMS.30
000256  000536                   BR     33$                       ;   7039
000260  104456         15$:      TRAP   56                        ;   7087
```

```
000262  000074                    .WORD   74
000264  000000G                   .WORD   EH.12
000266  000000G                   .WORD   EMS.30
000270  000531                    BR      33$                             ;       7039
000272  104456          16$:      TRAP    56                              ;       7088
000274  000075                    .WORD   75
000276  000000G                   .WORD   EH.13
000300  000000G                   .WORD   EMS.30
000302  000524                    BR      33$                             ;       7039
000304  104456          17$:      TRAP    56                              ;       7089
000306  000076                    .WORD   76
000310  000000G                   .WORD   EH.13
000312  000000G                   .WORD   EMS.30
000314  000517                    BR      33$                             ;       7039
000316  104456          18$:      TRAP    56                              ;       7090
000320  000077                    .WORD   77
000322  000000G                   .WORD   EH.13
000324  000000G                   .WORD   EMS.30
000326  000512                    BR      33$                             ;       7039
000330  104456          19$:      TRAP    56                              ;       7091
000332  000100                    .WORD   100
000334  000000G                   .WORD   EH.13
000336  000000G                   .WORD   EMS.30
000340  000505                    BR      33$                             ;       7039
000342  104456          20$:      TRAP    56                              ;       7092
000344  000101                    .WORD   101
000346  000000G                   .WORD   EH.13
000350  000000G                   .WORD   EMS.30
000352  000500                    BR      33$                             ;       7039
000354  104456          21$:      TRAP    56                              ;       7093
000356  000102                    .WORD   102
000360  000000G                   .WORD   EH.8
000362  000000G                   .WORD   EMS.30
000364  000473                    BR      33$                             ;       7039
000366  104456          22$:      TRAP    56                              ;       7094
000370  000103                    .WORD   103
000372  000000G                   .WORD   EH.7
000374  000000G                   .WORD   EMS.30
000376  000466                    BR      33$                             ;       7039
000400  104456          23$:      TRAP    56                              ;       7095
000402  000104                    .WORD   104
000404  000000G                   .WORD   EH.7
000406  000000G                   .WORD   EMS.30
000410  000461                    BR      33$                             ;       7039
000412  104456          24$:      TRAP    56                              ;       7096
000414  000105                    .WORD   105
000416  000000G                   .WORD   EH.7
000420  000000G                   .WORD   EMS.30
000422  000454                    BR      33$                             ;       7039
000424  104456          25$:      TRAP    56                              ;       7097
000426  000106                    .WORD   106
000430  000000G                   .WORD   EH.7
000432  000000G                   .WORD   EMS.30
```

```
000434  000447                          BR      33$                     ;               7039
000436  104456                  26$.    TRAP    56                      ;               7098
000440  000107                          .WORD   107
000442  000000G                         .WORD   EH.7
000444  000000G                         .WORD   EMS.30
000446  000442                  27$:    BR      33$                     ;               7039
000450  104456                  28$:    TRAP    56                      ;               7099
000452  000110                          .WORD   110
000454  000000G                         .WORD   EH.7
000456  000000G                         .WORD   EMS.30
000460  000435                          BR      33$                     ;               7039
000462  104456                  29$:    TRAP    56                      ;               7100
000464  000111                          .WORD   111
000466  000000G                         .WORD   EH.8
000470  000000G                         .WORD   EMS.30
000472  000430                  30$:    BR      33$                     ;               7035
000474  010746                  31$:    MOV     PC,-(SP)                ; PC,*          7106
000476  013746  000000G                 MOV     L$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                  ;               7108
000524  001411                          BEQ     32$
000526  012716  000000G                 MOV     #HRD.SUB,(SP)           ;               7111
000532  012746  000001                  MOV     #1,-(SP)
000536  010600                          MOV     SP,R0                   ; SP,*
000540  104414                          TRAP    14
000542  004737  000000G                 JSR     PC,EMS.ERR              ;               7112
000546  005726                          TST     (SP)+                   ;               7110
000550  062706  000012          32$:    ADD     #12,SP                  ;               7104
000554  020127  000043          33$:    CMP     R1,#43                  ;               7116
000560  001403                          BEQ     34$
000562  020127  000046                  CMP     R1,#46                  ;               7117
000566  001050                          BNE     37$
000570  032737  040000  000000G 34$:    BIT     #40000,SWP.FLAGS        ;               7120
000576  001420                          BEQ     36$
000600  020127  000043                  CMP     R1,#43                  ;               7126
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$                     ;               7123
000620  020127  000046          35$:    CMP     R1,#46                  ;               7128
000624  001031                          BNE     37$
000626  104456                          TRAP    56
000630  000046                          .WORD   46
000632  000000G                         .WORD   EGH.30
000634  000000G                         .WORD   EMS.30
000636  000424                          BR      37$                     ;               7123
000640  010746                  36$:    MOV     PC,-(SP)                ; PC,*          7133
```

```
000642  013746  000000G                 MOV     L$LUN,-(SP)
000646  010146                          MOV     R1,-(SP)
000650  012746  000000G                 MOV     #HRD.MSG,-(SP)
000654  012746  000004                  MOV     #4,-(SP)
000660  010600                          MOV     SP,R0                   ; SP,*
000662  104414                          TRAP    14
000664  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                                   7134
000670  012746  000001                  MOV     #1,-(SP)
000674  010600                          MOV     SP,R0                   ; SP,*
000676  104414                          TRAP    14
000700  004737  000000G                 JSR     PC,EMS.ERR              ;                                   7135
000704  062706  000014                  ADD     #14,SP                  ;                                   7131
000710  010200                  37$:    MOV     R2,R0                   ; CUR.PRIORITY,*                    7138
000712  104441                          TRAP    41
000714  000207                          RTS     PC                      ;                                   7010

; Routine Size:  231 words,     Routine Base:  $CODE$ + 26004
; Maximum stack depth per invocation:  11 words


000012                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAB:                                  ; CASE Table for ERR.HRD.RTNE+0076  7039
000012  000000                  3$:     .WORD   0                       ; [4$]
000014  000012                          .WORD   12                      ; [5$]
000016  000452                          .WORD   452                     ; [33$]
000020  000452                          .WORD   452                     ; [33$]
000022  000452                          .WORD   452                     ; [33$]
000024  000024                          .WORD   24                      ; [6$]
000026  000036                          .WORD   36                      ; [7$]
000030  000452                          .WORD   452                     ; [33$]
000032  000050                          .WORD   50                      ; [8$]
000034  000062                          .WORD   62                      ; [9$]
000036  000074                          .WORD   74                      ; [10$]
000040  000106                          .WORD   106                     ; [11$]
000042  000120                          .WORD   120                     ; [12$]
000044  000132                          .WORD   132                     ; [13$]
000046  000144                          .WORD   144                     ; [14$]
000050  000452                          .WORD   452                     ; [33$]
000052  000452                          .WORD   452                     ; [33$]
000054  000452                          .WORD   452                     ; [33$]
000056  000452                          .WORD   452                     ; [33$]
000060  000452                          .WORD   452                     ; [33$]
000062  000452                          .WORD   452                     ; [33$]
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  000156                          .WORD   156                     ; [15$]
```

```
000106  000170        .WORD   170     ; [16$]
000110  000202        .WORD   202     ; [17$]
000112  000214        .WORD   214     ; [18$]
000114  000226        .WORD   226     ; [19$]
000116  000240        .WORD   240     ; [20$]
000120  000252        .WORD   252     ; [21$]
000122  000264        .WORD   264     ; [22$]
000124  000276        .WORD   276     ; [23$]
000126  000310        .WORD   310     ; [24$]
000130  000322        .WORD   322     ; [25$]
000132  000334        .WORD   334     ; [26$]
000134  000346        .WORD   346     ; [28$]
000136  000360        .WORD   360     ; [29$]
```

```
;    7141  1      routine ERR_SOFT_RTNE (ERRNUM) : novalue =
;    7142  1
;    7143  1      !+
;    7144  1      ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;    7145  1      ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    7146  1      !-
;    7147  1
;    7148  2          begin
;    7149  2
;    7150  2          builtin
;    7151  2              PC;
;    7152  2
;    7153  3          if BIT_TST (SWP_FLAGS, SWF SFT)                    ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    7154  2          then
;    7155  2
;    7156  2              case .ERRNUM from 50 to 54 of
;    7157  2                  set
;    7158  2
;    7159  2                  [50]:       ERRSOFT (50, 0, 0);           ! CONTROLLER ERROR
;    7160  2
;    7161  2                  [51]:       ERRSOFT (51, 0, 0);           ! HOST MEMORY ACCESS ERROR
;    7162  2
;    7163  2                  [52]:       ERRSOFT (52, 0, 0);           ! DISK TRANSFER ERROR
;    7164  2
;    7165  2                  [53]:       ERRSOFT (53, 0, 0);           ! SDI ERROR
;    7166  2
;    7167  2                  [54]:       ERRSOFT (54, 0, 0);           ! SMALL DISK ERROR
;    7168  2                  tes
;    7169  2          else
;    7170  3              begin
;    7171  3              !****increment error count                       ! INCREMENT TOTAL ERROR COUNT
;    7172  3              PRINTB (SFT_MSG, .ERRNUM, .L$LUN, .PC);          ! PRINT ERROR LINE JUST LIKE DRS
;    7173  2              end;
;    7174  2
;    7175  1          end;


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

000000  032737  020000  000000G      ERR.SOFT.RTNE:
                                                  BIT     #20000,SWP.FLAGS              ;                  7153
000006  001440                                    BEQ     7$
000010  016600  000002                            MOV     2(SP),R0                      ; ERRNUM,*         7156
000014  162700  000062                            SUB     #62,R0
000020  006300                                    ASL     R0
000022  066007  000140'                           ADD     P.AAC(R0),PC                  ; Case dispatch
000026  104457                       2$:          TRAP    57                            ;                  7159
000030  000062                                    .WORD   62
000032  000000                                    .WORD   0
000034  000000                                    .WORD   0
000036  000207                                    RTS     PC                            ;                  7156
000040  104457                       3$:          TRAP    57                            ;                  7161
```

```
000042  000063                    .WORD   63
000044  000000                    .WORD   0
000046  000000                    .WORD   0
000050  000207                    RTS     PC                    ;                                7156
000052  104457             4$:    TRAP    57                    ;                                7163
000054  000064                    .WORD   64
000056  000000                    .WORD   0
000060  000000                    .WORD   0
000062  000207                    RTS     PC                    ;                                7156
000064  104457             5$:    TRAP    57                    ;                                7165
000066  000065                    .WORD   65
000070  000000                    .WORD   0
000072  000000                    .WORD   0
000074  000207                    RTS     PC                    ;                                7156
000076  104457             6$:    TRAP    57                    ;                                7167
000100  000066                    .WORD   66
000102  000000                    .WORD   0
000104  000000                    .WORD   0
000106  000207                    RTS     PC                    ;                                7153
000110  010746             7$:    MOV     PC,-(SP)              ; PC,*                           7172
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                ;                                7170
000142  000207                    RTS     PC                    ;                                7141

; Routine Size:  50 words,      Routine Base:  $CODE$ + 26722
; Maximum stack depth per invocation:  7 words


000140                            .PSECT  $PLIT$,  RO ,  D

                           P.AAC:                               ; CASE Table for ERR.SOFT.RTNE+0022     7156
000140  000000             1$:    .WORD   0                     ; [2$]
000142  000012                    .WORD   12                    ; [3$]
000144  000024                    .WORD   24                    ; [4$]
000146  000036                    .WORD   36                    ; [5$]
000150  000050                    .WORD   50                    ; [6$]
```

```
;    1    routine ERR_HRD_RTNE_APT (ERRNUM) : novalue =
;    1
;    1
;    1    !+
;    1    ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;    1    ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    1    !-
;    1
;    1
;    2        begin
;    2
;    2
;    2        local
;    2            CUR_PRIORITY;
;    2
;    2
;    2        builtin
;    2            PC;
;    2
;    2        GETPRI (CUR_PRIORITY);
;    2    !ZZZ     SETPRI (PRIO4);                              ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;    2        SETPRI (.BRLEVEL);                               ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW     ZZZ
;    2
;    2
;    2        if .APT_MODE
;    2        then
;    2
;    3            begin
;    3            .MAIL_BOX_TESTNUM = .RP_ADDR [LBN_LO];       ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    3            .MAIL_BOX_SUBTST = .RP_ADDR [DISK];          ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER UNDER APT ONLY
;    2            end;
;    2
;    2
;    2        if (.ERRNUM lequ 34) or                          ! FOR NON BAD BLOCK TYPE ERRORS
;    2           (.ERRNUM gtru 38) or
;    2           (.ERRNUM eql 36) or
;    3           (.ERRNUM eql 37)
;    3
;    2        then
;    2
;    3            if BIT_TST (SWP_FLAGS, SWF_HRD)              ! IF ERRORS TO BE TREATED NORMALLY
;    2            then
;    2
;    2                case .ERRNUM from 31 to 45 of
;    2                    set
;    2
;    2                    [31]:   ERRDF (31, EGH_30, EMS_30);  ! INVALID COMMAND
;    2
;    2                    [32]:   ERRDF (32, EGH_30, EMS_30);  ! COMMAND ABORTED
;    2
;    2                    [33]:   ;                            !
;    2
;    2                    [34]:   ;                            !
```

```
                        [35]:   ;                               ! MEDIA FORMAT ERROR

                        [36]:   ERRDF (36, EGH_30, EMS_30);     ! WRITE PROTECTED

                        [37]:   ERRDF (37, EGH_30, EMS_30);     ! COMPARE ERROR

                        [38]:   ;                               ! DATA ERROR

                        [39]:   ERRDF (39, EGH_30, EMS_30);     ! HOST BUFFER ACCESS ERROR

                        [40]:   ERRDF (40, EGH_30, EMS_30);     ! CONTROLLER ERROR

                        [41]:   ERRDF (41, EGH_30, EMS_30);     ! DRIVE ERROR

                        [42]:   ERRDF (42, EGH_30, 0);          ! HOST WRITE COMPARE ERROR

                        [43]:   ERRDF (43, EGH_30, EMS_30);     ! MESSAGE FROM INTERNAL DIAGNOSTICS

                        [44]:   ERRDF (44, EGH_30, EMS_30);     ! DUPLICATE UNIT NUMBER

                        [45]:   ERRDF (45, EGH_30, EMS_30);     ! INVALID END CODE
                        tes

                else

                    begin
                    !****increment error count               ! INCREMENT TOTAL ERROR COUNT
                    PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);     ! PRINT ERROR MESSAGE JUST LIKE DRS


                    if .ERRNUM neq 42

                    then
                        begin
                        PRINTB (HRD_SUB);                     ! NEXT LINE FOR NON-HOST COMPARE ERRORS
                        EMS_ERR ();                           ! PRINT REST OF THE INFORMATION
                        end;
                    end;

            if (.ERRNUM eql 35) or                            ! FOR BAD-BLOCK TYPE ERRORS
               (.ERRNUM eql 38)

            then

                if BIT_TST (SWP_FLAGS, SWF_BLK)               ! IF ERRORS TO BE TREATED NORMALLY
                then

                    selectoneu .ERRNUM of
                        set

                        [35]:   ERRDF (35, EGH_30, EMS_30);     ! MEDIA FORMAT ERROR

```

```
;    7282  2                   [38]:   ERRDF (38, EGH_30, EMS_30);      ! DATA ERROR
;    7283  2                   tes
;    7284  2
;    7285  2            else
;    7286  2
;    7287  3                begin
;    7288  3                !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;    7289  3                PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);      ! PRINT ERROR LINE JUST LIKE DRS
;    7290  3                PRINTB (HRD_SUB);                           ! PRINT NEXT LINE TOO
;    7291  3                EMS_ERR ();                                 ! PRINT REST OF THE INFORMATION
;    7292  2                end;
;    7293  2
;    7294  2
;    7295  2        SETPRI (.CUR_PRIORITY);                             ! PRIORITY BACK TO NORMAL
;    7296  2
;    7297  2
;    7298  1        end;


                                                .SBTTL  ERR.HRD.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
027066                                          .PSECT  $CODE$,  RO

000000  004137  000000G                 ERR.HRD.RTNE.APT:
                                                JSR     R1,$SAVE2                       ;                       7176
000004  104440                                  TRAP    40                              ;                       7195
000006  010002                                  MOV     R0,R2                           ; *,CUR.PRIORITY
000010  013700  000000G                         MOV     BRLEVEL,R0                      ;                       7197
000014  104441                                  TRAP    41
000016  032737  000001  001254'                 BIT     #1,APT.MODE                     ;                       7200
000024  001412                                  BEQ     1$
000026  013700  000000G                         MOV     RP.ADDR,R0                      ;                       7204
000032  016077  000050  001256'                 MOV     50(R0),@MAIL.BOX.TESTNUM
000040  013700  000000G                         MOV     RP.ADDR,R0                      ;                       7205
000044  016077  000010  001260'                 MOV     10(R0),@MAIL.BOX.SUBTST
000052  016601  000010          1$:             MOV     10(SP),R1                       ; ERRNUM,*              7209
000056  020127  000042                          CMP     R1,#42
000062  101411                                  BLOS    2$
000064  020127  000046                          CMP     R1,#46                          ;                       7210
000070  101006                                  BHI     2$
000072  020127  000044                          CMP     R1,#44                          ;                       7211
000076  001403                                  BEQ     2$
000100  020127  000045                          CMP     R1,#45                          ;                       7212
000104  001131                                  BNE     17$
000106  032737  010000  000000G 2$:             BIT     #10000,SWP.FLAGS                ;                       7216
000114  001475                                  BEQ     15$
000116  010100                                  MOV     R1,R0                           ;                       7219
000120  162700  000037                          SUB     #37,R0
000124  006300                                  ASL     R0
000126  066007  000152'                         ADD     P.AAD(R0),PC                    ; Case dispatch
000132  104455                  4$:             TRAP    55                              ;                       7222
000134  000037                                  .WORD   37
000136  000000G                                 .WORD   EGH.30
000140  000000G                                 .WORD   EMS.30
```

```
000142  000512                       BR      17$                         ;          7219
000144  104455               5$:     TRAP    55                          ;          7224
000146  000040                       .WORD   40
000150  000000G                      .WORD   EGH.30
000152  000000G                      .WORD   EMS.30
000154  000505                       BR      17$                         ;          7219
000156  104455               6$:     TRAP    55                          ;          7232
000160  000044                       .WORD   44
000162  000000G                      .WORD   EGH.30
000164  000000G                      .WORD   EMS.30
000166  000500                       BR      17$                         ;          7219
000170  104455               7$:     TRAP    55                          ;          7234
000172  000045                       .WORD   45
000174  000000G                      .WORD   EGH.30
000176  000000G                      .WORD   EMS.30
000200  000473                       BR      17$                         ;          7219
000202  104455               8$:     TRAP    55                          ;          7238
000204  000047                       .WORD   47
000206  000000G                      .WORD   EGH.30
000210  000000G                      .WORD   EMS.30
000212  000466                       BR      17$                         ;          7219
000214  104455               9$:     TRAP    55                          ;          7240
000216  000050                       .WORD   50
000220  000000G                      .WORD   EGH.30
000222  000000G                      .WORD   EMS.30
000224  000461                       BR      17$                         ;          7219
000226  104455               10$:    TRAP    55                          ;          7242
000230  000051                       .WORD   51
000232  000000G                      .WORD   EGH.30
000234  000000G                      .WORD   EMS.30
000236  000454                       BR      17$                         ;          7219
000240  104455               11$:    TRAP    55                          ;          7244
000242  000052                       .WORD   52
000244  000000G                      .WORD   EGH.30
000246  000000                       .WORD   0
000250  000447                       BR      17$                         ;          7219
000252  104455               12$:    TRAP    55                          ;          7246
000254  000053                       .WORD   53
000256  000000G                      .WORD   EGH.30
000260  000000G                      .WORD   EMS.30
000262  000442                       BR      17$                         ;          7219
000264  104455               13$:    TRAP    55                          ;          7248
000266  000054                       .WORD   54
000270  000000G                      .WORD   EGH.30
000272  000000G                      .WORD   EMS.30
000274  000435                       BR      17$                         ;          7219
000276  104455               14$:    TRAP    55                          ;          7250
000300  000055                       .WORD   55
000302  000000G                      .WORD   EGH.30
000304  000000G                      .WORD   EMS.30
000306  000430                       BR      17$                         ;          7216
000310  010746               15$:    MOV     PC,-(SP)                    ; PC.*     7257
000312  013746  000000G              MOV     L$LUN,-(SP)
```

```
000316  010146                          MOV     R1,-(SP)
000320  012746  000000G                 MOV     #DF.MSG,-(SP)
000324  012746  000004                  MOV     #4,-(SP)
000330  010600                          MOV     SP,R0                   ; SP,*
000332  104414                          TRAP    14
000334  020127  000052                  CMP     R1,#52                  ;                       7260
000340  001411                          BEQ     16$
000342  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                       7264
000346  012746  000001                  MOV     #1,-(SP)
000352  010600                          MOV     SP,R0                   ; SP,*
000354  104414                          TRAP    14
000356  004737  000000G                 JSR     PC,EMS.ERR              ;                       7265
000362  005726                          TST     (SP)+                   ;                       7263
000364  062706  000012          16$:    ADD     #12,SP                  ;                       7255
000370  020127  000043          17$:    CMP     R1,#43                  ;                       7269
000374  001403                          BEQ     18$
000376  020127  000046                  CMP     R1,#46                  ;                       7270
000402  001050                          BNE     21$
000404  032737  C40000  000000G 18$:    BIT     #40000,SWP.FLAGS        ;                       7274
000412  001420                          BEQ     20$
000414  020127  000043                  CMP     R1,#43                  ;                       7280
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$                     ;                       7277
000434  020127  000046          19$:    CMP     R1,#46                  ;                       7282
000440  001031                          BNE     21$
000442  104455                          TRAP    55
000444  000046                          .WORD   46
000446  000000G                         .WORD   EGH.30
000450  000000G                         .WORD   EMS.30
000452  000424                          BR      21$                     ;                       7277
000454  010746                  20$:    MOV     PC,-(SP)                ; PC,*                  7289
000456  013746  000000G                 MOV     L$LUN,-(SP)
000462  010146                          MOV     R1,-(SP)
000464  012746  000000G                 MOV     #DF.MSG,-(SP)
000470  012746  000004                  MOV     #4,-(SP)
000474  010600                          MOV     SP,R0                   ; SP,*
000476  104414                          TRAP    14
000500  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                       7290
000504  012746  000001                  MOV     #1,-(SP)
000510  010600                          MOV     SP,R0                   ; SP,*
000512  104414                          TRAP    14
000514  004737  000000G                 JSR     PC,EMS.ERR              ;                       7291
000520  062706  000014                  ADD     #14,SP                  ;                       7287
000524  010200                  21$:    MOV     R2,R0                   ; CUR.PRIORITY,*        7295
000526  104441                          TRAP    41
000530  000207                          RTS     PC                      ;                       7176

; Routine Size:  173 words,     Routine Base:  $CODE$ + 27066
; Maximum stack depth per invocation:  11 words
```

```
000152                          .PSECT  $PLIT$,  RO ,  D

                        P.AAD:                                  ; CASE Table for ERR.HRD.RTNE AP+0126    7219
000152  000000          3$:     .WORD   0                       ; [4$]
000154  000012                  .WORD   12                      ; [5$]
000156  000236                  .WORD   236                     ; [17$]
000160  000236                  .WORD   236                     ; [17$]
000162  000236                  .WORD   236                     ; [17$]
000164  000024                  .WORD   24                      ; [6$]
000166  000036                  .WORD   36                      ; [7$]
000170  000236                  .WORD   236                     ; [17$]
000172  000050                  .WORD   50                      ; [8$]
000174  000062                  .WORD   62                      ; [9$]
000176  000074                  .WORD   74                      ; [10$]
000200  000106                  .WORD   106                     ; [11$]
000202  000120                  .WORD   120                     ; [12$]
000204  000132                  .WORD   132                     ; [13$]
000206  000144                  .WORD   144                     ; [14$]


;    7299  1
;    7300  1
```

```
;  1      routine ERR_SOFT_RTNE APT (ERRNUM, index) : novalue =
;  1
;  1      !+
;  1      ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;  1      ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;  1      !-
;  1
;  2          begin
;  2
;  2          local
;  2              ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;  2
;  2          builtin
;  2              PC;
;  2
;  2          ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);        ! ADDRESS OF THE SAVED ERROR LOG INFORMATION
;  2
;  2          if .APT_MODE
;  2          then
;  3              begin
;  3              .MAIL_BOX_TESTNUM = .ELOG_ADDR [EL_BLOCK];       ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;  3              .MAIL_BOX_SUBTST = .ELOG_ADDR [EL_DK_NUM];       ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER IN APT ONLY
;  2              end;
;  2
;  3          if BIT_TST (SWP_FLAGS, SWF SFT)                       ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;  2          then
;  2
;  2              case .ERRNUM from 50 to 54 of
;  2                  set
;  2
;  2                  [50]:       ERRDF (50, 0, 0);                 ! CONTROLLER ERROR
;  2
;  2                  [51]:       ERRDF (51, 0, 0);                 ! HOST MEMORY ACCESS ERROR
;  2
;  2                  [52]:       ERRDF (52, 0, 0);                 ! DISK TRANSFER ERROR
;  2
;  2                  [53]:       ERRDF (53, 0, 0);                 ! SDI ERROR
;  2
;  2                  [54]:       ERRDF (54, 0, 0);                 ! SMALL DISK ERROR
;  2                  tes
;  2          else
;  3              begin
;  3              !****increment error count                        ! INCREMENT TOTAL ERROR COUNT
;  3              PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);            ! PRINT ERROR LINE JUST LIKE DRS
;  2              end;
;  2
;  1          end;

                                    .SBTTL  ERR.SOFT.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
027620                              .PSECT  $CODE$,  RO

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

```
                                         MOV     2(SP),-(SP)                  ; INDEX,*              7316
000004  012746  000102                   MOV     #102,-(SP)
000010  004737  000000G                  JSR     PC,BL$MUL
000014  062700  000000G                  ADD     #ELOG.PKT,R0
000020  032737  000001  001254'          BIT     #1,APT.MODE                  ;                      7318
000026  001406                           BEQ     1$
000030  016077  000056  001256'          MOV     56(R0),@MAIL.BOX.TESTNUM     ; *(ELOG.ADDR),*       7321
000036  016077  000012  001260'          MOV     12(R0),@MAIL.BOX.SUBTST      ; *(ELOG.ADDR),*       7322
000044  032737  020000  000000G  1$:     BIT     #20000,SWP.FLAGS             ;                      7325
000052  001440                           BEQ     8$
000054  016600  000010                   MOV     10(SP),R0                    ; ERRNUM,*             7328
000060  162700  000062                   SUB     #62,R0
000064  006300                           ASL     R0
000066  066007  000210'                  ADD     P.AAE(R0),PC                 ; Case dispatch
000072  104455                   3$:     TRAP    55                           ;                      7331
000074  000062                           .WORD   62
000076  000000                           .WORD   0
000100  000000                           .WORD   0
000102  000441                           BR      9$                           ;                      7328
000104  104455                   4$:     TRAP    55                           ;                      7333
000106  000063                           .WORD   63
000110  000000                           .WORD   0
000112  000000                           .WORD   0
000114  000434                           BR      9$                           ;                      7328
000116  104455                   5$:     TRAP    55                           ;                      7335
000120  000064                           .WORD   64
000122  000000                           .WORD   0
000124  000000                           .WORD   0
000126  000427                           BR      9$                           ;                      7328
000130  104455                   6$:     TRAP    55                           ;                      7337
000132  000065                           .WORD   65
000134  000000                           .WORD   0
000136  000000                           .WORD   0
000140  000422                           BR      9$                           ;                      7328
000142  104455                   7$:     TRAP    55                           ;                      7339
000144  000066                           .WORD   66
000146  000000                           .WORD   0
000150  000000                           .WORD   0
000152  000415                           BR      9$                           ;                      7325
000154  010716                   8$:     MOV     PC,(SP)                      ; PC,*                 7344
000156  013746  000000G                  MOV     L$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                       ;                      7342
000206  022626                   9$:     CMP     (SP)+,(SP)+                  ;                      7308
000210  000207                           RTS     PC                           ;                      7301

; Routine Size:  69 words,      Routine Base:  $CODE$ + 27620
; Maximum stack depth per invocation:  8 words
```

```
000210                                  .PSECT  $PLIT$,  RO ,  D

                              P.AAE:                                  ; CASE Table for ERR.SOFT.RTNE.A+0066    7328
000210  000000                2$:       .WORD   0                     ; [3$]
000212  000012                          .WORD   12                    ; [4$]
000214  000024                          .WORD   24                    ; [5$]
000216  000036                          .WORD   36                    ; [6$]
000220  000050                          .WORD   50                    ; [7$]


;    7348  1
;    7349  1
;    7350  1      end
;    7351  1
;    7352  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
;        $GGG$                  353        RO ,  I  ,  LCL,  REL,  CON
;        $CODE$                6157        RO ,  I  ,  LCL,  REL,  CON
;        $PLIT$                  73        RO ,  D  ,  LCL,  REL,  CON


;                        Library Statistics
;
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total     Loaded   Percent     Mapped      Time
;
;  DISK$USER2:[MLYNAR.ZRQ]ZRQAH0.L16;10    411       312        75         21         00:00.1


;                             COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAH0.BL2/LIST=ZRQAH0.LS2/OBJECT=ZRQAH0.OB2/SOURCE=PAGE:53
```

```
;        0      module ZRQAM4 (
;        0
;        0      %title 'RD/RX EXERCISER'
;        0                      ident = 'V01.9',
;        0                      addressing_mode (absolute),
;        0                      environment (noeis)
;        0                      ) =
;        0
;        1      begin
;        1
;        1      %sbttl 'LASTAD AND SETUP'
;        1
;        1      library 'ZRQAHO.L16';
;        1
;        1      !MMM require 'BLSMAC.REQ';                      ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;        1      require 'HSAXAO.BLB';                           ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;        1
;        2      LASTAD
;        2
;        2      BGNSETUP (4)                                                        !ZZZ
;        2
; P      2              BGNPTAB
; P      2                      INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000020', 0, 0, RD52_MAX_LBN, 0  !ZZZ
; P      2                              ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
; P      2
;        2              ENDPTAB
;        2
; P      2              BGNPTAB
; P      2                      INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000001', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P      2                              ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;        2              ENDPTAB
;        2
; P      2              BGNPTAB
; P      2                      INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000002', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P      2                              ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;        2              ENDPTAB
; P      2              BGNPTAB
; P      2                      INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000003', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P      2                              !HERE'S ONE FOR THE 4TH DRIVE                                           !ZZZ
;        2              ENDPTAB                                                                                 !ZZZ
;        2
;        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.AAA
        000010'             $REM5=             P.AAB
        000030'             $LAS4=             P.AAC
        000034'             $REM4=             P.AAD
        000054'             $LAS3=             P.AAE
        000060'             $REM3=             P.AAF
        000100'             $$LAS1=            P.AAG
        000104'             $REM2=             P.AAH
```

```
                                      .SBTTL  $END.LINK LASTAD AND SETUP
000000  000207                   $END.LINK::
                                        RTS     PC                          ;                              9108

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


;   9135  1     end
;   9136  1
;   9137  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:[MLYNAR.ZRQ]ZRQAH0.L16;10      411          7         1         21         00:00.1




;                                COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAH0.BL2/LIST=ZRQAH0.LS2/OBJECT=ZRQAH0.OB2/SOURCE=PAGE:53

; Size:          6158 code + 469 data words
; Run Time:         02:19.8
; Elapsed Time:     09:52.7
; Lines/CPU Min:     3922
; Lexemes/CPU-Min: 29675
; Memory Used:  501 pages
; Compilation Complete
```

```
Partition name : DUMMY
Identification : V02.3
Task  UIC      : [202,12]
Task attributes: -HD
Total address windows: 1.
Task  image  size  : 16704. words
Task address limits: 002000 103143
R-W disk blk limits: 000002 000103 000102 00066.

*** Root segment: ZRQAHO

R/W mem  limits: 002000 103143 101144 33380.
Disk blk limits: 000002 000103 000102 00066.

Memory allocation synopsis:

Section                                      Title  Ident  File
-------                                      -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON)    002000 000000 00000.
$CODE$:(RO,I,LCL,REL,CON)    002000 071746 29670.
                             002000 024110 10312. ZRQAM1 V02.3  ZRQAHO.OB1;3
                             026110 015160 06768. ZRQAM2 V02.3  ZRQAHO.OB1;3
                             043270 030032 12314. ZRQAM3 V02.3  ZRQAHO.OB2;3
                             073322 000316 00206. B16MUL 2.8    NOEIS.OLB;1
                             073640 000106 00070. B16SAV 2.4    NOEIS.OLB;1
$FFF$ :(RW,D,GBL,REL,CON)    073746 005056 02606.
                             073746 005056 02606. ZRQAM1 V02.3  ZRQAHO.OB1;3
$GGG$ :(RO,I,LCL,REL,CON)    101024 001302 00706.
                             101024 001302 00706. ZRQAM3 V02.3  ZRQAHO.OB2;3
$OWN$ :(RW,D,LCL,REL,CON)    102326 000244 00164.
                             102326 000244 00164. ZRQAM2 V02.3  ZRQAHO.OB1;3
$PLIT$:(RO,D,LCL,REL,CON)    102572 000222 00146.
                             102572 000222 00146. ZRQAM3 V02.3  ZRQAHO.OB2;3
$XYZ$ :(RO,I,LCL,REL,CON)    103014 000130 00088.
                             103014 000130 00088. ZRQAM4 V01.9  ZRQAHO.OB2;3

Global symbols:

ADDR.V 101023-R  BIT06  000100    BIT4   000020    BST    074126-R  CRLF   025742-R  DBM101 006226-R  DBM126 007260-R
ADR    000020    BIT07  000200    BIT5   000040    BUFF.A 100604-R  CRN.HI 100716-R  DBM104 006254-R  DBM127 007340-R
ASTERI 025754-R  BIT08  000400    BIT6   000100    BUFF.O 100624-R  CRN.LO 100714-R  DBM105 006316-R  DBM128 007416-R
AZINT  065106-R  BIT09  001000    BIT7   000200    BYTS.P 100674-R  CSR.AD 101004-R  DBM107 006354-R  DBM15  005204-R
AZINTO 065070-R  BIT1   000002    BIT8   000400    CCTLR  100654-R  CSR.ME 101002-R  DBM108 006412-R  DBM18  005234-R
BAL.IN 100762-R  BIT10  002000    BIT9   001000    CDISK  100656-R  CST    073746-R  DBM109 006502-R  DBM19  005306-R
BIT0   000001    BIT11  004000    BL$DIV 073546-R  CER.01 017342-R  CST.AD 074074-R  DBM111 006562-R  DBM20  005372-R
BIT00  000001    BIT12  010000    BL$LAS 103014-R  CER.02 017406-R  CTLR.C 100662-R  DBM112 006662-R  DBM21  005446-R
BIT01  000002    BIT13  020000    BL$MOD 073560-R  CLK.PR 100740-R  CTLR.I 045266-R  DBM12  005130-R  DBM22  005530-R
BIT02  000004    BIT14  040000    BL$MUL 073322-R  CLK.TI 100732-R  CUOFF  100660-R  DBM120 006764-R  DBM23  005574-R
BIT03  000010    BIT15  100000    BL$SHF 073572-R  CMD.TI 100710-R  C.ERR. 074546-R  DBM121 007056-R  DBM25  005632-R
BIT04  000020    BIT2   000004    BOE    000400    CNTR.E 024652-R  DASH   025746-R  DBM123 007146-R  DBM26  005700-R
BIT05  000040    BIT3   000010    BRLEVE 101012-R  CREDIT 100724-R  DATAGM 067524-R  DBM125 007210-R  DBM27  005732-R
```

DBM28A	006004-R	EH.10	013770-R	EX.ONL	015520-R	GP$5	026220-R	L$AUT	002070-R	L$SWLE	026012-R	QIO.SI	060240-R
DBM28B	006044-R	EH.12	014020-R	EX.OP	015544-R	GP$6	026230-R	L$AUTO	032242-R	L$TEST	002114-R	QIO.UN	055512-R
DBM29	006104-R	EH.13	014054-R	EX.PBN	016010-R	GP$7	026242-R	L$CCP	002106-R	L$TIML	002014-R	RANDOM	074506-R
DBM32	006152-R	EH.2	013452-R	EX.RBN	016166-R	GP$8	026254-R	L$CLEA	032542-R	L$UNIT	002012-R	RANDY	055234-R
DBM5	005102-R	EH.3	013512-R	EX.RD	015464-R	GP$9	026270-R	L$CO	002032-R	MAN.TS	026074-R	RDM.CN	074504-R
DCT	074076-R	EH.4	013550-R	EX.RP	016730-R	HARD.E	061456-R	L$DEPO	002011-R	MD.INI	053700-R	RDRX.A	074122-R
DCT.AD	074120-R	EH.5	013574-R	EX.SA	015350-R	HARD.I	046744-R	L$DESC	026130-R	MINUTE	100730-R	RDRX.E	025372-R
DFPTBL	025764-R	EH.6	013624-R	EX.SB	015422-R	HOE	100000	L$DESP	002076-R	MODULA	035632-R	RD.COU	101010-R
DF.MSG	025410-R	EH.7	013654-R	EX.SB0	015416-R	HOE.FL	100741-R	L$DEVP	002060-R	MSCP.P	074550-R	REG.EX	045714-R
DISK.R	066102-R	EH.8	013704-R	EX.SC	015366-R	HOST.W	064150-R	L$DISP	002124-R	MSG.01	010252-R	RETPKT	076276-R
DRIVER	044734-R	EH.9	013740-R	EX.TIM	017026-R	HOURS	100727-R	L$DLY	002116-R	MSG.02	010304-R	ROUND.	063604-R
DROP.C	034772-R	ELG.FM	015334-R	EX.WRD	017016-R	HRD.MS	025506-R	L$DTP	002040-R	MSG.03	010340-R	RPS.RE	064566-R
DRV.CT	035100-R	ELG.00	014720-R	EX.WRT	015474-R	HRD.SU	025704-R	L$DTYP	002034-R	MULTI.	053234-R	RPT1	010372-R
DR.ERR	052462-R	ELOG.P	077052-R	FATAL.	065306-R	HWPTE0	026000-R	L$DU	033146-R	NAME.H	026006-R	RPT10	011102-R
DR.RET	064754-R	EMS.BL	037504-R	FER.BC	101020-R	HWPTE1	026002-R	L$DUT	002072-R	NAME.L	026004-R	RPT11	011170-R
DUPROU	026072-R	EMS.CM	042052-R	FER.LB	101016-R	HWPTS0	025774-R	L$DVTY	026110-R	NEX	100712-R	RPT12	011236-R
DUP.FL	100652-R	EMS.EL	040430-R	FER0.L	100734-R	HWPTS1	025776-R	L$EF	002052-R	NEXT.P	100726-R	RPT13	011304-R
DUR	100664-R	EMS.ER	042440-R	FER1.L	100736-R	HWPT.B	025770-R	L$ENVI	002044-R	NEX.TR	033250-R	RPT14	011404-R
DU.MSG	007504-R	EMS.R1	040364-R	FILL.B	060450-R	HWPT.D	025772-R	L$ERRT	002126-R	NULL	005100-R	RPT15	011502-R
DU.RSN	010224-R	EMS.R2	040234-R	FORCED	101015-R	HWPT.I	025764-R	L$ETP	002102-R	OFF	000002	RPT16	011602-R
D$PCNT	002122-R	EMS.01	042650-R	FORCE.	100772-R	HWPT.V	025766-R	L$EXP1	002046-R	OF.RC	100704-R	RPT2	010456-R
D.FAIL	101014-R	EMS.10	042706-R	FREE.M	100672-R	HWQ1	002460-R	L$EXP4	002064-R	ON	000001	RPT3	010522-R
EBD.10	012766-R	EMS.12	042750-R	FSET.U	061332-R	HWQ2	002474-R	L$EXP5	002066-R	OUT.IO	034646-R	RPT4	010606-R
EBD.12	013026-R	EMS.13	043006-R	GET.IO	034466-R	HWQ3	002504-R	L$HARD	026154-R	OVF.CH	063530-R	RPT5	010652-R
EBD.13	013074-R	EMS.14	043050-R	GET.PK	033620-R	HWQ4	002546-R	L$HIME	002120-R	PARITY	033260-R	RPT6	010740-R
EBD.14	013126-R	EMS.18	043112-R	GET.RA	055124-R	HWQ5	002564-R	L$HPCP	002016-R	PKT.US	076262-R	RPT7	011004-R
EBD.18	013166-R	EMS.21	043160-R	GET.RE	034344-R	HWQ6A	002634-R	L$HPTP	002022-R	PNT	001000	RPT8	011022-R
EBD.19	013222-R	EMS.24	043204-R	GP$DIS	026476-R	HWQ6B	002706-R	L$HRDL	026152-R	POLL.C	065540-R	RPT9	011050-R
EBD.24	013302-R	EMS.30	043252-R	GP$1	026154-R	HWQ7A	002762-R	L$HW	025764-R	POLL.R	065640-R	RP.ADD	077050-R
EBS.01	012724-R	ENTRY.	100650-R	GP$10	026304-R	HWQ7B	003032-R	L$HWLE	025762-R	PRI	002000	RP.IND	077046-R
EF.CON	000036	EOP.FL	100651-R	GP$11	026314-R	HWQ8	003102-R	L$ICP	002104-R	PRI00	000000	RP.USE	077036-R
EF.NEW	000035	ERRBLK	002134-R	GP$12	026330-R	HWQ9	003160-R	L$INIT	032230-R	PRI01	000040	SA.REG	100706-R
EF.PWR	000034	ERRMSG	002132-R	GP$13	026342-R	IBE	010000	L$LADP	002026-R	PRI02	000100	SB.COD	100700-R
EF.RES	000037	ERRNBR	002130-R	GP$14	026354-R	IDU	000040	L$LAST	103020-R	PRI03	000140	SC.CLK	022610-R
EF.STA	000040	ERRTYP	002126-R	GP$15	026366-R	IER	020000	L$LOAD	002100-R	PRI04	000200	SC.CON	017506-R
EGD.10	011776-R	ERR.CO	014662-R	GP$16	026400-R	INIT.I	054040-R	L$LUN	002074-R	PRI05	000240	SC.CTO	022052-R
EGD.11	012036-R	ERR.00	014126-R	GP$17	026406-R	INIT.O	101022-R	L$MREV	002050-R	PRI06	000300	SC.DIS	020104-R
EGD.12	012062-R	EVL	000004	GP$18	026414-R	INIT.T	044576-R	L$NAME	002000-R	PRI07	000340	SC.DST	020514-R
EGD.13	012110-R	EX.ACC	015532-R	GP$19	026422-R	INI.CT	045570-R	L$NDHR	026324-R	PROC.R	060702-R	SC.DS2	020570-R
EGD.14	012136-R	EX.BBU	015670-R	GP$2	026164-R	INI.RR	047614-R	L$NDHW	026010-R	PTCH1	002136-R	SC.DUP	017530-R
EGD.15	012166-R	EX.BB1	015616-R	GP$20	026430-R	INT.GE	046454-R	L$NDSF	026624-R	PTCH2	002210-R	SC.ECC	020642-R
EGD.16	012204-R	EX.BB2	015550-R	GP$21	026436-R	IN.IOD	034704-R	L$NDSW	026100-R	PTCH3	002262-R	SC.ECD	020724-R
EGD.17	012234-R	EX.BC	016416-R	GP$22	026452-R	IODQ	100634-R	L$PRIO	002042-R	PTCH4	002334-R	SC.EC1	021174-R
EGD.18	012252-R	EX.BD	016472-R	GP$23	026462-R	IODQ.I	100644-R	L$PROT	026102-R	PTCH5	002406-R	SC.EC2	021224-R
EGD.19	012272-R	EX.BDR	016550-R	GP$24	026470-R	IODQ.O	100646-R	L$PRT	002112-R	PUTA.B	034606-R	SC.EC3	021254-R
EGD.20	012332-R	EX.BDW	016640-R	GP$25	026502-R	IO.RET	061062-R	L$REPP	002062-R	PUT.IO	034542-R	SC.EC4	021306-R
EGD.21	012420-R	EX.CBC	016226-R	GP$26	026516-R	IPKT.A	076260-R	L$REV	002010-R	PUT.PK	034204-R	SC.EC5	021336-R
EGD.22	012532-R	EX.CBR	016272-R	GP$27	026526-R	IRDRX.	074124-R	L$RPT	027462-R	PUT.RE	034452-R	SC.EC6	021366-R
EGD.23	012572-R	EX.CBW	016344-R	GP$28	026542-R	ISR	000100	L$SFTL	026326-R	P.INDE	101006-R	SC.EC7	021416-R
EGD.24	012634-R	EX.CMD	015444-R	GP$29	026560-R	IXE	004000	L$SOFT	026330-R	QIO	100670-R	SC.EC8	021450-R
EGH.30	012700-R	EX.CMP	015504-R	GP$3	026174-R	LOE	040000	L$SPC	002056-R	QIO.FU	057116-R	SC.EC9	021502-R
EGS.01	011664-R	EX.LB	017064-R	GP$30	026572-R	LOT	000010	L$SPCP	002020-R	QIO.GE	054524-R	SC.EDC	022202-R
EGS.02	011704-R	EX.LBN	015750-R	GP$31	026610-R	L$ACP	002110-R	L$SPTP	002024-R	QIO.LB	057436-R	SC.FCT	021126-R
EH.0	013356-R	EX.LBR	016050-R	GP$32	026616-R	L$APT	002036-R	L$STA	002030-R	QIO.OK	054244-R	SC.FER	020176-R
EH.1	013414-R	EX.LBW	016116-R	GP$4	026206-R	L$AU	033240-R	L$SW	026014-R	QIO.OU	054452-R	SC.FE2	020264 R

```
SC.FUL 020776-R  'C.PSP 022734-R  SC.VOL 017752-R  SWP.DP 026022-R  SWQ17  004126-R  TEMP1  100720-R  XX13   017130-R
SC.HWP 021604-R  SC.RCT 020756-R  SC.576 021052-R  SWP.ER 026014-R  SWQ19  004224-R  TEMP2  100722-R  XX23   017152-R
SC.IDS 022242-R  SC.RDY 022526-R  SEND   035156-R  SWP.FL 026020-R  SWQ2   003302-R  TIME   033342-R  XX32   017206-R
SC.INR 017622-R  SC.REF 023130-R  SEQUEN 066202-R  SWP.RA 026024-R  SWQ20  004314-R  TRK.SG 074500-R  XX33   017234-R
SC.INV 017654-R  SC.REP 023016-R  SET.CP 033424-R  SWP.TI 026026-R  SWQ21  004402-R  TST.PA 026076-R  XX34   017272-R
SC.IOP 020032-R  SC.RSP 022636-R  SET.CT 047716-R  SWP.UC 026030-R  SWQ22  004466-R  TYPER  100742-R  $END.L 103142-R
SC.ISH 020354-R  SC.SAF 021644-R  SE .UP 033520-R  SWP.UD 026032-R  SWQ23  004526-R  TYPEW  100752-R  $SAVE2 073640-R
SC.IS2 020434-R  SC.SDI 017462-R  SFPTBL 026014-R  SWP.XF 026016-R  SWQ24  004600-R  T$FREE 103140-R  $SAVE3 073654-R
SC.NBB 023060-R  SC.SDS 022124-R  SFT.MS 025604-R  SWQ1   003260-R  SWQ26  004644-R  T$PTHV 000004    $SAVE4 073672-R
SC.NXM 021762-R  SC.SON 017602-R  SOFT.E 070346-R  SWQ10  003532-R  SWQ27  005004-R  T.ADDR 074476-R  $SAVE5 073712-R
SC.ODA 021710-R  SC.SRI 022404-R  SPACE4 025736-R  SWQ11  003576-R  SWQ28  005030-R  T1     044562-R
SC.ODB 021740-R  SC.SRT 022312-R  STEP   100702-R  SWQ12  003630-R  SWQ4   003362-R  UAM    000200
SC.ONL 017560-R  SC.SUR 022704-R  ST.COD 100676-R  SWQ13  003726-R  SWQ7   003404-R  UPD.IO 062662-R
SC.PAR 022016-R  SC.SWP 021544-R  SWEEP  064434-R  SWQ14  004004-R  SWQ9   003456-R  VEC.BR 046126-R
SC.POE 022472-R  SC.UNK 017672-R  SWM1   004706-R  SWQ15  004056-R  TALLY  074146-R  WAIT   035614-R
```

```
*** Task builder statistics:

    Total work file references: 146628.
    Work  file  reads: 0.
    Work  file writes: 0.
    Size of core pool: 23454. words (91. pages)
    Size of work file: 5120. words (20. pages)

    Elapsed time:00:00:46
```

```
SYMBOL  VALUE       REFERENCES...

ADDR.V  101023-R  # ZRQAM1     ZRQAM2     ZRQAM3
ADR     000020    # ZRQAM1   # ZRQAM2   # ZRQAM3
ASTERI  025754-R  # ZRQAM1     ZRQAM2
AZINT   065106-R  # ZRQAM3
AZINTO  065070-R  # ZRQAM3
BAL.IN  100762-R  # ZRQAM1     ZRQAM2     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$DIV  073546-R  # B16MUL     ZRQAM2     ZRQAM3
BL$LAS  103014-R  # ZRQAM4
BL$MOD  073560-R  # B16MUL     ZRQAM2     ZRQAM3
BL$MUL  073322-R  # B16MUL     ZRQAM2     ZRQAM3
BL$SHF  073572-R  # B16MUL     ZRQAM3
BOE     000400    # ZRQAM1   # ZRQAM2   # ZRQAM3
BRLEVE  101012-R  # ZRQAM1     ZRQAM2     ZRQAM3
BST     074126-R  # ZRQAM1     ZRQAM2     ZRQAM3
BUFF.A  100604-R  # ZRQAM1     ZRQAM2     ZRQAM3
BUFF.O  100624-R  # ZRQAM1     ZRQAM2     ZRQAM3
BYTS.P  100674-R  # ZRQAM1     ZRQAM2     ZRQAM3
CCTLR   100654-R  # ZRQAM1     ZRQAM2     ZRQAM3
CDISK   100656-R  # ZRQAM1     ZRQAM2     ZRQAM3
CER.01  017342-R  # ZRQAM1     ZRQAM2
CER.02  017406-R  # ZRQAM1     ZRQAM2
CLK.PR  100740-R  # ZRQAM1     ZRQAM2     ZRQAM3
CLK.TI  100732-R  # ZRQAM1     ZRQAM2     ZRQAM3
CMD.TI  100710-R  # ZRQAM1     ZRQAM2     ZRQAM3
CNTR.E  024652-R  # ZRQAM1     ZRQAM2
CREDIT  100724-R  # ZRQAM1     ZRQAM2     ZRQAM3
```

```
ZRQAHO     CREATED BY  TKB    ON 10-OCT-85 AT 09:56     PAGE 2
GLOBAL CROSS REFERENCE                                  CREF    V02
SYMBOL  VALUE      REFERENCES...

CRLF    025742-R  # ZRQAM1    ZRQAM2    ZRQAM3
CRN.HI  100716-R  # ZRQAM1    ZRQAM2    ZRQAM3
CRN.LO  100714-R  # ZRQAM1    ZRQAM2    ZRQAM3
CSR.AD  101004-R  # ZRQAM1    ZRQAM2    ZRQAM3
CSR.ME  101002-R  # ZRQAM1    ZRQAM2    ZRQAM3
CST     073746-R  # ZRQAM1    ZRQAM2    ZRQAM3
CST.AD  074074-R  # ZRQAM1    ZRQAM2    ZRQAM3
CTLR.C  100662-R  # ZRQAM1    ZRQAM2    ZRQAM3
CTLR.I  045266-R  # ZRQAM3
CUOFF   100660-R  # ZRQAM1    ZRQAM2    ZRQAM3
C.ERR.  074546-R  # ZRQAM1    ZRQAM2    ZRQAM3
DASH    025746-R  # ZRQAM1    ZRQAM2
DATAGM  067524-R  # ZRQAM3
DBM101  006226-R  # ZRQAM1
DBM104  006254-R  # ZRQAM1
DBM105  006316-R  # ZRQAM1
DBM107  006354-R  # ZRQAM1    ZRQAM2
DBM108  006412-R  # ZRQAM1    ZRQAM3
DBM109  006502-R  # ZRQAM1    ZRQAM3
DBM111  006562-R  # ZRQAM1    ZRQAM3
DBM112  006662-R  # ZRQAM1    ZRQAM3
DBM12   005130-R  # ZRQAM1    ZRQAM3
DBM120  006764-R  # ZRQAM1    ZRQAM3
DBM121  007056-R  # ZRQAM1    ZRQAM3
DBM123  007146-R  # ZRQAM1    ZRQAM3
DBM125  [illegible]-R  # ZRQAM1    ZRQAM2    ZRQAM3
DBM126  007260-R  # ZRQAM1    ZRQAM2    ZRQAM3
DBM127  007340-R  # ZRQAM1    ZRQAM3
DBM128  007416-R  # ZRQAM1    ZRQAM3
DBM15   005204-R  # ZRQAM1
DBM18   005234-R  # ZRQAM1    ZRQAM3
DBM19   005306-R  # ZRQAM1    ZRQAM3
DBM20   005372-R  # ZRQAM1    ZRQAM3
DBM21   005446-R  # ZRQAM1    ZRQAM3
DBM22   005530-R  # ZRQAM1    ZRQAM3
DBM23   005574-R  # ZRQAM1    ZRQAM3
DBM25   005632-R  # ZRQAM1    ZRQAM3
DBM26   005700-R  # ZRQAM1    ZRQAM3
DBM27   005732-R  # ZRQAM1    ZRQAM3
DBM28A  006004-R  # ZRQAM1
DBM28B  006044-R  # ZRQAM1
DBM29   006104-R  # ZRQAM1    ZRQAM3
DBM32   006152-R  # ZRQAM1
DBM5    005102-R  # ZRQAM1    ZRQAM2
DCT     074076-R  # ZRQAM1    ZRQAM2    ZRQAM3
DCT.AD  074120-R  # ZRQAM1    ZRQAM2    ZRQAM3
DFPTBL  025764-R  # ZRQAM1
DF.MSG  0[illegible]3410-R  # ZRQAM1    ZRQAM3
DISK.R  066102-R  # ZRQAM3
DRIVER  044734-R  # ZRQAM3
DROP.C  034772-R  # ZRQAM2    ZRQAM3
DRV.CT  035100-R  # ZRQAM2    ZRQAM3
```

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
DR.ERR	052462-R	# ZRQAM3		
DR.RET	064754-R	# ZRQAM3		
DUPROU	026072-R	# ZRQAM1		
DUP.FL	100652-R	# ZRQAM1	ZRQAM2	ZRQAM3
DUR	100664-R	# ZRQAM1	ZRQAM2	ZRQAM3
DU.MSG	007504-R	# ZRQAM1	ZRQAM2	
DU.RSN	010224-R	# ZRQAM1	ZRQAM2	
D$PCNT	002122-R	# ZRQAM1		
D.FAIL	101014-R	# ZRQAM1	ZRQAM2	ZRQAM3
EBD.10	012766-R	# ZRQAM1	ZRQAM2	
EBD.12	013026-R	# ZRQAM1	ZRQAM2	
EBD.13	013074-R	# ZRQAM1	ZRQAM2	
EBD.14	013126-R	# ZRQAM1	ZRQAM2	
EBD.18	013166-R	# ZRQAM1	ZRQAM2	
EBD.19	013222-R	# ZRQAM1	ZRQAM2	
EBD.24	013302-R	# ZRQAM1	ZRQAM2	
EBS.01	012724-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	011776-R	# ZRQAM1	ZRQAM3	
EGD.11	012036-R	# ZRQAM1	ZRQAM3	
EGD.12	012062-R	# ZRQAM1	ZRQAM3	
EGD.13	012110-R	# ZRQAM1	ZRQAM3	
EGD.14	012136-R	# ZRQAM1	ZRQAM3	
EGD.15	012166-R	# ZRQAM1	ZRQAM3	
EGD.16	012204-R	# ZRQAM1	ZRQAM3	
EGD.17	012234-R	# ZRQAM1	ZRQAM3	
EGD.18	012252-R	# ZRQAM1	ZRQAM3	
EGD.19	012272-R	# ZRQAM1	ZRQAM3	
EGD.20	012332-R	# ZRQAM1	ZRQAM3	
EGD.21	012420-R	# ZRQAM1	ZRQAM3	
EGD.22	012532-R	# ZRQAM1	ZRQAM3	
EGD.23	012572-R	# ZRQAM1	ZRQAM3	
EGD.24	012634-R	# ZRQAM1	ZRQAM3	
EGH.30	012700-R	# ZRQAM1	ZRQAM3	
EGS.01	011664-R	# ZRQAM1	ZRQAM2	
EGS.02	011704-R	# ZRQAM1	ZRQAM3	
EH.0	013356-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.1	013414-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.10	013770-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.12	014020-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.13	014054-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.2	013452-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.3	013512-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.4	013550-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.5	013574-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.6	013624-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.7	013654-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.8	013704-R	# ZRQAM1	ZRQAM2	ZRQAM3

```
SYMBOL  VALUE      REFERENCES...

EH.9    013740 R  # ZRQAM1    ZRQAM2    ZRQAM3
ELG FM  015374-R  # ZRQAM1    ZRQAM2
ELG.00  014720 R  # ZRQAM1    ZRQAM2
ELOG.P  077052 R  # ZRQAM1    ZRQAM2    ZRQAM3
EMS.BL  037504 R  # ZRQAM2
EMS.CM  042052 R  # ZRQAM2    ZRQAM3
EMS.EL  040430-R  # ZRQAM2    ZRQAM3
EMS.ER  042440 R  # ZRQAM2    ZRQAM3
EMS.R1  040364-R  # ZRQAM2    ZRQAM3
EMS.R2  040234 R  # ZRQAM2    ZRQAM3
EMS.01  042650-R  # ZRQAM2
EMS.10  042706-R  # ZRQAM2    ZRQAM3
EMS.12  042750 R  # ZRQAM2    ZRQAM3
EMS.13  043006-R  # ZRQAM2    ZRQAM3
EMS.14  043050-R  # ZRQAM2    ZRQAM3
EMS.18  043112-R  # ZRQAM2    ZRQAM3
EMS.21  043160-R  # ZRQAM2    ZRQAM3
EMS.24  043204-R  # ZRQAM2    ZRQAM3
EMS.30  043252-R  # ZRQAM2    ZRQAM3
ENTRY.  100650-R  # ZRQAM1    ZRQAM2    ZRQAM3
EOP.FL  100651-R  # ZRQAM1    ZRQAM2    ZRQAM3
ERRBLK  002134-R  # ZRQAM1
ERRMSG  002132-R  # ZRQAM1
ERRNBR  002130-R  # ZRQAM1
ERRTYP  002126-R  # ZRQAM1
ERR.CO  014662-R  # ZRQAM1    ZRQAM2
ERR.00  014126-R  # ZRQAM1    ZRQAM2
EVL     000004    # ZRQAM1  # ZRQAM2  # ZRQAM3
EX.ACC  015532-R  # ZRQAM1    ZRQAM2
EX.BBU  015670-R  # ZRQAM1    ZRQAM2
EX.BB1  015616-R  # ZRQAM1    ZRQAM2
EX.BB2  015550-R  # ZRQAM1    ZRQAM2
EX.BC   016416-R  # ZRQAM1    ZRQAM2
EX.BD   016472-R  # ZRQAM1    ZRQAM2
EX.BDR  016550-R  # ZRQAM1    ZRQAM2
EX.BDW  016640-R  # ZRQAM1    ZRQAM2
EX.CBC  016226-R  # ZRQAM1    ZRQAM2
EX.CBR  016272-R  # ZRQAM1    ZRQAM2
EX.CBW  016344-R  # ZRQAM1    ZRQAM2
EX.CMD  015444-R  # ZRQAM1    ZRQAM2
EX.CMP  015504-R  # ZRQAM1    ZRQAM2
EX.LB   017064-R  # ZRQAM1    ZRQAM2
EX.LBN  015750-R  # ZRQAM1    ZRQAM2
EX.LBR  016050-R  # ZRQAM1    ZRQAM2
EX.LBW  016116-R  # ZRQAM1    ZRQAM2
EX.ONL  015520-R  # ZRQAM1    ZRQAM2
EX.OP   015544-R  # ZRQAM1    ZRQAM2
EX.PBN  016010-R  # ZRQAM1    ZRQAM2
EX.RBN  016166-R  # ZRQAM1    ZRQAM2
EX.RD   015464-R  # ZRQAM1    ZRQAM2
EX.RP   016730-R  # ZRQAM1    ZRQAM2
EX.SA   015350-R  # ZRQAM1    ZRQAM2
```

```
GLOBAL CROSS REFERENCE                                     CREF   V02

SYMBOL   VALUE       REFERENCES...

EX.SB    015422-R   # ZRQAM1     ZRQAM2
EX.SBO   015416 R   # ZRQAM1     ZRQAM2
EX.SC    015366-R   # ZRQAM1     ZRQAM2
EX.TIM   017026 R   # ZRQAM1     ZRQAM2
EX.WRD   017016-R   # ZRQAM1     ZRQAM2
EX.WRT   015474-R   # ZRQAM1     ZRQAM2
FATAL.   065306-R   # ZRQAM3
FER.BC   101020-R   # ZRQAM1     ZRQAM2     ZRQAM3
FER.LB   101016-R   # ZRQAM1     ZRQAM2
FER0.L   100734-R   # ZRQAM1     ZRQAM2     ZRQAM3
FER1.L   100736-R   # ZRQAM1     ZRQAM2     ZRQAM3
FILL.B   060450-R   # ZRQAM3
FORCED   101015-R   # ZRQAM1     ZRQAM2     ZRQAM3
FORCE.   100772-R   # ZRQAM1     ZRQAM2     ZRQAM3
FREE.M   100672-R   # ZRQAM1     ZRQAM2     ZRQAM3
FSET.U   061332-R   # ZRQAM3
GET.IO   034466-R   # ZRQAM2     ZRQAM3
GET.PK   033620-R   # ZRQAM2     ZRQAM3
GET.RA   055124-R   # ZRQAM3
GET.RE   034344-R   # ZRQAM2     ZRQAM3
GP$DIS   026476-R   # ZRQAM2
GP$1     026154-R   # ZRQAM2
GP$10    026304-R   # ZRQAM2
GP$11    026314-R   # ZRQAM2
GP$12    026330-R   # ZRQAM2
GP$13    026342-R   # ZRQAM2
GP$14    026354-R   # ZRQAM2
GP$15    026366-R   # ZRQAM2
GP$16    026400-R   # ZRQAM2
GP$17    026406-R   # ZRQAM2
GP$18    026414-R   # ZRQAM2
GP$19    026422-R   # ZRQAM2
GP$2     026164-R   # ZRQAM2
GP$20    026430-R   # ZRQAM2
GP$21    026436-R   # ZRQAM2
GP$22    026452-R   # ZRQAM2
GP$23    026462-R   # ZRQAM2
GP$24    026470-R   # ZRQAM2
GP$25    026502-R   # ZRQAM2
GP$26    026516-R   # ZRQAM2
GP$27    026526-R   # ZRQAM2
GP$28    026542-R   # ZRQAM2
GP$29    026560-R   # ZRQAM2
GP$3     026174-R   # ZRQAM2
GP$30    026572-R   # ZRQAM2
GP$31    026610-R   # ZRQAM2
GP$32    026616-R   # ZRQAM2
GP$4     026206-R   # ZRQAM2
GP$5     026220-R   # ZRQAM2
GP$6     026230-R   # ZRQAM2
GP$7     026242-R   # ZRQAM2
GP$8     026254-R   # ZRQAM2
```

GLOBAL CROSS REFERENCE   CREF   V02

```
SYMBOL  VALUE      REFERENCES...

GP$9     026270 R   # ZRQAM2
HARD.E   061456-R   # ZRQAM3
HARD.I   046744-R   # ZRQAM3
HOE      100000     # ZRQAM1   # ZRQAM2   # ZRQAM3
HOE.FL   100741-R   # ZRQAM1     ZRQAM2     ZRQAM3
HOST.W   064150-R   # ZRQAM3
HOURS    100727-R   # ZRQAM1     ZRQAM2     ZRQAM3
HRD.MS   025506-R   # ZRQAM1     ZRQAM3
HRD.SU   025704-R   # ZRQAM1     ZRQAM3
HWPTE0   026000-R   # ZRQAM1
HWPTE1   026002-R   # ZRQAM1
HWPTS0   025774-R   # ZRQAM1
HWPTS1   025776-R   # ZRQAM1
HWPT.B   025770-R   # ZRQAM1
HWPT.D   025772-R   # ZRQAM1
HWPT.I   025764-R   # ZRQAM1
HWPT.V   025766-R   # ZRQAM1
HWQ1     002460-R   # ZRQAM1     ZRQAM2
HWQ2     002474 R   # ZRQAM1     ZRQAM2
HWQ3     002504-R   # ZRQAM1     ZRQAM2
HWQ4     002546-R   # ZRQAM1     ZRQAM2
HWQ5     002564-R   # ZRQAM1     ZRQAM2
HWQ6A    002634-R   # ZRQAM1     ZRQAM2
HWQ6B    002706-R   # ZRQAM1     ZRQAM2
HWQ7A    002762-R   # ZRQAM1     ZRQAM2
HWQ7B    003032-R   # ZRQAM1     ZRQAM2
HWQ8     003102-R   # ZRQAM1     ZRQAM2
HWQ9     003160-R   # ZRQAM1     ZRQAM2
IBE      010000     # ZRQAM1   # ZRQAM2   # ZRQAM3
IDU      000040     # ZRQAM1   # ZRQAM2   # ZRQAM3
IER      020000     # ZRQAM1   # ZRQAM2   # ZRQAM3
INIT.I   054040-R   # ZRQAM3
INIT.O   101022-R   # ZRQAM1     ZRQAM2     ZRQAM3
INIT.T   044576-R   # ZRQAM3
INI.CT   045570-R   # ZRQAM3
INI.RR   047614-R   # ZRQAM3
INT.GE   046454-R   # ZRQAM3
IN.IOD   034704-R   # ZRQAM2     ZRQAM3
IODQ     100634-R   # ZRQAM1     ZRQAM2     ZRQAM3
IODQ.I   100644-R   # ZRQAM1     ZRQAM2     ZRQAM3
IODQ.O   100646-R   # ZRQAM1     ZRQAM2     ZRQAM3
IO.RET   061062-R   # ZRQAM3
IPKT.A   076260-R   # ZRQAM1     ZRQAM2     ZRQAM3
IRDRX.   074124-R   # ZRQAM1     ZRQAM2     ZRQAM3
ISR      000100     # ZRQAM1   # ZRQAM2   # ZRQAM3
IXE      004000     # ZRQAM1   # ZRQAM2   # ZRQAM3
LOE      040000     # ZRQAM1   # ZRQAM2   # ZRQAM3
LOT      000010     # ZRQAM1   # ZRQAM2   # ZRQAM3
L$ACP    002110-R   # ZRQAM1
L$APT    002036-R   # ZRQAM1
L$AU     033240-R     ZRQAM1   # ZRQAM2
L$AUT    002070-R   # ZRQAM1
```

```
SYMBOL   VALUE      REFERENCES...

L$AUTO  032242-R     ZRQAM1  # ZRQAM2
L$CCP   002106 R   # ZRQAM1
L$CLEA  032542-R     ZRQAM1  # ZRQAM2
L$CO    002032-R   # ZRQAM1
L$DEPO  002011-R   # ZRQAM1
L$DESC  026130-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$DU    033146-R     ZRQAM1  # ZRQAM2
L$DUT   002072-R   # ZRQAM1
L$DVTY  026110-R     ZRQAM1  # ZRQAM2
L$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$HARD  026154-R     ZRQAM1  # ZRQAM2
L$HIME  002120-R   # ZRQAM1    ZRQAM2
L$HPCP  002016-R   # ZRQAM1
L$HPTP  002022-R   # ZRQAM1
L$HRDL  026152-R   # ZRQAM2
L$HW    025764-R   # ZRQAM1
L$HWLE  025762-R   # ZRQAM1
L$ICP   002104-R   # ZRQAM1
L$INIT  032230-R     ZRQAM1  # ZRQAM2
L$LADP  002026-R   # ZRQAM1
L$LAST  103020-R     ZRQAM1  # ZRQAM4
L$LOAD  002100-R   # ZRQAM1
L$LUN   002074-R   # ZRQAM1    ZRQAM2    ZRQAM3
L$MREV  002050-R   # ZRQAM1
L$NAME  002000-R   # ZRQAM1
L$NDHR  026324-R   # ZRQAM2
L$NDHW  026010-R   # ZRQAM1
L$NDSF  026624-R   # ZRQAM2
L$NDSW  026100-R   # ZRQAM1
L$PRIO  002042-R   # ZRQAM1
L$PROT  026102-R   # ZRQAM1
L$PRT   002112-R   # ZRQAM1
L$REPP  002062-R   # ZRQAM1
L$REV   002010-R   # ZRQAM1
L$RPT   027462-R     ZRQAM1  # ZRQAM2
L$SFTL  026326-R   # ZRQAM2
L$SOFT  026330-R     ZRQAM1  # ZRQAM2
L$SPC   002056-R   # ZRQAM1
L$SPCP  002020-R   # ZRQAM1
L$SPTP  002024-R   # ZRQAM1
```

```
SYMBOL   VALUE       REFERENCES...

L$STA    002030-R   # ZRQAM1
L$SW     026014-R   # ZRQAM1
L$SWLE   026012-R   # ZRQAM1
L$TEST   002114-R   # ZRQAM1
L$TIML   002014-R   # ZRQAM1
L$UNIT   002012-R   # ZRQAM1     ZRQAM2     ZRQAM3
MAN.TS   026074-R   # ZRQAM1     ZRQAM3
MD.INI   053700-R   # ZRQAM3
MINUTE   100730-R   # ZRQAM1     ZRQAM2     ZRQAM3
MODULA   035632-R   # ZRQAM2     ZRQAM3
MSCP.P   074550-R   # ZRQAM1     ZRQAM2     ZRQAM3
MSG.01   010252-R   # ZRQAM1     ZRQAM2
MSG.02   010304-R   # ZRQAM1     ZRQAM3
MSG.03   010340-R   # ZRQAM1     ZRQAM3
MULTI.   053234-R   # ZRQAM3
NAME.H   026006-R   # ZRQAM1
NAME.L   026004-R   # ZRQAM1
NEX      100712-R   # ZRQAM1     ZRQAM2     ZRQAM3
NEXT.P   100726-R   # ZRQAM1     ZRQAM2     ZRQAM3
NEX.TR   033250-R   # ZRQAM2     ZRQAM3
NULL     005100-R   # ZRQAM1     ZRQAM2
OFF      000002     # ZRQAM1   # ZRQAM2   # ZRQAM3
OF.RC    100704-R   # ZRQAM1     ZRQAM2     ZRQAM3
ON       000001     # ZRQAM1   # ZRQAM2   # ZRQAM3
OUT.IO   034646-R   # ZRQAM2     ZRQAM3
OVF.CH   063530-R   # ZRQAM3
PARITY   033260-R   # ZRQAM2     ZRQAM3
PKT.US   076262-R   # ZRQAM1     ZRQAM2     ZRQAM3
PNT      001000     # ZRQAM1   # ZRQAM2   # ZRQAM3
POLL.C   065540-R   # ZRQAM3
POLL.R   065640-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   060702-R   # ZRQAM3
PTCH1    002136-R   # ZRQAM1
PTCH2    002210-R   # ZRQAM1
PTCH3    002262-R   # ZRQAM1
PTCH4    002334-R   # ZRQAM1
PTCH5    002406-R   # ZRQAM1
PUTA.B   034606-R   # ZRQAM2     ZRQAM3
PUT.IO   034542-R   # ZRQAM2     ZRQAM3
PUT.PK   034204-R   # ZRQAM2     ZRQAM3
PUT.RE   034452-R   # ZRQAM2     ZRQAM3
P.INDE   101006-R   # ZRQAM1     ZRQAM2     ZRQAM3
QIO      100670-R   # ZRQAM1     ZRQAM2     ZRQAM3
```

```
SYMBOL  VALUE      REFERENCES...

QIO.FU  057116-R  # ZRQAM3
QIO.GE  054524-R  # ZRQAM3
QIO.LB  057436-R  # ZRQAM3
QIO.OK  054244-R  # ZRQAM3
QIO.OU  054452-R  # ZRQAM3
QIO.SI  060240-R  # ZRQAM3
QIO.UN  055512-R  # ZRQAM3
RANDOM  074506-R  # ZRQAM1     ZRQAM2     ZRQAM3
RANDY   055234-R  # ZRQAM3
RDM.CN  074504-R  # ZRQAM1     ZRQAM2     ZRQAM3
RDRX.A  074122-R  # ZRQAM1     ZRQAM2     ZRQAM3
RDRX.E  025372-R  # ZRQAM1     ZRQAM2
RD.COU  101010-R  # ZRQAM1     ZRQAM2     ZRQAM3
REG.EX  045714-R  # ZRQAM3
RETPKT  076276-R  # ZRQAM1     ZRQAM2     ZRQAM3
ROUND.  063604-R  # ZRQAM3
RPS.RE  064566-R  # ZRQAM3
RPT1    010372-R  # ZRQAM1     ZRQAM2
RPT10   011102-R  # ZRQAM1     ZRQAM2
RPT11   011170-R  # ZRQAM1     ZRQAM2
RPT12   011236-R  # ZRQAM1     ZRQAM2
RPT13   011304-R  # ZRQAM1     ZRQAM2
RPT14   011404-R  # ZRQAM1     ZRQAM2
RPT15   011502-R  # ZRQAM1     ZRQAM2
RPT16   011602-R  # ZRQAM1     ZRQAM2
RPT2    010456-R  # ZRQAM1     ZRQAM2
RPT3    010522-R  # ZRQAM1     ZRQAM2
RPT4    010606-R  # ZRQAM1     ZRQAM2
RPT5    010652-R  # ZRQAM1     ZRQAM2
RPT6    010740-R  # ZRQAM1     ZRQAM2
RPT7    011004-R  # ZRQAM1     ZRQAM2
RPT8    011022-R  # ZRQAM1     ZRQAM2
RPT9    011050-R  # ZRQAM1     ZRQAM2
RP.ADD  077050-R  # ZRQAM1     ZRQAM2     ZRQAM3
RP.IND  077046-R  # ZRQAM1     ZRQAM2     ZRQAM3
RP.USE  077036-R  # ZRQAM1     ZRQAM2     ZRQAM3
SA.REG  100706-R  # ZRQAM1     ZRQAM2     ZRQAM3
SB.COD  100700-R  # ZRQAM1     ZRQAM2     ZRQAM3
SC.CLK  022610-R  # ZRQAM1     ZRQAM2
SC.CON  017506-R  # ZRQAM1     ZRQAM2
SC.CTO  022052-R  # ZRQAM1     ZRQAM2
SC.DIS  020104-R  # ZRQAM1     ZRQAM2
SC.DST  020514-R  # ZRQAM1     ZRQAM2
SC.DS2  020570-R  # ZRQAM1     ZRQAM2
SC.DUP  017530-R  # ZRQAM1     ZRQAM2
SC.ECC  020642-R  # ZRQAM1     ZRQAM2
SC.ECD  020724-R  # ZRQAM1     ZRQAM2
SC.EC1  021174-R  # ZRQAM1     ZRQAM2
SC.EC2  021224-R  # ZRQAM1     ZRQAM2
SC.EC3  021254-R  # ZRQAM1     ZRQAM2
SC.EC4  021306-R  # ZRQAM1     ZRQAM2
SC.EC5  021336-R  # ZRQAM1     ZRQAM2
```

```
SYMBOL  VALUE      REFERENCES...

SC.EC6  021366-R  # ZRQAM1    ZRQAM2
SC.EC7  021416-R  # ZRQAM1    ZRQAM2
SC.EC8  021450 R  # ZRQAM1    ZRQAM2
SC.EC9  021502-R  # ZRQAM1    ZRQAM2
SC.EDC  022202-R  # ZRQAM1    ZRQAM2
SC.FCT  021126-R  # ZRQAM1    ZRQAM2
SC.FER  020176-R  # ZRQAM1    ZRQAM2
SC.FE2  020264-R  # ZRQAM1    ZRQAM2
SC.FUL  020776-R  # ZRQAM1    ZRQAM2
SC.HWP  021604-R  # ZRQAM1    ZRQAM2
SC.IDS  022242-R  # ZRQAM1    ZRQAM2
SC.INR  017622-R  # ZRQAM1    ZRQAM2
SC.INV  017654-R  # ZRQAM1    ZRQAM2
SC.IOP  020032-R  # ZRQAM1    ZRQAM2
SC.ISH  020354-R  # ZRQAM1    ZRQAM2
SC.IS2  020434-R  # ZRQAM1    ZRQAM2
SC.NBB  023060-R  # ZRQAM1    ZRQAM2
SC.NXM  021762-R  # ZRQAM1    ZRQAM2
SC.ODA  021710-R  # ZRQAM1    ZRQAM2
SC.ODB  021740-R  # ZRQAM1    ZRQAM2
SC.ONL  017560-R  # ZRQAM1    ZRQAM2
SC.PAR  022016-R  # ZRQAM1    ZRQAM2
SC.POE  022472-R  # ZRQAM1    ZRQAM2
SC.PSP  022734-R  # ZRQAM1    ZRQAM2
SC.RCT  020756-R  # ZRQAM1    ZRQAM2
SC.RDY  022526-R  # ZRQAM1    ZRQAM2
SC.REF  023130-R  # ZRQAM1    ZRQAM2
SC.REP  023016-R  # ZRQAM1    ZRQAM2
SC.RSP  022636-R  # ZRQAM1    ZRQAM2
SC.SAF  021644-R  # ZRQAM1    ZRQAM2
SC.SDI  017462-R  # ZRQAM1    ZRQAM2
SC.SDS  022124-R  # ZRQAM1    ZRQAM2
SC.SON  017602-R  # ZRQAM1    ZRQAM2
SC.SRI  022404-R  # ZRQAM1    ZRQAM2
SC.SRT  022312-R  # ZRQAM1    ZRQAM2
SC.SUR  022704-R  # ZRQAM1    ZRQAM2
SC.SWP  021544-R  # ZRQAM1    ZRQAM2
SC.UNK  017672-R  # ZRQAM1    ZRQAM2
SC.VOL  017752-R  # ZRQAM1    ZRQAM2
SC.576  021052-R  # ZRQAM1    ZRQAM2
SEND    035156-R  # ZRQAM2    ZRQAM3
SEQUEN  066202-R  # ZRQAM3
SET.CP  033424-R  # ZRQAM2    ZRQAM3
SET.CT  047716-R  # ZRQAM3
SET.UP  033520-R  # ZRQAM2    ZRQAM3
SFPTBL  026014-R  # ZRQAM1
SFT.MS  025604-R  # ZRQAM1    ZRQAM3
SOFT.E  070346-R  # ZRQAM3
SPACE4  025736-R  # ZRQAM1    ZRQAM2
STEP    100702-R  # ZRQAM1    ZRQAM2    ZRQAM3
ST.COD  100676-R  # ZRQAM1    ZRQAM2    ZRQAM3
SWEEP   064434-R  # ZRQAM3
```

```
SYMBOL   VALUE      REFERENCES...

SWM1     004706 R  # ZRQAM1     ZRQAM2
SWP.DP   026022-R  # ZRQAM1     ZRQAM3
SWP.ER   026014-R  # ZRQAM1     ZRQAM3
SWP.FL   026020-R  # ZRQAM1     ZRQAM2     ZRQAM3
SWP.RA   026024-R  # ZRQAM1     ZRQAM3
SWP.TI   026026-R  # ZRQAM1     ZRQAM3
SWP.UC   026030 R  # ZRQAM1     ZRQAM3
SWP.UD   026032-R  # ZRQAM1     ZRQAM3
SWP.XF   026016-R  # ZRQAM1     ZRQAM3
SWQ1     003260-R  # ZRQAM1     ZRQAM2
SWQ10    003532-R  # ZRQAM1     ZRQAM2
SWQ11    003576-R  # ZRQAM1     ZRQAM2
SWQ12    003630-R  # ZRQAM1     ZRQAM2
SWQ13    003726-R  # ZRQAM1     ZRQAM2
SWQ14    004004-R  # ZRQAM1     ZRQAM2
SWQ15    004056-R  # ZRQAM1     ZRQAM2
SWQ17    004126-R  # ZRQAM1     ZRQAM2
SWQ19    004224-R  # ZRQAM1     ZRQAM2
SWQ2     003302-R  # ZRQAM1     ZRQAM2
SWQ20    004314-R  # ZRQAM1     ZRQAM2
SWQ21    004402-R  # ZRQAM1     ZRQAM2
SWQ22    004466-R  # ZRQAM1     ZRQAM2
SWQ23    004526-R  # ZRQAM1     ZRQAM2
SWQ24    004600-R  # ZRQAM1     ZRQAM2
SWQ26    004644-R  # ZRQAM1     ZRQAM2
SWQ27    005004-R  # ZRQAM1     ZRQAM2
SWQ28    005030-R  # ZRQAM1     ZRQAM2
SWQ4     003362-R  # ZRQAM1     ZRQAM2
SWQ7     003404-R  # ZRQAM1     ZRQAM2
SWQ9     003456-R  # ZRQAM1     ZRQAM2
TALLY    074146-R  # ZRQAM1     ZRQAM2     ZRQAM3
TEMP1    100720-R  # ZRQAM1     ZRQAM2     ZRQAM3
TEMP2    100722-R  # ZRQAM1     ZRQAM2     ZRQAM3
TIME     033342-R  # ZRQAM2     ZRQAM3
TRK.SG   074500-R  # ZRQAM1     ZRQAM2     ZRQAM3
TST.PA   026076-R  # ZRQAM1     ZRQAM3
TYPER    100742-R  # ZRQAM1     ZRQAM2     ZRQAM3
TYPEW    100752-R  # ZRQAM1     ZRQAM2     ZRQAM3
T$FREE   103140-R  # ZRQAM4
T$PTHV   000004      ZRQAM1   # ZRQAM4
T.ADDR   074476-R  # ZRQAM1     ZRQAM2     ZRQAM3
T1       044562-R    ZRQAM1   # ZRQAM3
UAM      000200    # ZRQAM1   # ZRQAM2   # ZRQAM3
UPD.IO   062662-R  # ZRQAM3
VEC.BR   046126-R  # ZRQAM3
WAIT     035614-R  # ZRQAM2     ZRQAM3
XX13     017130-R  # ZRQAM1     ZRQAM2
XX23     017152-R  # ZRQAM1     ZRQAM2
XX32     017206-R  # ZRQAM1     ZRQAM2
XX33     017234-R  # ZRQAM1     ZRQAM2
XX34     017272-R  # ZRQAM1     ZRQAM2
$END.L   103142-R  # ZRQAM4
```

```
SYMBOL  VALUE      REFERENCES...

$SAVE2  073640-R    B16MUL  # B16SAV   ZRQAM2   ZRQAM3
$SAVE3  073654-R  # B16SAV    ZRQAM2   ZRQAM3
$SAVE4  073672 R  # B16SAV    ZRQAM2   ZRQAM3
$SAVE5  073712-R    B16MUL  # B16SAV   ZRQAM2   ZRQAM3
```