RQDX1 RX50 RQDX1/RUX50 EXER

RD51/52 RUX50 CZRQAFO

AH-T399F-MC

FICHE 01 OF 03

APR 1985

RQDX1 RX50
RD51/52 RUX50

RQDX1/RUX50 EXER
CZRQAFO

AH-T399F-MC
FICHE 02 OF 03

APR 1985
digital
Made In USA

RQDX1 RX50
RD51/52 RUX50

RQDX1/RUX50 EXER
CZRQAF0

AH-T399F-MC
FICHE 03 OF 03

APR 1985
digital
Made In USA

IDENTIFICATION

PRODUCT CODE: AC-T398F-MC

PRODUCT NAME: CZRQAFO RQDX1/RUX50 EXERCISER

PRODUCT DATE: 27-DEC-84

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: RAVINDER K. KARWAN
BOB POWERS

```
!*************************************************************************
!                                                                        *
!                        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

        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

        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 .

!
!****** RING SIZES
!
        CR_LOG          = 2,                                ! LOG2 LENGTH OF COMMAND RING
        RR_LOG          = 2,                                ! LOG2 LENGTH OF RESPONSE RING
        CRING_LEN       = 1 ↑ CR_LOG,                       ! COMMAND RING LENGTH
        RRING_LEN       = 1 ↑ RR_LOG,                       ! RESPONSE RING LENGTH
!
!****** 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          TOO_MANY_READS   = 2,                                        !FOR READ/WRITE BALANCE WITH HOST READ COMPARES  ZZZ
;  0          DESC_SIZ         = 4,                                        !NO. OF BYTES IN A PACKET DESCRIPTOR             ZZZ
;  0   !
;  0   !****** SW P-TABLE FLAGS (SWP_FLAGS)
;  0   !
;  0   !ZZZ   SWF_TRC          = %o'000001',                               ! DIAGNOSTIC TRACE
;  0          SWF_APT          = %o'000001',                               !RUNNING UNDER A.P.T. MONITOR                    ZZZ
;  0          SWF_RDM          = %o'000002',                               ! RANDOM SEEK MODE
;  0          SWF_CRC          = %o'000004',                               ! READ-COMPARE AT CONTROLLER
;  0   !      SWF_DCC          = %o'000010',                               ! DRIVE COMPLEMENT COMPLETE
;  0          SWF_CWC          = %o'000020',                               ! WRITE-COMPARE AT CONTROLLER
;  0          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
        !
            INI_ATT          = 2,                                       ! NO. OF HW INIT ATTEMPTS BEFORE FAILURE IS ASSUMED
            WR_RING          = ((%o'200') or (CR_LOG ^ 3) or (RR_LOG)),
                                                                        ! WR-BIT-AND-RING-LENGTH (STEP 1 WRITE/STEP 2 READ)
            QIO_PER_CTLR     = CRING_LEN * 2,                           ! MAXIMUM NUMBER OF OUTSTANDING QIOS PER CONTROLLER
        !   MAX_XFER         = 256,                                     ! MAXIMUM SIZE (WORDS) OF AN I/O TRANSFER
            REMOVABLE_BIT    = %o'0',                                   ! BIT IN HARDWARE TABLES MARKING A REMOVABLE DISK
            FIXED_BIT        = %o'20',                                  ! BIT IN HARDWARE TABLES MARKING A FIXED     DISK
            REMOVABLE        = 0,                                       ! NUMBER FOR REMOVABLE DISK WHEN SHIFTED RIGHT
            FIXED            = 1,                                       ! NUMBER FOR FIXED     DISK WHEN SHIFTED RIGHT
```

```
;  0      RX_50          = 0,                  !D_TYPE FLAG = 0 FOR RX50 ZZZ
;  0      RD_51          = 1,                  !D_TYPE FLAG = 1 FOR RD51 ZZZ
;  0      RD_52          = 2,                  !              2 FOR RD52 ZZZ
;  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
;  0      MD_EXP         = %o'100000',                   ! EXPRESS REQUEST
;  0   !
;  0   !****** CONNECTION ID VALUES (MSCP_PKT, RETPKT)
;  0   !            (SERVE AS SOURCES AND DESTINATIONS OF MSCP MESSAGES)
;  0   !
;  0      CID_DISK       = 0,                            ! DISK MSCP
;  0      CID_MSCP       = 0,                            !DISK MSCP
!ZZZ
;  0      CID_TAPE       = 1,                            ! TAPE MSCP
;  0      CID_DUP        = 2,                            ! DIAGNOSTIC AND UTILITIES PROTOCOL
;  0      CID_DRIVER     = 3,                            ! EXERCISER "DRIVER"
```

```
;  0    !
;  0    !****** MESSAGE TYPE VALUES
;  0    !
;  0            MT_SEQ          = 0,                            ! SEQUENTIAL (FROM PORT)
;  0            MT_DG           = 1,                            ! DATAGRAM (FROM PORT)
;  0            MT_CRD          = 2,                            ! CREDIT NOTIFICATION (FROM PORT)
;  0            MT_FATAL        = 3,                            ! FATAL DEVICE ERROR (FROM "DRIVER")
;  0            MT_TIMEOUT      = 4,                            ! COMMAND TIMEOUT (FROM "DRIVER")
;  0    !
;  0    !****** CONTROLLER FLAGS
;  0    !               (IN SET CONTROLLER CHARACTERISTICS COMMAND AND RESPONSE)
;  0    !
;  0            CF_ATN          = %o'000200',                   ! ENABLE ATTENTION MESSAGES
;  0            CF_MSC          = %o'000100',                   ! ENABLE MISCELLANEOUS ERROR LOG MESSAGES
;  0            CF_OTH          = %o'000040',                   ! ENABLE OTHER HOST'S ERROR LOG MESSAGES
;  0            CF_THS          = %o'000020',                   ! ENABLE THIS HOST'S ERROR LOG MESSAGES
;  0    !       CF_MASK         = CF_ATN or CF_MSC or CF_THS,
;  0            CF_MASK         = CF_MSC or CF_THS,             ! RELEVENT BITS IN CTLR FLAGS WORD
;  0    !
;  0    !****** UNIT FLAGS
;  0    !               (IN ONLINE COMMAND AND RESPONSE)
;  0    !
;  0            UF_REMOVABLE    = %o'000200',                   ! REMOVABLE MEDIA
;  0            UF_WPH          = %o'020000',                   ! WRITE PROTECT (HARDWARE)
;  0    !
;  0    !****** STATUS / EVENT CODE DEFINITIONS
;  0    !
;  0            ST_SUC          = %o'0',                        ! SUCCESS
;  0            ST_CMD          = %o'1',                        ! INVALID COMMAND
;  0            ST_ABO          = %o'2',                        ! COMMAND ABORTED
;  0            ST_OFL          = %o'3',                        ! UNIT OFFLINE
;  0            ST_AVL          = %o'4',                        ! DRIVE AVAILABLE
;  0            ST_MFE          = %o'5',                        ! MEDIA FORMAT ERROR
;  0            ST_WPT          = %o'6',                        ! WRITE PROTECTED
;  0            ST_CMP          = %o'7',                        ! COMPARE ERROR
;  0            ST_DAT          = %o'10',                       ! DATA ERROR
;  0            ST_HST          = %o'11',                       ! HOST BUFFER ACCESS ERROR
;  0            ST_CNT          = %o'12',                       ! CONTROLLER ERROR
;  0            ST_DRV          = %o'13',                       ! DRIVE ERROR
;  0            ST_DIA          = %o'37',                       ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  0    !
;  0    !****** END MESSAGE FLAGS
;  0    !
;  0            EF_BBR          = %o'200',                      ! BAD BLOCK REPORTED
;  0            EF_BBU          = %o'100',                      ! BAD BLOCK NOT REPORTED
;  0    !
;  0    !****** RDRX LITERALS
;  0    !
;  0            RCIP            = 0,                            ! IP REGISTER
;  0            RCSA            = 1,                            ! SA REGISTER
;  0    !
;  0    !****** COMMON SA REGISTER BIT DEFINITIONS
;  0    !
```

```
;  0              SA_S1           = %o'004000',                    ! STEP 1 STATUS BIT
;  0              SA_S2           = %o'010000',                    !   :  2
;  0              SA_S3           = %o'020000',                    !   :  3
;  0              SA_S4           = %o'040000',                    !   V  4
;  0              SA_ERR          = %o'100000',                    ! ERROR INDICATOR
;  0              SA_INT          = %o'000200',                    ! INTERRUPT ENABLE DURING INITIALIZATION
;  0              SA_GO           = %o'000001',                    ! GO BIT TO START FIRMWARE
;  0      !
;  0      !****** INITIALIZATION STEP READ MASKS
;  0      !
;  0              S1_MASK         = %o'176000',                    ! STEP 1 READ BITS
;  0              S2_MASK         = %o'174377',                    !   :  2
;  0              S3_MASK         = %o'174377',                    !   :  3
;  0              S4_MASK         = %o'174000',                    !   V  4
;  0      !
;  0      !****** COMMAND TYPES
;  0      !
;  0              IMM_CMD         = 0,                             ! IMMEDIATE COMMAND
;  0              SEQ_CMD         = 1,                             ! SEQUENTIAL COMMAND
;  0              NON_SEQ_CMD     = 2,                             ! NON-SEQUENTIAL COMMAND
;  0      !
;  0      !****** ERROR-LOG FORMAT TYPES
;  0      !
;  0              FORMAT_CNTR     = %o'0',                         ! CONTROLLER ERROR
;  0              FORMAT_HOST     = %o'1',                         ! HOST MEMORY ACCESS ERROR
;  0              FORMAT_XFER     = %o'2',                         ! DISK TRANSFER ERROR
;  0              FORMAT_SDI      = %o'3',                         ! 'STANDARD DISK INTECONNECT' ERROR
;  0              FORMAT_SDE      = %o'4',                         ! SMALL DISK ERROR
;  0      !
;  0      !****** ERROR-LOG BLOCK NUMBER INFORMATION
;  0      !
;  0              TYPE_LBN        = %o'0000',                      ! LOGICAL BLOCK NUMBER
;  0              TYPE_RBN        = %o'0110',                      ! REPLACEMENT BLOCK NUMBER
;  0      !
;  0      !****** MSCP DISK MODEL CODES
;  0      !
;  0      !       MODEL_RX50      = 7,                             ! RX50
;  0      !       MODEL_RD51      = 6,                             ! RD51
;  0      !       MODEL_RD52      = 8,                             ! RD52
;  0      !
;  0      !****** 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              NOT_PRESENT     = 0,                             ! DISK IS NOT PRESENT IN CONTROLLER
;  0              UNPROTECTED     = 1,                             ! DISK HAS UNPROTECTED CUSTOMER LBN'S
```

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

```
!*************************************************************************
!                                                                       *
!                        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
        TES,                                     !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_BEGO       = [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_ENDO       = [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_NAMEO      = [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)
```

```
;   0          PKT_Q           = [1, 2, 4, 0],            ! PACKET DESCRIPTOR (HI ORDER Q-BUS BITS)
;   0          PKT_F           = [1, 14, 1, 0],           ! PACKET DESCRIPTOR FLAG BIT
;   0          PKT_O           = [1, 15, 1, 0],           ! PACKET DESCRIPTOR OWNERSHIP BIT
;   0          CMD_TYPE        = [2, 0, 8, 0],            ! COMMAND TYPE
;   0          RSP_RECEIVED    = [2, 8, 8, 0],            ! FLAG SET IF RESPONSE TO COMMAND RECEIVED
;   0          MSGLEN          = [3, 0, 16, 0],           ! MESSAGE LENGTH
;   0          CREDITS         = [4, 0, 4, 0],            ! CREDITS
;   0          MSGTYP          = [4, 4, 4, 0],            ! MESSAGE TYPE
;   0          CONNID          = [4, 8, 8, 0],            ! CONNECTION ID
;   0     !
;   0     !    GENERIC COMMAND PACKET AND END PACKET HEADER FILEDS
;   0     !
;   0          CRN_LO          = [5, 0, 16, 0],           ! COMMAND REF NUMBER (LO ORDER)
;   0          CRN_HI          = [6, 0, 16, 0],           ! COMMAND REF NUMBER (HI ORDER)
;   0          DK_NUM          = [7, 0, 16, 0],           ! DISK ADDRESS (RD/RX DISK NUMBER)
;   0          OPCODE          = [9, 0, 8, 0],            ! OPCODE AND ENDCODE
;   0          MODIFY          = [10, 0, 16, 0],          ! COMMAND MODIFIERS
;   0          STATUS_CODE     = [10, 0, 5, 0],           ! STATUS (PART OF RESPONSE PACKET)
;   0          STATUS_SUBCODE  = [10, 5, 11, 0],          ! SUBCODE (PART OF RESPONSE PACKET)
;   0     !
;   0     !    READ, WRITE, AND ACCESS COMMAND FIELDS (FOR COMMAND AND END PACKETS)
;   0     !
;   0          BC_LO           = [11, 0, 16, 0],          ! BYTE COUNT (LO ORDER)
;   0          BC_HI           = [12, 0, 16, 0],          ! BYTE COUNT (HI ORDER)
;   0          BUF_0           = [13, 0, 16, 0],          ! I/O BUFFER DESCRIPTOR
;   0          BUF_1           = [14, 0, 16, 0],          !     :
;   0          BUF_2           = [15, 0, 16, 0],          !     :
;   0          BUF_3           = [16, 0, 16, 0],          !     :
;   0          BUF_4           = [17, 0, 16, 0],          !     :
;   0          BUF_5           = [18, 0, 16, 0],          !     V
;   0          LBN_L           = [19, 0, 16, 0],          ! LOGICAL BLOCK NUMBER (LO ORDER)
;   0          LBN_H           = [20, 0, 16, 0],          ! LOGICAL BLOCK NUMBER (HI ORDER)
;   0
;   0     !
;   0     ! DUP PROGRAM LETTER FIELDS (FOR EXECUTE LOCAL PROGRAM CMD)          ZZZ
;   0     !                                                                   ZZZ
;   0          L1              = [11, 0, 8, 0],     !LETTER NO 1            ZZZ
;   0          L2              = [11, 8, 8, 0],     !LETTER NO 2            ZZZ
;   0          L3              = [12, 0, 8, 0],     !LETTER NO 3            ZZZ
;   0          L4              = [12, 8, 8, 0],     !LETTER NO 4            ZZZ
;   0          L5              = [13, 0, 8, 0],     !LETTER NO 5            ZZZ
;   0          L6              = [13, 8, 8, 0],     !LETTER NO 6            ZZZ
;   0     !
;   0     !    SET CONTROLLER CHARACTERISTICS COMMAND FIELDS
;   0     !
;   0          C_FLAGS         = [12, 0, 16, 0],          ! CONTROLLER FLAGS
;   0     !
;   0     !    ONLINE COMMAND FIELDS
;   0     !
;   0          U_FLAGS         = [12, 0, 16, 0],          ! UNIT FLAGS
;   0          DDPAR           = [19, 0, 16, 0]           ! DEVICE-DEPENDENT PARAMETERS
;   0          tes,
;   0     !
```

```
;  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 SIZE (LO ORDER)
```

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

```
;      0            set
;      0            WORD0              = [0, 0, 16, 0],          ! ALL FIELDS IN WORD 0
;      0            CRING_CNT          = [0, 0, 8, 0],           ! NUMBER OF SLOTS IN CRING NOT YET RETURNED TO HOST
;      0            IG_INT             = [0, 14, 1, 0],          ! IGNORE INTERRUPT BIT
;      0            STAT               = [0, 15, 1, 0],          ! ONLINE / OFFLINE STATUS
;      0            SA_SAVE            = [1, 0, 16, 0],          ! SA REGISTER SAVE WORD
;      0            RR_BEG             = [2, 0, 16, 0],          ! FIXED ADDRESSES OF START AND
;      0            RR_END             = [3, 0, 16, 0],          !        END OF EACH RING
;      0            CR_BEG             = [4, 0, 16, 0],          !              :
;      0            CR_END             = [5, 0, 16, 0],          !              V
;      0            RR_POLL            = [6, 0, 16, 0],          ! ADDR OF NEXT RRING SLOT TO BE POLLED
;      0            CR_POLL            = [7, 0, 16, 0],          ! ADDR OF NEXT CRING SLOT TO BE POLLED
;      0            CR_NEXT            = [8, 0, 16, 0]           ! ADDR OF NEXT AVAIL CRING SLOT
;      0            tes,
;      0    !
;      0    !****** ERROR LOG PACKET SAVE AREA FIELDS
;      0    !
;      0    EP_FIELDS =
;      0            set
;      0            EL_CNTR            = [0, 0, 8, 0],           ! CONTROLLER NUMBER
;      0            EL_CONTENTS        = [0, 8, 8, 0],           ! FLAG INDICATES IF PACKET CONTENTS ALREADY PRINTED
;      0            EL_MSGLEN          = [1, 0, 16, 0],          ! PACKET LENGTH
;      0            EL_CRN_LO          = [3, 0, 16, 0],          ! COMMAND REFERENCE NUMBER
;      0            EL_CRN_HI          = [4, 0, 16, 0],          !
;      0            EL_DK_NUM          = [5, 0, 16, 0],          ! DISK ADDRESS (RD/RX DISK NUMBER)
;      0            EL_FORMAT          = [7, 0, 8, 0],           ! FORMAT
;      0            EL_CONTINUE        = [7, 14, 1, 0],          ! CONTINUE FLAG
;      0            EL_SUCCESS         = [7, 15, 1, 0],          ! SUCCESS FLAG
;      0            EL_CODE            = [8, 0, 5, 0],           ! ERROR CODE
;      0            EL_SUBCODE         = [8, 5, 11, 0],          ! SUB CODE
;      0            EL_RETRY           = [20, 8, 8, 0],          ! RETRY COUNT
;      0            EL_BLOCK           = [23, 0, 16, 0],         ! BLOCK NUMBER
;      0            EL_BLOCK_TYPE      = [24, 12, 4, 0]          ! TYPE OF BLOCK NUMBER INFO RETURNED
;      0            tes,
;      0    !
;      0    !****** INFORMATION ABOUT LAST RESPONSE PACKET
;      0    !
;      0    LAST_PKT_FIELDS =
;      0            set
;      0            LAST_HRD_ERR       = [0, 0, 16, 0],          ! FLAG INDICATES IF HARD ERROR OCCURED
;      0            LAST_CRN_LO        = [1, 0, 16, 0],          ! COMMAND REFERENCE NUMBER
;      0            LAST_CRN_HI        = [2, 0, 16, 0]           !
;      0            tes,
;      0    !
;      0    !****** RDRX REGISTER FIELDS
;      0    !
;      0    RC_REG =
;      0            set
;      0            RC_ALL             = [0, 16, 0]              ! DEFINE ALL BITS
;      0            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 ZRQAFO.REQ/LIST=ZRQAFO.LIS/LIBRARY=ZRQAFO.L16/SOURCE=PAGE:53

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

```
module ZRQAM1 (

%title 'RD/RX EXERCISER'
                ident = 'V02.1',
                addressing_mode (absolute),
                environment (noeis)
                ) =

begin


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

        PRODUCT CODE:           AC-T398F-MC

        PRODUCT NAME:           CZRQAFO RQDX1/RUX50 EXERCISER

        PRODUCT DATE:           27-DEC-84

        MAINTAINER:             DIAGNOSTIC ENGINEERING

        AUTHOR:                 RAVINDER K. KARWAN
                                BOB POWERS


        Copyright (C) 1983, 1984, 1985

        Digital Equipment Corporation, Maynard, Massachusetts  01754

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

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

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


        The following are trademarks of Digital Equipment Corporation:

        DIGITAL         PDP             UNIBUS          MASSBUS
        DEC             DECUS           DECTAPE
```

```
        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.
```

TABLE OF CONTENTS

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


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

      This program  will functionally verify  and exercise RQDX1
      or RUX50 Controller/Disk Drive subsystems. It is designed
      to verify that the subsystem is functioning correctly and
      operating within design specifications.



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

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

      LSI - 11/23 processor with 28K or more of memory,  console
      device (eg. VT100) and RQDX1 or RUX50 controller board and
      attached RD51 or RD52 WINCHESTER drive(s) and RX-50 FLOPPY
      drive(s)

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

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


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

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

1.4   DIAGNOSTIC HIERARCHY PREREQUISITES
      ----------------------------------

      NONE


1.5   ASSUMPTIONS
      -----------

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

```
        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 RQDX1 or RUX50 as addressed by the processor with memory management turned off. The program expects an even 16-bit address in the range of 160000 to 177774. 172150 is the default.

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

    Answer with the interrupt vector ofthe same RQDX1 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-RQDX1 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 RQDX1 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 diifferent 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 sec-
              tion.  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  de-
              tected, 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 indi-
              cated.  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 RQDX1 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 control-
     ler  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 success-
     ful.  Normally, the whole sequence of retries is classi-
     fied 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 sequen-
     tially.

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 con-
     troller 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 address, if applicable. In some cases where a drive-fatal error applies to a controller, the controller's IP address will be printed.

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

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

3.2 INITIALIZATION ERRORS

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

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

1. Register Existence Test failure (no drive present)

2. Vector Test failure

3. BR Level Test failure

4. Initialization sequence failure

5. Online failed

6. Access failed

3.3 EXERCISER ERRORS

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

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

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

3.4 ERROR LOG MESSAGES

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

3.5 MSCP ERRORS

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

3.6 SAMPLE ERROR STATEMENT

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

```
(example)                          (comments)

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

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

```
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 ques-
      tions. 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, depend-
      ing 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 an 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 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  Control-
                                         ler 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'      See #50.
    error.

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

```
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 1068  1           Pattern 9       0000      000000       0 000 000 000 000 000
; C 1069  1                           0000      000000       0 000 000 000 000 000
; C 1070  1                           0000      000000       0 000 000 000 000 000
; C 1071  1                           FFFF      177777       1 111 111 111 111 111
; C 1072  1                           FFFF      177777       1 111 111 111 111 111
; C 1073  1                           FFFF      177777       1 111 111 111 111 111
; C 1074  1                           0000      000000       0 000 000 000 000 000
; C 1075  1                           0000      000000       0 000 000 000 000 000
; C 1076  1                           FFFF      177777       1 111 111 111 111 111
; C 1077  1                           FFFF      177777       1 111 111 111 111 111
; C 1078  1                           0000      000000       0 000 000 000 000 000
; C 1079  1                           FFFF      177777       1 111 111 111 111 111
; C 1080  1                           0000      000000       0 000 000 000 000 000
; C 1081  1                           FFFF      177777       1 111 111 111 111 111
; C 1082  1                           0000      000000       0 000 000 000 000 000
; C 1083  1                           FFFF      177777       1 111 111 111 111 111
; C 1084  1
; C 1085  1           Pattern 10      B6D9      133331       1 011 011 011 011 001
; C 1086  1
; C 1087  1           Pattern 11      5555      052525       0 101 010 101 010 101
; C 1088  1                           5555      052525       0 101 010 101 010 101
; C 1089  1                           5555      052525       0 101 010 101 010 101
; C 1090  1                           AAAA      125252       1 010 101 010 101 010
; C 1091  1                           AAAA      125252       1 010 101 010 101 010
; C 1092  1                           AAAA      125252       1 010 101 010 101 010
; C 1093  1                           5555      052525       0 101 010 101 010 101
; C 1094  1                           5555      052525       0 101 010 101 010 101
; C 1095  1                           AAAA      125252       1 010 101 010 101 010
; C 1096  1                           AAAA      125252       1 010 101 010 101 010
; C 1097  1                           5555      052525       0 101 010 101 010 101
; C 1098  1                           AAAA      125252       1 010 101 010 101 010
; C 1099  1                           5555      052525       0 101 010 101 010 101
; C 1100  1                           AAAA      125252       1 010 101 010 101 010
; C 1101  1                           5555      052525       0 101 010 101 010 101
; C 1102  1                           AAAA      125252       1 010 101 010 101 010
```

```
; C 1103  1     Pattern 12    2D2D    026455     0 010 110 100 101 101
; C 1104  1                   2D2D    026455     0 010 110 100 101 101
; C 1105  1                   2D2D    026455     0 010 110 100 101 101
; C 1106  1                   D2D2    151322     1 101 001 011 010 010
; C 1107  1                   D2D2    151322     1 101 001 011 010 010
; C 1108  1                   D2D2    151322     1 101 001 011 010 010
; C 1109  1                   2D2D    026455     0 010 110 100 101 101
; C 1110  1                   2D2D    026455     0 010 110 100 101 101
; C 1111  1                   D2D2    151322     1 101 001 011 010 010
; C 1112  1                   D2D2    151322     1 101 001 011 010 010
; C 1113  1                   2D2D    026455     0 010 110 100 101 101
; C 1114  1                   2D2D    026455     0 010 110 100 101 101
; C 1115  1                   D2D2    151322     1 101 001 011 010 010
; C 1116  1                   2D2D    026455     0 010 110 100 101 101
; C 1117  1                   D2D2    151322     1 101 001 011 010 010
; C 1118  1                   2D2D    026455     0 010 110 100 101 101
; C 1119  1                   D2D2    151322     1 101 001 011 010 010
; C 1120  1                   2D2D    026455     0 010 110 100 101 101
; C 1121  1                   D2D2    151322     1 101 001 011 010 010
; C 1122  1                   2D2D    026455     0 010 110 100 101 101
; C 1123  1
; C 1124  1     Pattern 13    6DB6    066666     0 110 110 110 110 110
; C 1125  1
; C 1126  1     Pattern 14    0001    000001     0 000 000 000 000 001
; C 1127  1                   0002    000002     0 000 000 000 000 010
; C 1128  1                   0004    000004     0 000 000 000 000 100
; C 1129  1                   0008    000010     0 000 000 000 001 000
; C 1130  1                   0010    000020     0 000 000 000 010 000
; C 1131  1                   0020    000040     0 000 000 000 100 000
; C 1132  1                   0040    000100     0 000 000 001 000 000
; C 1133  1                   0080    000200     0 000 000 010 000 000
; C 1134  1                   0100    000400     0 000 000 100 000 000
; C 1135  1                   0200    001000     0 000 001 000 000 000
; C 1136  1                   0400    002000     0 000 010 000 000 000
; C 1137  1                   0800    004000     0 000 100 000 000 000
; C 1138  1                   1000    010000     0 001 000 000 000 000
; C 1139  1                   2000    020000     0 010 000 000 000 000
; C 1140  1                   4000    040000     0 100 000 000 000 000
; C 1141  1                   8000    100000     1 000 000 000 000 000
```

```
; C 1142  1        Pattern 15      FFFE     177776        1 111 111 111 111 110
; C 1143  1                        FFFD     177775        1 111 111 111 111 101
; C 1144  1                        FFFB     177773        1 111 111 111 111 011
; C 1145  1                        FFF7     177767        1 111 111 111 110 111
; C 1146  1                        FFEF     177757        1 111 111 111 101 111
; C 1147  1                        FFDF     177737        1 111 111 111 011 111
; C 1148  1                        FFBF     177677        1 111 111 110 111 111
; C 1149  1                        FF7F     177577        1 111 111 101 111 111
; C 1150  1                        FEFF     177377        1 111 111 011 111 111
; C 1151  1                        FDFF     176777        1 111 110 111 111 111
; C 1152  1                        FBFF     175777        1 111 101 111 111 111
; C 1153  1                        F7FF     173777        1 111 011 111 111 111
; C 1154  1                        EFFF     167777        1 110 111 111 111 111
; C 1155  1                        DFFF     157777        1 101 111 111 111 111
; C 1156  1                        BFFF     137777        1 011 111 111 111 111
; C 1157  1                        7FFF     077777        0 111 111 111 111 111
; C 1158  1
; C 1159  1        Pattern 16      B6D9     133331        1 011 011 011 011 001
; C 1160  1                        B6D9     133331        1 011 011 011 011 001
; C 1161  1                        B6D9     133331        1 011 011 011 011 001
; C 1162  1                        DB6C     155554        1 101 101 101 101 100
; C 1163  1                        DB6C     155554        1 101 101 101 101 100
; C 1164  1                        DB6C     155554        1 101 101 101 101 100
; C 1165  1                        B6D9     133331        1 011 011 011 011 001
; C 1166  1                        B6D9     133331        1 011 011 011 011 001
; C 1167  1                        DB6C     155554        1 101 101 101 101 100
; C 1168  1                        DB6C     155554        1 101 101 101 101 100
; C 1169  1                        B6D9     133331        1 011 011 011 011 001
; C 1170  1                        DB6C     155554        1 101 101 101 101 100
; C 1171  1                        B6D9     133331        1 011 011 011 011 001
; C 1172  1                        DB6C     155554        1 101 101 101 101 100
; C 1173  1                        B6D9     133331        1 011 011 011 011 001
; C 1174  1                        DB6C     155554        1 101 101 101 101 100
```

```
; C   1          Pattern 17      (LBN)*     (LBN)                (LBN)
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1
; C   1                   * This word position contains the number of the locical
; C   1                     block to be written.
; C   1
; C   1
; C   1
; C   1          Pattern 18      8D36       106466       1 000 110 100 110 110
; C   1                          (LBN)      (LBN)               (LBN)
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          8D36       106466       1 000 110 100 110 110
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
; C   1                          72C9       071311       0 111 001 011 001 001
```

```
; C   1      Pattern 19     (LBN)    (LBN)               (LBN)
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1
; C   1      Pattern 20     B999     134631       1 011 100 110 011 001
; C   1                     (LBN)    (LBN)               (LBN)
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     B999     134631       1 011 100 110 011 001
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1                     4666     043146       0 100 011 001 100 110
; C   1
; C   1      Pattern 21     (LBN)    (LBN)               (LBN)
; C   1
;     1   )%
```

```
%sbttl 'PROGRAM HEADER'

library 'ZRQAFO.L16';                         ! RDRX EXERCISER GLOBAL LIBRARY

require 'BLSMAC.REQ';                         ! DIAGNOSTIC SUPERVISOR LIBRARY

literal
    DS$NBR_OF_TESTS = 1;                      ! NUMBER OF TESTS IN THIS DIAGNOSTIC

EQUALS;

POINTER (ALL);

!+
! THE PROGRAM HEADER IS THE INTERFACE BETWEEN
! THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
!-

!ZZZ HEADER (%ascii'ZRQA', %ascii'F', %ascii'0', 32767, 1, PRIOO);
HEADER (%ascii'ZRQA', %ascii'F', %ascii'0', 32000, 1, PRIOO);        !ZZZ NEED POSITIVE NUMBER
```

```
%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

    DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),        !BUFFER FOR DUP  ZZZ
                        !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)
    CLK_PRESENT : byte,                                  ! FLAG INDICATES IF LINE-CLOCK PRESENT
    HOE_FLAG : byte,                                     ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET

    S_PATTERN : WORD,                 !PATTERN FOR DUP WRITES              ZZZ
    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
    FER_LBN : word,                                      ! LBN OF THE "FORCED ERROR" BLOCK
    FER_BC : word,                                       ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
    INIT_OCCURED : byte initial (byte (FALSE)),          ! EXERCISER INITIALIZATION COMPLETE
    ADDR_VECT_OK : byte initial (byte (FALSE));          ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED

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
!
    HWQ1 = uplit (%asciz'IP address'),
    HWQ2 = uplit (%asciz'Vector'),
    HWQ3 = uplit (%asciz'BR Level [usually 4-RQDX1 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'),
    HWQ10 = uplit (%asciz'Also run DUP exerciser'),                      !ZZZ
    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
    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
    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)'),
!
! 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   READS      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'),
    EGD_14 = uplit (%asciz'FATAL CONTROLLER ERROR'),
    EGD_15 = uplit (%asciz'ONLINE FAILED'),
    EGD_16 = uplit (%asciz'WRITE-PROTECT CONFLICT'),
    EGD_17 = uplit (%asciz'ACCESS FAILED'),
    EGD_18 = uplit (%asciz'FATAL I/O ERROR'),
!   EGD_19 = uplit (%asciz'CONTROLLER TIMEOUT'),
    EGD_19 = uplit (%asciz'DISK TYPE UNKNOWN TO EXERCISER'),
    EGD_20 = uplit (%asciz'FAILED TO SEND SET-CONTROLLER-CHARACTERISTICS COMMAND'),
    EGD_21 = uplit (%asciz'SET-CONTROLLER-CHARACTERISTICS RESPONSE HAS BAD ENDCODE OR FLAGS IN ERROR'),
    EGD_22 = uplit (%asciz'FAILED TO SEND ON-LINE COMMAND'),
    EGD_23 = uplit (%asciz'ON-LINE RESPONSE HAS BAD ENDCODE'),
    EGD_24 = uplit (%asciz'ON-LINE RESPONSE HAS UNKNOWN DEVICE'),
!
!       HARD or SOFT (ERRHRD or ERRSOFT)
!
    EGH_30 = uplit (%asciz'I/O REQUEST FAILED'),
```

```
!
! BASIC ERROR MESSAGES (PRINTB)
!
!       SYSTEM FATAL (ERRSF)
!
    EBS_01 = uplit (%asciz'%AMORE THAN %D2%A UNITS SPECIFIED'),
!
!       DRIVE FATAL (ERRDF)
!
    EBD_10 = uplit (%asciz'%A* NO RESPONSE AT ADDRESS %O6'),
    EBD_12 = uplit (%asciz'%A* INCORRECT BR LEVEL FOR DRIVE %O6'),
    EBD_13 = uplit (%asciz'%A* STEP %D1%A READ ERROR'),
    EBD_14 = uplit (%asciz'%A* BAD SA CODE FROM DRIVE %O6'),
    EBD_18 = uplit (%asciz'%A* DISK%D2%A WENT OFFLINE'),
    EBD_19 = uplit (%asciz'%A* DRIVE %O6%A NOT PROCESSING COMMAND PACKETS'),
    EBD_24 = uplit (%asciz'%A* DISK%D2%A WENT TO THE "AVAILABLE" STATE'),
!
!
!       HARD or SOFT (ERRHRD or ERRSOFT)
!
EH_0 = UPLIT (%ASCIZ' - UNRECOGNIZED MESSAGE TYPE'),                 !ZZZ
EH_1 = UPLIT (%ASCIZ' - UNRECOGNIZED CONNECTION ID'),                !ZZZ
EH_2 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN MESSAGE'),               !ZZZ
EH_3 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN PACKET'),                !ZZZ
EH_4 = UPLIT (%ASCIZ' - UNRECOGNIZED CRN'),                          !ZZZ
EH_5 = UPLIT (%ASCIZ' - UNRECOGNIZED OPCODE'),                       !ZZZ
EH_6 = UPLIT (%ASCIZ' - MSCP STATUS CODE ERR'),                      !ZZZ
EH_7 = UPLIT (%ASCIZ' - DUP STATUS CODE ERR'),                       !ZZZ
EH_8 = UPLIT (%ASCIZ' - UNRECOGNIZED STATUS CODE'),                  !ZZZ
EH_9 = UPLIT (%ASCIZ' - LBN HOST COMPARE ERR'),                      !ZZZ
EH_10 = UPLIT (%ASCIZ' - DBN HOST COMPARE ERR'),                     !ZZZ
EH_12 = UPLIT (%ASCIZ' - UNABLE TO LOAD DUP MEDIA'),                 !ZZZ
EH_13 = UPLIT (%ASCIZ' - ERR IN DUP PKT WHEN USING CTLR LC PROG'),   !ZZZ
!
    ERR_00  = uplit (%asciz'%A* DISK%D2'),
    ERR_COD = uplit (
              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')) : vector [14],
!
!       ERROR LOG MESSAGE (ERRSOFT)
!
```

```
;    1        ELG_00  = uplit (%asciz'%AERROR LOG MESSAGE RECEIVED:%N'),
;    1        ELG_FMT = uplit (
;    1                  uplit (%asciz'%A* CONTROLLER ERROR%N'),
;    1                  uplit (%asciz'%A* HOST MEMORY ACCESS ERROR%N'),
;    1                  uplit (%asciz'%A* DISK%D2%A - DISK TRANSFER ERROR%N'),
;    1                  uplit (%asciz'%A* DISK%D2%A - "STANDARD DISK INTERCONNECT" ERROR%N'),
;    1                  uplit (%asciz'%A* DISK%D2%A - "SMALL DISK" ERROR%N')) : vector [5],
;    1    !
;    1    ! EXTENDED ERROR MESSAGES (PRINTX)
;    1    !
;    1        EX_SA  = uplit (%asciz'%N%A* SA: %O6'),
;    1        EX_SC  = uplit (%asciz'%N%A* STATUS CODE: %O2'),
;    1        EX_SBO = uplit (%asciz'%O4'),
;    1        EX_SB  = uplit (%asciz'%N%A* SUB_CODE: '),
;    1        EX_CMD = uplit (%asciz'%N%A* COMMAND: '),
;    1        EX_RD  = uplit (%asciz'%AREAD'),
;    1        EX_WRT = uplit (%asciz'%AWRITE'),
;    1        EX_CMP = uplit (%asciz'%A-COMPARE'),
;    1        EX_ONL = uplit (%asciz'%AONLINE'),
;    1        EX_ACC = uplit (%asciz'%AACCESS'),
;    1        EX_OP  = uplit (%asciz'%O3'),
;    1        EX_BB  = uplit (%asciz'%N%A* BAD BLOCK (Host replaceable): %D5%A. (OCT %O6%A)'),
;    1        EX_BB1 = uplit (%asciz'%N%A* 1st BAD BLOCK (Host replaceable): %D5%A. (OCT %O6%A)'),
;    1        EX_BBU = uplit (%asciz'%N%A* BAD BLOCK REPORTED (Replaced): %D%A. (OCT %O6%A)'),
;    1        EX_LBN = uplit (%asciz'%N%A* LBN: %D5%A. (OCT %O6%A)'),
;    1        EX_PBN = uplit (%asciz'%N%A* PBN: %D5%A. (OCT %O6%A)'),
;    1        EX_LBR = uplit (%asciz'%N%A* LBN: (READ) %D5%A. (OCT %O6%A)'),
;    1        EX_LBW = uplit (%asciz'%N%A* LBN: (WRITE) %D5%A. (OCT %O6%A)'),
;    1        EX_RBN = uplit (%asciz'%N%A* REPLACEMENT BLOCK NO. %D5%A. (OCT %O6%A)'),
;    1        EX_CBC = uplit (%asciz'%N%A* BYTE COUNT IN COMMAND: %D5%A.'),
;    1        EX_CBR = uplit (%asciz'%N%A* BYTE COUNT IN READ COMMAND: %D5%A.'),
;    1        EX_CBW = uplit (%asciz'%N%A* BYTE COUNT IN WRITE COMMAND: %D5%A.'),
;    1        EX_BC  = uplit (%asciz'%N%A* ACTUAL # OF BYTES TRANSFERRED: %D5%A.'),
;    1        EX_BD  = uplit (%asciz'%N%A* I/O BUFFER ADDRESS (32 bits): %O6%A %O6'),
;    1        EX_BDR = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR READ (32 bits): %O6%A %O6'),
;    1        EX_BDW = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR WRITE (32 bits): %O6%A %O6'),
;    1        EX_RP  = uplit (%asciz'%N%ACONTENTS OF COMMAND/RESPONSE PACKET SAVE AREA:%N'),
;    1        EX_WRD = uplit (%asciz'%A %O6'),
;    1        EX_TIM = uplit (%asciz'%N%A* TIME: %Z2%A:%Z2%A HOURS%N'),
;    1
;    1
;    1        XX13 = UPLIT (%ASCIZ'%N%A * DISK : %D2'),                            !ZZZ
;    1        XX23 = UPLIT (%ASCIZ'%N%ADBN: %D5%A. (OCT %O6%A)'),                  !ZZZ
;    1        XX32 = UPLIT (%ASCIZ'%N%ABYTE NUMBER: %D3'),                         !ZZZ
;    1        XX33 = UPLIT (%ASCIZ'%N%ARANDOM WRITTEN WORD :%B16'),                !ZZZ
;    1        XX34 = UPLIT (%ASCIZ'%N%ARANDOM READ WORD bin:%B16%A oct:%O6'),      !ZZZ
;    1
;    1    !
;    1    ! CONFIGURATION ERROR MESSAGES (PRINTF)
;    1    !
;    1        CER_01 = uplit (%asciz'%N%ADUPLICATE UNIT:%D2%A AT IP: %O6'),
;    1        CER_02 = uplit (%asciz'%N%AMORE THAN %D1%A DIFFERENT IP ADDRESSES'),
;    1    !
```

```
! ERROR/EVENT SUB CODES (PRINTX)
!
    SC_SDI = uplit (%ascic'%ASPIN-DOWN IGNORED'),
    SC_CON = uplit (%ascic'%ASTILL CONNECTED'),
    SC_DUP = uplit (%ascic'%ADUPLICATE UNIT NUMBER'),
    SC_ONL = uplit (%ascic'%AALREADY ONLINE'),
    SC_SON = uplit (%ascic'%ASTILL ONLINE'),
    SC_UNK = uplit (%ascic'%AUNIT UNKNOWN OR ONLINE TO ANOTHER CONTROLLER'),
    SC_VOL = uplit (%ascic'%ANO VOLUME MOUNTED OR DRIVE DISABLED BY SWITCH'),
    SC_IOP = uplit (%ascic'%AUNIT INOPERATIVE (RD51/52 write fault)'),
    SC_DIS = uplit (%ascic'%AUNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTICS'),
    SC_FER = uplit (%ascic'%A"FORCED ERROR" DETECTED WHILE ACCESSING FCT OR RCT'),
    SC_FE2 = uplit (%ascic'%ASECTOR HAD BEEN WRITTEN WITH "FORCED ERROR" MODIFIER'),
    SC_ISH = uplit (%ascic'%AFCT OR RCT UNREADABLE - INVALID SECTOR HEADER'),
    SC_IS2 = uplit (%ascic'%AHEADER COMPARE ERROR (Valid header not found)'),
    SC_DST = uplit (%ascic'%AFCT OR RCT UNREADABLE - DATA SYNC TIMEOUT'),
    SC_DS2 = uplit (%ascic'%ADATA SYNC NOT FOUND (Data sync timeout)'),
    SC_ECC = uplit (%ascic'%AFCT OR RCT UNREADABLE - UNCORRECTABLE ECC ERROR'),
    SC_ECD = uplit (%ascic'%AUNCORRECTABLE ECC ERROR'),
    SC_RCT = uplit (%ascic'%ARCT CORRUPTED'),
    SC_FUL = uplit (%ascic'%ANO REPLACEMENT BLOCK AVAILABLE (RCT full)'),
    SC_576 = uplit (%ascic'%ADISK NOT FORMATTED WITH 512 BYTE SECTORS'),
    SC_FCT = uplit (%ascic'%ADISK NOT FORMATTED OR FCT CORRUPTED'),
    SC_EC1 = uplit (%ascic'%AONE SYMBOL ECC ERROR'),
    SC_EC2 = uplit (%ascic'%ATWO SYMBOL ECC ERROR'),
    SC_EC3 = uplit (%ascic'%ATHREE SYMBOL ECC ERROR'),
    SC_EC4 = uplit (%ascic'%AFOUR SYMBOL ECC ERROR'),
    SC_EC5 = uplit (%ascic'%AFIVE SYMBOL ECC ERROR'),
    SC_EC6 = uplit (%ascic'%ASIX SYMBOL ECC ERROR'),
    SC_EC7 = uplit (%ascic'%ASEVEN SYMBOL ECC ERROR'),
    SC_EC8 = uplit (%ascic'%AEIGHT SYMBOL ECC ERROR'),
    SC_EC9 = uplit (%ascic'%ACORRECTABLE ERROR IN ECC FIELD'),
    SC_SWP = uplit (%ascic'%AUNIT SOFTWARE WRITE PROTECTED'),
    SC_HWP = uplit (%ascic'%AUNIT HARDWARE WRITE PROTECTED'),
    SC_ODA = uplit (%ascic'%AODD TRANSFER ADDRESS'),
    SC_ODB = uplit (%ascic'%AODD BYTE COUNT'),
    SC_NXM = uplit (%ascic'%ANON-EXISTENT HOST MEMORY'),
    SC_PAR = uplit (%ascic'%AHOST MEMORY PARITY ERROR'),
    SC_CTO = uplit (%ascic'%ACOMMAND TIMOUT OR RETRY LIMIT EXCEEDED'),
    SC_SDS = uplit (%ascic'%ASERIALIZER/DESERIALIZER OVERRUN OR UNDERRUN'),
    SC_EDC = uplit (%ascic'%A"ERROR DETECTION CODE" ERROR'),
    SC_IDS = uplit (%ascic'%AINCONSISTENT INTERNAL DATA STRUCTURE'),
    SC_SRT = uplit (%ascic'%ADRIVE COMMAND TIMEOUT (No response or seek incomplete)'),
    SC_SRI = uplit (%ascic'%ACONTROLLER DETECTED TRANSMISSION OR PROTOCOL ERROR'),
    SC_POE = uplit (%ascic'%APOSITION ERROR (Mis-seek)'),
    SC_RDY = uplit (%ascic'%ALOST READ/WRITE READY DURING/BETWEEN TRANSFERS'),
    SC_CLK = uplit (%ascic'%ADRIVE CLOCK DROPOUT'),
    SC_RSP = uplit (%ascic'%ALOST RECEIVER READY BETWEEN SECTORS'),
    SC_SUR = uplit (%ascic'%ADRIVE DETECTED ERROR'),
    SC_PSP = uplit (%ascic'%ACONTROLLER DETECTED PULSE OR STATE PARITY ERROR'),
!
! 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'),
		uplit (%asciz'%ACONTROLLER RAM ERROR (Non-parity)'),
		uplit (%asciz'%AINIT SEQUENCE ERROR'),
		uplit (%asciz'%AHIGHER LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
		uplit (%asciz'%APURGE/POLL HARDWARE FAILURE'),
		uplit (%asciz'%AMAPPING REGISTER READ FAILURE (Parity or timeout)')) : vector [23],
!
! RD/RX CONTROLLER DEPENDENT ERRORS CODES
!
	RDRX_ERR = uplit (
		uplit (%asciz'%AT11 CPU FAILURE'),
		uplit (%asciz'%ANON-PARITY RAM ERROR'),
		uplit (%asciz'%ASTATE MACHINE FAILURE - T11 ADDRESS REGISTER'),
		uplit (%asciz'%ASTATE MACHINE FAILURE - Q-BUS ADDRESS REGISTER'),
		uplit (%asciz'%ASTATE MACHINE FAILURE - CRC REGISTER'),
		uplit (%asciz'%ASTATE MACHINE FAILURE - SERIALIZER/DESERIALIZER REGISTER'),
		uplit (%asciz'%ASTATE MACHINE FAILURE - WRONG HARDWARE VERSION')) : vector [7],
!
! PRINTOUTS THAT FAKE THE DRS ERROR MESSAGES
!
	DF_MSG  = uplit (%asciz'%N%AZRQA DEV FTL  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
	HRD_MSG = uplit (%asciz'%N%AZRQA HRD ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
	SFT_MSG = uplit (%asciz'%N%AZRQA SFT ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06%N'),
	HRD_SUB = uplit (%asciz'%N%AI/O REQUEST FAILED%N'),


!
! MISCELLANEOUS
!
	SPACE4   = uplit (%asciz'%S4'),
	CRLF     = uplit (%asciz'%N'),
	DASH     = uplit (%asciz'%A - '),
	ASTERISK = uplit (%asciz'%A* ');
```

```
;  1  %sbttl 'DEFAULT HARDWARE P-TABLE'
;  1
;  1  !+
;  1  ! THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
;  1  ! THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
;  1  ! IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
;  1  ! AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
;  1  !-
;  1
;  1  BGNHW (DFPTBL);
;  1
;  1  global
;  1      HWPT_IP_ADDR : word initial (INIT_IP_ADDR),          ! IP ADDRESS
;  1      HWPT_VECTOR : word initial (INIT_INTR_VECT),         ! VECTOR ADDRESS
;  1      HWPT_BR_LEVEL : word initial (INIT_BR_LEVEL),        ! BR LEVEL
;  1      HWPT_DISK : WORD INITIAL (%o'000200'),               !PROTECT, WHOLE DISK, NO DUP   ZZZ
;  1                                                           ! DK 0        ZZZ
;  1      HWPTS0_LBN : word initial (0),                       ! STARTING TRACK LO ZZZ
;  1      HWPTS1_LBN : word initial (0),                       ! STARTING TRACK HI ZZZ
;  1      HWPTE0_LBN : word initial (%o'177777'),              ! ENDING TRACK LO   ZZZ
;  1      HWPTE1_LBN : word initial (0),                       ! ENDING TRACK HI   ZZZ
;  1      NAME_LO    : WORD INITIAL (%o'020040'),              !DISK TYPE          ZZZ
;  1      NAME_HI    : WORD INITIAL (%o'020040');              !DISK TYPE          ZZZ
;  1
;  1  ENDHW;
```

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

```
;        1      %sbttl 'PROTECTION TABLE'
;        1
;        1      !+
;        1      ! THIS TABLE IS USED BY THE RUNTIME SERVICES
;        1      ! TO PROTECT THE LOAD MEDIA.
;        1      !-
;        1
;        1      BGNPROT (0, -1, 6);
;        1
;        1      !1ST ARG =      OFFSET INTO P-TABLE FOR CSR ADDRESS
;        1      !2ND ARG =      OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;        1      !3RD ARG =      OFFSET INTO P-TABLE FOR DRIVE NUMBER
;        1
;        1      ENDPROT;
;        1      end
;        1
;        0      eludom
```

```
                                           .TITLE  ZRQAM1 RD/RX EXERCISER
                                           .IDENT  /V02.1/
                                           .ENABL  AMA

000000                                     .PSECT  $CODE$,  RO
000000     132     122     121    L$NAME::.ASCII  /ZRQ/
000003     101                            .ASCII  /A/
000004     000                            .BYTE   0
000005     000                            .BYTE   0
000006     000                            .BYTE   0
000007     000                            .BYTE   0
000010                            L$REV::
000010     106                            .ASCII  /F/
000011     060                            .ASCII  /0/
000012  000000G                   L$UNIT::.WORD   T$PTHV
000014  076400                    L$TIML::.WORD   76400
000016  000000G                   L$HPCP::.WORD   L$HARD
000020  000000G                   L$SPCP::.WORD   L$SOFT
000022  022564'                   L$HPTP::.WORD   L$HW
000024  022614'                   L$SPTP::.WORD   L$SW
000026  000000G                   L$LADP::.WORD   L$LAST
000030  000000                    L$STA:: .WORD   0
000032  000000                    L$CO::  .WORD   0
000034  000001                    L$DTYP::.WORD   1
000036  000000                    L$APT:: .WORD   0
000040  000124'                   L$DTP:: .WORD   L$DISPATCH
000042  000000                    L$PRIO::.WORD   0
000044  000000                    L$ENVI::.WORD   0
000046  000000                    L$EXP1::.WORD   0
000050                            L$MREV::
000050     003                            .BYTE   3
000051     003                            .BYTE   3
000052  000000                    L$EF::  .WORD   0
```

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

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

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

```
000740   101   154   163     P.AAL:  .ASCII  /Als/
000743   157   040   162             .ASCII  /o r/
000746   165   156   040             .ASCII  /un /
000751   104   125   120             .ASCII  /DUP/
000754   040   145   170             .ASCII  ' ex/
000757   145   162   143             .ASCII  /erc/
000762   151   163   145             .ASCII  /ise/
000765   162   000   000             .ASCII  /r/<00><00>
000770   127   162   151     P.AAM:  .ASCII  /Wri/
000773   164   145   040             .ASCII  /te /
000776   157   156   040             .ASCII  /on /
001001   144   151   141             .ASCII  /dia/
001004   147   156   157             .ASCII  /gno/
001007   163   164   151             .ASCII  /sti/
001012   143   040   141             .ASCII  /c a/
001015   162   145   141             .ASCII  /rea/
001020   000   000                   .ASCII  <00><00>
001022   110   141   162     P.AAN:  .ASCII  /Har/
001025   144   040   145             .ASCII  /d e/
001030   162   162   157             .ASCII  /rro/
001033   162   040   154             .ASCII  /r l/
001036   151   155   151             .ASCII  /imi/
001041   164   000   000             .ASCII  /t/<00><00>
001044   124   162   141     P.AAO:  .ASCII  /Tra/
001047   156   163   146             .ASCII  /nsf/
001052   145   162   040             .ASCII  /er /
001055   154   151   155             .ASCII  /lim/
001060   151   164   040             .ASCII  /it /
001063   151   156   040             .ASCII  /in /
001066   155   145   147             .ASCII  /meg/
001071   141   142   171             .ASCII  /aby/
001074   164   145   163             .ASCII  /tes/
001077   040   050   060             .ASCII  / (0/
001102   040   146   157             .ASCII  / fo/
001105   162   040   161             .ASCII  /r q/
001110   165   151   143             .ASCII  /uic/
001113   153   040   160             .ASCII  /k p/
001116   141   163   163             .ASCII  /ass/
001121   051   000   000             .ASCII  /)/<00><00>
001124   122   141   156     P.AAP:  .ASCII  /Ran/
001127   144   157   155             .ASCII  /dom/
001132   040   163   145             .ASCII  / se/
001135   145   153   040             .ASCII  /ek /
001140   155   157   144             .ASCII  /mod/
001143   145   000   000             .ASCII  /e/<00><00>
001146   122   145   141     P.AAQ:  .ASCII  /Rea/
001151   144   055   143             .ASCII  /d-c/
001154   157   155   160             .ASCII  /omp/
001157   141   162   145             .ASCII  /are/
001162   163   040   160             .ASCII  /s p/
001165   145   162   146             .ASCII  /erf/
001170   157   162   155             .ASCII  /orm/
001173   145   144   040             .ASCII  /ed /
```

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

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

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

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

```
002356  163  040  110          .ASCII  /s H/
002361  110  115  115          .ASCII  /HMM/
002364  040  050  145          .ASCII  / (e/
002367  170  141  155          .ASCII  /xam/
002372  160  154  145          .ASCII  /ple/
002375  072  040  061          .ASCII  /: 1/
002400  063  060  065          .ASCII  /305/
002403  051  000  000          .ASCII  /)/<00><00>
002406  103  157  165  P.ABF:  .ASCII  /Cou/
002411  156  164  040          .ASCII  /nt /
002414  145  141  143          .ASCII  /eac/
002417  150  040  162          .ASCII  /h r/
002422  145  164  162          .ASCII  /etr/
002425  171  040  141          .ASCII  /y a/
002430  163  040  141          .ASCII  /s a/
002433  040  163  145          .ASCII  / se/
002436  160  145  162          .ASCII  /per/
002441  141  164  145          .ASCII  /ate/
002444  040  163  157          .ASCII  / so/
002447  146  164  040          .ASCII  /ft /
002452  145  162  162          .ASCII  /err/
002455  157  162  000          .ASCII  /or/<00>
002460  122  165  156  P.ABG:  .ASCII  /Run/
002463  156  151  156          .ASCII  /nin/
002466  147  040  165          .ASCII  /g u/
002471  156  144  145          .ASCII  /nde/
002474  162  040  164          .ASCII  /r t/
002477  150  145  040          .ASCII  /he /
002502  101  056  120          .ASCII  /A.P/
002505  056  124  056          .ASCII  /.T./
002510  040  115  157          .ASCII  / Mo/
002513  156  151  164          .ASCII  /nit/
002516  157  162  000          .ASCII  /or/<00>
002521  000                    .ASCII  <00>
002522  124  150  145  P.ABH:  .ASCII  /The/
002525  040  162  145          .ASCII  / re/
002530  155  141  151          .ASCII  /mai/
002533  156  151  156          .ASCII  /nin/
002536  147  040  161          .ASCII  /g q/
002541  165  145  163          .ASCII  /ues/
002544  164  151  157          .ASCII  /tio/
002547  156  163  040          .ASCII  /ns /
002552  157  156  154          .ASCII  /onl/
002555  171  040  141          .ASCII  /y a/
002560  160  160  154          .ASCII  /ppl/
002563  171  040  164          .ASCII  /y t/
002566  157  040  165          .ASCII  /o u/
002571  156  160  162          .ASCII  /npr/
002574  157  164  145          .ASCII  /ote/
002577  143  164  145          .ASCII  /cte/
002602  144  040  144          .ASCII  /d d/
002605  151  163  153          .ASCII  /isk/
002610  040  165  156          .ASCII  / un/
```

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

```
003045  122  072  040          .ASCII  /R: /
003050  103  141  156          .ASCII  /Can/
003053  047  164  040          .ASCII  /'t /
003056  146  151  156          .ASCII  /fin/
003061  144  040  144          .ASCII  /d d/
003064  151  163  153          .ASCII  /isk/
003067  040  045  104          .ASCII  / %D/
003072  063  045  101          .ASCII  /3%A/
003075  040  151  156          .ASCII  / in/
003100  040  103  123          .ASCII  / CS/
003103  124  040  045          .ASCII  /T %/
003106  104  061  000          .ASCII  /D1/<00>
003111  000                    .ASCII  <00>
003112  045  116  045  P.ABO:  .ASCII  /%N%/
003115  101  052  052          .ASCII  /A**/
003120  040  102  141          .ASCII  / Ba/
003123  144  040  143          .ASCII  /d c/
003126  157  156  156          .ASCII  /onn/
003131  040  111  104          .ASCII  / ID/
003134  040  045  117          .ASCII  / %O/
003137  066  045  101          .ASCII  /6%A/
003142  040  162  145          .ASCII  / re/
003145  143  145  151          .ASCII  /cei/
003150  166  145  144          .ASCII  /ved/
003153  040  146  162          .ASCII  / fr/
003156  157  155  040          .ASCII  /om /
003161  045  117  066          .ASCII  /%O6/
003164  000  000               .ASCII  <00><00>
003166  045  116  045  P.ABP:  .ASCII  /%N%/
003171  101  052  052          .ASCII  /A**/
003174  040  115  145          .ASCII  / Me/
003177  163  163  141          .ASCII  /ssa/
003202  147  145  040          .ASCII  /ge /
003205  164  171  160          .ASCII  /typ/
003210  145  040  045          .ASCII  /e %/
003213  117  062  045          .ASCII  /O2%/
003216  101  040  162          .ASCII  /A r/
003221  145  143  145          .ASCII  /ece/
003224  151  166  145          .ASCII  /ive/
003227  144  040  151          .ASCII  /d i/
003232  156  040  115          .ASCII  /n M/
003235  123  103  120          .ASCII  /SCP/
003240  040  160  141          .ASCII  / pa/
003243  143  153  145          .ASCII  /cke/
003246  164  000               .ASCII  /t/<00>
003250  045  116  045  P.ABQ:  .ASCII  /%N%/
003253  101  052  052          .ASCII  /A**/
003256  040  123  105          .ASCII  / SE/
003261  121  125  105          .ASCII  /QUE/
003264  116  072  040          .ASCII  /N: /
003267  122  105  124          .ASCII  /RET/
003272  120  113  124          .ASCII  /PKT/
003275  040  156  157          .ASCII  / no/
```

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

J6


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

K6


```
003770  045  117  066           .ASCII  /%O6/
003773  000                     .ASCII  <00>
003774  045  116  045   P.ACA:  .ASCII  /%N%/
003777  101  052  052           .ASCII  /A**/
004002  040  122  145           .ASCII  / Re/
004005  155  157  166           .ASCII  /mov/
004010  141  142  154           .ASCII  /abl/
004013  145  040  144           .ASCII  /e d/
004016  151  163  153           .ASCII  /isk/
004021  040  151  163           .ASCII  / is/
004024  040  163  145           .ASCII  / se/
004027  154  145  143           .ASCII  /lec/
004032  164  145  144           .ASCII  /ted/
004035  000                     .ASCII  <00>
004036  045  116  045   P.ACB:  .ASCII  /%N%/
004041  101  052  052           .ASCII  /A**/
004044  040  106  151           .ASCII  / Fi/
004047  170  145  144           .ASCII  /xed/
004052  040  144  151           .ASCII  / di/
004055  163  153  040           .ASCII  /sk /
004060  151  163  040           .ASCII  /is /
004063  163  145  154           .ASCII  /sel/
004066  145  143  164           .ASCII  /ect/
004071  145  144  000           .ASCII  /ed/<00>
004074  045  116  045   P.ACC:  .ASCII  /%N%/
004077  101  052  052           .ASCII  /A**/
004102  040  111  154           .ASCII  / Il/
004105  154  145  147           .ASCII  /leg/
004110  141  154  040           .ASCII  /al /
004113  146  165  156           .ASCII  /fun/
004116  143  164  151           .ASCII  /cti/
004121  157  156  072           .ASCII  /on:/
004124  040  045  117           .ASCII  / %O/
004127  066  000  000           .ASCII  /6/<00><00>
004132  045  116  045   P.ACD:  .ASCII  /%N%/
004135  101  052  052           .ASCII  /A**/
004140  040  103  157           .ASCII  / Co/
004143  155  155  141           .ASCII  /mma/
004146  156  144  040           .ASCII  /nd /
004151  162  145  146           .ASCII  /ref/
004154  040  043  040           .ASCII  / # /
004157  045  117  066           .ASCII  /%O6/
004162  045  101  057           .ASCII  /%A/<57>
004165  045  117  066           .ASCII  /%O6/
004170  045  101  040           .ASCII  /%A /
004173  050  117  143           .ASCII  /(Oc/
004176  164  051  040           .ASCII  /t) /
004201  156  157  164           .ASCII  /not/
004204  040  163  145           .ASCII  / se/
004207  156  164  040           .ASCII  /nt /
004212  142  171  040           .ASCII  /by /
004215  110  157  163           .ASCII  /Hos/
004220  164  000                .ASCII  /t/<00>
```

```
004222  045  116  045   P.ACE:  .ASCII  /$N$/
004225  101  052  052           .ASCII  /A**/
004230  040  125  156           .ASCII  / Un/
004233  153  156  157           .ASCII  /kno/
004236  167  156  040           .ASCII  /wn /
004241  105  162  162           .ASCII  /Err/
004244  157  162  040           .ASCII  /or /
004247  114  157  147           .ASCII  /Log/
004252  040  146  157           .ASCII  / fo/
004255  162  155  141           .ASCII  /rma/
004260  164  040  045           .ASCII  /t $/
004263  117  063  045           .ASCII  /O3$/
004266  101  040  162           .ASCII  /A r/
004271  145  143  145           .ASCII  /ece/
004274  151  166  145           .ASCII  /ive/
004277  144  000  000           .ASCII  /d/<00><00>
004302  045  116  045   P.ACF:  .ASCII  /$N$/
004305  101  052  052           .ASCII  /A**/
004310  040  117  160           .ASCII  / Op/
004313  055  143  157           .ASCII  /-co/
004316  144  145  040           .ASCII  /de /
004321  045  117  063           .ASCII  /$O3/
004324  045  101  054           .ASCII  /$A,/
004327  040  105  156           .ASCII  / En/
004332  144  055  143           .ASCII  /d-c/
004335  157  144  145           .ASCII  /ode/
004340  040  045  117           .ASCII  / $O/
004343  063  045  101           .ASCII  /3$A/
004346  040  146  157           .ASCII  / fo/
004351  162  040  162           .ASCII  /r r/
004354  145  146  040           .ASCII  /ef /
004357  043  040  045           .ASCII  /# $/
004362  117  066  045           .ASCII  /O6$/
004365  101  057  045           .ASCII  /A/<57>/$/
004370  117  066  045           .ASCII  /O6$/
004373  101  040  050           .ASCII  /A (/
004376  070  051  000           .ASCII  /8)/<00>
004401  000                     .ASCII  <00>
004402  045  116  045   P.ACG:  .ASCII  /$N$/
004405  101  052  052           .ASCII  /A**/
004410  040  103  155           .ASCII  / Cm/
004413  144  055  142           .ASCII  /d-b/
004416  143  040  045           .ASCII  /c $/
004421  117  066  045           .ASCII  /O6$/
004424  101  057  045           .ASCII  /A/<57>/$/
004427  117  066  045           .ASCII  /O6$/
004432  101  040  122           .ASCII  /A R/
004435  163  160  055           .ASCII  /sp-/
004440  142  143  040           .ASCII  /bc /
004443  045  117  066           .ASCII  /$O6/
004446  045  101  057           .ASCII  /$A/<57>
004451  045  117  066           .ASCII  /$O6/
004454  045  101  040           .ASCII  /$A /
```

```
004457   146   157   162              .ASCII  /for/
004462   040   045   117              .ASCII  / %O/
004465   066   045   101              .ASCII  /6%A/
004470   057   045   117              .ASCII  <57>/%O/
004473   066   045   101              .ASCII  /6%A/
004476   040   050   070              .ASCII  / (8/
004501   051   000   000              .ASCII  /)/<00><00>
004504   045   116   045   P.ACH:     .ASCII  /%N%/
004507   101   052   052              .ASCII  /A**/
004512   040   122   145              .ASCII  / Re/
004515   163   160   157              .ASCII  /spo/
004520   156   163   145              .ASCII  /nse/
004523   040   141   154              .ASCII  / al/
004526   162   145   141              .ASCII  /rea/
004531   144   171   040              .ASCII  /dy /
004534   162   145   143              .ASCII  /rec/
004537   145   151   166              .ASCII  /eiv/
004542   145   144   040              .ASCII  /ed /
004545   146   157   162              .ASCII  /for/
004550   040   143   155              .ASCII  / cm/
004553   144   040   045              .ASCII  /d %/
004556   117   066   045              .ASCII  /O6%/
004561   101   057   045              .ASCII  /A/<57>/%/
004564   117   066   045              .ASCII  /O6%/
004567   101   040   050              .ASCII  /A (/
004572   070   051   000              .ASCII  /8)/<00>
004575   000                          .ASCII  <00>
004576   045   116   045   P.ACI:     .ASCII  /%N%/
004601   101   052   052              .ASCII  /A**/
004604   040   106   141              .ASCII  / Fa/
004607   151   154   165              .ASCII  /ilu/
004612   162   145   040              .ASCII  /re /
004615   164   157   040              .ASCII  /to /
004620   163   145   156              .ASCII  /sen/
004623   144   040   143              .ASCII  /d c/
004626   157   155   155              .ASCII  /omm/
004631   141   156   144              .ASCII  /and/
004634   040   141   146              .ASCII  / af/
004637   164   145   162              .ASCII  /ter/
004642   040   043   040              .ASCII  / # /
004645   045   117   066              .ASCII  /%O6/
004650   045   101   057              .ASCII  /%A/<57>
004653   045   117   066              .ASCII  /%O6/
004656   045   101   040              .ASCII  /%A /
004661   050   070   051              .ASCII  /(8)/
004664   000   000                    .ASCII  <00><00>
004666   045   116   045   P.ACJ:     .ASCII  /%N%/
004671   101   125   116              .ASCII  /AUN/
004674   111   124   045              .ASCII  /IT%/
004677   104   062   045              .ASCII  /D2%/
004702   101   040   104              .ASCII  /A D/
004705   122   117   120              .ASCII  /ROP/
004710   120   105   104              .ASCII  /PED/
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
012222    105    122    122              .ASCII  /ERR/
012225    117    122    045              .ASCII  /OR%/
012230    116    000                     .ASCII  /N/<00>
012232    045    101    052    P.AFX:    .ASCII  /%A*/
012235    040    104    111              .ASCII  / DI/
012240    123    113    045              .ASCII  /SK%/
012243    104    062    045              .ASCII  /D2%/
012246    101    040    055              .ASCII  /A -/
012251    040    042    123              .ASCII  / "S/
012254    124    101    116              .ASCII  /TAN/
012257    104    101    122              .ASCII  /DAR/
012262    104    040    104              .ASCII  /D D/
012265    111    123    113              .ASCII  /ISK/
012270    040    111    116              .ASCII  / IN/
012273    124    105    122              .ASCII  /TER/
012276    103    117    116              .ASCII  /CON/
012301    116    105    103              .ASCII  /NEC/
012304    124    042    040              .ASCII  /T" /
012307    105    122    122              .ASCII  /ERR/
012312    117    122    045              .ASCII  /OR%/
012315    116    000    000              .ASCII  /N/<00><00>
012320    045    101    052    P.AFY:    .ASCII  /%A*/
012323    040    104    111              .ASCII  / DI/
012326    123    113    045              .ASCII  /SK%/
012331    104    062    045              .ASCII  /D2%/
012334    101    040    055              .ASCII  /A -/
012337    040    042    123              .ASCII  / "S/
012342    115    101    114              .ASCII  /MAL/
012345    114    040    104              .ASCII  /L D/
012350    111    123    113              .ASCII  /ISK/
012353    042    040    105              .ASCII  /" E/
012356    122    122    117              .ASCII  /RRO/
012361    122    045    116              .ASCII  /R%N/
012364    000    000                     .ASCII  <00><00>
012366 012074'                 P.AFT:    .WORD   P.AFU
012370 012124'                           .WORD   P.AFV
012372 012164'                           .WORD   P.AFW
012374 012232'                           .WORD   P.AFX
012376 012320'                           .WORD   P.AFY
012400    045    116    045    P.AFZ:    .ASCII  /%N%/
012403    101    052    040              .ASCII  /A* /
012406    123    101    072              .ASCII  /SA:/
012411    040    045    117              .ASCII  / %O/
012414    066    000                     .ASCII  /6/<00>
012416    045    116    045    P.AGA:    .ASCII  /%N%/
012421    101    052    040              .ASCII  /A* /
012424    123    124    101              .ASCII  /STA/
012427    124    125    123              .ASCII  /TUS/
012432    040    103    117              .ASCII  / CO/
012435    104    105    072              .ASCII  /DE:/
012440    040    045    117              .ASCII  / %O/
012443    062    000    000              .ASCII  /2/<00><00>
012446    045    117    064    P.AGB:    .ASCII  /%O4/
```

```
012451  000                        .ASCII  <00>
012452  045   116   045   P.AGC:   .ASCII  /$N$/
012455  101   052   040            .ASCII  /A* /
012460  123   125   102            .ASCII  /SUB/
012463  137   103   117            .ASCII  /_CO/
012466  104   105   072            .ASCII  /DE:/
012471  040   000   000            .ASCII  / /<00><00>
012474  045   116   045   P.AGD:   .ASCII  /$N$/
012477  101   052   040            .ASCII  /A* /
012502  103   117   115            .ASCII  /COM/
012505  115   101   116            .ASCII  /MAN/
012510  104   072   040            .ASCII  /D: /
012513  000                        .ASCII  <00>
012514  045   101   122   P.AGE:   .ASCII  /$AR/
012517  105   101   104            .ASCII  /EAD/
012522  000   000                  .ASCII  <00><00>
012524  045   101   127   P.AGF:   .ASCII  /$AW/
012527  122   111   124            .ASCII  /RIT/
012532  105   000                  .ASCII  /E/<00>
012534  045   101   055   P.AGG:   .ASCII  /$A-/
012537  103   117   115            .ASCII  /COM/
012542  120   101   122            .ASCII  /PAR/
012545  105   000   000            .ASCII  /E/<00><00>
012550  045   101   117   P.AGH:   .ASCII  /$AO/
012553  116   114   111            .ASCII  /NLI/
012556  116   105   000            .ASCII  /NE/<00>
012561  000                        .ASCII  <00>
012562  045   101   101   P.AGI:   .ASCII  /$AA/
012565  103   103   105            .ASCII  /CCE/
012570  123   123   000            .ASCII  /SS/<00>
012573  000                        .ASCII  <00>
012574  045   117   063   P.AGJ:   .ASCII  /$O3/
012577  000                        .ASCII  <00>
012600  045   116   045   P.AGK:   .ASCII  /$N$/
012603  101   052   040            .ASCII  /A* /
012606  102   101   104            .ASCII  /BAD/
012611  040   102   114            .ASCII  / BL/
012614  117   103   113            .ASCII  /OCK/
012617  040   050   110            .ASCII  / (H/
012622  157   163   164            .ASCII  /ost/
012625  040   162   145            .ASCII  / re/
012630  160   154   141            .ASCII  /pla/
012633  143   145   141            .ASCII  /cea/
012636  142   154   145            .ASCII  /ble/
012641  051   072   040            .ASCII  /): /
012644  045   104   065            .ASCII  /$D5/
012647  045   101   056            .ASCII  /$A./
012652  040   050   117            .ASCII  / (O/
012655  103   124   040            .ASCII  /CT /
012660  045   117   066            .ASCII  /$O6/
012663  045   101   051            .ASCII  /$A)/
012666  000   000                  .ASCII  <00><00>
012670  045   116   045   P.AGL:   .ASCII  /$N$/
```

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

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

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

K8


```
013624   106   106   105             .ASCII  /FFE/
013627   122   040   101             .ASCII  /R A/
013632   104   104   122             .ASCII  /DDR/
013635   105   123   123             .ASCII  /ESS/
013640   040   050   063             .ASCII  / (3/
013643   062   040   142             .ASCII  /2 b/
013646   151   164   163             .ASCII  /its/
013651   051   072   040             .ASCII  /): /
013654   045   117   066             .ASCII  /%O6/
013657   045   101   040             .ASCII  /%A /
013662   045   117   066             .ASCII  /%O6/
013665   000                         .ASCII  <00>
013666   045   116   045   P.AGX:    .ASCII  /%N%/
013671   101   052   040             .ASCII  /A* /
013674   111   057   117             .ASCII  /I/<57>/O/
013677   040   102   125             .ASCII  / BU/
013702   106   106   105             .ASCII  /FFE/
013705   122   040   101             .ASCII  /R A/
013710   104   104   122             .ASCII  /DDR/
013713   105   123   123             .ASCII  /ESS/
013716   040   106   117             .ASCII  / FO/
013721   122   040   122             .ASCII  /R R/
013724   105   101   104             .ASCII  /EAD/
013727   040   050   063             .ASCII  / (3/
013732   062   040   142             .ASCII  /2 b/
013735   151   164   163             .ASCII  /its/
013740   051   072   040             .ASCII  /): /
013743   045   117   066             .ASCII  /%O6/
013746   045   101   040             .ASCII  /%A /
013751   045   117   066             .ASCII  /%O6/
013754   000   000                   .ASCII  <00><00>
013756   045   116   045   P.AGY:    .ASCII  /%N%/
013761   101   052   040             .ASCII  /A* /
013764   111   057   117             .ASCII  /I/<57>/O/
013767   040   102   125             .ASCII  / BU/
013772   106   106   105             .ASCII  /FFE/
013775   122   040   101             .ASCII  /R A/
014000   104   104   122             .ASCII  /DDR/
014003   105   123   123             .ASCII  /ESS/
014006   040   106   117             .ASCII  / FO/
014011   122   040   127             .ASCII  /R W/
014014   122   111   124             .ASCII  /RIT/
014017   105   040   050             .ASCII  /E (/
014022   063   062   040             .ASCII  /32 /
014025   142   151   164             .ASCII  /bit/
014030   163   051   072             .ASCII  /s):/
014033   040   045   117             .ASCII  / %O/
014036   066   045   101             .ASCII  /6%A/
014041   040   045   117             .ASCII  / %O/
014044   066   000                   .ASCII  /6/<00>
014046   045   116   045   P.AGZ:    .ASCII  /%N%/
014051   101   103   117             .ASCII  /ACO/
014054   116   124   105             .ASCII  /NTE/
```

```
014057  116  124  123            .ASCII  /NTS/
014062  040  117  106            .ASCII  / OF/
014065  040  103  117            .ASCII  / CO/
014070  115  115  101            .ASCII  /MMA/
014073  116  104  057            .ASCII  /ND/<57>
014076  122  105  123            .ASCII  /RES/
014101  120  117  116            .ASCII  /PON/
014104  123  105  040            .ASCII  /SE /
014107  120  101  103            .ASCII  /PAC/
014112  113  105  124            .ASCII  /KET/
014115  040  123  101            .ASCII  / SA/
014120  126  105  040            .ASCII  /VE /
014123  101  122  105            .ASCII  /ARE/
014126  101  072  045            .ASCII  /A:%/
014131  116  000  000            .ASCII  /N/<00><00>
014134  045  101  040   P.AHA:   .ASCII  /%A /
014137  045  117  066            .ASCII  /%O6/
014142  000  000                 .ASCII  <00><00>
014144  045  116  045   P.AHB:   .ASCII  /%N%/
014147  101  052  040            .ASCII  /A* /
014152  124  111  115            .ASCII  /TIM/
014155  105  072  040            .ASCII  /E: /
014160  045  132  062            .ASCII  /%Z2/
014163  045  101  072            .ASCII  /%A:/
014166  045  132  062            .ASCII  /%Z2/
014171  045  101  040            .ASCII  /%A /
014174  110  117  125            .ASCII  /HOU/
014177  122  123  045            .ASCII  /RS%/
014202  116  000                 .ASCII  /N/<00>
014204  045  116  045   P.AHC:   .ASCII  /%N%/
014207  101  040  052            .ASCII  /A */
014212  040  104  111            .ASCII  / DI/
014215  123  113  040            .ASCII  /SK /
014220  072  040  045            .ASCII  /: %/
014223  104  062  000            .ASCII  /D2/<00>
014226  045  116  045   P.AHD:   .ASCII  /%N%/
014231  101  104  102            .ASCII  /ADB/
014234  116  072  040            .ASCII  /N: /
014237  045  104  065            .ASCII  /%D5/
014242  045  101  056            .ASCII  /%A./
014245  040  050  117            .ASCII  / (O/
014250  103  124  040            .ASCII  /CT /
014253  045  117  066            .ASCII  /%O6/
014256  045  101  051            .ASCII  /%A)/
014261  000                      .ASCII  <00>
014262  045  116  045   P.AHE:   .ASCII  /%N%/
014265  101  102  131            .ASCII  /ABY/
014270  124  105  040            .ASCII  /TE /
014273  116  125  115            .ASCII  /NUM/
014276  102  105  122            .ASCII  /BER/
014301  072  040  045            .ASCII  /: %/
014304  104  063  000            .ASCII  /D3/<00>
014307  000                      .ASCII  <00>
```

```
014310  045  116  045     P.AHF:  .ASCII  /%N%/
014313  101  122  101             .ASCII  /ARA/
014316  116  104  117             .ASCII  /NDO/
014321  115  040  127             .ASCII  /M W/
014324  122  111  124             .ASCII  /RIT/
014327  124  105  116             .ASCII  /TEN/
014332  040  127  117             .ASCII  / WO/
014335  122  104  040             .ASCII  /RD /
014340  072  045  102             .ASCII  /:%B/
014343  061  066  000             .ASCII  /16/<00>
014346  045  116  045     P.AHG:  .ASCII  /%N%/
014351  101  122  101             .ASCII  /ARA/
014354  116  104  117             .ASCII  /NDO/
014357  115  040  122             .ASCII  /M R/
014362  105  101  104             .ASCII  /EAD/
014365  040  127  117             .ASCII  / WO/
014370  122  104  040             .ASCII  /RD /
014373  142  151  156             .ASCII  /bin/
014376  072  045  102             .ASCII  /:%B/
014401  061  066  045             .ASCII  /16%/
014404  101  040  157             .ASCII  /A o/
014407  143  164  072             .ASCII  /ct:/
014412  045  117  066             .ASCII  /%O6/
014415  000                       .ASCII  <00>
014416  045  116  045     P.AHH:  .ASCII  /%N%/
014421  101  104  125             .ASCII  /ADU/
014424  120  114  111             .ASCII  /PLI/
014427  103  101  124             .ASCII  /CAT/
014432  105  040  125             .ASCII  /E U/
014435  116  111  124             .ASCII  /NIT/
014440  072  045  104             .ASCII  /:%D/
014443  062  045  101             .ASCII  /2%A/
014446  040  101  124             .ASCII  / AT/
014451  040  111  120             .ASCII  / IP/
014454  072  040  045             .ASCII  /: %/
014457  117  066  000             .ASCII  /O6/<00>
014462  045  116  045     P.AHI:  .ASCII  /%N%/
014465  101  115  117             .ASCII  /AMO/
014470  122  105  040             .ASCII  /RE /
014473  124  110  101             .ASCII  /THA/
014476  116  040  045             .ASCII  /N %/
014501  104  061  045             .ASCII  /D1%/
014504  101  040  104             .ASCII  /A D/
014507  111  106  106             .ASCII  /IFF/
014512  105  122  105             .ASCII  /ERE/
014515  116  124  040             .ASCII  /NT /
014520  111  120  040             .ASCII  /IP /
014523  101  104  104             .ASCII  /ADD/
014526  122  105  123             .ASCII  /RES/
014531  123  105  123             .ASCII  /SES/
014534  000  000                  .ASCII  <00><00>
014536  045  101  123     P.AHJ:  .ASCII  /%AS/
014541  120  111  116             .ASCII  /PIN/
```

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

B9


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
022545   000                          .ASCII  <00>
022546   045     101     040  P.AKR:  .ASCII  /%A /
022551   055     040     000          .ASCII  /- /<00>
022554   045     101     052  P.AKS:  .ASCII  /%A*/
022557   040     000     000          .ASCII  / /<00><00>
022562   00000C               L$HWLEN::
                                      .WORD   <<L$NDHW-L$HWLEN>/2>
022564   172150               HWPT.IP.ADDR::
                                      .WORD   -5630
022566   000154               HWPT.VECTOR::
                                      .WORD   154
022570   000004               HWPT.BR.LEVEL::
                                      .WORD   4
022572   000200               HWPT.DISK::
                                      .WORD   200
022574   000000               HWPTS0.LBN::
                                      .WORD   0
022576   000000               HWPTS1.LBN::
                                      .WORD   0
022600   177777               HWPTE0.LBN::
                                      .WORD   -1
022602   000000               HWPTE1.LBN::
                                      .WORD   0
022604   020040               NAME.LO::
                                      .WORD   20040
022606   020040               NAME.HI::
                                      .WORD   20040
022610                        L$NDHW::.BLKW   1
022612   00000C               L$SWLEN::
                                      .WORD   <<L$NDSW-L$SWLEN>/2>
022614   000040               SWP.ERROR::
                                      .WORD   40
022616   000000               SWP.XFER::
                                      .WORD   0
022620   054046               SWP.FLAGS::
                                      .WORD   54046
022622   000000               SWP.DPAT::
                                      .WORD   0
022624   000143               SWP.RAT::
                                      .WORD   143
022626   000000               SWP.TIME::
                                      .WORD   0
022630   000013               DUPROUND::
                                      .WORD   13
022632   000020               SWP.UCNT::
                                      .WORD   20
022634                        SWP.UDPAT::
                                      .BLKW   20
022674                        L$NDSW::.BLKW   1
022676   000000               L$PROT::.WORD   0
022700   177777                       .WORD   -1
022702   000006                       .WORD   6
```

```
000000                          .PSECT  $FFF$,  D  ,  GBL
000000                  CST::   .BLKW   53
000126                  CST.ADDR::
                                .BLKW   1
000130                  DCT::   .BLKW   11
000152                  DCT.ADDR::
                                .BLKW   1
000154                  RDRX.ADDR::
                                .BLKW   1
000156                  IRDRX.ADDR::
                                .BLKW   1
000160                  BST::   .BLKW   10
000200                  TALLY:: .BLKW   154
000530                  T.ADDR::.BLKW   1
000532                  DUPPKT::.BLKW   401
001534                  TRK.SGN::
001534     001                  .BYTE   1
001535     001                  .BYTE   1
001536     001                  .BYTE   1
001537     001                  .BYTE   1
001540  000020          RDM.CNT::
                                .WORD   20
001542                  RANDOM::.BLKW   20
001602                  C.ERR.TBL::
                                .BLKW   1
001604                  MSCP.PKT::
                                .BLKW   644
003314                  IPKT.ADDR::
                                .BLKW   1
003316                  PKT.USE::
                                .BLKW   6
003332                  RETPKT::.BLKW   260
004072                  RP.USE::.BLKW   4
004102                  RP.INDX::
                                .BLKW   1
004104                  RP.ADDR::
                                .BLKW   1
004106                  ELOG.PKT::
                                .BLKW   655
005640                  BUFF.ADDR::
                                .BLKW   10
005660                  BUFF.OWN::
                                .BLKW   4
005670                  IODQ::  .BLKW   4
005700                  IODQ.IN::
                                .BLKW   1
005702                  IODQ.OUT::
                                .BLKW   1
005704                  ENTRY.REASON::
                                .BLKB   1
005705                  EOP.FLAG::
```

```
                                          .BLKB    1
005706                           DUP.FLAGS::
                                          .BLKW    1
005710                           CCTLR:: .BLKW    1
005712                           CDISK:: .BLKW    1
005714                           CUOFF:: .BLKW    1
005716                           CTLR.CNT::
                                          .BLKW    1
005720                           DUR::    .BLKW    2
005724                           QIO::    .BLKB    1
                                          .EVEN
005726                           FREE.MEM.ADDR::
                                          .BLKW    1
005730                           BYTS.PER.QIO::
                                          .BLKW    1
005732                           ST.CODE::
                                          .BLKW    1
005734                           SB.CODE::
                                          .BLKW    1
005736                           STEP::   .BLKW    1
005740                           OF.RC::  .BLKW    1
005742                           SA.REG::.BLKW    1
005744                           CMD.TIME::
                                          .BLKW    1
005746                           NEX::    .BLKW    1
005750                           CRN.LOW::
                                          .BLKW    1
005752                           CRN.HIGH::
                                          .BLKW    1
005754                           TEMP1:: .BLKW    1
005756                           TEMP2:: .BLKW    1
005760                           CREDIT.BAL::
                                          .BLKW    1
005762                           NEXT.PKT.USE::
                                          .BLKB    1
005763                           HOURS:: .BLKB    1
005764                           MINUTES::
                                          .BLKB    1
                                          .EVEN
005766                           CLK.TICKS::
                                          .BLKW    1
005770                           CLK.PRESENT::
                                          .BLKB    1
005771                           HOE.FLAG::
                                          .BLKB    1
005772                           S.PATTERN::
                                          .BLKW    1
005774                           S.DUPPKT::
                                          .BLKW    1
005776                           P.INDEX::
                                          .BLKW    1
006000  000000                   RD.COUNT::
                                          .WORD    0
```

K10


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

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

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

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

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

```
007066'   EGS.02==    P.ADQ
007160'   EGD.10==    P.ADR
007220'   EGD.11==    P.ADS
007244'   EGD.12==    P.ADT
007272'   EGD.13==    P.ADU
007320'   EGD.14==    P.ADV
007350'   EGD.15==    P.ADW
007366'   EGD.16==    P.ADX
007416'   EGD.17==    P.ADY
007434'   EGD.18==    P.ADZ
007454'   EGD.19==    P.AEA
007514'   EGD.20==    P.AEB
007602'   EGD.21==    P.AEC
007714'   EGD.22==    P.AED
007754'   EGD.23==    P.AEE
010016'   EGD.24==    P.AEF
010062'   EGH.30==    P.AEG
010106'   EBS.01==    P.AEH
010150'   EBD.10==    P.AEI
010210'   EBD.12==    P.AEJ
010256'   EBD.13==    P.AEK
010310'   EBD.14==    P.AEL
010350'   EBD.18==    P.AEM
010404'   EBD.19==    P.AEN
010464'   EBD.24==    P.AEO
010540'   EH.0==      P.AEP
010576'   EH.1==      P.AEQ
010634'   EH.2==      P.AER
010674'   EH.3==      P.AES
010732'   EH.4==      P.AET
010756'   EH.5==      P.AEU
011006'   EH.6==      P.AEV
011036'   EH.7==      P.AEW
011066'   EH.8==      P.AEX
011122'   EH.9==      P.AEY
011152'   EH.10==     P.AEZ
011202'   EH.12==     P.AFA
011236'   EH.13==     P.AFB
011310'   ERR.00==    P.AFC
012000'   ERR.COD==   P.AFD
012034'   ELG.00==    P.AFS
012366'   ELG.FMT==   P.AFT
012400'   EX.SA==     P.AFZ
012416'   EX.SC==     P.AGA
012446'   EX.SBO==    P.AGB
012452'   EX.SB==     P.AGC
012474'   EX.CMD==    P.AGD
012514'   EX.RD==     P.AGE
012524'   EX.WRT==    P.AGF
012534'   EX.CMP==    P.AGG
012550'   EX.ONL==    P.AGH
012562'   EX.ACC==    P.AGI
012574'   EX.OP==     P.AGJ
```

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

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


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



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

```
;                    COMMAND QUALIFIERS

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

```
;  0    module ZRQAM2 (
;  0
;  0    %title 'RD/RX EXERCISER'
;  0                    ident = 'V01.9',
;  0                    addressing_mode (absolute),
;  0                    environment (noeis)
;  0                    ) =
;  0
;  1    begin
;  1
;  1    %sbttl 'DECLARATIONS'
;  1
;  1    library 'ZRQAFO.L16';                              ! RDRX EXERCISER GLOBAL LIBRARY
;  1
;  1    require 'BLSMAC.REQ';                              ! DIAGNOSTIC SUPERVISOR LIBRARY
;  1
;  1    forward routine
;  1        NEX_TRAP : L$ISR novalue,
;  1        EMS_01 : novalue,
;  1        EMS_TIM : novalue,
;  1        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        DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),        !BUFFER FOR DUP  ZZZ
;  1                        !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
;  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        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          P_INDEX : SIGNED WORD,          !CURRENT MESSAGE PACKET INDEX     ZZZ
;  1          S_PATTERN : WORD,               !PATTERN FOR DUP WRITES           ZZZ
;  1          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_00,
;  1          ELG_FMT : vector [5],
;  1          EX_TIM,
;  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_BB,
;     1          EX_BB1,
;     1          EX_BBU,
;     1          EX_LBN,
;     1          EX_PBN,
;     1          EX_LBR,
;     1          EX_LBW,
;     1          EX_RBN,
;     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_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_EC1,
;     1          SC_EC2,
;     1          SC_EC3,
;     1          SC_EC4,
;     1          SC_EC5,
```

```
;      1          SC_EC6,
;      1          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          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,
;      1          HWQ6A,
;      1          HWQ6B,
;      1          HWQ7A,
;      1          HWQ7B,
;      1          HWQ8,
;      1          HWQ9,
;      1          HWQ10,                                          !ZZZ
;      1          HWQ11,                                          !ZZZ
;      1          SWQ1,
;      1          SWQ2,
;      1          SWQ4,
;      1          SWQ7,
;      1          SWQ9,
;      1          SWQ10,
;      1          SWQ11,
;      1          SWQ12,
;      1          SWQ13,
;      1          SWQ14,
;      1          SWQ15,
;      1          SWQ17,
```

```
;      1            SWQ19,
;      1            SWQ20,
;      1            SWQ21,
;      1            SWQ22,
;      1            SWQ23,
;      1            SWQ24,
;      1            SWQ25,
;      1            SWQ26,                                              !ZZZ
;      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
;      1            EH_7,                                               !ZZZ
;      1            EH_8,                                               !ZZZ
;      1            EH_9,                                               !ZZZ
;      1            EH_10,                                              !ZZZ
;      1            EH_12,                                              !ZZZ
;      1            EH_13,                                              !ZZZ
;      1            SWM1,
;      1            NULL,
;      1            SWP_FLAGS : word,
;      1            L$HIMEM,
;      1            L$LUN,
;      1            L$UNIT;
;      1        !   O_BRK;
;      1
;      1        own
;      1            TBL_SUC : vector [17] initial (NULL, SC_SDI, SC_CON, NULL, SC_DUP, NULL, NULL,
;      1                      NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SON),
;      1            TBL_OFL : vector [9] initial (SC_UNK, SC_VOL, SC_IOP, NULL, SC_DUP, NULL, NULL,
;      1                      NULL, SC_DIS),
;      1            TBL_MFE : vecter [11] initial (SC_FER, NULL, SC_ISH, SC_DST, SC_EC9, SC_576,
;      1                      SC_FCT, SC_ECC, SC_RCT, SC_FUL, SC_EC1),
;      1            TBL_WPT : vector [3] initial (NULL, SC_SWP, SC_HWP),
;      1            TBL_DAT : vector [16] initial (SC_FE2, NULL, SC_IS2, SC_DS2, SC_EC9, NULL, NULL,
;      1                      SC_ECD, SC_EC1, SC_EC2, SC_EC3, SC_EC4, SC_EC5, SC_EC6, SC_EC7, SC_EC8),
;      1            TBL_HST : vector [5] initial (NULL, SC_ODA, SC_ODB, SC_NXM, SC_PAR),
;      1            TBL_CNT : vector [4] initial (SC_CTO, SC_SDS, SC_EDC, SC_IDS),
;      1            TBL_DRV : vector [9] initial (NULL, SC_SRT, SC_SRI, SC_POE, SC_RDY, SC_CLK, SC_RSP,
;      1                      SC_SUR, SC_PSP);
;      1
;      1
```

```
%sbttl 'TYPE AND DESCRIPTION'

EQUALS;

DEVTYP (%asciz'RQDX1 or RUX50');                    ! NAME OF DEVICE SUPPORTED BY PROGRAM
DESCRIPT (%asciz'RD/RX EXERCISER');     ! TEST DESCRIPTION
```

```
;  1   %sbttl 'HARDWARE PARAMETER CODING SECTION'
;  1
;  1   !+
;  1   ! THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;  1   ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;  1   ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;  1   ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;  1   ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;  1   ! WITH THE OPERATOR.
;  1   !-
;  1
;  1   BGNHRD;
;  1
;  1   GPRMA (HWQ1, 0, 0, %o'160000', %o'177777', YES, 1);                    ! IP ADDRESS
;  1   GPRMA (HWQ2, 2, 0, %o'4', %o'774', YES, 1);                            ! VECTOR
;  1   GPRMD (HWQ3, 4, 0, %o'377', %o'0', %o'7', YES, 1);                     ! BR LEVEL
;  1   GPRMD (HWQ4, 6, D, %o'17', %decimal'0', %decimal'15', YES, 1);         ! RDRX DRIVE NUMBER
;  1   GPRML (HWQ10, 6, %o'000040', YES, 1);                                  !ALSO RUN DUP EXERCISER       ZZZ
;  1   XFERF (NODU);                                                          !                             ZZZ
;  1   GPRML (HWQ11, 6, %o'000100', YES, 1);                                  !WRITE DIAG AREA              ZZZ
;  1   $L (NODU);                                                             !                             ZZZ
;  1   GPRML (HWQ5, 6, %o'000200', YES, 1);                                   !TEST ENTIRE CUSTOMER AREA?   ZZZ
;  1   XFERT (TOQ8);                                                          !BR IF YES                    ZZZ
;  1   GPRMD (HWQ6A, 8, 0, %o'177777', %decimal'0', %o'177777', YES, 1);      ! STARTING LBN LO             ZZZ
;  1   GPRMD (HWQ6B, 10, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! STARTING LBN HI             ZZZ
;  1   GPRMD (HWQ7A, 12, 0, %o'177777', GP$ATLO (8), %o'177777', YES, 1);     ! ENDING LBN LO               ZZZ
;  1   GPRMD (HWQ7B, 14, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! ENDING LBN HI               ZZZ
;  1   $L (TOQ8);                                                             !                             ZZZ
;  1   GPRML (HWQ8, 6, %o'100000', NO, 0);                                    ! WRITE ON CUST DATA AREA
;  1   XFERF (HWDONE);                                                        ! NO - DONE
;  1   GPRML (HWQ9, 6, %o'100000', NO, 1);                                    ! ** WARNING / CONFIRM
;  1   $L (HWDONE);
;  1
;  1   ENDHRD;
```

```
;   1   %sbttl 'SOFTWARE PARAMETER CODING SECTION'
;   1
;   1   !+
;   1   ! THE SOFTWARE PARAMETER CODIMG SECTION CONTAINS MACROS
;   1   ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;   1   ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;   1   ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;   1   ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;   1   ! WITH THE OPERATOR.
;   1   !-
;   1
;   1   BGNSFT;
;   1
;   1   !GPRML (SWQ16, 4, SWF_TRC, YES, 1);                          ! ENABLE DIAGNOSTIC TRACE
;   1   GPRMD (SWQ24, 10, D, %o'177777', 0, 2359, YES, 1);          ! START TIME
;   1   GPRMD (SWQ1, 0, D, %o'177777', 0, 65535, YES, 1);           ! ERROR LIMIT
;   1   GPRMD (SWQ2, 2, D, %o'177777', 0, 99, YES, 1);              ! TRANSFER LIMIT
;   1   GPRMD (SWQ17, 8, D, %o'177777', 0, 100, YES, 1);            ! PERCENT OF RD OPERATIONS
;   1   GPRML (SWQ15, 4, SWF_CST, YES, 1);                          ! CLEAR STATISTICAL TABLES ?
;   1   GPRML (SWQ20, 4, SWF_FER, YES, 1);                          ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT SET?
;   1   GPRML (SWQ23, 4, SWF_BLK, YES, 1);                          ! HALT ON BAD-BLOCK TYPE ERRORS WITH 'HOE' FLAG?
;   1   GPRML (SWQ21, 4, SWF_HRD, YES, 1);                          ! HALT ON HARD ERRORS WITH 'HOE' FLAG SET?
;   1   GPRML (SWQ22, 4, SWF_SFT, YES, 1);                          ! HALT ON SOFT ERRORS WITH 'HOE' FLAG SET?
;   1   GPRML (SWQ25, 4, SWF_TRY, YES, 1);                          ! COUNT EACH RETRY AS ANOTHER SOFT-ERROR?
;   1   GPRML (SWQ4, 4, SWF_RDM, YES, 1);                           ! RANDOM SEEK MODE ?
;   1   XFERF (SW1);                                                ! IF NO, DO NEXT QUESTION
;   1   XFER (SW2);                                                 !
;   1   $L (SW1);                                                   !
;   1   GPRML (SWQ19, 4, SWF_SEQ, YES, 1);                          ! RANDOM OR SEQUENTIAL SELECTION OF DRIVES
;   1   $L (SW2);                                                   !
;   1   GPRML (SWQ7, 4, SWF_CRC, YES, 1);                           ! READ-COMPARES AT CONTROLLER ?
;   1   GPRML (SWQ26, 4, SWF_APT, YES, 1);                          !RUNNING UNDER A.P.T. MONITOR?          ZZZ
;   1   DISPLAY (SWM1);                                             ! REMAINING QUESTIONS ONLY APPLY ...
;   1   GPRML (SWQ9, 4, SWF_CWC, YES, 1);                           ! WRITE-COMPARES AT CONTROLLER ?
;   1   XFERF (SW3);                                                ! IF NO, DO NEXT QUESTION
;   1   XFER (SW4);                                                 !
;   1   $L (SW3);                                                   !
;   1   GPRML (SWQ10, 4, SWF_HWC, YES, 1);                          ! CHECK WRITES AT HOST BY READING ?
;   1   $L (SW4);                                                   !
;   1   GPRML (SWQ11, 4, SWF_UDP, YES, 1);                          ! USER-DEFINED DATA PATTERN ?
;   1   XFERF (SW5);                                                ! IF NO, DO NEXT QUESTION
;   1   XFER (SW6);                                                 !
;   1   $L (SW5);                                                   !
;   1   GPRMD (SWQ12, 6, D, %o'177777', 0, DP_CNT, YES, 1);         ! SELECT PRE-DEFINED DATA PATTERN
;   1   XFER (SW7);                                                 ! DONE
;   1   $L (SW6);                                                   !
;   1   GPRMD (SWQ13, 12, D, %o'177777', 1, MAX_UDP_CNT, YES, 1);   ! NO. OF WORDS IN USER DATA PATTERN
;   1   GPRMD (SWQ14, 14, O, %o'177777', 0, %o'177777', NO, 12);    ! PATTERN VALUES
;   1   $L (SW7);
;   1   ENDSFT;
```

```
#sbttl 'REPORT CODING SECTION'


!+
! THE REPORT CODING SECTION CONTAINS THE
! "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
!-


BGNRPT;

local
    CUR_PRIORITY : word;

GETPRI (CUR_PRIORITY);
!ZZZ SETPRI (PRIO4);                                         !ZZZ
SETPRI (.BRLEVEL);                                 !ZZZ

PRINTS (RPT1);
PRINTS (RPT2);
PRINTS (RPT3);
PRINTS (RPT4);
PRINTS (RPT5);
PRINTS (RPT6);

incr CTLR from 0 to MAX_CTLR - 1 do

    begin
    SET_CPAR (.CTLR);

    incr DISK from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do

        begin
        SET_UPAR (.DISK);


        if .CST_ADDR [.DISK + OF_DATA, D_PRES] eql PRESENT
        then

            begin
            PRINTS (RPT7,
                .L$LUN, .CST_ADDR [.DISK + OF_DATA, D_DISK_NUM], CST [.CTLR, .DISK + OF_NAME_0, D_NAME_0]);
            PRINTS (RPT8,
                .T_ADDR [TOT_READS_HI], .T_ADDR [TOT_READS_LO],
                .T_ADDR [MTOT_BYT_RED], .T_ADDR [TOT_BYT_RED_HI], .T_ADDR [TOT_BYT_RED_LO]);
            PRINTS (RPT8,
                .T_ADDR [TOT_WRITES_HI], .T_ADDR [TOT_WRITES_LO],
                .T_ADDR [MTOT_BYT_WRT], .T_ADDR [TOT_BYT_WRT_HI], .T_ADDR [TOT_BYT_WRT_LO]);
            PRINTS (RPT9,
                .T_ADDR [ERR_HRD_SEK], .T_ADDR [ERR_HRD_DAT], .T_ADDR [ERR_HRD_DRV], .T_ADDR [ERR_HRD_HST],
                .T_ADDR [ERR_SFT_SEK], .T_ADDR [ERR_SFT_DAT], .T_ADDR [ERR_SFT_DRV], .T_ADDR [ERR_SFT_HST]);
```

B12


```
                        end;
                    end;


                if .CST [.CTLR, STATE] eql PRESENT
                then

                    begin
                    PRINTS (RPT10);
                    PRINTS (RPT11, .C_ERR_TBL [.CTLR, C_ERR_HRD], .C_ERR_TBL [.CTLR, C_ERR_SFT]);
                    end;


                end;

            SETPRI (.CUR_PRIORITY);

            IF .RD_COUNT NEQ 0                                          !IF THERE IS A WINCHESTER      ZZZ
            THEN                                                        !THEN OUTPUT EXTRA LINES       ZZZ

            begin                                                       ! PRINTS DUP DATA
            prints(crlf);
            PRINTS(RPT13);
            PRINTS(RPT14);
            PRINTS(RPT15);
            INCR CTLR FROM 0 TO MAX_CTLR-1 DO
                    BEGIN
                    SET_CPAR(.CTLR);
                    INCR DISK FROM (0+OF_UN) TO (3*UNIT_SIZE+OF_UN) BY UNIT_SIZE DO
                            BEGIN
                            SET_UPAR(.DISK);
                            IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_51 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
                            THEN
                        PRINTS (RPT16,
                            .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
                            .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);

            !ZZZ            IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_52 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
            !ZZZ            THEN
            !ZZZ        PRINTS (RPT18,
            !ZZZ            .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
            !ZZZ            .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
                            END;
                    END;
                        PRINTS (CRLF);
            END;

            ENDRPT;


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

```
000000                                        .PSECT  $CODE$,  RO
000000     122      121      104     L$DVTYP::
                                              .ASCII  /RQD/
000003     130      061      040              .ASCII  /X1 /
000006     157      162      040              .ASCII  /or /
000011     122      125      130              .ASCII  /RUX/
000014     065      060      000              .ASCII  /50/<00>
000017     000                                .ASCII  <00>
000020     122      104      057     L$DESC:: .ASCII  /RD/<57>
000023     122      130      040              .ASCII  /RX /
000026     105      130      105              .ASCII  /EXE/
000031     122      103      111              .ASCII  /RCI/
000034     123      105      122              .ASCII  /SER/
000037     000                                .ASCII  <00>
000040                                        .BLKB   2
000042  000000C                      L$HRDLN::
                                              .WORD   <<<L$NDHRD-L$HRDLN>/2>-1>
000044  000031                       GP$1::   .WORD   31
000046  000000G                               .WORD   HWQ1
000050  160000                                .WORD   -20000
000052  177777                                .WORD   -1
000054  001031                       GP$2::   .WORD   1031
000056  000000G                               .WORD   HWQ2
000060  000004                                .WORD   4
000062  000774                                .WORD   774
000064  002032                       GP$3::   .WORD   2032
000066  000000G                               .WORD   HWQ3
000070  000377                                .WORD   377
000072  000000                                .WORD   0
000074  000007                                .WORD   7
000076  003052                       GP$4::   .WORD   3052
000100  000000G                               .WORD   HWQ4
000102  000017                                .WORD   17
000104  000000                                .WORD   0
000106  000017                                .WORD   17
000110  003130                       GP$5::   .WORD   3130
000112  000000G                               .WORD   HWQ10
000114  000040                                .WORD   40
000116  000000C                      $NODU:   .WORD   <<<<$LNODU-$NODU>*400>+4>+40>
000120  003130                       GP$6::   .WORD   3130
000122  000000G                               .WORD   HWQ11
000124  000100                                .WORD   100
000126  001004                       $LNODU:  .WORD   1004
000130  003130                       GP$7::   .WORD   3130
000132  000000G                               .WORD   HWQ5
000134  000200                                .WORD   200
000136  000000C                      $TOQ8:   .WORD   <<<<$LTOQ8-$TOQ8>*400>+4>+20>
000140  004032                       GP$8::   .WORD   4032
000142  000000G                               .WORD   HWQ6A
000144  177777                                .WORD   -1
000146  000000                                .WORD   0
```

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

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

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


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

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

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

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

     100000                BIT15==          -100000
```

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

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

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

000000  004137  000000G         LRPT:   JSR     R1,$SAVE4               ;                               5220
000004  104440                          TRAP    40                      ;                               5237
000006  010004                          MOV     R0,R4                   ; *.CUR.PRIORITY
000010  013700  000000G                 MOV     BRLEVEL,R0              ;                               5239
000014  104441                          TRAP    41
000016  012746  000000G                 MOV     #RPT1,-(SP)             ;                               5241
000022  012746  000001                  MOV     #1,-(SP)
000026  010600                          MOV     SP,R0                   ; SP,*
000030  104416                          TRAP    16
000032  012716  000000G                 MOV     #RPT2,(SP)              ;                               5242
000036  012746  000001                  MOV     #1,-(SP)
000042  010600                          MOV     SP,R0                   ; SP,*
000044  104416                          TRAP    16
000046  012716  000000G                 MOV     #RPT3,(SP)              ;                               5243
000052  012746  000001                  MOV     #1,-(SP)
000056  010600                          MOV     SP,R0                   ; SP,*
000060  104416                          TRAP    16
000062  012716  000000G                 MOV     #RPT4,(SP)              ;                               5244
000066  012746  000001                  MOV     #1,-(SP)
000072  010600                          MOV     SP,R0                   ; SP,*
000074  104416                          TRAP    16
000076  012716  000000G                 MOV     #RPT5,(SP)              ;                               5245
000102  012746  000001                  MOV     #1,-(SP)
000106  010600                          MOV     SP,R0                   ; SP,*
000110  104416                          TRAP    16
000112  012716  000000G                 MOV     #RPT6,(SP)              ;                               5246
000116  012746  000001                  MOV     #1,-(SP)
000122  010600                          MOV     SP,R0                   ; SP,*
000124  104416                          TRAP    16
000126  005002                          CLR     R2                      ; CTLR                          5248
000130  010216                  1$:     MOV     R2,(SP)                 ; CTLR,*                        5251
000132  004737  000000V                 JSR     PC,SET.CPAR
000136  012703  000003                  MOV     #3,R3                   ; *,DISK                        5253
000142  010316                  2$:     MOV     R3,(SP)                 ; DISK,*                        5256
000144  004737  000000V                 JSR     PC,SET.UPAR
000150  010301                          MOV     R3,R1                   ; DISK,*                        5259
000152  006301                          ASL     R1
000154  063701  000000G                 ADD     CST.ADDR,R1
000160  032711  040000                  BIT     #40000,(R1)
000164  001535                          BEQ     3$
000166  010216                          MOV     R2,(SP)                 ; CTLR,*                        5264
000170  012746  000053                  MOV     #53,-(SP)
000174  004737  000000G                 JSR     PC,BL$MUL
000200  060300                          ADD     R3,R0                   ; DISK,*
000202  006300                          ASL     R0
000204  062700  000000G                 ADD     #CST,R0
000210  010016                          MOV     R0,(SP)
```

```
000212 062716 000012          ADD     #12,(SP)
000216 111146                 MOVB    (R1),-(SP)
000220 042716 177760          BIC     #177760,(SP)
000224 013746 000000G         MOV     L$LUN,-(SP)
000230 012746 000000G         MOV     #RPT7,-(SP)
000234 012746 000004          MOV     #4,-(SP)
000240 010600                 MOV     SP,R0              ; SP,*
000242 104416                 TRAP    16
000244 013700 000000G         MOV     T.ADDR,R0          ;                  5267
000250 016016 000032          MOV     32(R0),(SP)
000254 016046 000034          MOV     34(R0),-(SP)
000260 016046 000036          MOV     36(R0),-(SP)
000264 016046 000016          MOV     16(R0),-(SP)
000270 016046 000020          MOV     20(R0),-(SP)
000274 012746 000000G         MOV     #RPT8,-(SP)
000300 012746 000006          MOV     #6,-(SP)
000304 010600                 MOV     SP,R0              ; SP,*
000306 104416                 TRAP    16
000310 013700 000000G         MOV     T.ADDR,R0          ;                  5270
000314 016016 000040          MOV     40(R0),(SP)
000320 016046 000042          MOV     42(R0),-(SP)
000324 016046 000044          MOV     44(R0),-(SP)
000330 016046 000024          MOV     24(R0),-(SP)
000334 016046 000026          MOV     26(R0),-(SP)
000340 012746 000000G         MOV     #RPT8,-(SP)
000344 012746 000006          MOV     #6,-(SP)
000350 010600                 MOV     SP,R0              ; SP,*
000352 104416                 TRAP    16
000354 013700 000000G         MOV     T.ADDR,R0          ;                  5273
000360 005016                 CLR     (SP)
000362 116016 000055          MOVB    55(R0),(SP)
000366 005046                 CLR     -(SP)
000370 116016 000054          MOVB    54(R0),(SP)
000374 005046                 CLR     -(SP)
000376 116016 000053          MOVB    53(R0),(SP)
000402 005046                 CLR     -(SP)
000404 116016 000052          MOVB    52(R0),(SP)
000410 005046                 CLR     -(SP)
000412 116016 000051          MOVB    51(R0),(SP)
000416 005046                 CLR     -(SP)
000420 116016 000050          MOVB    50(R0),(SP)
000424 005046                 CLR     -(SP)
000426 116016 000047          MOVB    47(R0),(SP)
000432 005046                 CLR     -(SP)
000434 116016 000046          MOVB    46(R0),(SP)
000440 012746 000000G         MOV     #RPT9,-(SP)
000444 012746 000011          MOV     #11,-(SP)
000450 010600                 MOV     SP,R0              ; SP,*
000452 104416                 TRAP    16
000454 062706 000064          ADD     #64,SP             ;                  5262
000460 062703 000012     3$:  ADD     #12,R3             ; *,DISK           5253
000464 020327 000041          CMP     R3,#41             ; DISK,*
000470 003624                 BLE     2$
```

```
000472  010216                          MOV     R2,(SP)                 ; CTLR,*                5278
000474  012746  000126                  MOV     #126,-(SP)
000500  004737  000000G                 JSR     PC,BL$MUL
000504  005726                          TST     (SP)+
000506  005760  000002G                 TST     CST+2(RO)
000512  100026                          BPL     4$
000514  012716  000000G                 MOV     #RPT10,(SP)             ;                       5282
000520  012746  000001                  MOV     #1,-(SP)
000524  010600                          MOV     SP,RO                   ; SP,*
000526  104416                          TRAP    16
000530  010200                          MOV     R2,RO                   ; CTLR,*                5283
000532  006300                          ASL     RO
000534  005016                          CLR     (SP)
000536  116016  000001G                 MOVB    C.ERR.TBL+1(RO),(SP)
000542  005046                          CLR     -(SP)
000544  116016  000000G                 MOVB    C.ERR.TBL(RO),(SP)
000550  012746  000000G                 MOV     #RPT11,-(SP)
000554  012746  000003                  MOV     #3,-(SP)
000560  010600                          MOV     SP,RO                   ; SP,*
000562  104416                          TRAP    16
000564  062706  000010                  ADD     #10,SP                  ;                       5281
000570  005202                  4$:     INC     R2                      ; CTLR                  5248
000572  000243                          .WORD CLV!CLC

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

```
000720  006301                          ASL     R1
000722  063701  000000G                 ADD     CST.ADDR,R1
000726  132711  000020                  BITB    #20,(R1)
000732  001432                          BEQ     8$
000734  032711  040000                  BIT     #40000,(R1)
000740  001427                          BEQ     8$
000742  013700  000000G                 MOV     T.ADDR,R0                       ;                       5309
000746  016016  000056                  MOV     56(R0),(SP)
000752  016046  000060                  MOV     60(R0),-(SP)
000756  016046  000062                  MOV     62(R0),-(SP)
000762  016046  000064                  MOV     64(R0),-(SP)
000766  111146                          MOVB    (R1),-(SP)
000770  042716  177760                  BIC     #177760,(SP)
000774  013746  000000G                 MOV     L$LUN,-(SP)
001000  012746  000000G                 MOV     #RPT16,-(SP)
001004  012746  000007                  MOV     #7,-(SP)
001010  010600                          MOV     SP,R0                           ; SP,*
001012  104416                          TRAP    16
001014  062706  000016                  ADD     #16,SP
001020  062702  000012          8$:     ADD     #12,R2                          ; *,DISK                5302
001024  020227  000041                  CMP     R2,#41                          ; DISK,*
001030  003727                          BLE     7$
001032  005203                          INC     R3                              ; CTLR                  5299
001034  000243                          .WORD CLV!CLC

001036  003717                          BLE     6$
001040  012716  000000G                 MOV     #CRLF,(SP)                      ;                       5318
001044  012746  000001                  MOV     #1,-(SP)
001050  010600                          MOV     SP,R0                           ; SP,*
001052  104416                          TRAP    16
001054  062706  000012                  ADD     #12,SP                          ;                       5294
001060  062706  000016          9$:     ADD     #16,SP                          ;                       5220
001064  000207                          RTS     PC

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



                                        .SBTTL  L$RPT REPORT CODING SECTION
000000  004737  000530'         L$RPT:: JSR     PC,LRPT                         ;                       5319
000004  104425                          TRAP    25
000006  000207                          RTS     PC

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


;     5322  1
```

```
;  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 (PRI07);                                         ! 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_01);                                        ! "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
```

```
                        CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =     !ZZZ
                                (.HWPT_REF [HWP_DISK_DUPWT]);            !ZZZ
                        CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;      !ZZZ
                        FLAG = FOUND;
                        exitloop;
                        end
                    else
                        begin                                                        ! DUPLICATE UNIT
                        PRINTF (CER_01, .HWPT_REF [HWP_DISK_NUM], .HWPT_REF [HWP_IP_ADDR]);
                                                                                     ! "DUPLICATE UNIT: X AT IP: XXXXXX"
                        DUR [.UNIT] = DU_CONF;                                       ! CONFIGURATION ERROR
                        DODU (.UNIT);                                                ! DROP UNIT
                        FLAG = FOUND;
                        exitloop;
                        end;

            if .FLAG eql NOT_FOUND                                                   ! IF NO IP MATCH TO EXISTING CST
            then
                begin

                incr CTLR from 0 to (MAX_CTLR - 1) do                                ! LOOP THROUGH EACH CST

                    if .CST [.CTLR, IP_ADDR] eqla 0                                  ! IF EMPTY CST FOUND
                    then
                        begin
                        CST [.CTLR, IP_ADDR] = .HWPT_REF [HWP_IP_ADDR];
                        CST [.CTLR, VEC_ADDR] = .HWPT_REF [HWP_VECTOR];
                        CST [.CTLR, BR_LEV] = .HWPT_REF [HWP_BR_LEVEL];
                        TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
                        CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
                                                                                     ! COPY DISK ADDR AND PROT BIT
                        CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
                        CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
                        CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;

                        IF .HWPT_REF [HWP_ENTIRE] EQL TRUE               !ZZZ IF DEFAULT TEST RANGE,
                        THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;         !ZZZ MAKE HI ADDR ALL ONES

                        CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =            !ZZZ
                                .HWPT_REF [HWP_BEG_TRK];                 !ZZZ
                        CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =           !ZZZ
                                .HWPT_REF [HWP_BEG_TRK1];                !ZZZ
                        CST [.CTLR, .TEMP + OF_END, D_END0] =            !ZZZ
                                .HWPT_REF [HWP_END_TRK];                 !ZZZ
                        CST [.CTLR, .TEMP + OF_END1, D_END1] =           !ZZZ
                                .HWPT_REF [HWP_END_TRK1];                !ZZZ

                        CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;         !ZZZ BLANK NAME
                        CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;         !ZZZ BLANK NAME


                        CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;     !ZZZ
                        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
```

```
;  3                  CTLR_CNT = .CTLR_CNT + 1;                                          ! INCREMENT CONTROLLER COUNT
;  3
;  3          MEMORY (FREE_MEM_ADDR);                                                    ! GET START OF FREE MEMORY
;  3
;  2          end;                                                                       ! END OF "START" INITIALIZATION
;  2
;  2      !+
;  2      !       CLEAR STATISTICS TABLES
;  2      !-
;  2
;  2      incr UNITS from 0 to MAX_UNITS - 1 do                                          ! CLEAR CURRENT STATISTICS
;  2          incr COUNT from 0 to TALLY_CLEAR - 1 do
;  2              TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;
;  2
;  2      if .CLEAR_TABLES                                                               ! IF CLEAR TABLES ON EVERY PASS
;  2      then
;  2          incr UNITS from 0 to MAX_UNITS - 1 do
;  2              incr COUNT from TALLY_CLEAR to TALLY_LEN - 1 do                        ! INITIALIZE TOTALS
;  2                  TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;                           !
;  2
;  2      if .CLEAR_TABLES
;  2      then
;  2          incr CTLR from 0 to MAX_CTLR - 1 do
;  3              begin
;  3              C_ERR_TBL [.CTLR, C_ERR_HRD] = 0;                                      ! INITALIZE CONTROLLER ERRORS
;  3              C_ERR_TBL [.CTLR, C_ERR_SFT] = 0;                                      !
;  2              end;
;  2
;  2      !+
;  2      !       MISCELLANEOUS INITIALIZATON
;  2      !-
;  2
;  2      incr CTLR from 0 to (MAX_CTLR - 1) do                                          ! INIT NO. OF OUTSTANDING QIOs
;  2          QIO [.CTLR] = 0;
;  2
;  2      incr COUNT from 0 to (RP_CNT - 1) do                                           ! INITIALIZE RETURN PACKET POOL
;  2          RP_USE [.COUNT] = -1;
;  2
;  2      if .CLK_PRESENT                                                                ! STOP CLOCK IF PRESENT
;  2      then
;  2          LINE_CLOCK = 0;
;  2
;  2      IODQ_IN = IODQ_OUT = 0;                                                        ! INIT I/O DONE QUEUE POINTERS
;  2      CRN_LOW = CRN_HIGH = 0;                                                        ! INIT COMMAND REFERENCE NUMBER
;  2      SETPRI (PRIO0);                                                                ! SET PROGRAM PRIORITY TO 0
;  2
;  1      ENDINIT;

                              .GLOBL  L$DLY

                              .SBTTL  LINIT INITIALIZE SECTION
```

G13


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

H13


```
000276  005066  000024          9$:     CLR     24(SP)                  ; $$TMP
000302  005300                          DEC     R0                      ; $$TMP1
000304  001374                          BNE     9$
000306  005301                  10$:    DEC     R1                      ; $$TMP2
000310  000766                          BR      8$
000312  104422                  11$:    TRAP    22
000314  005302                          DEC     R2                      ; COUNT                         5391
000316  001360                          BNE     7$
000320  022626                          CMP     (SP)+,(SP)+             ;                               5384
000322  023727  000000G 000004  12$:    CMP     L$UNIT,#4               ;                               5408
000330  101405                          BLOS    13$
000332  104454                          TRAP    54                      ;                               5411
000334  000001                          .WORD   1
000336  000000G                         .WORD   EGS.01
000340  000000V                         .WORD   EMS.01
000342  104444                          TRAP    44
000344  123727  000000G 000005  13$:    CMPB    ENTRY.REASON,#5         ;                               5421
000352  001002                          BNE     14$
000354  000137  003726'                 JMP     43$
000360  112766  000377  000010  14$:    MOVB    #377,10(SP)             ; *,SMALLEST.DRIVE              5424
000366  005000                          CLR     R0                      ; COUNT                         5426
000370  005060  000000G         15$:    CLR     CST(R0)                 ; *(COUNT)                      5427
000374  062700  000002                  ADD     #2,R0                   ; *,COUNT                       5426
000400  020027  000124                  CMP     R0,#124                 ; COUNT,*
000404  003771                          BLE     15$
000406  013704  000000G                 MOV     L$UNIT,R4               ;                               5429
000412  005003                          CLR     R3                      ; UNIT
000414  000435                          BR      18$
000416  010302                  16$:    MOV     R3,R2                   ; UNIT,*                        5431
000420  006302                          ASL     R2
000422  012700  000022                  MOV     #22,R0
000426  060600                          ADD     SP,R0                   ; HWPT.ADDRESS,*
000430  060002                          ADD     R0,R2
000432  010300                          MOV     R3,R0                   ; UNIT,*
000434  104442                          TRAP    42
000436  010001                          MOV     R0,R1                   ; *,HWPT.REF
000440  010112                          MOV     R1,(R2)                 ; HWPT.REF,*
000442  001421                          BEQ     17$
000444  005002                          CLR     R2                      ;                               5434
000446  156602  000010                  BISB    10(SP),R2               ; SMALLEST.DRIVE,*
000452  116100  000006                  MOVB    6(R1),R0                ; *(HWPT.REF),*
000456  042700  177760                  BIC     #177760,R0
000462  020002                          CMP     R0,R2
000464  103010                          BHIS    17$
000466  116100  000006                  MOVB    6(R1),R0                ; *(HWPT.REF),*                 5436
000472  042700  177760                  BIC     #177760,R0
000476  105066  000010                  CLRB    10(SP)                  ; SMALLEST.DRIVE
000502  050066  000010                  BIS     R0,10(SP)               ; *,SMALLEST.DRIVE
000506  005203                  17$:    INC     R3                      ; UNIT                          5429
000510  020304                  18$:    CMP     R3,R4                   ; UNIT,*
000512  002741                          BLT     16$
000514  013766  000000G 000016          MOV     L$UNIT,16(SP)           ;                               5438
000522  005004                          CLR     R4                      ; UNIT
```

```
000524  000137  003704'                    JMP     41$
000530  010400                     19$:    MOV     R4,R0                  ; UNIT,*                    5440
000532  006300                             ASL     R0
000534  012703  000022                     MOV     #22,R3
000540  060603                             ADD     SP,R3                  ; HWPT.ADDRESS,*
000542  060300                             ADD     R3,R0
000544  005710                             TST     (R0)
000546  001002                             BNE     20$
000550  000137  003702'                    JMP     40$
000554  105066  000006             20$:    CLRB    6(SP)                  ; FLAG                      5443
000560  011001                             MOV     (R0),R1                ; *,HWPT.REF                5444
000562  005066  000002                     CLR     2(SP)                  ; CTLR                      5446
000566  016646  000002             21$:    MOV     2(SP),-(SP)            ; CTLR,*                    5448
000572  012746  000126                     MOV     #126,-(SP)
000576  004737  000000G                    JSR     PC,BL$MUL
000602  022626                             CMP     (SP)+,(SP)+
000604  026011  000000G                    CMP     CST(R0),(R1)           ; *,HWPT.REF
000610  001402                             BEQ     22$
000612  000137  003124'                    JMP     28$
000616  012766  000001  000014     22$:    MOV     #1,14(SP)              ;                           5485
000624  112766  000001  000006             MOVB    #1,6(SP)               ; *,FLAG
000632  012705  000006                     MOV     #6,R5                  ;                           5451
000636  060105                             ADD     R1,R5                  ; HWPT.REF,*
000640  111546                             MOVB    (R5),-(SP)
000642  042716  177760                     BIC     #177760,(SP)
000646  005000                             CLR     R0
000650  156600  000012                     BISB    12(SP),R0              ; SMALLEST.DRIVE,*
000654  160016                             SUB     R0,(SP)
000656  012746  000012                     MOV     #12,-(SP)
000662  004737  000000G                    JSR     PC,BL$MUL
000666  010066  000010                     MOV     R0,10(SP)
000672  005726                             TST     (SP)+
000674  016616  000004                     MOV     4(SP),(SP)             ; CTLR,*                    5452
000700  012746  000053                     MOV     #53,-(SP)
000704  004737  000000G                    JSR     PC,BL$MUL
000710  010003                             MOV     R0,R3
000712  022626                             CMP     (SP)+,(SP)+
000714  066600  000004                     ADD     4(SP),R0
000720  006300                             ASL     R0
000722  032760  040000  000006G            BIT     #40000,CST+6(R0)
000730  001140                             BNE     27$
000732  016602  000004                     MOV     4(SP),R2               ; *,TEMP                    5455
000736  062702  000003                     ADD     #3,R2                  ; *,TEMP
000742  010300                             MOV     R3,R0                  ;                           5456
000744  060200                             ADD     R2,R0                  ; TEMP,*
000746  006300                             ASL     R0
000750  062700  000000G                    ADD     #CST,R0
000754  011510                             MOV     (R5),(R0)
000756  010446                             MOV     R4,-(SP)               ; UNIT,*                    5458
000760  000316                             SWAB    (SP)
000762  042716  170377                     BIC     #170377,(SP)
000766  042710  007400                     BIC     #7400,(R0)
000772  052610                             BIS     (SP)+,(R0)
```

```
000774  042710  010000                  BIC     #10000,(R0)             ;                               5459
001000  052710  040000                  BIS     #40000,(R0)             ;                               5460
001004  105715                          TSTB    (R5)                    ;                               5462
001006  100003                          BPL     23$
001010  012761  177777  000016          MOV     #-1,16(R1)              ; *,*(HWPT.REF)                 5463
001016  010300                  23$:    MOV     R3,R0                   ;                               5465
001020  060200                          ADD     R2,R0                   ; TEMP,*
001022  006300                          ASL     R0
001024  016160  000010  000002G         MOV     10(R1),CST+2(R0)        ; *(HWPT.REF),*
001032  010300                          MOV     R3,R0                   ;                               5467
001034  060200                          ADD     R2,R0                   ; TEMP,*
001036  006300                          ASL     R0
001040  016160  000012  000004G         MOV     12(R1),CST+4(R0)        ; *(HWPT.REF),*
001046  010300                          MOV     R3,R0                   ;                               5469
001050  060200                          ADD     R2,R0                   ; TEMP,*
001052  006300                          ASL     R0
001054  016160  000014  000006G         MOV     14(R1),CST+6(R0)        ; *(HWPT.REF),*
001062  010300                          MOV     R3,R0                   ;                               5471
001064  060200                          ADD     R2,R0                   ; TEMP,*
001066  006300                          ASL     R0
001070  016160  000016  000010G         MOV     16(R1),CST+10(R0)       ; *(HWPT.REF),*
001076  010300                          MOV     R3,R0                   ;                               5474
001100  060200                          ADD     R2,R0                   ; TEMP,*
001102  006300                          ASL     R0
001104  011660  000012G                 MOV     (SP),CST+12(R0)         ; BLANKS,*
001110  010300                          MOV     R3,R0                   ;                               5475
001112  060200                          ADD     R2,R0                   ; TEMP,*
001114  006300                          ASL     R0
001116  011660  000014G                 MOV     (SP),CST+14(R0)         ; BLANKS,*
001122  010300                          MOV     R3,R0                   ;                               5478
001124  060200                          ADD     R2,R0                   ; TEMP,*
001126  006300                          ASL     R0
001130  062700  000020G                 ADD     #CST+20,R0
001134  105010                          CLRB    (R0)
001136  111546                          MOVB    (R5),-(SP)              ;                               5480
001140  005046                          CLR     -(SP)
001142  032766  000040  000002          BIT     #40,2(SP)
001150  001401                          BEQ     24$
001152  005216                          INC     (SP)
001154  005116                  24$:    COM     (SP)
001156  011646                          MOV     (SP),-(SP)
001160  042710  100000                  BIC     #100000,(R0)
001164  006026                          ROR     (SP)+
001166  103002                          BCC     25$
001170  052710  100000                  BIS     #100000,(R0)
001174  005726                  25$:    TST     (SP)+
001176  111516                          MOVB    (R5),(SP)               ;                               5482
001200  042710  010000                  BIC     #10000,(R0)
001204  032726  000100                  BIT     #100,(SP)+
001210  001402                          BEQ     26$
001212  052710  010000                  BIS     #10000,(R0)
001216  010300                  26$:    MOV     R3,R0                   ;                               5484
001220  060200                          ADD     R2,R0                   ; TEMP,*
```

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

001304  003002                                     BGT     29$
001306  000137  002414'                            JMP     21$
001312  105766  000006                     29$:    TSTB    6(SP)                      ; FLAG                         5498
001316  001402                                     BEQ     30$
001320  000137  003702'                            JMP     40$
001324  005066  000014                     30$:    CLR     14(SP)                     ; CTLR                         5502
001330  016646  000014                     31$:    MOV     14(SP),-(SP)               ; CTLR,*                       5504
001334  012746  000126                             MOV     #126,-(SP)
001340  004737  000000G                            JSR     PC,BL$MUL
001344  022626                                     CMP     (SP)+,(SP)+
001346  005760  000000G                            TST     CST(R0)
001352  001402                                     BEQ     32$
001354  000137  003622'                            JMP     37$
001360  011160  000000G                    32$:    MOV     (R1),CST(R0)               ; HWPT.REF,*                   5507
001364  016103  000002                             MOV     2(R1),R3                   ; *(HWPT.REF),*                5508
001370  042703  177000                             BIC     #177000,R3
001374  042760  000777  000002G                    BIC     #777,CST+2(R0)
001402  050360  000002G                            BIS     R3,CST+2(R0)
001406  116160  000004  000004G                    MOVB    4(R1),CST+4(R0)            ; *(HWPT.REF),*                5509
001414  012705  000006                             MOV     #6,R5                      ;                              5510
001420  060105                                     ADD     R1,R5                      ; HWPT.REF,*
001422  111546                                     MOVB    (R5),-(SP)
001424  042716  177760                             BIC     #177760,(SP)
001430  005000                                     CLR     R0
001432  156600  000012                             BISB    12(SP),R0                  ; SMALLEST.DRIVE,*
001436  160016                                     SUB     R0,(SP)
001440  012746  000012                             MOV     #12,-(SP)
001444  004737  000000G                            JSR     PC,BL$MUL
001450  005726                                     TST     (SP)+
001452  010002                                     MOV     R0,R2                      ; *,TEMP
001454  062702  000003                             ADD     #3,R2                      ; *,TEMP
001460  016616  000016                             MOV     16(SP),(SP)                ; CTLR,*                       5511
001464  012746  000053                             MOV     #53,-(SP)
001470  004737  000000G                            JSR     PC,BL$MUL
001474  010003                                     MOV     R0,R3
```

```
001476  005726                          TST     (SP)+
001500  060200                          ADD     R2,R0                   ; TEMP,*
001502  006300                          ASL     R0
001504  062700  000000G                 ADD     #CST,R0
001510  011510                          MOV     (R5),(R0)
001512  010416                          MOV     R4,(SP)                 ; UNIT,*                        5513
001514  000316                          SWAB    (SP)
001516  042716  170377                  BIC     #170377,(SP)
001522  042710  007400                  BIC     #7400,(R0)
001526  052610                          BIS     (SP)+,(R0)
001530  042710  010000                  BIC     #10000,(R0)             ;                               5514
001534  052710  040000                  BIS     #40000,(R0)             ;                               5515
001540  105715                          TSTB    (R5)                    ;                               5517
001542  100003                          BPL     33$
001544  012761  177777  000016          MOV     #-1,16(R1)              ; *,*(HWPT.REF)                 5518
001552  010300                  33$:    MOV     R3,R0                   ;                               5520
001554  060200                          A )D    R2,R0                   ; TEMP,*
001556  006300                          ASL     R0
001560  016160  000010  000002G         MOV     10(R1),CST+2(R0)        ; *(HWPT.REF),*
001566  010300                          MOV     R3,R0                   ;                               5522
001570  060200                          ADD     R2,R0                   ; TEMP,*
001572  006300                          ASL     R0
001574  016160  000012  000004G         MOV     12(R1),CST+4(R0)        ; *(HWPT.REF),*
001602  010300                          MOV     R3,R0                   ;                               5524
001604  060200                          ADD     R2,R0                   ; TEMP,*
001606  006300                          ASL     R0
001610  016160  000014  000006G         MOV     14(R1),CST+6(R0)        ; *(HWPT.REF),*
001616  010300                          MOV     R3,R0                   ;                               5526
001620  060200                          ADD     R2,R0                   ; TEMP,*
001622  006300                          ASL     R0
001624  016160  000016  000010G         MOV     16(R1),CST+10(R0)       ; *(HWPT.REF),*
001632  010300                          MOV     R3,R0                   ;                               5529
001634  060200                          ADD     R2,R0                   ; TEMP,*
001636  006300                          ASL     R0
001640  011660  000012G                 MOV     (SP),CST+12(R0)         ; BLANKS,*
001644  010300                          MOV     R3,R0                   ;                               5530
001646  060200                          ADD     R2,R0                   ; TEMP,*
001650  006300                          ASL     R0
001652  011660  000014G                 MOV     (SP),CST+14(R0)         ; BLANKS,*
001656  010300                          MOV     R3,R0                   ;                               5533
001660  060200                          ADD     R2,R0                   ; TEMP,*
001662  006300                          ASL     R0
001664  062700  000020G                 ADD     #CST+20,R0
001670  105010                          CLRB    (R0)
001672  111546                          MOVB    (R5),-(SP)              ;                               5535
001674  005046                          CLR     -(SP)
001676  032766  000040  000002          BIT     #40,2(SP)
001704  001401                          BEQ     34$
001706  005216                          INC     (SP)
001710  005116                  34$:    COM     (SP)
001712  011646                          MOV     (SP),-(SP)
001714  042710  100000                  BIC     #100000,(R0)
001720  006026                          ROR     (SP)+
```

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

002002  003002                                BGT     38$
002004  000137  003156'                       JMP     31$
002010  105766  000006                38$:    TSTB    6(SP)                   ; FLAG                  5543
002014  001017                        39$:    BNE     40$
002016  012746  000001                        MOV     #1,-(SP)                ;                       5546
002022  012746  000000G                       MOV     #CER.02,-(SP)
002026  012746  000002                        MOV     #2,-(SP)
002032  010600                                MOV     SP,R0                   ; SP,*
002034  104417                                TRAP    17
002036  112764  000001  000000G               MOVB    #1,DUR(R4)              ; *,*(UNIT)             5547
002044  010400                                MOV     R4,R0                   ; UNIT,*                5548
002046  104451                                TRAP    51
002050  062706  000006                        ADD     #6,SP                   ;                       5545
002054  005204                        40$:    INC     R4                      ; UNIT                  5438
002056  020466  000016                41$:    CMP     R4,16(SP)               ; UNIT,*
002062  002002                                BGE     42$
002064  000137  002356'                       JMP     19$
002070  123727  000000G 000005        42$:    CMPB    ENTRY.REASON,#5         ;                       5561
002076  001051                                BNE     48$
002100  013703  000000G               43$:    MOV     L$UNIT,R3               ;                       5565
002104  005004                                CLR     R4                      ; UNIT
002106  000404                                BR      45$
002110  010400                        44$:    MOV     R4,R0                   ; UNIT,*                5566
002112  104442                                TRAP    42
002114  010001                                MOV     R0,R1                   ; *,HWPT.REF
002116  005204                                INC     R4                      ; UNIT                  5565
002120  020403                        45$:    CMP     R4,R3                   ; UNIT,*
002122  002772                                BLT     44$
002124  005003                                CLR     R3                      ; CTLR                  5568
002126  010346                        46$:    MOV     R3,-(SP)                ; CTLR,*                5570
002130  012746  000126                        MOV     #126,-(SP)
002134  004737  000000G                       JSR     PC,BL$MUL
002140  105060  000005G                       CLRB    CST+5(R0)
002144  010316                                MOV     R3,(SP)                 ; CTLR,*                5573
002146  012746  000053                        MOV     #53,-(SP)
002152  004737  000000G                       JSR     PC,BL$MUL
```

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

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

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

```
002410  001411                          BEQ     57$
002412  005000                          CLR     R0                              ; CTLR                          5610
002414  105060  000000G         56$:    CLRB    C.ERR.TBL(R0)                   ; .(CTLR)                       5612
002420  105060  000001G                 CLRB    C.ERR.TBL+1(R0)                 ; .(CTLR)                       5613
002424  062700  000002                  ADD     #2,R0                           ; .,CTLR                        5610
002430  000243                          .WORD CLV!CLC

002432  003770                          BLE     56$
002434  005000                  57$:    CLR     R0                              ; CTLR                          5620
002436  105060  000000G         58$:    CLRB    QIO(R0)                         ; .(CTLR)                       5621
002442  005200                          INC     R0                              ; CTLR                          5620
002444  000243                          .WORD CLV!CLC

002446  003773                          BLE     58$
002450  005000                          CLR     R0                              ; COUNT                         5623
002452  112760  000377  000000G 59$:    MOVB    #377,RP.USE(R0)                 ; .,.(COUNT)                    5624
002460  005200                          INC     R0                              ; COUNT                         5623
002462  020027  000007                  CMP     R0,#7                           ; COUNT,.
002466  003771                          BLE     59$
002470  132737  000001  000000G         BITB    #1,CLK.PRESENT                  ;                               5626
002476  001402                          BEQ     60$
002500  005037  177546                  CLR     @#177546                        ;                               5628
002504  005037  000000G         60$:    CLR     IODQ.OUT                        ;                               5630
002510  005037  000000G                 CLR     IODQ.IN
002514  005037  000000G                 CLR     CRN.HIGH                        ;                               5631
002520  005037  000000G                 CLR     CRN.LOW
002524  005000                          CLR     R0                              ;                               5632
002526  104441                          TRAP    41
002530  062706  000032                  ADD     #32,SP                          ;                               5321
002534  000207                          RTS     PC

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




                                        .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  001626'         L$INIT::JSR     PC,LINIT                        ;                               5632
000004  104411                          TRAP    11
000006  000207                          RTS     PC

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

```
;       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       ! SEE 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                              ;               5634

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


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

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

```
%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

                incr COUNT from 1 to 10000 do
                    begin
                    DELAY (1);

                    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;
```

```
;   5706  4               WRT_RDRX (RCIP, RC_ALL, ALL_ONES);          ! WRITE IP TO STOP DEVICE
;   5707  3               end;
;   5708  3
;   5709  3           CLRVEC (.CST[.CTLR, VEC_ADDR]);                 ! RETURN CONTROLLER'S TRAP VECTOR TO SUPERVISOR
;   5710  2           end;
;   5711  2
;   5712  1       ENDCLN;


                                                .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  004137  000000G                 LCLEAN: JSR     R1,$SAVE5                       ;                               5652
000004  005746                                  TST     -(SP)
000006  104424                                  TRAP    24                              ;                               5663
000010  132737  000001  000000G                 BITB    #1,CLK.PRESENT                  ;                               5669
000016  001402                                  BEQ     1$
000020  005037  177546                          CLR     @#177546                        ;                               5672
000024  005005                          1$:     CLR     R5                              ; CTLR                          5676
000026  010546                          2$:     MOV     R5,-(SP)                        ; CTLR,*                        5678
000030  012746  000126                          MOV     #126,-(SP)
000034  004737  000000G                         JSR     PC,BL$MUL
000040  010003                                  MOV     R0,R3
000042  022626                                  CMP     (SP)+,(SP)+
000044  016337  000000G 000000G                 MOV     CST(R3),RDRX.ADDR
000052  001477                                  BEQ     13$
000054  132737  000001  000000G                 BITB    #1,ADDR.VECT.OK                 ;                               5682
000062  001466                                  BEQ     12$
000064  010546                                  MOV     R5,-(SP)                        ; CTLR,*                        5686
000066  012746  000022                          MOV     #22,-(SP)
000072  004737  000000G                         JSR     PC,BL$MUL
000076  022626                                  CMP     (SP)+,(SP)+
000100  005760  000000G                         TST     DCT(R0)
000104  100051                                  BPL     11$
000106  012704  023420                          MOV     #23420,R4                       ; *,COUNT                       5689
000112  012702  000001                  3$:     MOV     #1,R2                           ; *,$$TMP2                      5691
000116  001410                          4$:     BEQ     7$
000120  013701  000000G                         MOV     L$DLY,R1                        ; *,$$TMP1
000124  001403                                  BEQ     6$
000126  005016                          5$:     CLR     (SP)                            ; $$TMP
000130  005301                                  DEC     R1                              ; $$TMP1
000132  001375                                  BNE     5$
000134  005302                          6$:     DEC     R2                              ; $$TMP2
000136  000767                                  BR      4$
000140  105760  000000G                 7$:     TSTB    DCT(R0)                         ;                               5693
000144  001027                                  BNE     10$
000146  005001                                  CLR     R1                              ; Z                             5696
000150  016037  000004G 000000G         8$:     MOV     DCT+4(R0),TEMP1                 ;                               5698
000156  060137  000000G                         ADD     R1,TEMP1                        ; Z,*
000162  017737  000000G 000000G                 MOV     @TEMP1,TEMP2                    ;                               5699
000170  027727  000000G 000000G                 CMP     @TEMP2,#CRN.LOW                 ;                               5700
000176  001005                                  BNE     9$
000200  012702  177777                          MOV     #-1,R2                          ; *,RC.REG                      5702
000204  010277  000000G                         MOV     R2,@RDRX.ADDR                   ; RC.REG,*
000210  000413                                  BR      12$
```

F14


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

000256  003663                           BLE     2$
000260  005726                           TST     (SP)+                      ;                   5652
000262  000207                           RTS     PC

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



                                         .SBTTL  L$CLEAN CLEANUP CODING SECTION
000000  004737  004406'         L$CLEAN::
                                         JSR     PC,LCLEAN                  ;                   5710
000004  104412                           TRAP    12
000006  000207                           RTS     PC

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

```
;  1    %sbttl 'DROP UNIT SECTION'
;  1
;  1    !+
;  1    ! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;  1    ! TO NO LONGER BE TESTED.
;  1    !-
;  1
;  2    BGNDU;
;  2
;  2    local
;  2         UNIT : word,                                                              ! UNIT NUMBER
;  2         PRINT : byte initial (byte (FALSE));                                      ! NO PRINTING
;  2
;  2    label
;  2         SEARCH;
;  2
;  3    begin
;  3
;  3    register
;  3         INPUT = 0;                                                                ! R0 = UNIT NO.
;  3
;  3    UNIT = .INPUT;                                                                 ! GET UNIT NUMBER
;  2    end;                                                                           ! UNDECLARE R0
;  2
;  2    !ZZZif BIT_TST (SWP_FLAGS, SWF_TRC)
;  2    !ZZZthen
;  2    !ZZZ    PRINTF (DBM5, .UNIT);
;  2
;  2    SEARCH :                                                                       ! SEARCH BLOCK
;  3    begin
;  3
;  3    incr CTLR from 0 to (MAX_CTLR - 1) do                                          ! FOR EACH CNTR
;  3
;  3        incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do   ! FOR EACH UNIT
;  3
;  3            if (.CST [.CTLR, .OFFSET + OF_DATA, D_UNIT] eql .UNIT) and             ! IF UNIT MATCHES
;  4               (.CST [.CTLR, .OFFSET + OF_DATA, D_PRES] eql PRESENT)
;  3            then
;  4                begin
;  4
;  4                if (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE) or         ! IF UNIT ALIVE
;  4                   (.DUR [.UNIT] eql DU_ONLINE) or
;  5                   (.DUR [.UNIT] eql DU_PROTECT)
;  4                then
;  5                    begin
;  5                    PRINT = TRUE;                                                  ! O.K. TO PRINT
;  5
;  5                    if (.CST [.CTLR, U_CNT] gtru 0) and
;  6                       (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;  5                    then
;  5                        CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] - 1;             ! DECREMENT COUNT
;  5
;  5                    if (.CST [.CTLR, U_CNT] eql 0) and
```

```
;  6                    (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;  5                then
;  5                    EOP_FLAG = TRUE;                                                    ! ALL UNITS OFFLINE
;  5
;  5                CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;                       ! MARK UNIT OFFLINE
;  4                end;                                                                    ! IF UNIT ALIVE
;  4
;  4            leave SEARCH;                                                               ! EXIT SEARCH BLOCK
;  3            end;                                                                        ! IF UNIT FOUND
;  3
;  2    end;
;  2
;  2    if .PRINT or                                                                        ! IF OK TO PRINT
;  2       (.DUR [.UNIT] eql DU_CONF) or
;  2       (.DUR [.UNIT] eql DU_INIT) or
;  2       (.DUR [.UNIT] eql DU_ONLINE) or
;  3       (.DUR [.UNIT] eql DU_PROTECT)
;  2    then
;  3        begin
;  3        PRINTF (DU_MSG, .UNIT);                                                         ! "UNIT XX DROPPED"
;  3        PRINTF (.DU_RSN [.DUR [.UNIT]]);                                                ! REASON
;  2        end;
;  2
;  1    ENDDU;

                                             .SBTTL  LDU DROP UNIT SECTION
000000  004137  000000G          LDU:        JSR     R1,$SAVE5               ;                      5712
000004  024646                               CMP     -(SP),-(SP)
000006  105066  000002                       CLRB    2(SP)                   ; PRINT
000012  010001                               MOV     R0,R1                   ; INPUT,UNIT           5734
000014  005005                               CLR     R5                      ; CTLR                 5744
000016  010546                   1$:         MOV     R5,-(SP)                ; CTLR,*               5748
000020  012746  000053                       MOV     #53,-(SP)
000024  004737  000000G                      JSR     PC,BL$MUL
000030  010066  000004                       MOV     R0,4(SP)
000034  012703  000003                       MOV     #3,R3                   ; *,OFFSET             5746
000040  010300                   2$:         MOV     R3,R0                   ; OFFSET,*             5748
000042  066600  000004                       ADD     4(SP),R0
000046  006300                               ASL     R0
000050  012702  000000G                      MOV     #CST,R2
000054  060002                               ADD     R0,R2
000056  010104                               MOV     R1,R4                   ; UNIT,*
000060  011200                               MOV     (R2),R0
000062  000300                               SWAB    R0
000064  042700  177760                       BIC     #177760,R0
000070  020004                               CMP     R0,R4
000072  001055                               BNE     8$
000074  032712  040000                       BIT     #40000,(R2)             ;                      5749
000100  001452                               BEQ     8$
000102  005004                               CLR     R4                      ;                      5753
000104  032712  020000                       BIT     #20000,(R2)
000110  001402                               BEQ     3$
```

```
000112  005204                             INC     R4
000114  000410                             BR      4$
000116  126127  000000G 000007     3$:     CMPB    DUR(R1),#7                 ; *(UNIT),*            5754
000124  001404                             BEQ     4$
000126  126127  000000G 000011             CMPB    DUR(R1),#11                ; *(UNIT),*            5755
000134  001032                             BNE     7$
000136  112766  000001  000006     4$:     MOVB    #1,6(SP)                   ; *,PRINT              5758
000144  010516                             MOV     R5,(SP)                    ; CTLR,*               5760
000146  012746  000126                     MOV     #126,-(SP)
000152  004737  000000G                    JSR     PC,BL$MUL
000156  005726                             TST     (SP)+
000160  062700  000004G                    ADD     #CST+4,R0
000164  105760  000001                     TSTB    1(R0)
000170  001404                             BEQ     5$
000172  006004                             ROR     R4                         ;                      5761
000174  105660  000001                     SBCB    1(R0)                      ;                      5763
000200  001006                             BNE     6$                         ;                      5765
000202  032712  020000             5$:     BIT     #20000,(R2)                ;                      5766
000206  001403                             BEQ     6$
000210  112737  000001  000000G            MOVB    #1,EOP.FLAG                ;                      5768
000216  042712  020000             6$:     BIC     #20000,(R2)                ;                      5770
000222  022626                     7$:     CMP     (SP)+,(SP)+                ;                      5751
000224  000411                             BR      9$
000226  062703  000012             8$:     ADD     #12,R3                     ; *,OFFSET             5746
000232  020327  000041                     CMP     R3,#41                     ; OFFSET,*
000236  003700                             BLE     2$
000240  022626                             CMP     (SP)+,(SP)+
000242  005205                             INC     R5                         ; CTLR                 5744
000244  000243                             .WORD CLV!CLC

000246  003663                             BLE     1$
000250  032766  000001  000002     9$:     BIT     #1,2(SP)                   ; *,PRINT              5778
000256  001020                             BNE     10$
000260  126127  000000G 000001             CMPB    DUR(R1),#1                 ; *(UNIT),*            5779
000266  001414                             BEQ     10$
000270  126127  000000G 000002             CMPB    DUR(R1),#2                 ; *(UNIT),*            5780
000276  001410                             BEQ     10$
000300  126127  000000G 000007             CMPB    DUR(R1),#7                 ; *(UNIT),*            5781
000306  001404                             BEQ     10$
000310  126127  000000G 000011             CMPB    DUR(R1),#11                ; *(UNIT),*            5782
000316  001024                             BNE     11$
000320  010146                     10$:    MOV     R1,-(SP)                   ; UNIT,*               5785
000322  012746  000000G                    MOV     #DU.MSG,-(SP)
000326  012746  000002                     MOV     #2,-(SP)
000332  010600                             MOV     SP,R0                      ; SP,*
000334  104417                             TRAP    17
000336  116101  000000G                    MOVB    DUR(R1),R1                 ; *(UNIT),*            5786
000342  042701  177400                     BIC     #177400,R1
000346  006301                             ASL     R1
000350  016116  000000G                    MOV     DU.RSN(R1),(SP)
000354  012746  000001                     MOV     #1,-(SP)
000360  010600                             MOV     SP,R0                      ; SP,*
000362  104417                             TRAP    17
```

```
000364  062706  000010                  ADD     #10,SP                          ;       5784
000370  022626                  11$:    CMP     (SP)+,(SP)+                     ;       5712
000372  000207                          RTS     PC

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



                                        .SBTTL  L$DU DROP UNIT SECTION
000000  004737  004702'         L$DU::  JSR     PC,LDU                          ;       5787
000004  104453                          TRAP    53
000006  000207                          RTS     PC

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

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


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

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

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



                                        .SBTTL  L$AU ADD UNIT SECTION
000000  004737  005306'          L$AU::  JSR     PC,LAU                            ;                              5816
000004  104452                           TRAP    52
000006  000207                           RTS     PC

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

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


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

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

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


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

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

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

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

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

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


                                                .SBTTL  SET.UPAR GLOBAL ROUTINES
000000  010146                          SET.UPAR::
                                                MOV     R1,-(SP)                ;                       5886
000002  016637  000004  000000G                 MOV     4(SP),CUOFF             ; OFFSET,*              5905
000010  016600  000004                          MOV     4(SP),R0                ; CUOFF,*               5906
000014  006300                                  ASL     R0
000016  063700  000000G                         ADD     CST.ADDR,R0
000022  111037  000000G                         MOVB    (R0),CDISK
000026  042737  177760  000000G                 BIC     #177760,CDISK
000034  011001                                  MOV     (R0),R1                 ;                       5907
000036  000301                                  SWAB    R1
000040  042701  177760                          BIC     #177760,R1
000044  010137  000000G                         MOV     R1,L$LUN
000050  010146                                  MOV     R1,-(SP)                ; L$LUN,*               5908
000052  012746  000066                          MOV     #66,-(SP)
000056  004737  000000G                         JSR     PC,BL$MUL
000062  062700  000000G                         ADD     #TALLY,R0
000066  010037  000000G                         MOV     R0,T.ADDR
000072  022626                                  CMP     (SP)+,(SP)+             ;                       5904
000074  012601                                  MOV     (SP)+,R1                ;                       5886
000076  000207                                  RTS     PC

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

```
;  1
;  1    global routine GET_PKT (CTLR) =
;  1
;  1    !+
;  1    !       THIS ROUTINE SEARCHES THE MSCP PACKET POOL ALLOCATION TABLE (PKT_USE)
;  1    !       FOR A FREE MSCP PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;  1    !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED
;  1    !       TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;  1    !
;  1    !       INPUTS:
;  1    !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;  1    !-
;  1
;  2        begin
;  2
;  2        local
;  2            index : signed word initial (-1),
;  2            RING_ADDR : word,
;  2            PACKET_OWNED : byte,
;  2            NEXT_PACKET : byte;
;  2
;  2
;  2        NEXT_PACKET = .NEXT_PKT_USE;                            ! NEXT PACKET TO TRY
;  2
;  2        incr COUNT from 0 to (PKT_CNT - 1) do                   ! FOR EACH ENTRY IN ALLOCATION TABLE
;  3            begin
;  3            PACKET_OWNED = FALSE;
;  3
;  3            if .PKT_USE [.NEXT_PACKET] lss 0                    ! IF ENTRY INDICATES FREE PACKET
;  3
;  3            then
;  4                begin
;  4                RING_ADDR = .DCT_ADDR [RR_BEG];                 ! FIRST RESPONSE PACKET'S ADDRESS
;  4
;  4                incr I from 1 to (RRING_LEN + CRING_LEN) do ! FOR EACH PACKET ADDRESS
;  4
;  4                    if (..RING_ADDR eqla .MSCP_PKT [.NEXT_PACKET, PKT_LO]) and
;  5                       (((.RING_ADDR + 2) and ED_OWN) eql ED_OWN)
;  5
;  4                    then
;  5                        begin                                   ! CHECK ADDRESS AND OWNERSHIP
;  5                        PACKET_OWNED = TRUE;                    ! PACKET OWNED BY CONTROLLER
;  5                        exitloop;
;  5                        end
;  4                    else
;  4                        RING_ADDR = .RING_ADDR + 4;             ! ADDRESS OF NEXT PACKET IN RING
;  4
;  4                if not .PACKET_OWNED                            ! IF NOT ALREADY USED
;  4
;  4                then
;  5                    begin
;  5                    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,$SAVE5                       ;                       5911
000004  162706  000006                  SUB     #6,SP
000010  012704  177777                  MOV     #-1,R4                          ; *,INDEX               5923
000014  113766  000000G 000004          MOVB    NEXT.PKT.USE,4(SP)              ; *,NEXT.PACKET         5932
000022  012766  000014  000002          MOV     #14,2(SP)                       ; *,COUNT               5934
000030  105016                  1$:     CLRB    (SP)                            ; PACKET.OWNED          5936
000032  005001                          CLR     R1                              ;                       5938
000034  156601  000004                  BISB    4(SP),R1                        ; NEXT.PACKET,*
000040  105761  000000G                 TSTB    PKT.USE(R1)
000044  002072                          BGE     7$
000046  013700  000000G                 MOV     DCT.ADDR,R0                     ;                       5942
000052  016005  000004                  MOV     4(R0),R5                        ; *,RING.ADDR
```

```
000056 010146                         MOV   R1,-(SP)                  ;                        5946
000060 012746 000106                  MOV   #106,-(SP)
000064 004737 000000G                 JSR   PC,BL$MUL
000070 012702 000010                  MOV   #10,R2                    ; *,I                    5944
000074 021560 000000G          2$:    CMP   (R5),MSCP.PKT(R0)         ; RING.ADDR,*            5946
000100 001014                         BNE   3$
000102 012703 000002                  MOV   #2,R3                     ;                        5947
000106 060503                         ADD   R5,R3                     ; RING.ADDR,*
000110 042703 077777                  BIC   #77777,R3
000114 020327 100000                  CMP   R3,#-100000
000120 001004                         BNE   3$
000122 112766 000001 000004           MOVB  #1,4(SP)                  ; *,PACKET.OWNED         5951
000130 000404                         BR    4$                        ;                        5950
000132 062705 000004           3$:    ADD   #4,R5                     ; *,RING.ADDR            5955
000136 005302                         DEC   R2                        ; I                      5944
000140 001355                         BNE   2$
000142 032766 000001 000004    4$:    BIT   #1,4(SP)                  ; *,PACKET.OWNED         5957
000150 001027                         BNE   6$
000152 116661 000030 000000G          MOVB  30(SP),PKT.USE(R1)        ; CTLR,*                 5961
000160 010104                         MOV   R1,R4                     ; *,INDEX                5962
000162 010116                         MOV   R1,(SP)                   ;                        5965
000164 012746 000043                  MOV   #43,-(SP)
000170 004737 000000G                 JSR   PC,BL$MUL
000174 005726                         TST   (SP)+
000176 012702 000002                  MOV   #2,R2                     ; *,J                    5964
000202 010003                  5$:    MOV   R0,R3                     ;                        5965
000204 060203                         ADD   R2,R3                     ; J,*
000206 006303                         ASL   R3
000210 005063 000000G                 CLR   MSCP.PKT(R3)
000214 005202                         INC   R2                        ; J                      5964
000216 020227 000042                  CMP   R2,#42                    ; J,*
000222 003767                         BLE   5$
000224 022626                         CMP   (SP)+,(SP)+               ;                        5960
000226 000414                         BR    9$
000230 022626                  6$:    CMP   (SP)+,(SP)+               ;                        5941
000232 105266 000004           7$:    INCB  4(SP)                     ; NEXT.PACKET            5973
000236 126627 000004 000014           CMPB  4(SP),#14                 ; NEXT.PACKET,*          5975
000244 103402                         BLO   8$
000246 105066 000004                  CLRB  4(SP)                     ; NEXT.PACKET            5978
000252 005366 000002           8$:    DEC   2(SP)                     ; COUNT                  5934
000256 001264                         BNE   1$
000260 005704                  9$:    TST   R4                        ; INDEX                  5982
000262 002435                         BLT   11$
000264 105764 000000G                 TSTB  PKT.USE(R4)               ; *(INDEX)               5983
000270 002432                         BLT   11$
000272 010446                         MOV   R4,-(SP)                  ; INDEX,*                5987
000274 012746 000106                  MOV   #106,-(SP)
000300 004737 000000G                 JSR   PC,BL$MUL
000304 012760 000040 000006G          MOV   #40,MSCP.PKT+6(R0)
000312 142760 000017 000010G          BICB  #17,MSCP.PKT+10(R0)       ;                        5988
000320 152760 000001 000010G          BISB  #1,MSCP.PKT+10(R0)
000326 005000                         CLR   R0                        ;                        5989
000330 156600 000010                  BISB  10(SP),R0                 ; NEXT.PACKET,*
```

G15


```
000334  005200                          INC     R0
000336  110037  000000G                 MOVB    R0,NEXT.PKT.USE
000342  120027  000014                  CMPB    R0,#14                  ; NEXT.PKT.USE,*                5991
000346  103402                          BLO     10$
000350  105037  000000G                 CLRB    NEXT.PKT.USE            ;                               5993
000354  022626                  10$:    CMP     (SP)+,(SP)+             ;                               5986
000356  010400                  11$:    MOV     R4,R0                   ; INDEX,*                       5923
000360  062706  000006                  ADD     #6,SP                   ;                               5911
000364  000207                          RTS     PC

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


;   6000  1
;   6001  1
```

```
;  1
;  1
;  1     global routine PUT_PKT (index) : novalue =
;  1
;  1     !+
;  1     !       THE MSCP PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;  1     !       ROUTINE.
;  1     !-
;  1
;  1
;  2         begin
;  2
;  2
;  2         local
;  2             RING_ADDR : word,
;  2             OWNER : word;
;  2
;  2         RING_ADDR = .DCT_ADDR [RR_BEG];                               ! ADDRESS IN FIRST RESPONSE RING
;  2
;  2         incr COUNT from 1 to (RRING_LEN + CRING_LEN) do               ! FOR EACH ADDRESS IN THE RINGS
;  3             begin
;  3
;  3             if .MSCP_PKT [.index, PKT_LO] eqla ..RING_ADDR            ! IF ADDRESS MATCHES
;  3
;  3             then
;  4                 begin
;  4                 OWNER = .RING_ADDR + 2;                               ! ADDRESS OF OWNERSHIP WORD
;  4                 .OWNER = ..OWNER and (not (ED_OWN)) and (not (ED_FLAG)) ;  ! GIVE OWNERSHIP TO HOST
;  3                 end;
;  3
;  3
;  3             RING_ADDR = .RING_ADDR + 4;                               ! LOOK AT NEXT PACKET ADDRESS IN RING
;  2             end;
;  2
;  2
;  2         PKT_USE [.index] = -1;
;  2
;  1         end;


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

I15


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

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


;     6040  1
;     6041  1
```

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


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

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

```
;  1      global routine GET_RETPKT (CTLR) =
;  1
;  1      !+
;  1      !       THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
;  1      !       FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;  1      !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
;  1      !       THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;  1      !
;  1      !       INPUTS:
;  1      !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;  1      !-
;  1
;  2          begin
;  2
;  2          local
;  2              index : signed word initial (-1);     ! ASSUME NONE AVAILABLE
;  2
;  2          incr COUNT from 0 to (RP_CNT - 1) do     ! FOR EACH ENTRY IN TABLE
;  2
;  2              if .RP_USE [.COUNT] lss 0             ! IF FREE RETPKT IS FOUND
;  2              then
;  3                  begin
;  3                  RP_USE [.COUNT] = .CTLR;          ! ALLOCATE RETURN PACKET TO CONTROLLER
;  3                  index = .COUNT;
;  3
;  3                  incr J from 0 to (RP_LEN - 1) do  ! ZERO OUT RETPKT
;  3                      RETPKT [.COUNT, .J, 0, 16, 0] = 0;
;  3
;  3                  exitloop;                         ! DONE
;  3                  end;
;  2
;  2          return .index;                            ! RETURN PACKET INDEX (OR -1) TO CALLER
;  1          end;


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

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

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

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


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

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

```
global routine GET_IO_BUFF (ADDR) : novalue =


!+
!       THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
!       POOL.
!
!       INPUTS:
!               ADDR - ADDRESS TO STORE THE 2-WORD BUFFER DESCRIPTOR
!
!       IMPLICIT INPUTS:
!               CCTLR - CURRENT CONTROLLER NUMBER
!
!       OUTPUTS:
!               THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
!               WORDS AT "ADDR" AND "ADDR + 2". OTHERWISE, A ZERO IS RETURNED
!               AT "ADDR" IF NO BUFFERS ARE AVAILABLE.
!-



    begin
    .ADDR = 0;                                                  ! ASSUME FAILURE

    incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE

        if .BUFF_OWN [.COUNT] lss 0                             ! IF BUFFER IS FREE

        then

            begin
            BUFF_OWN [.COUNT] = .CCTLR;                         ! ALLOCATE BUFFER TO CONTROLLER
            .ADDR = .BUFF_ADDR [.COUNT];                        ! RETURN BUFFER DESCRIPTOR
            exitloop;                                           ! DONE
            end;


    end;                                                        ! ROUTINE GET_IO_BUFF


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

B16


```
000036  000404                        BR      3$                          ;                    6130
000040  005201               2$:      INC     R1                          ; COUNT              6124
000042  020127  000007                CMP     R1,#7                       ; COUNT,*
000046  003760                        BLE     1$
000050  012601               3$:      MOV     (SP)+,R1                    ;                    6100
000052  000207                        RTS     PC

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


;    6138  1
;    6139  1
```

C16


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


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

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

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


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

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

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


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

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

F16


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


                                                .SBTTL  IN.IODQ GLOBAL ROUTINES
000000  010146                         IN.IODQ::
                                                MOV     R1,-(SP)                        ;                       6199
000002  013701  000000G                         MOV     IODQ.IN,R1                      ;                       6206
000006  010100                                  MOV     R1,R0
000010  005200                                  INC     R0
000012  020037  000000G                         CMP     R0,IODQ.OUT
000016  001421                                  BEQ     1$
000020  010100                                  MOV     R1,R0                           ;                       6207
000022  162700  000007                          SUB     #7,R0
000026  020037  000000G                         CMP     R0,IODQ.OUT
000032  001413                                  BEQ     1$                              ;                       6209
000034  116661  000004  000000G                 MOVB    4(SP),IODQ(R1)                  ; INDEX,*               6212
000042  005237  000000G                         INC     IODQ.IN                         ;                       6213
000046  023727  000000G 000010                  CMP     IODQ.IN,#10                     ;                       6215
000054  103402                                  BLO     1$
000056  005037  000000G                         CLR     IODQ.IN                         ;                       6217
000062  012601                         1$:      MOV     (SP)+,R1                        ;                       6199
000064  000207                                  RTS     PC

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

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

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

H16


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

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


;    6250  1
;    6251  1
```

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

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

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

```
;  1   global routine SEND (index) =
;  1
;  1   !+
;  1   !       IF THE CURRENT RDRX IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
;  1   !       ROUTINE "SENDS" A COMMAND TO THE RDRX BY LOADING THE PACKET
;  1   !       DESCRIPTOR OF AN MSCP PACKET INTO THE COMMAND RING AND READING THE
;  1   !       DEVICE'S IP REGISTER.  IF THE
;  1   !       CURRENT RDRX IN NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
;  1   !       THE CALLER, AND NO ACTION IS TAKEN.
;  1   !
;  1   !       INPUTS:
;  1   !               INDEX - INDEX OF MSCP PACKET CONTAINING THE COMMAND TO
;  1   !                       BE SENT
;  1   !
;  1   !       IMPLICIT INPUTS:
;  1   !               CCTLR - CURRENT CONTROLLER NUMBER
;  1   !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;  1   !-
;  1
;  2       begin
;  2
;  2       local
;  2           SLOT_ADDR,
;  2           TEMP : word,
;  2           CUR_PRIORITY : word;
;  2
;  2       if (.DCT_ADDR [CRING_CNT] lssu CRING_LEN) and                  ! IF CRING IS NOT FULL AND
;  3          ((.DCT_ADDR [STAT] eql ONLINE) or                           !   IF DEVICE IS ONLINE OR
;  3           (.MSCP_PKT [.index, OPCODE] eql OP_SCC))                   !   IT IS A SET-CTRL-CHAR COMMAND
;  2       then
;  2
;  4           if (not ((.MSCP_PKT [.index, OPCODE] eql OP_ACC) or (.MSCP_PKT [.index, OPCODE] eql OP_ONL) or
;  4                   (.MSCP_PKT [.index, OPCODE] eql OP_RD) or (.MSCP_PKT [.index, OPCODE] eql OP_SCC) or
;  4           (.MSCP_PKT [.INDEX, OPCODE] EQL OP_SDD) OR                     !ZZZ
;  4           (.MSCP_PKT [.INDEX, OPCODE] EQL OP_RCD) OR                     !ZZZ
;  4           (.MSCP_PKT [.INDEX, OPCODE] EQL OP_GDS) OR                     !ZZZ
;  4           (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ELP) OR                     !ZZZ
;  4           (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ABT) OR                     !ZZZ
;  4           (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ESP) OR                     !ZZZ
;  3                   (.MSCP_PKT [.index, OPCODE] eql OP_WRT)))
;  2           then
;  3               begin
;  3               PRINTF (DBM107, .MSCP_PKT [.index, OPCODE]);
;  3               return FAILURE;
;  3               end
;  2           else
;  3               begin
;  3
;  3               do
;  3                   BREAK                                              ! LOOP TILL CREDIT BALANCE POSITIVE
;  4               until ((.MSCP_PKT [.index, CMD_TYPE] eql IMM_CMD) and
;  3                     (.CREDIT_BAL gequ 1)) or
;  3                     (.CREDIT_BAL gtru 1);
```

```
;    6329  3
;    6330  3                  MSCP_PKT [.index, CRN_LO] = (CRN_LOW = .CRN_LOW + 1);          ! ASSIGN CMD REF NUM
;    6331  3
;    6332  3                  if .CRN_LOW eql 0
;    6333  3                  then
;    6334  3                      CRN_HIGH = .CRN_HIGH + 1;                                  ! CMD REF NUM (HIGH ORDER)
;    6335  3
;    6336  3                  MSCP_PKT [.index, CRN_HI] = .CRN_HIGH;                         !
;    6337  3                  SLOT_ADDR = .DCT_ADDR [CR_NEXT];                               ! ADDR OF NEXT COMMAND SLOT
;    6338  3
;    6339  3                  do                                                             ! WAIT TILL NEXT SLOT HOST OWNED
;    6340  3                      BREAK                                                      !
;    6341  3                  until ((.(.SLOT_ADDR + 2) and ED_OWN) eql 0);                  !
;    6342  3
;    6343  3                  GETPRI (CUR_PRIORITY);                                         ! NO INTERRUPTS WHILE POINTERS UPDATED
;    6344  3        !ZZZ      SETPRI (PRIO4);                                                !
;    6345  3                  SETPRI (.BRLEVEL);                                             !          ZZZ
;    6346  3
;    6347  3                  .SLOT_ADDR = .MSCP_PKT [.index, PKT_LO];                       ! LOAD BUFF DESC (LO) INTO COMMAND SLOT
;    6348  3                  SLOT_ADDR = .SLOT_ADDR + 2;                                    ! ADVANCE TO NEXT WORD
;    6349  3                  .SLOT_ADDR = .MSCP_PKT [.index, PKT_HI];                       ! LOAD BUFF DESC (HI) INTO COMMAND SLOT
;    6350  3                  .SLOT_ADDR = ..SLOT_ADDR and (not (ED_FLAG));                  ! CLEAR INTERRUPT FLAG IN CASE SET
;    6351  3                  .SLOT_ADDR = ..SLOT_ADDR or ED_OWN;                            ! GIVE OWNERSHIP TO CONTROLLER
;    6352  3                  SLOT_ADDR = .SLOT_ADDR + 2;                                    ! ADVANCE TO NEXT COMMAND SLOT
;    6353  3
;    6354  3                  if .SLOT_ADDR gtra .DCT_ADDR [CR_END]                          ! IF BEYOND END OF CRING
;    6355  3                  then
;    6356  3                      SLOT_ADDR = .DCT_ADDR [CR_BEG];                            ! CYCLE BACK TO BEGINNING
;    6357  3
;    6358  3                  DCT_ADDR [CR_NEXT] = .SLOT_ADDR;                               ! RESTORE CR_NEXT POINTER IN DCT
;    6359  3                  DCT_ADDR [CRING_CNT] = .DCT_ADDR [CRING_CNT] + 1;              ! INCR # OF COMMANDS IN CRING
;    6360  4                  IF (.MSCP_PKT [.INDEX, CONNID] EQL CID_MSCP)                   !IF MSCP COMMAND                ZZZ
;    6361  3                  THEN (CREDIT_BAL = .CREDIT_BAL - 1);                           !DECR CREDIT BALANCE            ZZZ
;    6362  3                  TEMP = .RDRX_ADDR [RCIP, RC_ALL];                              ! READ IP TO FORCE PORT TO POLL
;    6363  3                  SETPRI (.CUR_PRIORITY);                                        ! LOWER PRIORITY
;    6364  3                  return SUCCESS;
;    6365  3                  end
;    6366  3
;    6367  2          else
;    6368  2              return FAILURE;                                                    ! IF DEVICE IS NOT ONLINE
;    6369  2
;    6370  1          end;                                                                   ! ROUTINE SEND


                                                  .SBTTL  SEND GLOBAL ROUTINES
000000  004137  000000G                SEND::     JSR     R1,$SAVE3                          ;                              6276
000004  005746                                    TST     -(SP)
000006  127727  000000G 000004                    CMPB    @DCT.ADDR,#4                       ;                              6302
000014  103100                                    BHIS    2$
000016  005777  000000G                           TST     @DCT.ADDR                          ;                              6303
000022  100413                                    BMI     1$
000024  016646  000014                            MOV     14(SP),-(SP)                       ; INDEX,*                      6304
000030  012746  000106                            MOV     #106,-(SP)
```

```
000034  004737  000000G                JSR     PC,BL$MUL
000040  022626                         CMP     (SP)+,(SP)+
000042  126027  000022G 000004         CMPB    MSCP.PKT+22(R0),#4
000050  001167                         BNE     10$
000052  016646  000014          1$:    MOV     14(SP),-(SP)                ; INDEX,*                 6307
000056  012746  000106                 MOV     #106,-(SP)
000062  004737  000000G                JSR     PC,BL$MUL
000066  010002                         MOV     R0,R2
000070  022626                         CMP     (SP)+,(SP)+
000072  005000                         CLR     R0
000074  156200  000022G                BISB    MSCP.PKT+22(R2),R0
000100  020027  000020                 CMP     R0,#20
000104  001445                         BEQ     3$
000106  020027  000011                 CMP     R0,#11
000112  001442                         BEQ     3$
000114  020027  000041                 CMP     R0,#41                      ;                         6308
000120  001437                         BEQ     3$
000122  020027  000004                 CMP     R0,#4
000126  001434                         BEQ     3$
000130  020027  000005                 CMP     R0,#5                       ;                         6310
000134  001431                         BEQ     3$
000136  020027  000001                 CMP     R0,#1                       ;                         6311
000142  001426                         BEQ     3$
000144  020027  000003                 CMP     R0,#3                       ;                         6312
000150  001423                         BEQ     3$
000152  020027  000006                 CMP     R0,#6                       ;                         6313
000156  001420                         BEQ     3$
000160  020027  000002                 CMP     R0,#2                       ;                         6314
000164  001415                         BEQ     3$
000166  020027  000042                 CMP     R0,#42                      ;                         6315
000172  001412                         BEQ     3$
000174  010046                         MOV     R0,-(SP)                    ;                         6318
000176  012746  000000G                MOV     #DBM107,-(SP)
000202  012746  000002                 MOV     #2,-(SP)
000206  010600                         MOV     SP,R0                       ; SP,*
000210  104417                         TRAP    17
000212  062706  000006                 ADD     #6,SP                       ;                         6317
000216  000504                  2$:    BR      10$                         ;                         6307
000220  104422                  3$:    TRAP    22                          ;                         6324
000222  105762  000004G                TSTB    MSCP.PKT+4(R2)              ;                         6326
000226  001003                         BNE     4$
000230  005737  000000G                TST     CREDIT.BAL                  ;                         6327
000234  001004                         BNE     5$
000236  023727  000000G 000001  4$:    CMP     CREDIT.BAL,#1               ;                         6328
000244  101765                         BLOS    3$
000246  013700  000000G         5$:    MOV     CRN.LOW,R0                  ;                         6330
000252  005200                         INC     R0
000254  010037  000000G                MOV     R0,CRN.LOW
000260  010062  000012G                MOV     R0,MSCP.PKT+12(R2)
000264  001002                         BNE     6$                          ;                         6332
000266  005237  000000G                INC     CRN.HIGH                    ;                         6334
000272  013762  000000G 000014G 6$:    MOV     CRN.HIGH,MSCP.PKT+14(R2)    ;                         6336
000300  013700  000000G                MOV     DCT.ADDR,R0                 ;                         6337
```

```
000304  016001  000020                 MOV     20(R0),R1                  ; *,SLOT.ADDR
000310  104422                 7$:     TRAP    22                         ;                        6339
000312  032761  100000  000002         BIT     #-100000,2(R1)             ; *,*(SLOT.ADDR)         6341
000320  001373                         BNE     7$
000322  104440                         TRAP    40                         ;                        6343
000324  010003                         MOV     R0,R3                      ; *,CUR.PRIORITY
000326  013700  000000G                MOV     BRLEVEL,R0                 ;                        6345
000332  104441                         TRAP    41
000334  016221  000000G                MOV     MSCP.PKT(R2),(R1)+         ; *,SLOT.ADDR            6347
000340  016211  000002G                MOV     MSCP.PKT+2(R2),(R1)        ; *,SLOT.ADDR            6349
000344  042711  040000                 BIC     #40000,(R1)                ; *,SLOT.ADDR            6350
000350  052721  100000                 BIS     #100000,(R1)+              ; *,SLOT.ADDR            6351
000354  013700  000000G                MOV     DCT.ADDR,R0                ;                        6354
000360  020160  000012                 CMP     R1,12(R0)                  ; SLOT.ADDR,*
000364  101402                         BLOS    8$
000366  016001  000010                 MOV     10(R0),R1                  ; *,SLOT.ADDR            6356
000372  010160  000020         8$:     MOV     R1,20(R0)                  ; SLOT.ADDR,*            6358
000376  105210                         INCB    (R0)                       ;                        6359
000400  105762  000011G                TSTB    MSCP.PKT+11(R2)            ;                        6360
000404  001002                         BNE     9$
000406  005337  000000G                DEC     CREDIT.BAL                 ;                        6361
000412  017716  000000G        9$:     MOV     @RDRX.ADDR,(SP)            ; *,RC.REG               6362
000416  010300                         MOV     R3,R0                      ; CUR.PRIORITY,*         6363
000420  104441                         TRAP    41
000422  012700  000001                 MOV     #1,R0                      ;                        6307
000426  000401                         BR      11$                        ;                        6368
000430  005000                 10$:    CLR     R0
000432  005726                 11$:    TST     (SP)+                      ;                        6276
000434  000207                         RTS     PC

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

```
;    6371  1     global routine WAIT : novalue =
;    6372  1
;    6373  1     !+
;    6374  1     !       THE PURPOSE OF THIS ROUTINE IS TO KILL TIME UNTIL AN RDRX INTERRUPT
;    6375  1     !       RESULTS IN A RETURN PACKET INDEX BEING DEPOSITED INTO THE I/O DONE
;    6376  1     !       QUEUE (IODQ).
;    6377  1     !-
;    6378  1
;    6379  1         do
;    6380  1             BREAK                              ! BREAK FOR ACT
;    6381  1         until .IODQ_IN neq .IODQ_OUT;


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

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


;    6382  1
```

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


007700                                    X:        .BLKW    1


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

```
000042  013700  007700'          MOV     X,R0                            ;                       6404
000046  006300                   ASL     R0
000050  020127  077777           CMP     R1,#77777                       ; SIZE,*                6403
000054  101011                   BHI     2$
000056  016046  000000G          MOV     RANDOM(R0),-(SP)                ;                       6404
000062  042716  100000           BIC     #100000,(SP)
000066  010146                   MOV     R1,-(SP)                        ; SIZE,*
000070  005216                   INC     (SP)
000072  004737  000000G          JSR     PC,BL$MOD
000076  000431                   BR      3$                              ;                       6403
000100  006201           2$:     ASR     R1                              ; SIZE                  6408
000102  016046  000000G          MOV     RANDOM(R0),-(SP)                ;                       6409
000106  042716  100000           BIC     #100000,(SP)
000112  010146                   MOV     R1,-(SP)                        ; SIZE,*
000114  005216                   INC     (SP)
000116  004737  000000G          JSR     PC,BL$MOD
000122  006300                   ASL     R0                              ; ANSWER                6411
000124  013701  007700'          MOV     X,R1                            ;                       6412
000130  006301                   ASL     R1
000132  116102  000002G          MOVB    RANDOM+2(R1),R2
000136  042702  177776           BIC     #177776,R2
000142  060200                   ADD     R2,R0                           ; *,ANSWER
000144  012701  000004           MOV     #4,R1                           ;                       6414
000150  060601                   ADD     SP,R1                           ; SAVESZ,*
000152  020001                   CMP     R0,R1                           ; ANSWER,*
000154  101402                   BLOS    3$
000156  016600  000004           MOV     4(SP),R0                        ; SAVESZ,ANSWER         6415
000162  062706  000006   3$:     ADD     #6,SP                           ;                       6384
000166  000207                   RTS     PC

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

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


                                     .SBTTL  EMS.SA ERROR MESSAGE SUBROUTINES
000000  010146                  EMS.SA: MOV     R1,-(SP)                 ;          6421
000002  013701  000000G                 MOV     SA.REG,R1                ;          6430
000006  020127  177777                  CMP     R1,#-1
000012  001023                          BNE     1$
000014  012746  000000G                 MOV     #CRLF,-(SP)              ;          6433
000020  012746  000001                  MOV     #1,-(SP)
000024  010600                          MOV     SP,R0                    ; SP,*
000026  104415                          TRAP    15
000030  012716  000000G                 MOV     #ASTERISK,(SP)           ;          6434
```

```
000034  012746  000001                  MOV     #1,-(SP)
000040  010600                          MOV     SP,R0                           ; SP,*
000042  104415                          TRAP    15
000044  013716  000000G                 MOV     CNTR.ERR,(SP)                   ;               6435
000050  012746  000001                  MOV     #1,-(SP)
000054  010600                          MOV     SP,R0                           ; SP,*
000056  104415                          TRAP    15
000060  000475                          BR      3$                              ;               6432
000062  010100                  1$:     MOV     R1,R0                           ;               6439
000064  042700  174000                  BIC     #174000,R0
000070  020027  000026                  CMP     R0,#26
000074  101030                          BHI     2$
000076  012746  000000G                 MOV     #CRLF,-(SP)                     ;               6442
000102  012746  000001                  MOV     #1,-(SP)
000106  010600                          MOV     SP,R0                           ; SP,*
000110  104415                          TRAP    15
000112  012716  000000G                 MOV     #ASTERISK,(SP)                  ;               6443
000116  012746  000001                  MOV     #1,-(SP)
000122  010600                          MOV     SP,R0                           ; SP,*
000124  104415                          TRAP    15
000126  013700  000000G                 MOV     SA.REG,R0                       ;               5444
000132  042700  174000                  BIC     #174000,R0
000136  006300                          ASL     R0
000140  016016  000000G                 MOV     CNTR.ERR(R0),(SP)
000144  012746  000001                  MOV     #1,-(SP)
000150  010600                          MOV     SP,R0                           ; SP,*
000152  104415                          TRAP    15
000154  000437                          BR      3$                              ;               6441
000156  010100                  2$:     MOV     R1,R0                           ;               6448
000160  042700  174000                  BIC     #174000,R0
000164  162700  000620                  SUB     #620,R0
000170  020027  000006                  CMP     R0,#6
000174  101031                          BHI     4$
000176  012746  000000G                 MOV     #CRLF,-(SP)                     ;               6451
000202  012746  000001                  MOV     #1,-(SP)
000206  010600                          MOV     SP,R0                           ; SP,*
000210  104415                          TRAP    15
000212  012716  000000G                 MOV     #ASTERISK,(SP)                  ;               6452
000216  012746  000001                  MOV     #1,-(SP)
000222  010600                          MOV     SP,R0                           ; SP,*
000224  104415                          TRAP    15
000226  013700  000000G                 MOV     SA.REG,R0                       ;               6453
000232  042700  174000                  BIC     #174000,R0
000236  006300                          ASL     R0
000240  016016  176340G                 MOV     RDRX.ERR-1440(R0),(SP)
000244  012746  000001                  MOV     #1,-(SP)
000250  010600                          MOV     SP,R0                           ; SP,*
000252  104415                          TRAP    15
000254  005726                  3$:     TST     (SP)+                           ;               6450
000256  000407                          BR      5$                              ;               6448
000260  010146                  4$:     MOV     R1,-(SP)                        ;               6456
000262  012746  000000G                 MOV     #EX.SA,-(SP)
000266  012746  000002                  MOV     #2,-(SP)
```

```
000272  010600                          MOV     SP,R0                   ; SP,*
000274  104415                          TRAP    15
000276  004737  000000V         5$:     JSR     PC,EMS.TIM              ;                               6458
000302  062706  000006                  ADD     #6,SP                   ;                               6428
000306  012601                          MOV     (SP)+,R1                ;                               6421
000310  000207                          RTS     PC

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

```
;  1  routine EMS_SBC : novalue =
;  1
;  1  !+
;  1  !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "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      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          case .ST_CODE from ST_SUC to ST_DRV of
;  3              set
;  3
;  3              [ST_SUC]:           if .SB_CODE lequ 16                 ! SUCCESS SUB-CODES
;  3                                  then
;  3                                      PRINTX (.TBL_SUC [.SB_CODE]);
;  3
;  3              [ST_CMD]:           PRINTX (EX_SBO, .SB_CODE / 8);      ! INVALID COMMAND
;  3
;  3              [ST_ABO]:           ;                                   ! COMMAND ABORTED
;  3
;  3              [ST_OFL]:           if .SB_CODE lequ 8                  ! UNIT OFFLINE
;  3                                  then
;  3                                      PRINTX (.TBL_OFL [.SB_CODE]);
;  3
;  3              [ST_AVL]:           ;                                   ! UNIT AVAILABLE
;  3
;  3              [ST_MFE]:           if .SB_CODE lequ 10                 ! MEDIA FORMAT ERROR
;  3                                  then
;  3                                      PRINTX (.TBL_MFE [.SB_CODE]);
;  3
;  3              [ST_WPT]:           if (.SB_CODE / 128) lequ 2          ! WRITE PROTECTED
;  3                                  then
;  3                                      PRINTX (.TBL_WPT [(.SB_CODE / 128)]);
;  3
;  3              [ST_CMP]:           ;                                   ! COMPARE ERROR
;  3
;  3              [ST_DAT]:           if .SB_CODE lequ 15                 ! DATA ERROR
;  3                                  then
;  3                                      PRINTX (.TBL_DAT [.SB_CODE]);
;  3
;  3              [ST_HST]:           if .SB_CODE lequ 4                  ! HOST ACCESS ERROR
;  3                                  then
;  3                                      PRINTX (.TBL_HST [.SB_CODE]);
;  3
;  3              [ST_CNT]:           if .SB_CODE lequ 3                  ! CONTROLLER ERROR
;  3                                  then
```

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


                                            .SBTTL  EMS.SBC ERROR MESSAGE SUBROUTINES
000000  013700  000000G             EMS.SBC:MOV     ST.CODE,R0                  ;                      6471
000004  053700  000000G                     BIS     SB.CODE,R0
000010  001001                              BNE     1$
000012  000207                              RTS     PC
000014  012746  000000G             1$:     MOV     #EX.SB,-(SP)                ;                      6474
000020  012746  000001                      MOV     #1,-(SP)
000024  010600                              MOV     SP,R0                       ; SP,*
000026  104415                              TRAP    15
000030  013700  000000G                     MOV     ST.CODE,R0                  ;                      6476
000034  020027  000013                      CMP     R0,#13
000040  101003                              BHI     3$
000042  006300                              ASL     R0
000044  066007  000000'                     ADD     P.AAA(R0),PC                ; Case dispatch
000050  013716  000000G             3$:     MOV     SB.CODE,(SP)                ;                      6519
000054  012746  000000G                     MOV     #EX.SBO,-(SP)
000060  012746  000002                      MOV     #2,-(SP)
000064  010600                              MOV     SP,R0                       ; SP,*
000066  104415                              TRAP    15
000070  022626                              CMP     (SP)+,(SP)+
000072  000435                              BR      6$                          ;                      6476
000074  023727  000000G 000020      4$:     CMP     SB.CODE,#20                 ;                      6479
000102  101165                              BHI     14$
000104  013700  000000G                     MOV     SB.CODE,R0                  ;                      6481
000110  006300                              ASL     R0
000112  016016  000000'                     MOV     TBL.SUC(R0),(SP)
000116  012746  000001                      MOV     #1,-(SP)
000122  010600                              MOV     SP,R0                       ; SP,*
000124  104415                              TRAP    15
000126  000565                              BR      15$
000130  013716  000000G             5$:     MOV     SB.CODE,(SP)                ;                      6483
000134  012746  000010                      MOV     #10,-(SP)
000140  004737  000000G                     JSR     PC,BL$DIV
000144  010016                              MOV     R0,(SP)
000146  012746  000000G                     MOV     #EX.SBO,-(SP)
000152  012746  000002                      MOV     #2,-(SP)
000156  010600                              MOV     SP,R0                       ; SP,*
000160  104415                              TRAP    15
000162  062706  000006                      ADD     #6,SP
```

```
000166  000546                            6$:     BR      16$                                  ;               6476
000170  023727  000000G 000010            7$:     CMP     SB.CODE,#10                          ;               6487
000176  101142                                    BHI     16$
000200  013700  000000G                           MOV     SB.CODE,R0                           ;               6489
000204  006300                                    ASL     R0
000206  016016  000042'                           MOV     TBL.OFL(R0),(SP)
000212  012746  000001                            MOV     #1,-(SP)
000216  010600                                    MOV     SP,R0                                ; SP,*
000220  104415                                    TRAP    15
000222  000527                                    BR      15$
000224  023727  000000G 000012            8$:     CMP     SB.CODE,#12                          ;               6493
000232  101124                                    BHI     16$
000234  013700  000000G                           MOV     SB.CODE,R0                           ;               6495
000240  006300                                    ASL     R0
000242  016016  000064'                           MOV     TBL.MFE(R0),(SP)
000246  012746  000001                            MOV     #1,-(SP)
000252  010600                                    MOV     SP,R0                                ; SP,*
000254  104415                                    TRAP    15
000256  000511                                    BR      15$
000260  013716  000000G                   9$:     MOV     SB.CODE,(SP)                         ;               6497
000264  012746  000200                            MOV     #200,-(SP)
000270  004737  000000G                           JSR     PC,BL$DIV
000274  005726                                    TST     (SP)+
000276  020027  000002                            CMP     R0,#2
000302  101100                                    BHI     16$
000304  006300                                    ASL     R0                                   ;               6499
000306  016016  000112'                           MOV     TBL.WPT(R0),(SP)
000312  012746  000001                            MOV     #1,-(SP)
000316  010600                                    MOV     SP,R0                                ; SP,*
000320  104415                                    TRAP    15
000322  000467                                    BR      15$
000324  023727  000000G 000017            10$:    CMP     SB.CODE,#17                          ;               6503
000332  101064                                    BHI     16$
000334  013700  000000G                           MOV     SB.CODE,R0                           ;               6505
000340  006300                                    ASL     R0
000342  016016  000120'                           MOV     TBL.DAT(R0),(SP)
000346  012746  000001                            MOV     #1,-(SP)
000352  010600                                    MOV     SP,R0                                ; SP,*
000354  104415                                    TRAP    15
000356  000451                                    BR      15$
000360  023727  000000G 000004            11$:    CMP     SB.CODE,#4                           ;               6507
000366  101046                                    BHI     16$
000370  013700  000000G                           MOV     SB.CODE,R0                           ;               6509
000374  006300                                    ASL     R0
000376  016016  000160'                           MOV     TBL.HST(R0),(SP)
000402  012746  000001                            MOV     #1,-(SP)
000406  010600                                    MOV     SP,R0                                ; SP,*
000410  104415                                    TRAP    15
000412  000433                                    BR      15$
000414  023727  000000G 000003            12$:    CMP     SB.CODE,#3                           ;               6511
000422  101030                                    BHI     16$
000424  013700  000000G                           MOV     SB.CODE,R0                           ;               6513
000430  006300                                    ASL     R0
```

K1


```
000432  016016  000172'                 MOV     TBL.CNT(R0),(SP)
000436  012746  000001                  MOV     #1,-(SP)
000442  010600                          MOV     SP,R0                   ; SP,*
000444  104415                          TRAP    15
000446  000415                          BR      15$
000450  023727  000000G 000010  13$:    CMP     SB.CODE,#10             ;                          6515
000456  101012                  14$:    BHI     16$
000460  013700  000000G                 MOV     SB.CODE,R0              ;                          6517
000464  006300                          ASL     R0
000466  016016  000202'                 MOV     TBL.DRV(R0),(SP)
000472  012746  000001                  MOV     #1,-(SP)
000476  010600                          MOV     SP,R0                   ; SP,*
000500  104415                          TRAP    15
000502  005726                  15$:    TST     (SP)+
000504  022626                  16$:    CMP     (SP)+,(SP)+             ;                          6473
000506  000207                          RTS     PC                      ;                          6460

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


000000                                  .PSECT  $PLIT$,  RO ,  D

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

```
;  1   routine EMS_CMD : novalue =
;  1
;  1   !+
;  1   !       THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
;  1   !       PRESENT) OF THE CURRENT RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
;  1   !       INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
;  1   !
;  1   !       IMPLICIT INPUTS:
;  1   !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  1   !-
;  1
;  2       begin
;  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                           if .RP_ADDR [CMDMOD] neq 0
;  3                           then
;  3                               PRINTX (EX_CMP);                 ! COMPARE
;  3
;  2                           end;
;  2
;  3           [OP_WRT]:       begin
;  3                           PRINTX (EX_WRT);                     ! WRITE
;  3
;  3                           if .RP_ADDR [CMDMOD] neq 0
;  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_CMD


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

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

```
000020  013702  000000G              MOV     RP.ADDR,R2                 ;                        6539
000024  116201  000014               MOVB    14(R2),R1
000030  042701  177600               BIC     #177600,R1
000034  020127  000011               CMP     R1,#11                     ;                        6542
000040  001007                       BNE     1$
000042  012716  000000G              MOV     #EX.ONL,(SP)
000046  012746  000001               MOV     #1,-(SP)
000052  010600                       MOV     SP,R0                      ; SP,*
000054  104415                       TRAP    15
000056  000464                       BR      5$
000060  020127  000020       1$:     CMP     R1,#20                     ;                        6544
000064  001007                       BNE     2$
000066  012716  000000G              MOV     #EX.ACC,(SP)
000072  012746  000001               MOV     #1,-(SP)
000076  010600                       MOV     SP,R0                      ; SP,*
000100  104415                       TRAP    15
000102  000452                       BR      5$
000104  020127  000041       2$:     CMP     R1,#41                     ;                        6546
000110  001022                       BNE     3$
000112  012716  000000G              MOV     #EX.RD,(SP)                ;                        6547
000116  012746  000001               MOV     #1,-(SP)
000122  010600                       MOV     SP,R0                      ; SP,*
000124  104415                       TRAP    15
000126  013700  000000G              MOV     RP.ADDR,R0                 ;                        6549
000132  005760  000012               TST     12(R0)
000136  001434                       BEQ     5$
000140  012716  000000G              MOV     #EX.CMP,(SP)               ;                        6551
000144  012746  000001               MOV     #1,-(SP)
000150  010600                       MOV     SP,R0                      ; SP,*
000152  104415                       TRAP    15
000154  000424                       BR      4$
000156  020127  000042       3$:     CMP     R1,#42                     ;                        6555
000162  001024                       BNE     6$
000164  012716  000000G              MOV     #EX.WRT,(SP)               ;                        6556
000170  012746  000001               MOV     #1,-(SP)
000174  010600                       MOV     SP,R0                      ; SP,*
000176  104415                       TRAP    15
000200  013700  000000G              MOV     RP.ADDR,R0                 ;                        6558
000204  005760  000012               TST     12(R0)
000210  001407                       BEQ     5$
000212  012716  000000G              MOV     #EX.CMP,(SP)               ;                        6560
000216  012746  000001               MOV     #1,-(SP)
000222  010600                       MOV     SP,R0                      ; SP,*
000224  104415                       TRAP    15
000226  005726               4$:     TST     (SP)+
000230  005726               5$:     TST     (SP)+                      ;                        6555
000232  000412                       BR      7$                         ;                        6539
000234  005016               6$:     CLR     (SP)                       ;                        6564
000236  116216  000014               MOVB    14(R2),(SP)
000242  012746  000000G              MOV     #EX.OP,-(SP)
000246  012746  000002               MOV     #2,-(SP)
000252  010600                       MOV     SP,R0                      ; SP,*
000254  104415                       TRAP    15
```

```
000256  022626                       CMP     (SP)+,(SP)+
000260  022626               7$:     CMP     (SP)+,(SP)+                          ;                    6536
000262  000207                       RTS     PC                                   ;                    6525

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

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


                                                  .SBTTL  EMS.DBN ERROR MESSAGE SUBROUTINES
000000  013746  000000G                 EMS.DBN::
                                                  MOV     CDISK,-(SP)          ;              6577
000004  012746  000000G                           MOV     #XX13,-(SP)
000010  012746  000002                            MOV     #2,-(SP)
000014  010600                                    MOV     SP,R0                ; SP,*
000016  104414                                    TRAP    14
000020  013700  000000G                           MOV     CUOFF,R0             ;              6579
000024  006300                                    ASL     R0
000026  063700  000000G                           ADD     CST.ADDR,R0
000032  005016                                    CLR     (SP)
000034  116016  000020                            MOVB    20(R0),(SP)
000040  005046                                    CLR     -(SP)
000042  116016  000020                            MOVB    20(R0),(SP)
000046  012746  000000G                           MOV     #XX23,-(SP)
000052  012746  000003                            MOV     #3,-(SP)
000056  010600                                    MOV     SP,R0                ; SP,*
000060  104414                                    TRAP    14
000062  013716  000000G                           MOV     S.DUPPKT,(SP)        ;              6580
000066  162716  000002                            SUB     #2,(SP)
000072  012746  000000G                           MOV     #XX32,-(SP)
000076  012746  000002                            MOV     #2,-(SP)
000102  010600                                    MOV     SP,R0                ; SP,*
000104  104414                                    TRAP    14
000106  013716  000000G                           MOV     S.PATTERN,(SP)       ;              6581
000112  012746  000000G                           MOV     #XX33,-(SP)
000116  012746  000002                            MOV     #2,-(SP)
000122  010600                                    MOV     SP,R0                ; SP,*
000124  104414                                    TRAP    14
000126  013700  000000G                           MOV     S.DUPPKT,R0          ;              6582
000132  016016  000000G                           MOV     DUPPKT(R0),(SP)
000136  011646                                    MOV     (SP),-(SP)
000140  012746  000000G                           MOV     #XX34,-(SP)
000144  012746  000003                            MOV     #3,-(SP)
```

```
000150  010600                  MOV     SP,R0                   ; SP,*
000152  104414                  TRAP    14
000154  012716  000002G         MOV     #DUPPKT+2,(SP)          ;                       6583
000160  012746  000400          MOV     #400,-(SP)
000164  004737  000000V         JSR     PC,EMS.BLK
000170  062706  000034          ADD     #34,SP                  ;                       6576
000174  000207                  RTS     PC                      ;                       6568

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


;    6585  1
;    6586  1
```

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


                                                  .SBTTL  EMS.BLK ERROR MESSAGE SUBROUTINES
000000  010146                            EMS.BLK::
                                                  MOV     R1,-(SP)                 ;             6588
000002  012746  000000G                           MOV     #CRLF,-(SP)              ;             6600
000006  012746  000001                            MOV     #1,-(SP)
000012  010600                                    MOV     SP,R0                    ; SP,*
000014  104415                                    TRAP    15
000016  005001                                    CLR     R1                       ; COUNT       6601
000020  000445                                    BR      5$
000022  010100                            1$:     MOV     R1,R0                    ; COUNT,*     6603
000024  005300                                    DEC     R0
000026  032700  000007                            BIT     #7,R0
000032  001007                                    BNE     2$
000034  012716  000000G                           MOV     #SPACE4,(SP)             ;             6605
000040  012746  000001                            MOV     #1,-(SP)
000044  010600                                    MOV     SP,R0                    ; SP,*
000046  104415                                    TRAP    15
000050  005726                                    TST     (SP)+
000052  017616  000012                    2$:     MOV     @12(SP),(SP)             ; ADDR,*      6607
000056  012746  000000G                           MOV     #EX.WRD,-(SP)
000062  012746  000002                            MOV     #2,-(SP)
000066  010600                                    MOV     SP,R0                    ; SP,*
```

E2


```
000070  104415                                 TRAP    15
000072  062766  000002  000016                 ADD     #2,16(SP)                ; *,ADDR                          6608
000100  032701  000007                         BIT     #7,R1                    ; *,COUNT                         6610
000104  001403                                 BEQ     3$
000106  020166  000014                         CMP     R1,14(SP)                ; COUNT,LENGTH                    6611
000112  001007                                 BNE     4$
000114  012716  000000G               3$:      MOV     #CRLF,(SP)               ;                                 6613
000120  012746  000001                         MOV     #1,-(SP)
000124  010600                                 MOV     SP,R0                    ; SP,*
000126  104415                                 TRAP    15
000130  005726                                 TST     (SP)+
000132  022626                        4$:      CMP     (SP)+,(SP)+              ;                                 6602
000134  005201                        5$:      INC     R1                       ; COUNT                           6601
000136  020166  000010                         CMP     R1,10(SP)                ; COUNT,LENGTH
000142  003727                                 BLE     1$
000144  022626                                 CMP     (SP)+,(SP)+              ;                                 6596
000146  012601                                 MOV     (SP)+,R1                 ;                                 6588
000150  000207                                 RTS     PC

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


;     6616  1
;     6617  1
```

F2


```
;    6618  1      routine EMS_LBN : novalue =
;    6619  1
;    6620  1      !+
;    6621  1      !        THIS ROUTINE PRINTS (EXTENDED) ONE OF TWO BLOCK NUMBERS APPEARING IN
;    6622  1      !        THE CURRENT RETURN PACKET. NORMALLY, THE LBN FIELD IS PRINTED; THIS
;    6623  1      !        FIELD WAS COPIED INTO THE RETURN PACKET FROM THE ASSOCIATED COMMAND
;    6624  1      !        PACKET. HOWEVER, IF THE "FLAGS" FIELD OF THE CURRENT RETURN PACKET
;    6625  1      !        INDICATES "BAD BLOCK REPORTED", THEN THE "FIRST BAD BLOCK" FIELD IS
;    6626  1      !        PRINTED.
;    6627  1      !
;    6628  1      !        IMPLICIT INPUTS:
;    6629  1      !                RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    6630  1      !-
;    6631  1
;    6632  2          begin
;    6633  2
;    6634  2          if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and             ! IF NO BAD BLOCK FOUND
;    6635  3             (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    6636  2          then
;    6637  2              PRINTX (EX_LBN, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);
;    6638  2
;    6639  2          if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and             ! IF BAD BLOCKS FOUND AND REPLACED
;    6640  3             (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    6641  2          then
;    6642  2              PRINTX (EX_BBU, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    6643  2
;    6644  2          if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                 ! IF HOST REPLACEABLE BAD BLOCK FOUND
;    6645  3             (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    6646  2          then
;    6647  2              PRINTX (EX_BB, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    6648  2
;    6649  2          if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                 ! IF MORE THAN 1 HOST REPLACEABLE BAD BLOCK FOUND
;    6650  3             (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    6651  2          then
;    6652  2              PRINTX (EX_BB1, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    6653  1          end;


                                                 .SBTTL  EMS.LBN ERROR MESSAGE SUBROUTINES
000000  013700  000000G                 EMS.LBN:MOV     RP.ADDR,R0                 ;           6634
000004  105760  000015                          TSTB    15(R0)
000010  100417                                  BMI     1$
000012  132760  000100  000015                  BITB    #100,15(R0)                ;           6635
000020  001013                                  BNE     1$
000022  016046  000050                          MOV     50(R0),-(SP)               ;           6637
000026  011646                                  MOV     (SP),-(SP)
000030  012746  000000G                         MOV     #EX.LBN,-(SP)
000034  012746  000003                          MOV     #3,-(SP)
000040  010600                                  MOV     SP,R0                      ; SP,*
000042  104415                                  TRAP    15
000044  062706  000010                          ADD     #10,SP
000050  013700  000000G                 1$:     MOV     RP.ADDR,R0                 ;           6639
000054  105760  000015                          TSTB    15(R0)
```

```
000060  100417                          BMI     2$
000062  132760  000100  000015          BITB    #100,15(R0)             ;               6640
000070  001413                          BEQ     2$
000072  016046  000040                  MOV     40(R0),-(SP)            ;               6642
000076  011646                          MOV     (SP),-(SP)
000100  012746  000000G                 MOV     #EX.BBU,-(SP)
000104  012746  000003                  MOV     #3,-(SP)
000110  010600                          MOV     SP,R0                   ; SP,*
000112  104415                          TRAP    15
000114  062706  000010                  ADD     #10,SP
000120  013700  000000G         2$:     MOV     RP.ADDR,R0              ;               6644
000124  105760  000015                  TSTB    15(R0)
000130  100017                          BPL     3$
000132  132760  000100  000015          BITB    #100,15(R0)             ;               6645
000140  001013                          BNE     3$
000142  016046  000040                  MOV     40(R0),-(SP)            ;               6647
000146  011646                          MOV     (SP),-(SP)
000150  012746  000000G                 MOV     #EX.BB,-(SP)
000154  012746  000003                  MOV     #3,-(SP)
000160  010600                          MOV     SP,R0                   ; SP,*
000162  104415                          TRAP    15
000164  062706  000010                  ADD     #10,SP
000170  013700  000000G         3$:     MOV     RP.ADDR,R0              ;               6649
000174  105760  000015                  TSTB    15(R0)
000200  100017                          BPL     4$
000202  132760  000100  000015          BITB    #100,15(R0)             ;               6650
000210  001413                          BEQ     4$
000212  016046  000040                  MOV     40(R0),-(SP)            ;               6652
000216  011646                          MOV     (SP),-(SP)
000220  012746  000000G                 MOV     #EX.BB1,-(SP)
000224  012746  000003                  MOV     #3,-(SP)
000230  010600                          MOV     SP,R0                   ; SP,*
000232  104415                          TRAP    15
000234  062706  000010                  ADD     #10,SP
000240  000207                  4$:     RTS     PC                      ;               6618

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

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

                                        .SBTTL  EMS.BC ERROR MESSAGE SUBROUTINES
000000  013700  000000G         EMS.BC: MOV     RP.ADDR,R0                 ;               6666
000004  016046  000044                  MOV     44(R0),-(SP)
000010  012746  000000G                 MOV     #EX.CBC,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                      ; SP,*
000022  104415                          TRAP    15
000024  013700  000000G                 MOV     RP.ADDR,R0                 ;               6667
000030  016016  000020                  MOV     20(R0),(SP)
000034  012746  000000G                 MOV     #EX.BC,-(SP)
000040  012746  000002                  MOV     #2,-(SP)
000044  010600                          MOV     SP,R0                      ; SP,*
000046  104415                          TRAP    15
000050  062706  000012                  ADD     #12,SP                     ;               6665
000054  000207                          RTS     PC                         ;               6654

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

```
;    6669 1     routine EMS_BD : novalue =
;    6670 1
;    6671 1     !+
;    6672 1     !        THIS ROUTINE PRINTS (EXTENDED) THE TWO-WORD I/O BUFFER DESCRIPTOR
;    6673 1     !        APPEARING IN THE CURRENT RETURN PACKET.
;    6674 1     !
;    6675 1     !        IMPLICIT INPUTS:
;    6676 1     !                RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    6677 1     !-
;    6678 1
;    6679 1         PRINTX (EX_BD, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);       ! "I/O BUFFER DESCRIPTOR: XXXXXX XXXXXX"


                                          .SBTTL  EMS.BD ERROR MESSAGE SUBROUTINES
000000  013700  000000G          EMS.BD:  MOV     RP.ADDR,R0                      ;                 6679
000004  016046  000024                    MOV     24(R0),-(SP)
000010  016046  000026                    MOV     26(R0),-(SP)
000014  012746  000000G                   MOV     #EX.BD,-(SP)
000020  012746  000003                    MOV     #3,-(SP)
000024  010600                            MOV     SP,R0                           ; SP,*
000026  104415                            TRAP    15
000030  062706  000010                    ADD     #10,SP
000034  000207                            RTS     PC                              ;                 6669

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

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

                                            .SBTTL  EMS.RP ERROR MESSAGE SUBROUTINES
000000  010146                     EMS.RP:  MOV     R1,-(SP)                 ;          6682
000002  004737  010404'                     JSR     PC,EMS.SBC               ;          6692
000006  004737  011114'                     JSR     PC,EMS.CMD               ;          6693
000012  013700  000000G                     MOV     RP.ADDR,R0               ;          6695
000016  116000  000014                      MOVB    14(R0),R0
000022  042700  177600                      BIC     #177600,R0
000026  020027  000011                      CMP     R0,#11
000032  001402                              BEQ     1$
000034  004737  011750'                     JSR     PC,EMS.LBN               ;          6698
000040  013700  000000G            1$:      MOV     RP.ADDR,R0               ;          6700
000044  116001  000014                      MOVB    14(R0),R1
000050  042701  177600                      BIC     #177600,R1
000054  020127  000041                      CMP     R1,#41
000060  001407                              BEQ     2$
000062  116000  000014                      MOVB    14(R0),R0                ;          6701
000066  042700  177600                      BIC     #177600,R0
000072  020027  000042                      CMP     R0,#42
000076  001004                              BNE     3$
000100  004737  012212'            2$:      JSR     PC,EMS.BC                ;          6705
```

K2


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

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


;    6711  1
;    6712  1
```

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


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

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

```
;   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
;   3           PRINTB (.ELG_FMT [.REASON])                                 ! PRINT BASIC REASON
;   2       else
;   2           PRINTB (.ELG_FMT [.REASON], .DISK_NUM);                     ! PRINT BASIC REASON WITH DISK NUMBER
;   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 [12]);
;   2               end;
;   2
;   2       if (.ELOG_CODE eql ST_MFE) and
;   3          (.ELOG_SUB lequ 10)
;   2       then
;   3           begin
;   3           PRINTX (CRLF);
;   3           PRINTX (ASTERISK);
;   3           PRINTX (.TBL_MFE [.ELOG_SUB]);                              ! MEDIA FORMAT ERROR
```

```
;  2              end;
;  2
;  2          if (.ELOG_CODE eql ST_DAT) and
;  3             (.ELOG_SUB lequ 15)
;  2          then
;  3              begin
;  3              PRINTX (CRLF);
;  3              PRINTX (ASTERISK);
;  3              PRINTX (.TBL_DAT [.ELOG_SUB]);                              ! DATA ERROR
;  2              end;
;  2
;  2          if (.ELOG_CODE eql ST_HST) and
;  3             (.ELOG_SUB lequ 4)
;  2          then
;  3              begin
;  3              PRINTX (CRLF);
;  3              PRINTX (ASTERISK);
;  3              PRINTX (.TBL_HST [.ELOG_SUB]);                              ! HOST ACCESS ERROR
;  2              end;
;  2
;  2          if (.ELOG_CODE eql ST_CNT) and
;  3             (.ELOG_SUB lequ 3)
;  2          then
;  3              begin
;  3              PRINTX (CRLF);
;  3              PRINTX (ASTERISK);
;  3              PRINTX (.TBL_CNT [.ELOG_SUB]);                              ! CONTROLLER ERROR
;  2              end;
;  2
;  2          if (.ELOG_CODE eql ST_DRV) and
;  3             (.ELOG_SUB lequ 8)
;  2          then
;  3              begin
;  3              PRINTX (CRLF);
;  3              PRINTX (ASTERISK);
;  3              PRINTX (.TBL_DRV [.ELOG_SUB]);                              ! DRIVE ERROR
;  2              end;
;  2
;  2          if .REASON eql FORMAT_XFER                                      ! IF DISK XFER INVOLVED
;  2          then
;  2
;  2              if .ELOG_ADDR [EL_BLOCK_TYPE] eql TYPE_LBN                  ! PRINT PBN OR RBN
;  2              then
;  3                  PRINTX (EX_PBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK])
;  2              else
;  2                  PRINTX (EX_RBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK]);
;  2
;  2          EMS_TIM ();                                                     ! TIME
;  2          EMS_BLK ((.ELOG_ADDR + 2), ((.ELOG_ADDR [EL_MSGLEN] + 1) / 2) + 2); ! PRINTX CONTENTS OF PACKET
;  2          ELOG_ADDR [EL_CONTENTS] = EMPTY;                                ! DECLARE SAVE AREA FREE
;  2
;  1          end;
```

```
                                               .SBTTL  EMS.EL ERROR MESSAGE SUBROUTINES
000000  004137  000000G              EMS.EL::JSR     R1,$SAVE5                       ;                               6725
000004  005746                               TST     -(SP)
000006  016646  000020                       MOV     20(SP),-(SP)                    ; INDEX,*                       6741
000012  012746  000102                       MOV     #102,-(SP)
000016  004737  000000G                      JSR     PC,BL$MUL
000022  062700  000000G                      ADD     #ELOG.PKT,R0
000026  010001                               MOV     R0,R1                           ; *,ELOG.ADDR
000030  116166  000016  000004               MOVB    16(R1),4(SP)                    ; *(ELOG.ADDR),REASON           6742
000036  105066  000005                       CLRB    5(SP)                           ; REASON
000042  116105  000012                       MOVB    12(R1),R5                       ; *(ELOG.ADDR),DISK.NUM         6743
000046  116100  000020                       MOVB    20(R1),R0                       ; *(ELOG.ADDR),*                6744
000052  042700  177740                       BIC     #177740,R0
000056  105004                               CLRB    R4                              ; ELOG.CODE
000060  050004                               BIS     R0,R4                           ; *,ELOG.CODE
000062  016103  000020                       MOV     20(R1),R3                       ; *(ELOG.ADDR),ELOG.SUB         6745
000066  006203                               ASR     R3                              ; ELOG.SUB
000070  006203                               ASR     R3                              ; ELOG.SUB
000072  006203                               ASR     R3                              ; ELOG.SUB
000074  006203                               ASR     R3                              ; ELOG.SUB
000076  006203                               ASR     R3                              ; ELOG.SUB
000100  042703  174000                       BIC     #174000,R3                      ; *,ELOG.SUB
000104  012716  000000G                      MOV     #ELG.00,(SP)                    ;                               6746
000110  012746  000001                       MOV     #1,-(SP)
000114  010600                               MOV     SP,R0                           ; SP,*
000116  104414                               TRAP    14
000120  016602  000006                       MOV     6(SP),R2                        ; REASON,*                      6751
000124  006302                               ASL     R2
000126  005766  000006                       TST     6(SP)                           ; REASON                        6748
000132  001404                               BEQ     1$
000134  026627  000006  000001               CMP     6(SP),#1                        ; REASON,*                      6749
000142  001007                               BNE     2$
000144  016216  000000G              1$:     MOV     ELG.FMT(R2),(SP)                ;                               6751
000150  012746  000001                       MOV     #1,-(SP)
000154  010600                               MOV     SP,R0                           ; SP,*
000156  104414                               TRAP    14
000160  000411                               BR      3$                              ;                               6748
000162  005016                       2$:     CLR     (SP)                            ;                               6753
000164  110516                               MOVB    R5,(SP)                         ; DISK.NUM,*
000166  016246  000000G                      MOV     ELG.FMT(R2),-(SP)
000172  012746  000002                       MOV     #2,-(SP)
000176  010600                               MOV     SP,R0                           ; SP,*
000200  104414                               TRAP    14
000202  005726                               TST     (SP)+
000204  105704                       3$:     TSTB    R4                              ; ELOG.CODE                     6755
000206  001423                               BEQ     4$
000210  120427  000013                       CMPB    R4,#13                          ; ELOC.CODE,*                   6756
000214  101020                               BHI     4$
000216  012716  000000G                      MOV     #ASTERISK,(SP)                  ;                               6759
000222  012746  000001                       MOV     #1,-(SP)
000226  010600                               MOV     SP,R0                           ; SP,*
000230  104415                               TRAP    15
```

```
000232  005000                          CLR     R0                      ;                               6760
000234  150400                          BISB    R4,R0                   ; ELOG.CODE,*
000236  006300                          ASL     R0
000240  016016  177776G                 MOV     ERR.COD-2(R0),(SP)
000244  012746  000001                  MOV     #1,-(SP)
000250  010600                          MOV     SP,R0                   ; SP,*
000252  104415                          TRAP    15
000254  000417                          BR      5$                      ;                               6758
000256  120427  000037          4$:     CMPB    R4,#37                  ; ELOG.CODE,*                   6764
000262  001015                          BNE     6$
000264  012716  000000G                 MOV     #ASTERISK,(SP)          ;                               6767
000270  012746  000001                  MOV     #1,-(SP)
000274  010600                          MOV     SP,R0                   ; SP,*
000276  104415                          TRAP    15
000300  013716  000030G                 MOV     ERR.COD+30,(SP)         ;                               6768
000304  012746  000001                  MOV     #1,-(SP)
000310  010600                          MOV     SP,R0                   ; SP,*
000312  104415                          TRAP    15
000314  022626                  5$:     CMP     (SP)+,(SP)+             ;                               6766
000316  120427  000005          6$:     CMPB    R4,#5                   ; ELOG.CODE,*                   6771
000322  001031                          BNE     7$
000324  020327  000012                  CMP     R3,#12                  ; ELOG.SUB,*                    6772
000330  101026                          BHI     7$
000332  012716  000000G                 MOV     #CRLF,(SP)              ;                               6775
000336  012746  000001                  MOV     #1,-(SP)
000342  010600                          MOV     SP,R0                   ; SP,*
000344  104415                          TRAP    15
000346  012716  000000G                 MOV     #ASTERISK,(SP)          ;                               6776
000352  012746  000001                  MOV     #1,-(SP)
000356  010600                          MOV     SP,R0                   ; SP,*
000360  104415                          TRAP    15
000362  010300                          MOV     R3,R0                   ; ELOG.SUB,*                    6777
000364  006300                          ASL     R0
000366  016016  000064'                 MOV     TBL.MFE(R0),(SP)
000372  012746  000001                  MOV     #1,-(SP)
000376  010600                          MOV     SP,R0                   ; SP,*
000400  104415                          TRAP    15
000402  062706  000006                  ADD     #6,SP                   ;                               6774
000406  120427  000010          7$:     CMPB    R4,#10                  ; ELOG.CODE,*                   6780
000412  001031                          BNE     8$
000414  020327  000017                  CMP     R3,#17                  ; ELOG.SUB,*                    6781
000420  101026                          BHI     8$
000422  012716  000000G                 MOV     #CRLF,(SP)              ;                               6784
000426  012746  000001                  MOV     #1,-(SP)
000432  010600                          MOV     SP,R0                   ; SP,*
000434  104415                          TRAP    15
000436  012716  000000G                 MOV     #ASTERISK,(SP)          ;                               6785
000442  012746  000001                  MOV     #1,-(SP)
000446  010600                          MOV     SP,R0                   ; SP,*
000450  104415                          TRAP    15
000452  010300                          MOV     R3,R0                   ; ELOG.SUB,*                    6786
000454  006300                          ASL     R0
000456  016016  000120'                 MOV     TBL.DAT(R0),(SP)
```

```
000462  012746  000001                  MOV     #1,-(SP)
000466  010600                          MOV     SP,R0                   ; SP,*
000470  104415                          TRAP    15
000472  062706  000006                  ADD     #6,SP                   ;                       6783
000476  120427  000011          8$:     CMPB    R4,#11                  ; ELOG.CODE,*           6789
000502  001031                          BNE     9$
000504  020327  000004                  CMP     R3,#4                   ; ELOG.SUB,*            6790
000510  101026                          BHI     9$
000512  012716  000000G                 MOV     #CRLF,(SP)              ;                       6793
000516  012746  000001                  MOV     #1,-(SP)
000522  010600                          MOV     SP,R0                   ; SP,*
000524  104415                          TRAP    15
000526  012716  000000G                 MOV     #ASTERISK,(SP)          ;                       6794
000532  012746  000001                  MOV     #1,-(SP)
000536  010600                          MOV     SP,R0                   ; SP,*
000540  104415                          TRAP    15
000542  010300                          MOV     R3,R0                   ; ELOG.SUB,*            6795
000544  006300                          ASL     R0
000546  016016  000160'                 MOV     TBL.HST(R0),(SP)
000552  012746  000001                  MOV     #1,-(SP)
000556  010600                          MOV     SP,R0                   ; SP,*
000560  104415                          TRAP    15
000562  062706  000006                  ADD     #6,SP                   ;                       6792
000566  120427  000012          9$:     CMPB    R4,#12                  ; ELOG.CODE,*           6798
000572  001031                          BNE     10$
000574  020327  000003                  CMP     R3,#3                   ; ELOG.SUB,*            6799
000600  101026                          BHI     10$
000602  012716  000000G                 MOV     #CRLF,(SP)              ;                       6802
000606  012746  000001                  MOV     #1,-(SP)
000612  010600                          MOV     SP,R0                   ; SP,*
000614  104415                          TRAP    15
000616  012716  000000G                 MOV     #ASTERISK,(SP)          ;                       6803
000622  012746  000001                  MOV     #1,-(SP)
000626  010600                          MOV     SP,R0                   ; SP,*
000630  104415                          TRAP    15
000632  010300                          MOV     R3,R0                   ; ELOG.SUB,*            6804
000634  006300                          ASL     R0
000636  016016  000172'                 MOV     TBL.CNT(R0),(SP)
000642  012746  000001                  MOV     #1,-(SP)
000646  010600                          MOV     SP,R0                   ; SP,*
000650  104415                          TRAP    15
000652  062706  000006                  ADD     #6,SP                   ;                       6801
000656  120427  000013          10$:    CMPB    R4,#13                  ; ELOG.CODE,*           6807
000662  001031                          BNE     11$
000664  020327  000010                  CMP     R3,#10                  ; ELOG.SUB,*            6808
000670  101026                          BHI     11$
000672  012716  000000G                 MOV     #CRLF,(SP)              ;                       6811
000676  012746  000001                  MOV     #1,-(SP)
000702  010600                          MOV     SP,R0                   ; SP,*
000704  104415                          TRAP    15
000706  012716  000000G                 MOV     #ASTERISK,(SP)          ;                       6812
000712  012746  000001                  MOV     #1,-(SP)
000716  010600                          MOV     SP,R0                   ; SP,*
```

```
000720  104415                             TRAP    15
000722  010300                             MOV     R3,R0                   ; ELOG.SUB,*                    6813
000724  006300                             ASL     R0
000726  016016  000202'                    MOV     TBL.DRV(R0),(SP)
000732  012746  000001                     MOV     #1,-(SP)
000736  010600                             MOV     SP,R0                   ; SP,*
000740  104415                             TRAP    15
000742  062706  000006                     ADD     #6,SP                   ;                               6810
000746  026627  000010  000002    11$:     CMP     10(SP),#2               ; REASON,*                      6816
000754  001031                             BNE     14$
000756  032761  170000  000060             BIT     #170000,60(R1)          ; *,*(ELOG.ADDR)                6819
000764  001012                             BNE     12$
000766  016116  000056                     MOV     56(R1),(SP)             ; *(ELOG.ADDR),*                6821
000772  011646                             MOV     (SP),-(SP)
000774  012746  000000G                    MOV     #EX.PBN,-(SP)
001000  012746  000003                     MOV     #3,-(SP)
001004  010600                             MOV     SP,R0                   ; SP,*
001006  104415                             TRAP    15
001010  000411                             BR      13$                     ;                               6819
001012  016116  000056            12$:     MOV     56(R1),(SP)             ; *(ELOG.ADDR),*                6823
001016  011646                             MOV     (SP),-(SP)
001020  012746  000000G                    MOV     #EX.RBN,-(SP)
001024  012746  000003                     MOV     #3,-(SP)
001030  010600                             MOV     SP,R0                   ; SP,*
001032  104415                             TRAP    15
001034  062706  000006            13$:     ADD     #6,SP                   ;                               6819
001040  004737  000000V           14$:     JSR     PC,EMS.TIM              ;                               6825
001044  012716  000002                     MOV     #2,(SP)                 ;                               6826
001050  060116                             ADD     R1,(SP)                 ; ELOG.ADDR,*
001052  016146  000002                     MOV     2(R1),-(SP)             ; *(ELOG.ADDR),*
001056  005216                             INC     (SP)
001060  012746  000002                     MOV     #2,-(SP)
001064  004737  000000G                    JSR     PC,BL$DIV
001070  010066  000002                     MOV     R0,2(SP)
001074  062766  000002  000002             ADD     #2,2(SP)
001102  005726                             TST     (SP)+
001104  004737  011576'                    JSR     PC,EMS.BLK
001110  105061  000001                     CLRB    1(R1)                   ; *(ELOG.ADDR)                  6827
001114  062706  000014                     ADD     #14,SP                  ;                               6725
001120  000207                             RTS     PC

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

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

                                                .SBTTL  EMS.CMP ERROR MESSAGE SUBROUTINES
000000  010146                          EMS.CMP::
                                                MOV     R1,-(SP)                ;                               6830
000002  016601  000004                          MOV     4(SP),R1                ; ADDR,ORIG.ADDR                6842
000006  013746  000000G                         MOV     CDISK,-(SP)             ;                               6843
000012  012746  000000G                         MOV     #ERR.00,-(SP)
000016  012746  000002                          MOV     #2,-(SP)
000022  010600                                  MOV     SP,R0                   ; SP,*
000024  104414                                  TRAP    14
000026  012716  000000G                         MOV     #DASH,(SP)              ;                               6844
000032  012746  000001                          MOV     #1,-(SP)
000036  010600                                  MOV     SP,R0                   ; SP,*
000040  104414                                  TRAP    14
000042  013716  000030G                         MOV     ERR.COD+30,(SP)         ;                               6845
000046  012746  000001                          MOV     #1,-(SP)
000052  010600                                  MOV     SP,R0                   ; SP,*
000054  104414                                  TRAP    14
000056  016116  000050                          MOV     50(R1),(SP)             ; *(ORIG.ADDR),*                6846
000062  011646                                  MOV     (SP),-(SP)
000064  012746  000000G                         MOV     #EX.LBW,-(SP)
000070  012746  000003                          MOV     #3,-(SP)
000074  010600                                  MOV     SP,R0                   ; SP,*
000076  104415                                  TRAP    15
000100  013700  000000G                         MOV     RP.ADDR,R0              ;                               6847
000104  016016  000050                          MOV     50(R0),(SP)
```

```
000110  011646                  MOV     (SP),-(SP)
000112  012746  000000G         MOV     #EX.LBR,-(SP)
000116  012746  000003          MOV     #3,-(SP)
000122  010600                  MOV     SP,R0                   ; SP,*
000124  104415                  TRAP    15
000126  016116  000044          MOV     44(R1),(SP)             ; *(ORIG.ADDR),*                6848
000132  012746  000000G         MOV     #EX.CBW,-(SP)
000136  012746  000002          MOV     #2,-(SP)
000142  010600                  MOV     SP,R0                   ; SP,*
000144  104415                  TRAP    15
000146  016116  000020          MOV     20(R1),(SP)             ; *(ORIG.ADDR),*                6849
000152  012746  000000G         MOV     #EX.BC,-(SP)
000156  012746  000002          MOV     #2,-(SP)
000162  010600                  MOV     SP,R0                   ; SP,*
000164  104415                  TRAP    15
000166  013700  000000G         MOV     RP.ADDR,R0              ;                               6850
000172  016016  000044          MOV     44(R0),(SP)
000176  012746  000000G         MOV     #EX.CBR,-(SP)
000202  012746  000002          MOV     #2,-(SP)
000206  010600                  MOV     SP,R0                   ; SP,*
000210  104415                  TRAP    15
000212  013700  000000G         MOV     RP.ADDR,R0              ;                               6851
000216  016016  000020          MOV     20(R0),(SP)
000222  012746  000000G         MOV     #EX.BC,-(SP)
000226  012746  000002          MOV     #2,-(SP)
000232  010600                  MOV     SP,R0                   ; SP,*
000234  104415                  TRAP    15
000236  016116  000024          MOV     24(R1),(SP)             ; *(ORIG.ADDR),*                6852
000242  016146  000026          MOV     26(R1),-(SP)            ; *(ORIG.ADDR),*
000246  012746  000000G         MOV     #EX.BDW,-(SP)
000252  012746  000003          MOV     #3,-(SP)
000256  010600                  MOV     SP,R0                   ; SP,*
000260  104415                  TRAP    15
000262  013700  000000G         MOV     RP.ADDR,R0              ;                               6853
000266  016016  000024          MOV     24(R0),(SP)
000272  016046  000026          MOV     26(R0),-(SP)
000276  012746  000000G         MOV     #EX.BDR,-(SP)
000302  012746  000003          MOV     #3,-(SP)
000306  010600                  MOV     SP,R0                   ; SP,*
000310  104415                  TRAP    15
000312  004737  000000V         JSR     PC,EMS.TIM              ;                               6854
000316  062706  000062          ADD     #62,SP                  ;                               6837
000322  012601                  MOV     (SP)+,R1                ;                               6830
000324  000207                  RTS     PC

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

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

                                                .SBTTL  EMS.ERR ERROR MESSAGE SUBROUTINES
000000  013746  000000G                 EMS.ERR::
                                                MOV     CDISK,-(SP)             ;                       6862
000004  012746  000000G                         MOV     #ERR.00,-(SP)
000010  012746  000002                          MOV     #2,-(SP)
000014  010600                                  MOV     SP,R0                   ; SP,*
000016  104414                                  TRAP    14
000020  012716  000000G                         MOV     #DASH,(SP)              ;                       6863
000024  012746  000001                          MOV     #1,-(SP)
000030  010600                                  MOV     SP,R0                   ; SP,*
000032  104414                                  TRAP    14
000034  013700  000000G                         MOV     ST.CODE,R0              ;                       6865
000040  001413                                  BEQ     1$
000042  020027  000013                          CMP     R0,#13                  ;                       6866
000046  101010                                  BHI     1$
000050  006300                                  ASL     R0                      ;                       6868
000052  016016  177776G                         MOV     ERR.COD-2(R0),(SP)
000056  012746  000001                          MOV     #1,-(SP)
000062  010600                                  MOV     SP,R0                   ; SP,*
000064  104414                                  TRAP    14
000066  000422                                  BR      3$                      ;                       6865
000070  020027  000037                  1$:     CMP     R0,#37                  ;                       6871
000074  001007                                  BNE     2$
000076  013716  000026G                         MOV     ERR.COD+26,(SP)         ;                       6873
000102  012746  000001                          MOV     #1,-(SP)
000106  010600                                  MOV     SP,R0                   ; SP,*
000110  104414                                  TRAP    14
```

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

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

```
;   6880 1      routine EMS_TIM : novalue =
;   6881 1
;   6882 1      !+
;   6883 1      ! THIS ROUTINE PRINTS THE TIME-OF-DAY MESSAGE
;   6884 1      !-
;   6885 1
;   6386 1      PRINTX (EX_TIM, .HOURS, .MINUTES);


                                            .SBTTL  EMS.TIM ERROR MESSAGE SUBROUTINES
000000  005046                     EMS.TIM:CLR     -(SP)                          ;               6886
000002  113716  000000G                    MOVB    MINUTES,(SP)
000006  005046                             CLR     -(SP)
000010  113716  000000G                    MOVB    HOURS,(SP)
000014  012746  000000G                    MOV     #EX.TIM,-(SP)
000020  012746  000003                     MOV     #3,-(SP)
000024  010600                             MOV     SP,R0                          ; SP,*
000026  104415                             TRAP    15
000030  062706  000010                     ADD     #10,SP
000034  000207                             RTS     PC                             ;               6880

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

K3


```
;    6887  1      BGNMSG (EMS_01);


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

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


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


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

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

```
;   6890  1      BGNMSG (EMS_10);


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

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


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


                                                        .SBTTL  M$EMS.10 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                       M$EMS.10:
                                                        MOV     RDRX.ADDR,-(SP)                    ;                    6891
000004  063716  000000G                                 ADD     OF.RC,(SP)
000010  012746  000000G                                 MOV     #EBD.10,-(SP)
000014  012746  000002                                  MOV     #2,-(SP)
000020  010600                                          MOV     SP,R0                              ; SP,*
000022  104414                                          TRAP    14
000024  062706  000006                                  ADD     #6,SP                              ;                    6890
000030  000207                                          RTS     PC

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

```
;   6893  1      BGNMSG (EMS_12);

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

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

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

                                                  .SBTTL  M$EMS.12 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                 M$EMS.12:
                                                  MOV     RDRX.ADDR,-(SP)                      ;                      6894
000004  012746  000000G                           MOV     #EBD.12,-(SP)
000010  012746  000002                            MOV     #2,-(SP)
000014  010600                                    MOV     SP,R0                                ; SP,*
000016  104414                                    TRAP    14
000020  062706  000006                            ADD     #6,SP                                ;                      6893
000024  000207                                    RTS     PC

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

```
;    6896  1      BGNMSG (EMS_13);


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

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


;    6897  2      PRINTB (EBD_13, .STEP);                          ! "STEP X READ ERROR"
;    6898  2      EMS_SA ();                                       ! PRINTX SA CONTENTS
;    6899  1      ENDMSG;


                                                  .SBTTL  M$EMS.13 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                  M$EMS.13:
                                                  MOV     STEP,-(SP)                  ;                              6897
000004  012746  000000G                           MOV     #EBD.13,-(SP)
000010  012746  000002                            MOV     #2,-(SP)
000014  010600                                    MOV     SP,R0                       ; SP,*
000016  104414                                    TRAP    14
000020  004737  010072'                           JSR     PC,EMS.SA                   ;                              6898
000024  062706  000006                            ADD     #6,SP                       ;                              6896
000030  000207                                    RTS     PC

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

```
;    6900  1       BGNMSG (EMS_14);


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

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


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


                                                      .SBTTL  M$EMS.14 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                    M$EMS.14:
                                                   MOV     IRDRX.ADDR,-(SP)                   ;                     6901
000004  012746  000000G                            MOV     #EBD.14,-(SP)
000010  012746  000002                             MOV     #2,-(SP)
000014  010600                                     MOV     SP,R0                              ; SP,*
000016  104414                                     TRAP    14
000020  004737  010072'                            JSR     PC,EMS.SA                          ;                     6902
000024  062706  000006                             ADD     #6,SP                              ;                     6900
000030  000207                                     RTS     PC

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

```
;   6904  1      BGNMSG (EMS_18);

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

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

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

                                             .SBTTL  M$EMS.18 ERROR MESSAGE SUBROUTINES
000000  013746  000000G          M$EMS.18:
                                         MOV     CDISK,-(SP)                          ;                    6905
000004  012746  000000G                  MOV     #EBD.18,-(SP)
000010  012746  000002                   MOV     #2,-(SP)
000014  010600                           MOV     SP,R0                                ; SP,*
000016  104414                           TRAP    14
000020  004737  012326'                  JSR     PC,EMS.RP                            ;                    6906
000024  062706  000006                   ADD     #6,SP                                ;                    6904
000030  000207                           RTS     PC

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

```
;    6908 1      BGNMSG (EMS_21);

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

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


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

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

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

```
;   6911  1     BGNMSG (EMS_22)                    !CONTENTS OF DUP BUFFER          ZZZ

                                        .SBTTL  EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  000000V        EMS.22::JSR      PC,M$EMS.22                     ;                     6911
000004  104423                         TRAP     23
000006  000207                         RTS      PC

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

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

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

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

```
;    6914  1      BGNMSG (EMS_24);


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

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


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


                                               .SBTTL  M$EMS.24 ERROR MESSAGE SUBROUTINES
000000  013746  000000G              M$EMS.24:
                                             MOV     CDISK,-(SP)                    ;                     6915
000004  012746  000000G                      MOV     #EBD.24,-(SP)
000010  012746  000002                       MOV     #2,-(SP)
000014  010600                               MOV     SP,R0                          ; SP,*
000016  104414                               TRAP    14
000020  004737  012326'                      JSR     PC,EMS.RP                      ;                     6916
000024  062706  000006                       ADD     #6,SP                          ;                     6914
000030  000207                               RTS     PC

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

```
;   6918  1       BGNMSG (EMS_30);

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

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

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

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

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


;   6921  1
;   6922  1       end
;   6923  1
;   6924  0       eludom


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


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



;                               Library Statistics
;
;                                               -------- Symbols --------       Pages       Processing
;       File                                    Total    Loaded   Percent       Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQAFO.L16;11              406      297       73            21          00:00.1
```

```
;                                        COMMAND QUALIFIERS

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

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

```
module ZRQAM3 (

%title 'RD/RX EXERCISER'
                ident = 'V02.1',
                addressing_mode (absolute),
                environment (noeis)
                ) =

begin

%sbttl 'DECLARATIONS'

library 'ZRQAFO.L16';                           ! RDRX EXERCISER GLOBAL LIBRARY

require 'BLSMAC.REQ';                           ! DIAGNOSTIC SUPERVISOR LIBRARY

EQUALS;

forward routine                                 ! ROUTINES APPEAR IN THIS ORDER
    INIT_TEST : novalue,                        !    INDENTATION IMPLIES CALLED SUBROUTINE
        DRIVER_INIT : novalue,
            CTLR_INIT : novalue,
                INI_CTLR_DAT : novalue,
                REG_EXIST,
                VEC_BR_TEST,
                INT_GEN,
                HARD_INIT,
                INI_RRING : novalue,
                SET_CTLR_CHAR,
            UNIT_INIT : novalue,
                DR_ERR : novalue,
        ACCESS : novalue,
    MULTI_DRIVE : novalue,
        MD_INIT : novalue,
            INIT_IO_BUFF : novalue,
        FATAL_ERROR : novalue,
        QIO_OK,
            QIO_OUT,
            QIO_GEN : novalue,
                GET_RANDOM : novalue,
                QIO_UNIT : novalue,
                QIO_FUNC : novalue,
                   DUP : NOVALUE,                                !ZZZ
                         DUPWRTDBN : NOVALUE,                    !ZZZ
                         DUPREDDBN : NOVALUE,                    !ZZZ
                              DUPCOMMAND : NOVALUE,              !ZZZ
                              DUPIDLE : NOVALUE,                 !ZZZ
                QIO_LBN : novalue,
                QIO_SIZE : novalue,
                FILL_BUFF : novalue,
            PROC_RETPKT : novalue,
                DIO_RETPKT : NOVALUE,                            !ZZZ
```

```
                    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,
                DUP_RSP  : NOVALUE,
!ZZZ
                DISK_RSP : novalue,
                    SEQUEN : novalue,
                        SCAN_ERRLOG : novalue,
                            ERR_SOFT_RTNE : novalue,
                            ERR_SOFT_RTNE_APT : novalue,
                            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
    DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
```

```
;  1                  !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      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      S_PATTERN : WORD,              !PATTERN WRITTEN TO DBNS          ZZZ
;   1      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      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
;   1      EH_7,                                 !ZZZ
;   1      EH_8,                                 !ZZZ
;   1      EH_9,                                 !ZZZ
;   1      EH_10,                                !ZZZ
;   1      EH_12,                                !ZZZ
;   1      EH_13,                                !ZZZ
;   1      MSG_02,
;   1      MSG_03,
;   1      EGS_02,
;   1      EGD_10,
;   1      EGD_11,
;   1      EGD_12,
;   1      EGD_13,
;   1      EGD_14,
;   1      EGD_15,
;   1      EGD_16,
```

```
;  1        EGD_17,
;  1        EGD_18,
;  1        EGD_19,
;  1        EGD_20,
;  1        EGD_21,
;  1        EGD_22,
;  1        EGD_23,
;  1        EGD_24,
;  1        EGM_30,
;  1        DF_MSG,
;  1        HRD_MSG,
;  1        SFT_MSG,
;  1        HRD_SUB,
;  1        CRLF,
;  1        SWP_ERROR : word,                                   ! HARD ERROR LIMIT FOR DROPPING UNIT
;  1        SWP_XFER : word,                                    ! TRANSFER LIMIT FOR DROPPING UNIT
;  1        SWP_FLAGS : word,                                   ! FLAGS (SEE DOCUMENTATION)
;  1        DUPROUND  : WORD,                                   !DUP TESTING RATIO                 ZZZ
;  1        SWP_RAT : word,                                     ! RD51/52 OPERATION RATIO
;  1        SWP_DPAT : word,                                    ! DATA PATTERN NUMBER
;  1        SWP_UCNT : word,                                    ! USER DATA PATTERN COUNT
;  1        SWP_TIME : word,                                    ! TIME OF DAY
;  1        SWP_UDPAT : vector [MAX_UDP_CNT, word],             ! USER DATA PATTERN
;  1        L$LUN,
;  1        L$UNIT;
;  1
;  1    psect
;  1        own = $GGG$(read, nowrite, execute, local, concatenate);
;  1
;  1    own
;  1        COMM_AREA : blockvector [MAX_CTLR, COMM_LEN, word] field (COM_FIELDS),
;  1                            ! COMMUNICATIONS AREA BETWEEN HOST AND AZTEC CONTROLLERS
;  1    !!ZZZ    BST : vector [MAX_UNITS, word, signed],
;  1                            ! BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
;  1        DPST : vector [MAX_UNITS, byte],              ! DATA PATTERN SEQUENCE TABLE
;  1        MAX_LBN : vector [MAX_UNITS, word],           ! LARGEST LBN ALLOWED
;  1        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        RW_BALANCE : WORD INITIAL (3),                ! FLAGS TOO MANY READS IN RD/WR RATIO       ZZZ
;  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    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',
;  1            %o'177000', %o'176000', %o'174000', %o'170000',
;  1            %o'160000', %o'140000', %o'100000', %o'000000'),
;  1    PAT09 : vector [17] initial (16,                                ! PATTERN 9
;  1            rep 3 of (%o'000000'), rep 3 of (%o'177777'),
;  1            rep 2 of (%o'000000'), rep 2 of (%o'177777'),
;  1            %o'000000', %o'177777', %o'000000', %o'177777',
;  1            %o'000000', %o'177777'),
;  1    PAT10 : vector [2] initial (1,                                  ! PATTERN 10
;  1            %o'133331'),
;  1    PAT11 : vector [17] initial (16,                                ! PATTERN 11
;  1            rep 3 of (%o'052525'), rep 3 of (%o'125252'),
;  1            rep 2 of (%o'052525'), rep 2 of (%o'125252'),
;  1            %o'052525', %o'125252', %o'052525', %o'125252',
;  1            %o'052525', %o'125252'),
;  1    PAT12 : vector [21] initial (20,                                ! PATTERN 12
;  1            rep 3 of (%o'026455'), rep 3 of (%o'151322'),
;  1            rep 2 of (%o'026455'), rep 2 of (%o'151322'),
;  1            rep 2 of (%o'026455'),
;  1            %o'151322', %o'026455', %o'151322', %o'026455',
;  1            %o'151322', %o'026455', %o'151322', %o'026455'),
;  1    PAT13 : vector [2] initial (1,                                  ! PATTERN 13
;  1            %o'066666'),
;  1    PAT14 : vector [17] initial (16,                                ! PATTERN 14
;  1            %o'000001', %o'000002', %o'000004', %o'000010',
;  1            %o'000020', %o'000040', %o'000100', %o'000200',
;  1            %o'000400', %o'001000', %o'002000', %o'004000',
;  1            %o'010000', %o'020000', %o'040000', %o'100000'),
;  1    PAT15 : vector [17] initial (16,                                ! PATTERN 15
;  1            %o'177776', %o'177775', %o'177773', %o'177767',
;  1            %o'177757', %o'177737', %o'177677', %o'177577',
;  1            %o'177377', %o'176777', %o'175777', %o'173777',
```

```
;  1          %o'167777', %o'157777', %o'137777', %o'077777'),
;  1      PAT16 : vector [17] initial (16,                        ! PATTERN 16
;  1          rep 3 of (%o'133331'), rep 3 of (%o'155554'),
;  1          rep 2 of (%o'133331'), rep 2 of (%o'155554'),
;  1          %o'133331', %o'155554', %o'133331', %o'155554',
;  1          %o'133331', %o'155554'),
;  1      PAT17 : vector [22] initial (21,                        ! PATTERN 17
;  1          %o'000000', rep 2 of (%o'106466'),
;  1          rep 3 of (%o'071311'), rep 4 of (%o'106466'),
;  1          rep 5 of (%o'071311'), rep 6 of (%o'106466')),
;  1      PAT18 : vector [22] initial (21,                        ! PATTERN 18
;  1          %o'106466', %o'000000', %o'071311',
;  1          rep 3 of (%o'106466'), rep 4 of (%o'071311'),
;  1          rep 5 of (%o'106466'), rep 6 of (%o'071311')),
;  1      PAT19 : vector [22] initial (21,                        ! PATTERN 19
;  1          %o'000000', rep 2 of (%o'134631'),
;  1          rep 3 of (%o'043146'), rep 4 of (%o'134631'),
;  1          rep 5 of (%o'043146'), rep 6 of (%o'134631')),
;  1      PAT20 : vector [22] initial (21,                        ! PATTERN 20
;  1          %o'134631', %o'000000', %o'043146',
;  1          rep 3 of (%o'134631'), rep 4 of (%o'043146'),
;  1          rep 5 of (%o'134631'), rep 6 of (%o'043146')),
;  1      PAT21 : vector [2] initial (1,                          ! PATTERN 21
;  1          %o'000000'),                                        !    (LBN)
;  1      DPA_TBL : vector [DP_CNT] initial                       ! DATA PATTERN ADDRESS TABLE
;  1            (RDM_CNT, PAT02, PAT03, PAT04, PAT05,
;  1            PAT06, PAT07, PAT08, PAT09, PAT10, PAT11,
;  1            PAT12, PAT13, PAT14, PAT15, PAT16, PAT17,
;  1            PAT18, PAT19, PAT20, PAT21),
;  1      BST_CNT : word initial (0),               ! CURRENT SEQUENTIAL BLOCK COUNT
;  1      BST_DEV : word initial (0),               ! CURRENT SEQUENTIAL BLOCK DEVICE
;  1      CURRENT_VECTOR : word,                    ! CURRENT DEVICE'S VECTOR ADDRESS
;  1  !ZZZ    BRLEVEL : word,                       ! CURRENT DEVICE'S BR LEVEL              ZZZ
;  1      DUOFF   : WORD,                           !DUP OFFSET INTO CST                     ZZZ
;  1      DRS_START,                                ! START OF THE SUPERVISOR
;  1      APT_MODE : byte initial (byte (FALSE)),   ! FLAG SET IF EXERCISER RUNNING UNDER APT
;  1      MAIL_BOX_TESTNUM,                         ! ADDRESS OF TEST NUMBER LOCATION IN APT MAIL-BOX
;  1      MAIL_BOX_SUBTST,                          ! ADDRESS OF SUB-TEST NUMBER LOCATION IN APT MAIL-BOX
;  1      COMPARE_DATA : byte,                      ! FLAG CLEARED TO BYPASS HOST COMPARES
;  1      DRS_FLAGS: word,                          ! FLAGS USED IN START/RESTART OF THE EXERCISER
;  1      RD_MAX_SEQ_CNT : word,                    ! COUNT USED IN SEQUENTIAL ACCESS OPERATIONS
;  1      RX_MAX_SEQ_CNT : word;                    !
;  1
;  1  external routine
;  1      NEX_TRAP : L$ISR novalue,
;  1      TIME : L$ISR novalue,
;  1      SET_CPAR : novalue,
;  1      SET_UPAR : novalue,
;  1      OUT_IODQ,
;  1      IN_IODQ : novalue,
;  1      GET_PKT,
;  1      PUT_PKT : novalue,
;  1      GET_RETPKT,
```

C5


```
;      1          PUT_RETPKT : novalue,
;      1          GET_IO_BUFF : novalue,
;      1          PUT_IO_BUFF : novalue,
;      1          PUTA_BUFF : novalue,
;      1          SEND,
;      1          WAIT : novalue,
;      1          MODULAS,                                  !ZZZ
;      1          DROP_CTLR : novalue,
;      1          DRV_CTLERR : novalue,
;      1          EMS_RP1 : novalue,
;      1          EMS_EL : novalue,
;      1          EMS_CMP : novalue,
;      1          EMS_ERR : novalue,
;      1          EMS_10 : novalue,
;      1          EMS_12 : novalue,
;      1          EMS_13 : novalue,
;      1          EMS_14 : novalue,
;      1          EMS_18 : novalue,
;      1          EMS_21 : novalue,
;      1          EMS_22 : NOVALUE,                         !ZZZ
;      1          EMS_24 : novalue,
;      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
;      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
```

E5


```
;  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 WHETHER 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, PRI07);                                    ! 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
;  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    RFLAGS (DRS_FLAGS);                                               ! READ DRS FLAGS INTO LOC DRS_FLAGS
;    3
;    3    if BIT_TST (DRS_FLAGS, HOE) eql HOE
;    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  /V02.1/
                                        .ENABL  AMA


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

```
                                        .WORD   3
000110                          MX1:    .BLKW   1
000112                          MX2:    .BLKW   1
000114                          MAD1:   .BLKW   1
000116                          MAD2:   .BLKW   1
000120                          LAST.PKT:
                                        .BLKW   3
000126  000001                  PAT02:  .WORD   1
000130  000000                          .WORD   0
000132  000001                  PAT03:  .WORD   1
000134  177777                          .WORD   -1
000136  000001                  PAT04:  .WORD   1
000140  105613                          .WORD   -72165
000142  000001                  PAT05:  .WORD   1
000144  031463                          .WORD   31463
000146  000001                  PAT06:  .WORD   1
000150  030221                          .WORD   30221
000152  000020                  PAT07:  .WORD   20
000154  000001                          .WORD   1
000156  000003                          .WORD   3
000160  000007                          .WORD   7
000162  000017                          .WORD   17
000164  000037                          .WORD   37
000166  000077                          .WORD   77
000170  000177                          .WORD   177
000172  000377                          .WORD   377
000174  000777                          .WORD   777
000176  001777                          .WORD   1777
000200  003777                          .WORD   3777
000202  007777                          .WORD   7777
000204  017777                          .WORD   17777
000206  037777                          .WORD   37777
000210  077777                          .WORD   77777
000212  177777                          .WORD   -1
000214  000020                  PAT08:  .WORD   20
000216  177776                          .WORD   -2
000220  177774                          .WORD   -4
000222  177770                          .WORD   -10
000224  177760                          .WORD   -20
000226  177740                          .WORD   -40
000230  177700                          .WORD   -100
000232  177600                          .WORD   -200
000234  177400                          .WORD   -400
000236  177000                          .WORD   -1000
000240  176000                          .WORD   -2000
000242  174000                          .WORD   -4000
000244  170000                          .WORD   -10000
000246  160000                          .WORD   -20000
000250  140000                          .WORD   -40000
000252  100000                          .WORD   -100000
000254  000000                          .WORD   0
000256  000020                  PAT09:  .WORD   20
000260  000000                          .WORD   0
```

H5


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

I5


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

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

K5


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

```
001132  000506'                        .WORD   PAT15
001134  000550'                        .WORD   PAT16
001136  000612'                        .WORD   PAT17
001140  000666'                        .WORD   PAT18
001142  000742'                        .WORD   PAT19
001144  001016'                        .WORD   PAT20
001146  001072'                        .WORD   PAT21
001150  000000              BST.CNT:.WORD   0
001152  000000              BST.DEV:.WORD   0
001154                      CURRENT.VECTOR:
                                       .BLKW   1
001156                      DUOFF:     .BLKW   1
001160                      DRS.START:
                                       .BLKW   1
001162                      APT.MODE:
001162     000                         .BYTE   0
                                       .EVEN
001164                      MAIL.BOX.TESTNUM:
                                       .BLKW   1
001166                      MAIL.BOX.SUBTST:
                                       .BLKW   1
001170                      COMPARE.DATA:
                                       .BLKB   1
                                       .EVEN
001172                      DRS.FLAGS:
                                       .BLKW   1
001174                      RD.MAX.SEQ.CNT:
                                       .BLKW   1
001176                      RX.MAX.SEQ.CNT:
                                       .BLKW   1


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

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

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

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

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

000000  004137  000000G                 $T1:    JSR     R1,$SAVE3               ;               1883
000004  112737  000001  000000G                 MOVB    #1,EOP.FLAG             ;               1902
000012  112737  000001  001170'                 MOVB    #1,COMPARE.DATA         ;               1903
000020  042737  000002  000000G                 BIC     #2,DUP.FLAGS            ;               1904
000026  105037  000000G                         CLRB    HOE.FLAG                ;               1905
000032  105037  000000G                         CLRB    FORCED.ERROR            ;               1906
000036  005002                                  CLR     R2                      ; I             1909
000040  010246                          1$:     MOV     R2,-(SP)                ; I,*           1913
000042  012746  000043                          MOV     #43,-(SP)
000046  004737  000000G                         JSR     PC,BL$MUL
000052  005001                                  CLR     R1                      ; J             1912
000054  010003                          2$:     MOV     R0,R3                   ;               1913
000056  060103                                  ADD     R1,R3                   ; J,*
000060  006303                                  ASL     R3
000062  005063  000000G                         CLR     MSCP.PKT(R3)
000066  005201                                  INC     R1                      ; J             1912
000070  020127  000042                          CMP     R1,#42                  ; J,*
000074  003767                                  BLE     2$
000076  010216                                  MOV     R2,(SP)                 ; I,*           1915
000100  012746  000106                          MOV     #106,-(SP)
000104  004737  000000G                         JSR     PC,BL$MUL
000110  105060  000005G                         CLRB    MSCP.PKT+5(R0)
000114  062706  000006                          ADD     #6,SP                   ;               1910
000120  005202                                  INC     R2                      ; I             1909
000122  020227  000013                          CMP     R2,#13                  ; I,*
000126  003744                                  BLE     1$
```

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

```
000424  105037  000000G                   CLRB    CLK.PRESENT              ;   1958
000430  012746  000340                    MOV     #340,-(SP)               ;   1959
000434  012746  000000G                   MOV     #NEX.TRAP,-(SP)
000440  012746  000004                    MOV     #4,-(SP)
000444  012746  000003                    MOV     #3,-(SP)
000450  104437                            TRAP    37
000452  012700  000004                    MOV     #4,R0                    ;   1962
000456  104436                            TRAP    36
000460  032737  000001  000000G           BIT     #1,NEX                   ;   1965
000466  001060                            BNE     12$
000470  112737  000001  000000G           MOVB    #1,CLK.PRESENT           ;   1968
000476  005037  000000G                   CLR     CLK.TICKS                ;   1969
000502  013716  000000G                   MOV     SWP.TIME,(SP)            ;   1970
000506  012746  000144                    MOV     #144,-(SP)
000512  004737  000000G                   JSR     PC,BL$DIV
000516  110037  000000G                   MOVB    R0,HOURS
000522  013716  000000G                   MOV     SWP.TIME,(SP)            ;   1971
000526  012746  000144                    MOV     #144,-(SP)
000532  004737  000000G                   JSR     PC,BL$MOD
000536  010001                            MOV     R0,R1
000540  005201                            INC     R1
000542  110137  000000G                   MOVB    R1,MINUTES
000546  123727  000000G 000074    10$:    CMPB    MINUTES,#74              ;   1973
000554  103412                            BLO     11$
000556  005000                            CLR     R0                       ;   1975
000560  153700  000000G                   BISB    MINUTES,R0
000564  162700  000074                    SUB     #74,R0
000570  110037  000000G                   MOVB    R0,MINUTES
000574  105237  000000G                   INCB    HOURS                    ;   1976
000600  000762                            BR      10$                      ;   1973
000602  005016                    11$:    CLR     (SP)                     ;   1979
000604  113716  000000G                   MOVB    HOURS,(SP)
000610  012746  000030                    MOV     #30,-(SP)
000614  004737  000000G                   JSR     PC,BL$MOD
000620  110037  000000G                   MOVB    R0,HOURS
000624  062706  000006                    ADD     #6,SP                    ;   1967
000630  062706  000010            12$:    ADD     #10,SP                   ;   1941
000634  132737  000001  000000G   13$:    BITB    #1,CLK.PRESENT           ;   1985
000642  001416                            BEQ     14$
000644  012746  000300                    MOV     #300,-(SP)               ;   1988
000650  012746  000000G                   MOV     #TIME,-(SP)
000654  012746  000100                    MOV     #100,-(SP)
000660  012746  000003                    MOV     #3,-(SP)
000664  104437                            TRAP    37
000666  012737  000100  177546            MOV     #100,@#177546            ;   1989
000674  062706  000010                    ADD     #10,SP                   ;   1987
000700  104421                    14$:    TRAP    21                       ;   1992
000702  010037  001172'                   MOV     R0,DRS.FLAGS
000706  042700  077777                    BIC     #77777,R0                ;   1994
000712  020027  100000                    CMP     R0,#-100000
000716  001003                            BNE     15$
000720  012700  000001                    MOV     #1,R0
000724  000401                            BR      16$
```

D6


```
000726  005000                         15$:    CLR     R0
000730  020027  100000                 16$:    CMP     R0,#-100000
000734  001003                                 BNE     17$
000736  112737  000001  000000G                MOVB    #1,MOE.FLAG                     ;                           1996
000744  004737  000000V                17$:    JSR     PC,INIT.TEST                    ;                           1999
000750  005002                                 CLR     R2                              ; CTLR                      2001
000752  010246                         18$:    MOV     R2,-(SP)                        ; CTLR,*                    2003
000754  012746  000126                         MOV     #126,-(SP)
000760  004737  000000G                        JSR     PC,BL$MUL
000764  022626                                 CMP     (SP)+,(SP)+
000766  005760  000002G                        TST     CST+2(R0)
000772  100040                                 BPL     22$
000774  010246                                 MOV     R2,-(SP)                        ; CTLR,*                    2004
000776  012746  000022                         MOV     #22,-(SP)
001002  004737  000000G                        JSR     PC,BL$MUL
001006  022626                                 CMP     (SP)+,(SP)+
001010  005760  000000G                        TST     DCT(R0)
001014  100027                                 BPL     22$
001016  010246                                 MOV     R2,-(SP)                        ; CTLR,*                    2009
001020  012746  000053                         MOV     #53,-(SP)
001024  004737  000000G                        JSR     PC,BL$MUL
001030  012701  000003                         MOV     #3,R1                           ; *,OFFSET                  2007
001034  010003                         19$:    MOV     R0,R3                           ;                           2009
001036  060103                                 ADD     R1,R3                           ; OFFSET,*
001040  006303                                 ASL     R3
001042  032763  020000  000000G                BIT     #20000,CST(R3)
001050  001403                                 BEQ     20$
001052  105037  000000G                        CLRB    EOP.FLAG                        ;                           2012
001056  000405                                 BR      21$                             ;                           2011
001060  062701  000012                 20$:    ADD     #12,R1                          ; *,OFFSET                  2007
001064  020127  000042                         CMP     R1,#42                          ; OFFSET,*
001070  003761                                 BLE     19$
001072  022626                         21$:    CMP     (SP)+,(SP)+
001074  005202                         22$:    INC     R2                              ; CTLR                      2001
001076  000243                                 .WORD CLV!CLC

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

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



                                               .SBTTL  T1 TEST SECTION
000000  004737  000000'                T1::
000000                                 1$:     JSR     PC,$T1                          ;                           2018
000004  104466                                 TRAP    66
000006  006000                                 ROR     R0
```

```
000010  103773                          BLO     1$
000012  000207                          RTS     PC

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

```
%sbttl 'INITIALIZATION TEST ROUTINES'

GLOBAL routine INIT_TEST : novalue =

!+
!       THE INITIALIZATION TEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
!       DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
!       BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI-DRIVE TEST
!       OR THE DM EXERCISER.
!
!       BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
!       INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
!       MARKED OFFLINE IN THEIR DCT AND CST. FOR THOSE DEVICES WHICH SURVIVE
!       THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
!       EACH DISK VIA ROUTINE "ACCESS". THE INITIALIZATION TEST IS DEEMED A
!       SUCCESS IF A BLOCK ON THE INNER TRACK OF EACH DISK CAN BE ACCESSED.
!-

    begin
    DRIVER_INIT ();                            ! INIT DRIVER DATA AND DEVICES

    incr CTLR from 0 to (MAX_CTLR - 1) do      ! FOR EACH CONTROLLER
        begin
        SET_CPAR (.CTLR);                      ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS

        if .CST_ADDR [STATE] eql ONLINE        ! IF CONTROLLER IS STILL ALIVE
        then                                   ! FOR EACH DISK

            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
                   (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
                then
                    begin
                    SET_UPAR (.OFFSET);        ! SET UP UNIT-RELATED DATA ITEMS
                    IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)        !ZZZ
                       THEN ACCESS ();                               !ZZZ
                                              !SKIP IF DUP CAUSED INIT ZZZ

                    end;                       ! IF UNIT IS PRESENT AND ONLINE

        end;                                   ! CONTROLLER LOOP
    end;                                       ! ROUTINE INIT_TEST

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

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

000132  003727                       BLE    1$
000134  000207                       RTS    PC                   ;                2022

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

```
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
```

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

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

```
000232  006300                           ASL     R0
000234  060200                           ADD     R2,R0
000236  112760  000040  000012           MOVB    #40,12(R0)
000244  112760  000040  000013           MOVB    #40,13(R0)                 ;                       2111
000252  010100                           MOV     R1,R0                      ; OFFSET,*              2112
000254  006300                           ASL     R0
000256  060200                           ADD     R2,R0
000260  112760  000040  000014           MOVB    #40,14(R0)
000266  112760  000040  000015           MOVB    #40,15(R0)                 ;                       2113
000274  010116                           MOV     R1,(SP)                    ; OFFSET,*              2114
000276  004737  000000G                  JSR     PC,SET.UPAR
000302  004737  000000V                  JSR     PC,UNIT.INIT               ;                       2115
000306  062701  000012           4$:     ADD     #12,R1                     ; *,OFFSET              2104
000312  020127  000041                   CMP     R1,#41                     ; OFFSET,*
000316  003731                           BLE     3$
000320  022626                   5$:     CMP     (SP)+,(SP)+                ;                       2095
000322  005203                   6$:     INC     R3                         ; CTLR                  2091
000324  000243                           .WORD CLV!CLC

000326  003661                           BLE     2$
000330  000207                           RTS     PC                         ;                       2065

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

K6


```
GLOBAL routine CTLR_INIT : novalue =

!+
!       THIS "DRIVER" ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONTROLLER
!       CONFIGURED FOR TESTING. ITS GENERAL PURPOSE IS TO BRING THE RDRX ONLINE
!       TO THE HOST. SPECIFICALLY, IT IS WRITTEN TO:
!
!       1. INITIALIZE DRIVER CONTROLLER DATA, INCLUDING THE DCT,
!       2. SET UP THE DEVICE'S INTERRUPT VECTOR ADDRESS,
!       3. PERFORM A REGISTER EXISTENCE TEST TO VERIFY THE DEVICE'S PRESENCE,
!       4. PERFORM A VECTOR AND BR LEVEL TEST TO VERIFY THE DEVICE'S VECTOR
!               ADDRESS AND INTERRUPT REQUEST LEVEL,
!       5. DO A HARD INITIALIZATION (FOUR STEPS) ON THE DEVICE.
!
!       IF ANY OF THESE INITIAL TESTS FAIL, THEN ALL UNITS ASSOCIATED WITH THE
!       DEVICE ARE DROPPED.
!-

    begin

    local
        RESULT : byte;

    INI_CTLR_DAT ();                                        ! INITIALIZE CONTROLLER DATA
!ZZZ    SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], PRIO4);    ! SET DEVICE'S ASSUMED VECTOR ADDRESS
    SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], .BRLEVEL); ! SET DEVICE'S ASSUMED VECTOR ADDRESS    ZZZ
    DCT_ADDR [IG_INT] = TRUE;                               ! SET "IGNORE INTERRUPT" BIT
    L$LUN = .CST_ADDR [OF_UN + OF_DATA, D_UNIT];            ! GET FIRST UNIT NUMBER OF CONTROLLER
                                                            ! (USED BY DRS FOR DEVICE-FATAL CTLR ERRORS)
IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
THEN                                    !CAUSED INIT, SKIP THIS CODE  ZZZ

    if REG_EXIST () eql FAILURE                             ! REGISTER EXISTENCE TEST
    then
        begin
        DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
        return;
        end;

IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
THEN                                    !CAUSED INIT, SKIP THIS CODE  ZZZ

    if VEC_BR_TEST () eql FAILURE                           ! VECTOR ADDR AND BR LEVEL TEST
    then
        begin
        DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
        return;
        end;

    RESULT = HARD_INIT ();                                  ! ATTEMPT HARD DEVICE INIT
    DCT_ADDR [IG_INT] = FALSE;                              ! CLAER "IGNORE INTERRUPT" BIT

    if .RESULT eql SUCCESS                                  ! IF HARD INIT WAS SUCCESSFUL
```

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

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

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

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

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

                                          .SBTTL  INI.CTLR.DAT INITIALIZATION TEST ROUTINES
000000  004137  000000G           INI.CTLR.DAT::
                                          JSR     R1,$SAVE2                       ;       2194
000004  013701  000000G                   MOV     DCT.ADDR,R1                     ;       2207
000010  005011                            CLR     (R1)
000012  012702  000004                    MOV     #4,R2                           ;       2208
000016  060102                            ADD     R1,R2
000020  013746  000000G                   MOV     CCTLR,-(SP)
000024  012746  000050                    MOV     #50,-(SP)
000030  004737  000000G                   JSR     PC,BL$MUL
000034  062700  000010'                   ADD     #COMM.AREA+10,R0
000040  010012                            MOV     R0,(R2)
000042  010061  000006                    MOV     R0,6(R1)                        ;       2209
000046  062761  000014  000006            ADD     #14,6(R1)
000054  012700  000010                    MOV     #10,R0                          ;       2210
000060  060100                            ADD     R1,R0
000062  016110  000006                    MOV     6(R1),(R0)
000066  062710  000004                    ADD     #4,(R0)
000072  011061  000012                    MOV     (R0),12(R1)                     ;       2211
000076  062761  000014  000012            ADD     #14,12(R1)
000104  011261  000014                    MOV     (R2),14(R1)                     ;       2212
000110  011061  000020                    MOV     (R0),20(R1)                     ;       2213
000114  011061  000016                    MOV     (R0),16(R1)
000120  022626                            CMP     (SP)+,(SP)+                     ;       2206
000122  000207                            RTS     PC                              ;       2194

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

```
GLOBAL routine REG_EXIST =
!+
!       THIS IS THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VERIFY
!       THE PRESENCE OF AN RDRX DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
!       SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
!       ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE'S SA AND IP
!       REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
!       THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
!       DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
!-
    begin

    local
        DUMMY_0 : word,                         ! TEMP FOR READING SA AND IP
        DUMMY_1 : word;                         !

    if .ENTRY_REASON eql NEW_PASS
    then
        return SUCCESS;                         ! SKIP TEST FOR NEXT PASS

    OF_RC = 2;                                  ! SET UP TO READ SA FIRST

    do
        begin
        NEX = FALSE;                            ! SET TO "TRAP NOT RECEIVED"
        SETVEC (4, NEX_TRAP, PRIO7);            ! SET LOCATION 4 TRAP VECTOR ADDRESS
        DUMMY_0 = .(.RDRX_ADDR + .OF_RC);       ! READ REGISTER (THEN TRAP OR CONTINUE)
        DUMMY_1 = 0;                            ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
                                                !       (TRAP RETURNS TO NEXT INSTRUCTION)
        CLRVEC (4);                             ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS

        if .NEX                                 ! IF NEX TRAP OCCURRED
        then
            begin
            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 (10, EGD_10, EMS_10);         ! REGISTER EXISTENCE TEST FAILED
            SETPRI (PRIO0);                     ! LOWER PRIORITY
            return FAILURE;
            end
        else
            OF_RC = .OF_RC - 2;                 ! SET UP FOR IP REG OR QUIT

        end
    until .OF_RC lss 0;
```

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


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

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

```
;  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)
```

```
;     2323  3             then
;     2324  4                 begin                          ! IF INTERRUPT DID NOT OCCUR
;     2325  4                 SETPRI (.BRLEVEL - %o'40');    ! LOWER PRIORITY BY 1
;     2326  4                 DELAY (1);                     ! WAIT
;     2327  4
;     2328  4                 if .DCT_ADDR [SA_SAVE] neq 0   ! IF INTERRUPT DID OCCUR (SA_SAVE WOULD BE NON-ZERO)
;     2329  4                 then
;     2330  5                     begin
;     2331  5                     SETPRI (PRIO0);            ! RESTORE PROCESSOR PRIORITY TO 0
;     2332  5                     return SUCCESS;            ! ONLY SUCCESSFUL EXIT POINT
;     2333  4                     end;
;     2334  4
;     2335  3                 end;
;     2336  3
;     2337  2             end;
;     2338  2
;     2339  2         SETPRI (PRIO0);                        ! COME HERE ONLY FOR BR TEST FAILURE
;     2340  2         C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;     2341  2
;     2342  2         if .APT_MODE
;     2343  2         then
;     2344  3             begin
;     2345  3             .MAIL_BOX_TESTNUM = 1;
;     2346  3             .MAIL_BOX_SUBTST = 0;
;     2347  2             end;
;     2348  2
;     2349  2         ERRDF (12, EGD_12, EMS_12);
;     2350  2         return FAILURE;
;     2351  1         end;


                                               .GLOBL  L$DLY


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

```
000060  032737  000001  001162'          BIT     #1,APT.MODE              ;                2307
000066  001405                           BEQ     2$
000070  012777  000001  001164'          MOV     #1,@MAIL.BOX.TESTNUM     ;                2310
000076  005077  001166'                  CLR     @MAIL.BOX.SUBTST         ;                2311
000102  104455                   2$:     TRAP    55                       ;                2314
000104  000013                           .WORD   13
000106  000000G                          .WORD   EGD.11
000110  000000                           .WORD   0
000112  000477                           BR      11$                      ;                2315
000114  012716  000000G          3$:     MOV     #MSG.03,(SP)             ;                2319
000120  012746  000001                   MOV     #1,-(SP)
000124  010600                           MOV     SP,R0                    ; SP,*
000126  104417                           TRAP    17
000130  013700  000000G                  MOV     BRLEVEL,R0               ;                2320
000134  104441                           TRAP    41
000136  004737  000000V                  JSR     PC,INT.GEN               ;                2322
000142  005700                           TST     R0
000144  001035                           BNE     9$
000146  013700  000000G                  MOV     BRLEVEL,R0               ;                2325
000152  162700  000040                   SUB     #40,R0
000156  104441                           TRAP    41
000160  012701  000001                   MOV     #1,R1                    ; *,$$TMP2       2326
000164  001411                   4$:     BEQ     7$
000166  013700  000000G                  MOV     L$DLY,R0                 ; *,$$TMP1
000172  001404                           BEQ     6$
000174  005066  000006           5$:     CLR     6(SP)                    ; $$TMP
000200  005300                           DEC     R0                       ; $$TMP1
000202  001374                           BNE     5$
000204  005301                   6$:     DEC     R1                       ; $$TMP2
000206  000766                           BR      4$
000210  013700  000000G          7$:     MOV     DCT.ADDR,R0              ;                2328
000214  005760  000002                   TST     2(R0)
000220  001407                           BEQ     9$
000222  005000                           CLR     R0                       ;                2331
000224  104441                           TRAP    41
000226  062706  000006                   ADD     #6,SP                    ;                2332
000232  012700  000001           8$:     MOV     #1,R0                    ;                2330
000236  000427                           BR      12$
000240  005726                   9$:     TST     (SP)+                    ;                2318
000242  005000                           CLR     R0                       ;                2339
000244  104441                           TRAP    41
000246  013700  000000G                  MOV     CCTLR,R0                 ;                2340
000252  006300                           ASL     R0
000254  105260  000000G                  INCB    C.ERR.TBL(R0)
000260  032737  000001  001162'          BIT     #1,APT.MODE              ;                2342
000266  001405                           BEQ     10$
000270  012777  000001  001164'          MOV     #1,@MAIL.BOX.TESTNUM     ;                2345
000276  005077  001166'                  CLR     @MAIL.BOX.SUBTST         ;                2346
000302  104455                   10$:    TRAP    55                       ;                2349
000304  000014                           .WORD   14
000306  000000G                          .WORD   EGD.12
000310  000000G                          .WORD   EMS.12
000312  022626                   11$:    CMP     (SP)+,(SP)+              ;                2350
```

```
000314  005000                  CLR     R0                              ;                       2270
000316  005726          12$:    TST     (SP)+
000320  012601                  MOV     (SP)+,R1
000322  000207                  RTS     PC

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

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

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

```
000016  012700  177777          MOV     #-1,R0                  ; *,RC.REG              2370
000022  010077  000000G         MOV     R0,@RDRX.ADDR           ; RC.REG,*
000026  012701  000002          MOV     #2,R1                   ; *,$$TMP2              2371
000032  001411          1$:     BEQ     4$
000034  013700  000000G         MOV     L$DLY,R0                ; *,$$TMP1
000040  001404                  BEQ     3$
000042  005066  000002  2$:     CLR     2(SP)                   ; $$TMP
000046  005300                  DEC     R0                      ; $$TMP1
000050  001374                  BNE     2$
000052  005301          3$:     DEC     R1                      ; $$TMP2
000054  000766                  BR      1$
000056  013702  000000G 4$:     MOV     RDRX.ADDR,R2            ;                       2374
000062  012703  000764          MOV     #764,R3                 ; *,COUNT               2372
000066  016216  000002  5$:     MOV     2(R2),(SP)              ; *,RC.REG              2374
000072  011604                  MOV     (SP),R4                 ; RC.REG,SA
000074  010400                  MOV     R4,R0                   ; SA,*                  2375
000076  042700  001777          BIC     #1777,R0
000102  020027  004000          CMP     R0,#4000
000106  001416                  BEQ     10$                     ;                       2377
000110  012701  000001          MOV     #1,R1                   ; *,$$TMP2              2378
000114  001411          6$:     BEQ     9$
000116  013700  000000G         MOV     L$DLY,R0                ; *,$$TMP1
000122  001404                  BEQ     8$
000124  005066  000002  7$:     CLR     2(SP)                   ; $$TMP
000130  005300                  DEC     R0                      ; $$TMP1
000132  001374                  BNE     7$
000134  005301          8$:     DEC     R1                      ; $$TMP2
000136  000766                  BR      6$
000140  005303          9$:     DEC     R3                      ; COUNT                 2372
000142  001351                  BNE     5$
000144  013700  001154' 10$:    MOV     CURRENT.VECTOR,R0       ;                       2381
000150  006200                  ASR     R0
000152  006200                  ASR     R0
000154  010004                  MOV     R0,R4                   ; *,SA
000156  052704  111200          BIS     #111200,R4              ; *,SA
000162  010401                  MOV     R4,R1                   ; SA,RC.REG             2382
000164  010162  000002          MOV     R1,2(R2)                ; RC.REG,*
000170  012702  017500          MOV     #17500,R2               ; *,COUNT               2384
000174  012701  000001  11$:    MOV     #1,R1                   ; *,$$TMP2              2386
000200  001411          12$:    BEQ     15$
000202  013700  000000G         MOV     L$DLY,R0                ; *,$$TMP1
000206  001404                  BEQ     14$
000210  005066  000002  13$:    CLR     2(SP)                   ; $$TMP
000214  005300                  DEC     R0                      ; $$TMP1
000216  001374                  BNE     13$
000220  005301          14$:    DEC     R1                      ; $$TMP2
000222  000766                  BR      12$
000224  013700  000000G 15$:    MOV     DCT.ADDR,R0             ;                       2388
000230  005760  000002          TST     2(R0)
000234  001403                  BEQ     16$
000236  012700  000001          MOV     #1,R0                   ;                       2390
000242  000404                  BR      17$
000244  104422          16$:    TRAP    22
```

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

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

```
GLOBAL routine HARD_INIT =

!+
!       THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
!       INITIALIZE AN RDRX DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
!       THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
!       OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
!       IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
!-

    begin

    local
        IE_VEC : word;                                  ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
                                                        !   (USED IN STEP 1 WRITE AND STEP 3 READ)

    IE_VEC = .CURRENT_VECTOR ^ -2;                      ! GET VECTOR ADDR/4 (IE = 0)

    incr ATTEMPTS from 1 to INI_ATT do
        begin

        label
            STEP_1_READ,
            STEP_2_READ,
            STEP_3_READ,
            STEP_4_READ;

        WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
!
!       STEP 1 READ
!
        STEP = 1;
        STEP_1_READ:
            begin

            incr COUNT from 1 to 500 do
                begin
                DELAY (1);                              ! TOTAL DELAY COUNT OF 500 FOR STEP 1
                SA_REG = .RDRX_ADDR [RCSA, RC_ALL];     ! READ SA

                if (.SA_REG and S1_MASK) eql SA_S1      ! IF STEP 1 READ IS O.K.
                then
                    leave STEP_1_READ;

                BREAK;
                end;

            exitloop;
            end;

!
!       STEP 1 WRITE
!
```

```
;   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                   ;                       2397
000004  162706  000012                  SUB     #12,SP
000010  013704  001154'                 MOV     CURRENT.VECTOR,R4           ; *,IE.VEC              2413
000014  006204                          ASR     R4                          ; IE.VEC
000016  006204                          ASR     R4                          ; IE.VEC
```

```
000020  012705  000002                    MOV     #2,R5                   ; *,ATTEMPTS          2415
000024  012700  177777                    MOV     #-1,R0                  ; *,RC.REG            2424
000030  010077  000000G                   MOV     R0,@RDRX.ADDR           ; RC.REG,*
000034  012737  000001  000000G           MOV     #1,STEP                 ;                     2428
000042  012702  000764                    MOV     #764,R2                 ; *,COUNT             2432
000046  012701  000001            1$:     MOV     #1,R1                   ; *,$$TMP2            2434
000052  001411                    2$:     BEQ     5$
000054  013700  000000G                   MOV     L$DLY,R0                ; *,$$TMP1
000060  001404                            BEQ     4$
000062  005066  000010            3$:     CLR     10(SP)                  ; $$TMP
000066  005300                            DEC     R0                      ; $$TMP1
000070  001374                            BNE     3$
000072  005301                    4$:     DEC     R1                      ; $$TMP2
000074  000766                            BR      2$
000076  013700  000000G           5$:     MOV     RDRX.ADDR,R0            ;                     2435
000102  016016  000002                    MOV     2(R0),(SP)              ; *,RC.REG
000106  011637  000000G                   MOV     (SP),SA.REG             ; RC.REG,*
000112  011600                            MOV     (SP),R0                 ; SA.REG,*            2437
000114  042700  001777                    BIC     #1777,R0
000120  020027  004000                    CMP     R0,#4000
000124  001404                            BEQ     6$                      ;                     2439
000126  104422                            TRAP    22
000130  005302                            DEC     R2                      ; COUNT               2432
000132  001345                            BNE     1$
000134  000532                            BR      18$                     ;                     2416
000136  010437  000000G           6$:     MOV     R4,SA.REG               ; IE.VEC,*            2450
000142  052737  111000  000000G           BIS     #111000,SA.REG
000150  013701  000000G                   MOV     SA.REG,R1               ; *,RC.REG            2451
000154  013700  000000G                   MOV     RDRX.ADDR,R0
000160  010160  000002                    MOV     R1,2(R0)                ; RC.REG,*
000164  005237  000000G                   INC     STEP                    ;                     2455
000170  012702  023420                    MOV     #23420,R2               ; *,COUNT             2459
000174  012701  000001            7$:     MOV     #1,R1                   ; *,$$TMP2            2461
000200  001411                    8$:     BEQ     11$
000202  013700  000000G                   MOV     L$DLY,R0                ; *,$$TMP1
000206  001404                            BEQ     10$
000210  005066  000010            9$:     CLR     10(SP)                  ; $$TMP
000214  005300                            DEC     R0                      ; $$TMP1
000216  001374                            BNE     9$
000220  005301                    10$:    DEC     R1                      ; $$TMP2
000222  000766                            BR      8$
000224  013700  000000G           11$:    MOV     RDRX.ADDR,R0            ;                     2462
000230  016066  000002  000002            MOV     2(R0),2(SP)             ; *,RC.REG
000236  016637  000002  000000G           MOV     2(SP),SA.REG            ; RC.REG,*
000244  016600  000002                    MOV     2(SP),R0                ; SA.REG,*            2464
000250  042700  003400                    BIC     #3400,R0
000254  020027  010222                    CMP     R0,#10222
000260  001404                            BEQ     12$                     ;                     2466
000262  104422                            TRAP    22
000264  005302                            DEC     R2                      ; COUNT               2459
000266  001342                            BNE     7$
000270  000537                            BR      26$                     ;                     2416
000272  013700  000000G           12$:    MOV     DCT.ADDR,R0             ;                     2477
```

```
000276  016001  000004                    MOV     4(R0),R1                 ; *,RC.REG
000302  013700  000000G                   MOV     RDRX.ADDR,R0
000306  010160  000002                    MOV     R1,2(R0)                 ; RC.REG,*
000312  005237  000000G                   INC     STEP                     ;                          2481
000316  010403                            MOV     R4,R3                    ; IE.VEC,*                 2490
000320  052703  020000                    BIS     #20000,R3
000324  012702  023420                    MOV     #23420,R2                ; *,COUNT                  2485
000330  012701  000001            13$:    MOV     #1,R1                    ; *,$$TMP2                 2487
000334  001411                    14$:    BEQ     17$
000336  013700  000000G                   MOV     L$DLY,R0                 ; *,$$TMP1
000342  001404                            BEQ     16$
000344  005066  000010            15$:    CLR     10(SP)                   ; $$TMP
000350  005300                            DEC     R0                       ; $$TMP1
000352  001374                            BNE     15$
000354  005301                    16$:    DEC     R1                       ; $$TMP2
000356  000766                            BR      14$
000360  013700  000000G           17$:    MOV     RDRX.ADDR,R0             ;                          2488
000364  016066  000002  000004            MOV     2(R0),4(SP)              ; *,RC.REG
000372  016637  000004  000000G           MOV     4(SP),SA.REG             ; RC.REG,*
000400  016600  000004                    MOV     4(SP),R0                 ; SA.REG,*                 2490
000404  042700  003400                    BIC     #3400,R0
000410  020003                            CMP     R0,R3
000412  001404                            BEQ     19$                      ;                          2492
000414  104422                            TRAP    22
000416  005302                            DEC     R2                       ; COUNT                    2485
000420  001343                            BNE     13$
000422  000462                    18$:    BR      26$                      ;                          2416
000424  013700  000000G           19$:    MOV     RDRX.ADDR,R0             ;                          2503
000430  005060  000002                    CLR     2(R0)
000434  005237  000000G                   INC     STEP                     ;                          2507
000440  012703  023420                    MOV     #23420,R3                ; *,COUNT                  2511
000444  012701  000001            20$:    MOV     #1,R1                    ; *,$$TMP2                 2513
000450  001411                    21$:    BEQ     24$
000452  013700  000000G                   MOV     L$DLY,R0                 ; *,$$TMP1
000456  001404                            BEQ     23$
000460  005066  000010            22$:    CLR     10(SP)                   ; $$TMP
000464  005300                            DEC     R0                       ; $$TMP1
000466  001374                            BNE     22$
000470  005301                    23$:    DEC     R1                       ; $$TMP2
000472  000766                            BR      21$
000474  013700  000000G           24$:    MOV     RDRX.ADDR,R0             ;                          2514
000500  016066  000002  000006            MOV     2(R0),6(SP)              ; *,RC.REG
000506  016637  000006  000000G           MOV     6(SP),SA.REG             ; RC.REG,*
000514  016600  000006                    MOV     6(SP),R0                 ; SA.REG,*                 2516
000520  042700  003777                    BIC     #3777,R0
000524  020027  040000                    CMP     R0,#40000
000530  001404                            BEQ     25$                      ;                          2518
000532  104422                            TRAP    22
000534  005303                            DEC     R3                       ; COUNT                    2511
000536  001342                            BNE     20$
000540  000413                            BR      26$                      ;                          2416
000542  012737  000001  000000G   25$:    MOV     #1,CREDIT.BAL            ;                          2528
000550  005001                            CLR     R1                       ; RC.REG                   2529
```

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

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

```
;    2547  1    GLOBAL routine INI_RRING : novalue =
;    2548  1
;    2549  1    !+
;    2550  1    !       THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP PACKETS TO
;    2551  1    !       FILL AN RDRX RESPONSE RING. THE BUFFER DESCRIPTOR OF EACH PACKET
;    2552  1    !       (LOCATED IN FRONT OF THE PACKET ITSELF) IS LOADED INTO SUCCESSIVE
;    2553  1    !       RRING SLOTS. NOTE THAT THE BUFFER DESCRIPTORS HAVE BEEN INITIALIZED
;    2554  1    !       WITH THE FLAG AND OWNERSHIP BITS SET TO "1", MAKING EACH SLOT
;    2555  1    !       CONTROLLER-OWNED.
;    2556  1    !
;    2557  1    !       IMPLICIT INPUTS:
;    2558  1    !               CCTLR - CURRENT CONTROLLER NUMBER
;    2559  1    !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;    2560  1    !-
;    2561  1
;    2562  2        begin
;    2563  2
;    2564  2        local
;    2565  2            index : word,
;    2566  2            RRING_ADDR;
;    2567  2
;    2568  2        RRING_ADDR = .DCT_ADDR [RR_BEG];                      ! FIRST RESPONSE RING SLOT
;    2569  2
;    2570  2        incr COUNT from 1 to RRING_LEN do
;    2571  3            begin
;    2572  3            index = GET_PKT (.CCTLR);                         ! GET AN MSCP PACKET
;    2573  3            .RRING_ADDR = .MSCP_PKT [.index, PKT_LO];         ! LOAD LO-ORDER BUFF DESC INTO SLOT
;    2574  3            RRING_ADDR = .RRING_ADDR + 2;                     ! ADVANCE TO SECOND WORD
;    2575  3            .RRING_ADDR = .MSCP_PKT [.index, PKT_HI];         ! LOAD HI-ORDER BUFF DESC INTO SLOT
;    2576  3            PKT_USE [.index] = .CCTLR;                        ! PACKET IN USE
;    2577  3            .RRING_ADDR = ..RRING_ADDR or ED_OWN or ED_FLAG;! GIVE OWNERSHIP TO CONTRLLER
;    2578  3            RRING_ADDR = .RRING_ADDR + 2;                     ! ADVANCE TO NEXT SLOT
;    2579  2            end;
;    2580  2
;    2581  1        end;


                                          .SBTTL  INI.RRING INITIALIZATION TEST ROUTINES
000000  004137  000000G          INI.RRING::
                                          JSR     R1,$SAVE4                   ;                          2547
000004  013700  000000G                   MOV     DCT.ADDR,R0                 ;                          2568
000010  016001  000004                    MOV     4(R0),R1                    ; *,RRING.ADDR
000014  013703  000000G                   MOV     CCTLR,R3                    ;                          2572
000020  012704  000004                    MOV     #4,R4                       ; *,COUNT                  2570
000024  010346                   1$:      MOV     R3,-(SP)                    ;                          2572
000026  004737  000000G                   JSR     PC,GET.PKT
000032  010002                            MOV     R0,R2                       ; *,INDEX
000034  010216                            MOV     R2,(SP)                     ; INDEX,*                  2573
000036  012746  000106                    MOV     #106,-(SP)
000042  004737  000000G                   JSR     PC,BL$MUL
000046  016021  000000G                   MOV     MSCP.PKT(R0),(R1)+          ; *,RRING.ADDR
000052  016011  000002G                   MOV     MSCP.PKT+2(R0),(R1)         ; *,RRING.ADDR             2575
000056  013703  000000G                   MOV     CCTLR,R3                    ;                          2576
```

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

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

```
;  1    GLOBAL routine SET_CTLR_CHAR =
;  1
;  1    !+
;  1    !       THIS ROUTINE IS CALLED BY CTLR_INIT AFTER THE RDRX HAS BEEN HARD-
;  1    !       INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
;  1    !       CHARACTERISTICS" COMMAND, AND TO VALIDATE THE RESPONSE (END MESSAGE).
;  1    !
;  1    !       IMPLICIT INPUTS:
;  1    !               CCTLR - CURRENT CONTROLLER NUMBER
;  1    !-
;  1
;  2        begin
;  2
;  2        local
;  2            P_INDEX : word;
;  2
;  2
;  2    ! MISCELLANEOUS INITIALIZATION                                          !ZZZ
;  2                                                                            !ZZZ
;  2    QIO [.CCTLR] = 0;               !INIT NO OF OUTSTANDING QIOS            !ZZZ
;  2    CST [.CCTLR, U_CNT] = 0;        !CLEAR UNITS IN CST TABLE               !ZZZ
;  2    INCR COUNT FROM 0 TO (RP_CNT - 1) DO    !INIT RETURN PACKET POOL        !ZZZ
;  2         RP_USE [.COUNT] = -1;                                              !ZZZ
;  2                                                                            !ZZZ
;  2    IODQ_IN = IODQ_OUT = 0;         !INIT I/O DONE QUEUE POINTERS           !ZZZ
;  2
;  2
;  2        P_INDEX = GET_PKT (.CCTLR);                             ! GET AN MSCP PACKET
;  2        MSCP_PKT [.P_INDEX, MSGLEN] = SZ_SCC;                   ! PACKET SIZE
;  2        MSCP_PKT [.P_INDEX, OPCODE] = OP_SCC;                   ! OPCODE = SET CTLR CHAR
;  2        MSCP_PKT [.P_INDEX, C_FLAGS] = CF_MASK;                 ! CONTROLLER FLAGS
;  2        MSCP_PKT [.P_INDEX, CMD_TYPE] = IMM_CMD;                ! IMMEDIATE COMMAND
;  2
;  2        if SEND (.P_INDEX) eql FAILURE                          ! ATTEMPT SEND
;  2        then
;  3            begin                                               ! IF SEND WAS UNSUCCESSFUL
;  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 (20, EGD_20, 0);                              ! FATAL ERROR
;  3            PUT_PKT (.P_INDEX);                                 ! RETURN PACKET TO POOL
;  3            DROP_CTLR (.CCTLR, DU_CFATAL);                      ! DROP CONTROLLER
;  3            return FAILURE;
;  3            end
;  2        else
;  3            begin                                               ! IF SEND WAS SUCCESSFUL
;  3
```

```
;  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_ADCR [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;
;  6                      .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
```

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


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

```
000214  013716  000000G                     MOV     CCTLR,(SP)              ;               2629
000220  012746  000006                      MOV     #6,-(SP)
000224  004737  000000G                     JSR     PC,DROP.CTLR
000230  005726                              TST     (SP)+                   ;               2617
000232  005000                              CLR     R0                      ;               2633
000234  000554                              BR      12$
000236  004737  000000G             3$:     JSR     PC,WAIT                 ;               2637
000242  004737  000000G                     JSR     PC,OUT.IODQ             ;               2638
000246  010037  000000G                     MOV     R0,RP.INDX
000252  010016                              MOV     R0,(SP)                 ; RP.INDX,*     2639
000254  012746  000054                      MOV     #54,-(SP)
000260  004737  000000G                     JSR     PC,BL$MUL
000264  062700  000000G                     ADD     #RETPKT,R0
000270  010037  000000G                     MOV     R0,RP.ADDR
000274  132760  000360  000002              BITB    #360,2(R0)              ;               2641
000302  001404                              BEQ     4$
000304  013716  000000G                     MOV     RP.INDX,(SP)            ;               2643
000310  004737  000000G                     JSR     PC,PUT.RETPKT
000314  005726                      4$:     TST     (SP)+                   ;               2636
000316  013701  000000G                     MOV     RP.ADDR,R1              ;               2646
000322  005000                              CLR     R0
000324  126127  000003  000003              CMPB    3(R1),#3
000332  001002                              BNE     5$
000334  005200                              INC     R0
000336  000407                              BR      6$
000340  132761  000360  000002      5$:     BITB    #360,2(R1)              ;               2647
000346  001333                              BNE     3$
000350  105761  000014                      TSTB    14(R1)                  ;               2648
000354  100330                              BPL     3$
000356  006000                      6$:     ROR     R0                      ;               2650
000360  103015                              BCC     7$
000362  012716  000000G                     MOV     #DBM23,(SP)             ;               2653
000366  012746  000001                      MOV     #1,-(SP)
000372  010600                              MOV     SP,R0                   ; SP,*
000374  104417                              TRAP    17
000376  013716  000000G                     MOV     RP.INDX,(SP)            ;               2654
000402  004737  000000G                     JSR     PC,PUT.RETPKT
000406  004737  000000V                     JSR     PC,DR.ERR               ;               2655
000412  000447                              BR      10$                     ;               2656
000414  126127  000014  000204      7$:     CMPB    14(R1),#204             ;               2661
000422  001007                              BNE     8$
000424  016100  000022                      MOV     22(R1),R0               ;               2662
000430  042700  177657                      BIC     #177657,R0
000434  020027  000120                      CMP     R0,#120
000440  001437                              BEQ     11$
000442  013700  000000G             8$:     MOV     CCTLR,R0                ;               2665
000446  006300                              ASL     R0
000450  105260  000000G                     INCB    C.ERR.TBL(R0)
000454  032737  000001  001162'             BIT     #1,APT.MODE             ;               2667
000462  001405                              BEQ     9$
000464  012777  000001  001164'             MOV     #1,@MAIL.BOX.TESTNUM    ;               2670
000472  005077  001166'                     CLR     @MAIL.BOX.SUBTST        ;               2671
000476  104455                      9$:     TRAP    55                      ;               2674
```

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

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

```
;   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$LUN - CURRENT (DRS) UNIT NUMBER
;   1      !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;   1      !-
;   1
;   2          begin
;   2          local
;   2              MAX0_LBNS : WORD UNSIGNED,                ! UNIT'S MAXIMUM LO WORD LBN
;   2              MAX1_LBNS : WORD UNSIGNED;                ! UNIT'S MAXIMUM HI WORD LBN
;   2
;   2          P_INDEX = GET_PKT (.CCTLR);                   ! GET AN MSCP PACKET
;   2          MSCP_PKT [.P_INDEX, MSGLEN] = SZ_ONL;         ! PACKET SIZE
;   2          MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;         ! SET DISK ADDRESS (RD/RX DISK NUMBER)
;   2          MSCP_PKT [.P_INDEX, OPCODE] = OP_ONL;         ! OPCODE FOR "ONLINE"
;   2      !ZZZ MSCP_PKT [.P_INDEX, DDPAR] = BIT00;          ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
;   2          MSCP_PKT [.P_INDEX, CMD_TYPE] = SEQ_CMD;      ! SEQUENTIAL COMMAND
;   2
;   2          if SEND (.P_INDEX) eql FAILURE                ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
;   2          then
;   3              begin
;   3              T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;   3
;   3              if .APT_MODE                              !ZZZ
;   3              then
;   4                  begin
;   4                  .MAIL_BOX_TESTNUM = 1;
;   4                  .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;   3                  end;
;   3
;   3              CST_ADDR [.CUOFF, D_FATAL] = TRUE;        ! FATAL ERROR
;   3              ERRDF (22, EGD_22, 0);                    !
;   3              DUR [.L$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
;   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 (DBM26);                        ! "ERROR IN UNIT_INIT"
;  4              DR_ERR ();                             ! DROP CONTROLLER
;  4              end
;  3          else
;  3
;  4              if .RP_ADDR [ENDCOD] neq (OP_ONL or OP_END) ! IF RETPKT IS FROM DISK MSCP
;  3              then
;  4                  begin
;  4                  T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4
;  4                  if .APT_MODE                       !ZZZ
;  4                  then
;  5                      begin
;  5                      .MAIL_BOX_TESTNUM = 1;
;  5                      .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  4                      end;
;  4
;  4                  CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;  4                  ERRDF (23, EGD_23, EMS_21);        ! FATAL ERROR
;  4                  DUR [.L$LUN] = DU_ONLINE;          ! SETUP REASON TO DROP UNIT
;  4                  DODU (.L$LUN);                     ! DROP UNIT
;  4                  end
;  3              else
;  4                  begin                              ! RETPKT HAS GOOD ENDCODE
;  4                  ST_CODE = .RP_ADDR [STSCOD];       ! GET STATUS CODE
;  4                  SB_CODE = .RP_ADDR [SUBCOD];       ! GET SUB-CODE
;  4
;  4                  CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_0] = .RP_ADDR [NAME_0] + %o'100';                 ! UNIT NAME
ZZZ
;  4                  CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .RP_ADDR [NAME_1_HI] * 16;                   !ZZZ
;  4                  CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] +   !ZZZ
;  4                                                            .RP_ADDR [NAME_1_LO] + %o'100';              !ZZZ
;  4                  CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_2] = .RP_ADDR [NAME_NUM] / 10 + %o'60';           !ZZZ
;  4                  CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_3] = (.RP_ADDR [NAME_NUM] mod 10) + %o'60';       !ZZZ
;  4
;  4
;  4
;  4                  IF .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] EQL %O'104'         !IF NAME IS _D         !ZZZ
;  4                  THEN                                                                                   !ZZZ
;  4                  CST_ADDR [.CUOFF, D_TYPE] = FIXED                               !ITS FIXED.            !ZZZ
;  4                  ELSE                                                                                   !ZZZ
```

```
                CST_ADDR [.CUOFF, D_TYPE] = REMOVABLE;                              !OTHERWISE REMOVABLE    !ZZZ



                if .ST_CODE neq ST_SUC          ! IF STATUS CODE IS NOT SUCCESSFUL
                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 (15, EGD_15, EMS_30); ! ONLINE FAILED
                    DUR [.L$LUN] = DU_ONLINE;   ! SET UP REASON FOR DROPPING UNIT
                    DODU (.L$LUN);              ! DROP UNIT
                    end
                else
                    begin                       ! SUCCESFUL OPERATION

                    MAX0_LBNS = .RP_ADDR [SIZE0];                                   ! LOAD LOWER WORD OF UNIT SIZE
                    MAX1_LBNS = .RP_ADDR [SIZE1];                                   ! LOAD UPPER WORD OF UNIT SIZE

                    if (.MAX0_LBNS eql 0)                                           ! THIS SUBTRACTS ONE FROM THE TOTAL
                    then                                                            ! BECAUSE EVERYTHING STARTS AT 0
                        begin                                                       ! THROUGH (MAXIMUM - 1)
                        MAX0_LBNS = %O'177777';
                        MAX1_LBNS = .MAX1_LBNS - 1;
                        end
                    else
                        MAX0_LBNS = .MAX0_LBNS - 1;

                    if  (.CST_ADDR [.CUOFF + 2, D_BEG1] gtru .MAX1_LBNS) or             ! THIS SECTION CHECKS TO SEE IF LBNs LISTED

                        ((.CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .MAX1_LBNS) and           ! IN SOFTWARE QUESTIONS WERE TO LARGE FOR

                        (.CST_ADDR [.CUOFF + 1, D_BEG0] gtru (.MAX0_LBNS - 1)))         ! DEVICE SPECIFIED
                                                                                    ! note 1 less then max. or diagnostic will error
                    then                                                            ! operator error
                        begin
                        CST_ADDR [.CUOFF + 2, D_BEG1] = 0;
                        CST_ADDR [.CUOFF + 1, D_BEG0] = 0;                          ! change beginning lbn to 0
                        end;

                     if
                        (.CST_ADDR [.CUOFF + 4, D_END1] gtru .MAX1_LBNS) or

                        ((.CST_ADDR [.CUOFF + 4, D_END1] eqlu .MAX1_LBNS) and
                        (.CST_ADDR [.CUOFF + 3, D_END0] gtru .MAX0_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 (((.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;



                                                                                     ! THIS SECTION LOADS TYPE INTO  CST TABLE
              !ZZZ                            selectoneu .RP_ADDR [R_MODEL] of              ! 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)
                              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 (16, EGD_16, EMS_30);                                ! WRITE-PROTECT CONFLICT
                                  DUR [.L$LUN] = DU_PROTECT;                                 ! SET REASON TO DROP UNIT
```

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


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

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

```
000504  052776  010000  000004    7$:   BIS     #10000,@4(SP)          ;       2779
000512  104455                          TRAP    55                     ;       2780
000514  000027                          .WORD   27
000516  000000G                         .WORD   EGD.23
000520  000000G                         .WORD   EMS.21
000522  013700  000000G                 MOV     L$LUN,R0               ;       2781
000526  112760  000007  000000G         MOVB    #7,DUR(R0)
000534  104451                          TRAP    51                     ;       2782
000536  000137  006676'           8$:   JMP     25$                    ;       2767
000542  116237  000016  000000G   9$:   MOVB    16(R2),ST.CODE         ;       2786
000550  042737  177740  000000G         BIC     #177740,ST.CODE
000556  016200  000016                  MOV     16(R2),R0              ;       2787
000562  006200                          ASR     R0
000564  006200                          ASR     R0
000566  006200                          ASR     R0
000570  006200                          ASR     R0
000572  006200                          ASR     R0
000574  042700  174000                  BIC     #174000,R0
000600  010037  000000G                 MOV     R0,SB.CODE
000604  013701  000000G                 MOV     CUOFF,R1               ;       2789
000610  006301                          ASL     R1
000612  060501                          ADD     R5,R1
000614  012703  000012                  MOV     #12,R3
000620  060103                          ADD     R1,R3
000622  116200  000036                  MOVB    36(R2),R0
000626  006200                          ASR     R0
000630  042700  177740                  BIC     #177740,R0
000634  062700  000100                  ADD     #100,R0
000640  110013                          MOVB    R0,(R3)
000642  116200  000036                  MOVB    36(R2),R0              ;       2790
000646  042700  177776                  BIC     #177776,R0
000652  006300                          ASL     R0
000654  006300                          ASL     R0
000656  006300                          ASL     R0
000660  006300                          ASL     R0
000662  110063  000001                  MOVB    R0,1(R3)
000666  005000                          CLR     R0                     ;       2791
000670  156300  000001                  BISB    1(R3),R0
000674  016201  000034                  MOV     34(R2),R1
000700  006201                          ASR     R1
000702  006201                          ASR     R1
000704  006201                          ASR     R1
000706  006201                          ASR     R1
000710  000301                          SWAB    R1
000712  042701  177760                  BIC     #177760,R1
000716  060100                          ADD     R1,R0
000720  010001                          MOV     R0,R1                  ;       2792
000722  062701  000100                  ADD     #100,R1
000726  110163  000001                  MOVB    R1,1(R3)
000732  013701  000000G                 MOV     CUOFF,R1               ;       2793
000736  006301                          ASL     R1
000740  060501                          ADD     R5,R1
000742  116216  000034                  MOVB    34(R2),(SP)
```

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

F9


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

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

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

```
;  1     GLOBAL routine DR_ERR : novalue =
;  1
;  1     !+
;  1     !       THIS ROUTINE IS DESIGNED TO PROCESS RETURN PACKETS THAT ORIGINATE AT
;  1     !       THE "DRIVER" RATHER THAN THE DEVICE. DRIVER-ORIGINATED PACKETS INDICATE
;  1     !       EITHER A FATAL DEVICE ERROR OR A COMMAND TIMEOUT. SINCE THIS ROUTINE IS
;  1     !       ONLY CALLED DURING THE INITIALIZATION TEST, IT TREATS A COMMAND TIMEOUT
;  1     !       AS AN INITIALIZATION ERROR.
;  1     !
;  1     !       IMPLICIT INPUTS:
;  1     !               RP_ADDR - ADDRESS OF A RETPKT THAT ORIGINATED AT THE "DRIVER"
;  1     !                       (I.E., CONNECTION ID = CID_DRIVER)
;  1     !-
;  1
;  2         begin
;  2
;  2         local
;  2             REASON : word initial (DU_TIME);        ! ASSUME COMMAND TIMEOUT
;  2
;  2         if .RP_ADDR [MESTYP] eql MT_FATAL           ! IF FATAL DEVICE ERROR
;  2         then
;  2
;  2         DROP_CTLR (.CCTLR, .REASON);                ! DROP ALL UNITS ON CONTROLLER
;  1         end;


                                        .SBTTL  DR.ERR INITIALIZATION TEST ROUTINES
000000  010146                  DR.ERR::MOV     R1,-(SP)                        ;                       2922
000002  012701  000012                  MOV     #12,R1                          ; *,REASON              2936
000006  013700  000000G                 MOV     RP.ADDR,R0                      ;                       2941
000012  116000  000002                  MOVB    2(R0),R0
000016  042700  177417                  BIC     #177417,R0
000022  020027  000060                  CMP     R0,#60
000026  001006                          BNE     1$
000030  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       2944
000034  010146                          MOV     R1,-(SP)                        ; REASON,*
000036  004737  000000G                 JSR     PC,DROP.CTLR
000042  022626                          CMP     (SP)+,(SP)+
000044  012601                  1$:     MOV     (SP)+,R1                        ;                       2922
000046  000207                          RTS     PC

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

```
;  1  routine ACCESS : novalue =
;  1
;  1  !+
;  1  !       THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT DISK
;  1  !       CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
;  1  !       SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE DISK, AND CHECKING
;  1  !       THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
;  1  !
;  1  !       IMPLICIT INPUTS:
;  1  !               CCTLR - CURRENT CONTROLLER NUMBER
;  1  !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  1  !               L$LUN - CURRENT (DRS) UNIT NUMBER
;  1  !-
;  1
;  2      begin
;  2
;  2  local
;  2          RESULT : word initial (FAILURE),        ! GUILTY UNTIL PROVEN INNOCENT
;  2          LBN : word,
;  2          PASS : word initial (1);                ! LOOP PASS COUNT
;  2
;  2      ST_CODE = SB_CODE = 0;                      ! STATUS CODE AND SUB-CODE
;  2      LBN = (((.MAX_LBN [.L$LUN] + 1) ↑ -1) and %o'77777') - 1;
;  2                                                  ! START WITH LAST LBN ON TOP SURFACE: [(X+1)/2] -1
;  2
;  2      do
;  3          begin                                   ! LOOP STARTS HERE
;  3          P_INDEX = GET_PKT (.CCTLR);             ! GET AN MSCP PACKET
;  3          MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;   ! SET DISK ADDR (RD/RX DISK NUMBER)
;  3          MSCP_PKT [.P_INDEX, OPCODE] = OP_ACC;   ! ACCESS OPCODE
;  3          MSCP_PKT [.P_INDEX, BC_LO] = 512;       ! BYTE COUNT (1 BLOCK)
;  3          MSCP_PKT [.P_INDEX, LBN_L] = .LBN;      ! LOGICAL BLOCK NUMBER
;  3          MSCP_PKT [.P_INDEX, CMD_TYPE] = NON_SEQ_CMD;    ! NON-SEQUENTIAL COMMAND
;  3
;  3          if SEND (.P_INDEX) eql FAILURE          ! ATTEMPT TO SEND; IF CTLR NOT ONLINE
;  3          then
;  4              begin
;  4              PUT_PKT (.P_INDEX);                 ! RETURN PACKET TO POOL
;  4              PASS = 2;                           ! NO MORE TRIES
;  4              end
;  3          else
;  4              begin                               ! IF SEND WAS SUCCESSFUL
;  4
;  4              do
;  5                  begin
;  5                  WAIT ();                        ! WAIT FOR RESPONSE
;  5                  RP_INDX = OUT_IODQ ();          ! GET RETPKT (RESPONSE) INDEX
;  5                  RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
;  5
;  5                  if .RP_ADDR [MESTYP] neq MT_SEQ ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;  5                  then
;  5                      PUT_RETPKT (.RP_INDX);
;  5
```

```
;  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 + 1;                           ! ADVANCE TO FIRST LBN OF BOTTOM SURFACE
;  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                    ;                    2946
```

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

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

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

```
;   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 PROCESS. AFTER
;   1    !       INVOKING MD_INIT TO INITIALIZE MULTI-DRIVE TEST DATA, THIS ROUTINE
;   1    !       ENTERS A 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                                  RW_BALANCE = .RW_BALANCE + 1;         !INCR RD/WR RATIO COUNT
;  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                                  RW_BALANCE = 0;                       !INDICATE RD/WT PAIR WAS LAST ISSUED  ZZZ
;  7                              END                                       !ZZZ
;  7
;  6                              else
;  7                                  begin
;  7                                  PRINTF (DBM121, .CRN_HIGH, .CRN_LOW);
;  7                                  COMPARE_DATA = FALSE;                 ! NO SENSE IN COMPARING WRITE DATA
;  7                                  PUT_PKT (.MX2);                       ! RETURN PACKET TO POOL
;  6                                  end;
;  6
```

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


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

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

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

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

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


;     3173  1
```

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


                                                  .SBTTL  MD.INIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                  MD.INIT::
                                                  JSR     R1,$SAVE2                  ;                    3174
000004  132737  000001  000000G                   BITB    #1,INIT.OCCURED            ;                    3187
000012  001002                                    BNE     1$
000014  004737  000000V                           JSR     PC,INIT.IO.BUFF            ;                    3189
000020  123727  000000G 000003           1$:      CMPB    ENTRY.REASON,#3            ;                    3191
000026  001433                                    BEQ     4$
000030  123727  000000G 000005                    CMPB    ENTRY.REASON,#5            ;                    3192
000036  001427                                    BEQ     4$
000040  005002                                    CLR     R2                         ; CTLR               3195
000042  010246                           2$:      MOV     R2,-(SP)                   ; CTLR,*             3197
000044  004737  000000G                           JSR     PC,SET.CPAR
000050  012701  000003                            MOV     #3,R1                      ; *,DISK             3199
000054  010116                           3$:      MOV     R1,(SP)                    ; DISK,*             3202
```

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

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

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

;    3210  1
```

```
;  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      !-
;  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                         ;              3211
000004  013700  000000G                 MOV     FREE.MEM.ADDR,R0                  ;              3232
```

```
000010  062700  000003                      ADD     #3,R0
000014  010037  000000G                     MOV     R0,BUFF.ADDR
000020  042737  000001  000000G             BIC     #1,BUFF.ADDR
000026  032737  000037  000000G     1$:     BIT     #37,BUFF.ADDR               ;               3234
000034  001404                              BEQ     2$
000036  062737  000002  000000G             ADD     #2,BUFF.ADDR                ;               3235
000044  000770                              BR      1$                          ;               3234
000046  013746  001160'             2$:     MOV     DRS.START,-(SP)             ;               3237
000052  163716  000000G                     SUB     BUFF.ADDR,(SP)
000056  012746  000010                      MOV     #10,-(SP)
000062  004737  000000G                     JSR     PC,BL$DIV
000066  010037  000000G                     MOV     R0,BYTS.PER.QIO
000072  042737  000037  000000G             BIC     #37,BYTS.PER.QIO
000100  023727  000000G 001400              CMP     BYTS.PER.QIO,#1400          ;               3240
000106  101403                              BLOS    3$
000110  012737  001400  000000G             MOV     #1400,BYTS.PER.QIO          ;               3242
000116  023727  000000G 000040      3$:     CMP     BYTS.PER.QIO,#40            ;               3244
000124  103005                              BHIS    4$
000126  104454                              TRAP    54                          ;               3247
000130  000002                              .WORD   2
000132  000000G                             .WORD   EGS.02
000134  000000                              .WORD   0
000136  104444                              TRAP    44
000140  012702  000001              4$:     MOV     #1,R2                       ; *,INDEX       3251
000144  010200                      5$:     MOV     R2,R0                       ; INDEX,*       3255
000146  006300                              ASL     R0
000150  010201                              MOV     R2,R1                       ; INDEX,*
000152  006301                              ASL     R1
000154  016103  177776G                     MOV     BUFF.ADDR-2(R1),R3
000160  063703  000000G                     ADD     BYTS.PER.QIO,R3
000164  010360  000000G                     MOV     R3,BUFF.ADDR(R0)
000170  005202                              INC     R2                          ; INDEX         3251
000172  020227  000007                      CMP     R2,#7                       ; INDEX,*
000176  003762                              BLE     5$
000200  022626                              CMP     (SP)+,(SP)+                 ;               3231
000202  000207                              RTS     PC                          ;               3211

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

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


                                          .SBTTL  QIO.OK MULTI-DRIVE TEST ROUTINES
000000  013700  000000G          QIO.OK::MOV     CST.ADDR,R0                     ;          3280
000004  005760  000002                   TST     2(R0)
000010  100027                           BPL     1$
000012  132737  000001  000000G          BITB    #1,EOP.FLAG                     ;          3281
000020  001023                           BNE     1$
000022  013700  000000G                  MOV     CCTLR,R0                        ;          3282
000026  116000  000000G                  MOVB    QIO(R0),R0
000032  042700  177400                   BIC     #177400,R0
000036  062700  000002                   ADD     #2,R0
000042  020027  000010                   CMP     R0,#10
000046  101010                           BHI     1$
000050  013700  000000G                  MOV     CST.ADDR,R0                     ;          3283
000054  105760  000005                   TSTB    5(R0)
000060  001403                           BEQ     1$
000062  012700  000001                   MOV     #1,R0                           ;          3289
000066  000207                           RTS     PC
000070  005000                   1$:     CLR     R0
000072  000207                           RTS     PC                              ;          3258
```

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


;   3290  1
```

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


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

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

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

```
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                ;           3345
000006  013746  000000G                           MOV     CCTLR,-(SP)            ;           3347
000012  004737  000000G                           JSR     PC,GET.PKT
000016  010037  000110'                           MOV     R0,MX1
000022  005726                                    TST     (SP)+
000024  005700                                    TST     R0                     ; MX1
000026  002563                                    BLT     6$                     ;           3349
000030  013746  000000G                           MOV     CCTLR,-(SP)            ;           3351
000034  004737  000000G                           JSR     PC,GET.PKT
```

```
000040  010037  000112'                    MOV     R0,MX2
000044  005726                             TST     (SP)+
000046  005700                             TST     R0                      ; MX2
000050  002011                             BGE     1$
000052  013746  000110'                    MOV     MX1,-(SP)               ;                       3354
000056  004737  000000G                    JSR     PC,PUT.PKT
000062  012737  177777  000110'            MOV     #-1,MX1                 ;                       3355
000070  005726                             TST     (SP)+                   ;                       3356
000072  000207                             RTS     PC                      ;                       3353
000074  013746  000110'            1$:     MOV     MX1,-(SP)               ;                       3359
000100  012746  000106                     MOV     #106,-(SP)
000104  004737  000000G                    JSR     PC,BL$MUL
000110  062700  000000G                    ADD     #MSCP.PKT,R0
000114  010037  000114'                    MOV     R0,MAD1
000120  013716  000112'                    MOV     MX2,(SP)                ;                       3360
000124  012746  000106                     MOV     #106,-(SP)
000130  004737  000000G                    JSR     PC,BL$MUL
000134  062700  000000G                    ADD     #MSCP.PKT,R0
000140  010037  000116'                    MOV     R0,MAD2
000144  004737  000000V                    JSR     PC,GET.RANDOM           ;                       3361
000150  004737  000000V                    JSR     PC,QIO.UNIT             ;                       3362
000154  132737  000001  000000G            BITB    #1,EOP.FLAG             ;                       3364
000162  001103                             BNE     5$                      ;                       3315
000164  004737  000000V                    JSR     PC,QIO.FUNC             ;                       3368
000170  004737  000000V                    JSR     PC,QIO.LBN              ;                       3369
000174  004737  000000V                    JSR     PC,QIO.SIZE             ;                       3370
000200  013716  000114'                    MOV     MAD1,(SP)               ;                       3371
000204  062716  000032                     ADD     #32,(SP)
000210  004737  000000G                    JSR     PC,GET.IO.BUFF
000214  005737  000112'                    TST     MX2                     ;                       3373
000220  002437                             BLT     3$
000222  013716  000116'                    MOV     MAD2,(SP)               ;                       3376
000226  062716  000032                     ADD     #32,(SP)
000232  004737  000000G                    JSR     PC,GET.IO.BUFF
000236  013700  000116'                    MOV     MAD2,R0                 ;                       3378
000242  005760  000032                     TST     32(R0)
000246  001024                             BNE     3$
000250  013700  000114'                    MOV     MAD1,R0                 ;                       3382
000254  062700  000032                     ADD     #32,R0
000260  005710                             TST     (R0)
000262  001407                             BEQ     2$
000264  010016                             MOV     R0,(SP)                 ;                       3385
000266  004737  000000G                    JSR     PC,PUT.IO.BUFF
000272  013700  000114'                    MOV     MAD1,R0                 ;                       3386
000276  005060  000032                     CLR     32(R0)
000302  013716  000112'            2$:     MOV     MX2,(SP)                ;                       3389
000306  004737  000000G                    JSR     PC,PUT.PKT
000312  012737  177777  000112'            MOV     #-1,MX2                 ;                       3390
000320  013700  000114'            3$:     MOV     MAD1,R0                 ;                       3395
000324  005760  000032                     TST     32(R0)
000330  001010                             BNE     4$
000332  013716  000110'                    MOV     MX1,(SP)                ;                       3398
000336  004737  000000G                    JSR     PC,PUT.PKT
```

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

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

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


001200                                          .PSECT  $GGG$,  R0
001200  000255                          SEED:   .WORD   255
001202  000365                          NEXT.RANDNUM:
                                                .WORD   365


                                                .SBTTL  GET.RANDOM MULTI-DRIVE TEST ROUTINES
011276                                          .PSECT  $CODE$,  R0

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

```
000064  010002                   MOV     R0,R2                          ; NEXT,RANDNUM,*           3428
000066  010261  000000G          MOV     R2,RANDOM(R1)                  ; *,*(COUNT)
000072  022626                   CMP     (SP)+,(SP)+                    ;                          3425
000074  062701  000002           ADD     #2,R1                          ; *,COUNT                  3424
000100  020127  000036           CMP     R1,#36                         ; COUNT,*
000104  003744                   BLE     1$
000106  000207                   RTS     PC                             ;                          3408

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

```
GLOBAL routine QIO_UNIT : novalue =

!+
!       THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
!       CONFIGURED UNDER THE CURRENT CONTROLLER (CCTLR) TO BE USED FOR THE
!       CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
!       UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
!       RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
!
!       IMPLICIT INPUTS:
!               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
!
!       IMPLICIT OUTPUTS:
!               THE RD/RX DISK NUMBER (DISK ADDRESS) IS LOADED INTO THE
!                       APPROPRIATE FIELD OF BOTH MSCP PACKETS.
!-

    begin

    local
        MOD_COUNT : byte,
        TBL_COUNT : byte,
        SELECT_RD : byte initial (byte (TRUE)),
!ZZZ    RD_COUNT : word initial (0),
        RX_COUNT : word initial (0);

!
! THE UNITS WILL BE SELECTED ON AN ADJUSTABLE RATIO, RD51/52 TO RX50,
! SELECTED VIA THE SOFTWARE PARAMETERS
!
! THIS MODE IS FOR SELECTING DEVICES ON THE FOLLOWING SCHEME:
! CHOOSE A DEVICE AND KEEP IT SELECTED FOR A CONSTANT TIME, THEN
! MOVE TO THE NEXT. THIS IS NON-RANDOM, FIXED SEQUENTIAL OPERATIONAL
! MODE
!

            RD_COUNT = 0;                                                       !ZZZ
            RX_COUNT = 0;                                                       !ZZZ

            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
                   (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
                then

                    if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
                    then
                        RD_COUNT = .RD_COUNT + 1        ! NUMBER OF RD51/52s UNDER TEST
                    else
                        RX_COUNT = .RX_COUNT + 1;       ! NUMBER OF RX50s UNDER TEST

```

```
;  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 neq 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_TYPE] 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       !
```

```
;  3
;  3          if .MOD_COUNT neq 0                                         ! IF AT LEAST ONE RD51/52 PRESENT
;  3          then
;  4              begin
;  4              TBL_COUNT = 0;
;  4
;  4              do
;  5                  begin
;  5                  SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;  5                  TBL_COUNT = .TBL_COUNT + 1;
;  5                  end
;  5              until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;  5                     (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;  4                     (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;  4                    (.TBL_COUNT eql RDM_LEN);
;  4
;  4              MAD1 [DK_NUM] = .CDISK;
;  4              MAD2 [DK_NUM] = .CDISK;
;  4              return;
;  3              end;
;  3
;  2          end;
;  2
;  2  !
;  2  ! IF NO RD51/52 SELECTED, SELECT AN RX50
;  2  !
;  2  ! COUNT THE NUMBER OF RX50s
;  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
;  2             (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql REMOVABLE) 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  ! AND CHOOSE ONE!
;  2  !
;  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;
;   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;
```

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


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

```
000166 013700 001152'                MOV   BST.DEV,R0                 ;                    3493
000172 006300                        ASL   R0
000174 060200                        ADD   R2,R0
000176 032710 010000                 BIT   #10000,(R0)
000202 001022                        BNE   4$
000204 005337 001150'                DEC   BST.CNT                    ;                    3496
000210 013746 001152'                MOV   BST.DEV,-(SP)              ;                    3497
000214 004737 000000G                JSR   PC,SET.UPAR
000220 013700 000114'                MOV   MAD1,R0                    ;                    3498
000224 013760 000000G 000016         MOV   CDISK,16(R0)
000232 013700 000116'                MOV   MAD2,R0                    ;                    3499
000236 013760 000000G 000016         MOV   CDISK,16(R0)
000244 005726                        TST   (SP)+                      ;                    3500
000246 000207                        RTS   PC                         ;                    3495
000250 012702 000003          4$:    MOV   #3,R2                      ; *,OFFSET           3518
000254 013700 001152'         5$:    MOV   BST.DEV,R0                 ;                    3521
000260 001403                        BEQ   6$
000262 020027 000041                 CMP   R0,#41                     ;                    3522
000266 001004                        BNE   7$
000270 012737 000003 001152'  6$:    MOV   #3,BST.DEV                 ;                    3524
000276 000403                        BR    8$                         ;                    3521
000300 062737 000012 001152'  7$:    ADD   #12,BST.DEV                ;                    3526
000306 013700 001152'         8$:    MOV   BST.DEV,R0                 ;                    3528
000312 006300                        ASL   R0
000314 063700 000000G                ADD   CST.ADDR,R0
000320 032710 040000                 BIT   #40000,(R0)
000324 001451                        BEQ   12$
000326 032710 020000                 BIT   #20000,(R0)                ;                    3529
000332 001446                        BEQ   12$
000334 032710 010000                 BIT   #10000,(R0)                ;                    3530
000340 001043                        BNE   12$
000342 132710 000020                 BITB  #20,(R0)                   ;                    3534
000346 001004                        BNE   9$
000350 013746 001176'                MOV   RX.MAX.SEQ.CNT,-(SP)       ;                    3536
000354 010346                        MOV   R3,-(SP)                   ; RX.COUNT,*
000356 000404                        BR    10$
000360 013746 001174'         9$:    MOV   RD.MAX.SEQ.CNT,-(SP)       ;                    3538
000364 013746 000000G                MOV   RD.COUNT,-(SP)
000370 004737 000000G         10$:   JSR   PC,BL$DIV
000374 010037 001150'                MOV   R0,BST.CNT
000400 001003                        BNE   11$                        ;                    3540
000402 012737 000001 001150'         MOV   #1,BST.CNT                 ;                    3542
000410 013716 001152'         11$:   MOV   BST.DEV,(SP)               ;                    3544
000414 004737 000000G                JSR   PC,SET.UPAR
000420 013700 000114'                MOV   MAD1,R0                    ;                    3545
000424 013760 000000G 000016         MOV   CDISK,16(R0)
000432 013700 000116'                MOV   MAD2,R0                    ;                    3546
000436 013760 000000G 000016         MOV   CDISK,16(R0)
000444 022626                        CMP   (SP)+,(SP)+                ;                    3547
000446 000207                        RTS   PC                         ;                    3532
000450 062702 000012          12$:   ADD   #12,R2                     ; *,OFFSET           3518
000454 020227 000041                 CMP   R2,#41                     ; OFFSET,*
000460 003675                        BLE   5$
```

```
000462  013746  000036G         13$:    MOV     RANDOM+36,-(SP)         ;                               3561
000466  042716  100000                  BIC     #100000,(SP)
000472  012746  000144                  MOV     #144,-(SP)
000476  004737  000000G                 JSR     PC,BL$MOD
000502  022626                          CMP     (SP)+,(SP)+
000504  020037  000000G                 CMP     R0,SWP.RAT
000510  103401                          BLO     14$
000512  105004                          CLRB    R4                      ; SELECT.RD                     3563
000514  006004                  14$:    ROR     R4                      ; SELECT.RD                     3571
000516  103105                          BCC     19$
000520  105003                          CLRB    R3                      ; MOD.COUNT                     3574
000522  012701  000003                  MOV     #3,R1                   ; *,OFFSET                      3576
000526  010100                  15$:    MOV     R1,R0                   ; OFFSET,*                      3578
000530  006300                          ASL     R0
000532  063700  000000G                 ADD     CST.ADDR,R0
000536  032710  040000                  BIT     #40000,(R0)
000542  001417                          BEQ     16$
000544  032710  020000                  BIT     #20000,(R0)             ;                               3579
000550  001414                          BEQ     16$
000552  132710  000020                  BITB    #20,(R0)                ;                               3580
000556  001411                          BEQ     16$
000560  032710  010000                  BIT     #10000,(R0)             ;                               3581
000564  001006                          BNE     16$
000566  005000                          CLR     R0                      ;                               3584
000570  150300                          BISB    R3,R0                   ; MOD.COUNT,*
000572  006300                          ASL     R0
000574  010160  000064'                 MOV     R1,STORAGE(R0)          ; OFFSET,*
000600  105203                          INCB    R3                      ; MOD.COUNT                     3585
000602  062701  000012          16$:    ADD     #12,R1                  ; *,OFFSET                      3576
000606  020127  000041                  CMP     R1,#41                  ; OFFSET,*
000612  003745                          BLE     15$
000614  105703                          TSTB    R3                      ; MOD.COUNT                     3592
000616  001445                          BEQ     19$
000620  105002                          CLRB    R2                      ; TBL.COUNT                     3595
000622  005000                  17$:    CLR     R0                      ;                               3599
000624  150200                          BISB    R2,R0                   ; TBL.COUNT,*
000626  006300                          ASL     R0
000630  016046  000000G                 MOV     RANDOM(R0),-(SP)
000634  042716  100000                  BIC     #100000,(SP)
000640  005046                          CLR     -(SP)
000642  110316                          MOVB    R3,(SP)                 ; MOD.COUNT,*
000644  004737  000000G                 JSR     PC,BL$MOD
000650  006300                          ASL     R0
000652  016016  000064'                 MOV     STORAGE(R0),(SP)
000656  004737  000000G                 JSR     PC,SET.UPAR
000662  105202                          INCB    R2                      ; TBL.COUNT                     3600
000664  022626                          CMP     (SP)+,(SP)+             ;                               3598
000666  013700  000000G                 MOV     CUOFF,R0                ;                               3602
000672  006300                          ASL     R0
000674  063700  000000G                 ADD     CST.ADDR,R0
000700  032710  040000                  BIT     #40000,(R0)
000704  001406                          BEQ     18$
000706  032710  020000                  BIT     #20000,(R0)             ;                               3603
```

```
000712  001403                            BEQ     18$
000714  032710  010000                    BIT     #10000,(R0)              ;                        3604
000720  001510                            BEQ     24$
000722  120227  000020           18$:     CMPB    R2,#20                   ; TBL.COUNT,*            3605
000726  001335                            BNE     17$
000730  000504                            BR      24$                      ;                        3607
000732  105003                   19$:     CLRB    R3                       ; MOD.COUNT              3620
000734  012701  000003                    MOV     #3,R1                    ; *,OFFSET               3622
000740  010100                   20$:     MOV     R1,R0                    ; OFFSET,*               3624
000742  006300                            ASL     R0
000744  063700  000000G                   ADD     CST.ADDR,R0
000750  032710  040000                    BIT     #40000,(R0)
000754  001417                            BEQ     21$
000756  032710  020000                    BIT     #20000,(R0)              ;                        3625
000762  001414                            BEQ     21$
000764  132710  000020                    BITB    #20,(R0)                 ;                        3626
000770  001011                            BNE     21$
000772  032710  010000                    BIT     #10000,(R0)              ;                        3627
000776  001006                            BNE     21$
001000  005000                            CLR     R0                       ;                        3630
001002  150300                            BISB    R3,R0                    ; MOD.COUNT,*
001004  006300                            ASL     R0
001006  010160  000064'                   MOV     R1,STORAGE(R0)           ; OFFSET,*
001012  105203                            INCB    R3                       ; MOD.COUNT              3631
001014  062701  000012           21$:     ADD     #12,R1                   ; *,OFFSET               3622
001020  020127  000041                    CMP     R1,#41                   ; OFFSET,*
001024  003745                            BLE     20$
001026  105703                            TSTB    R3                       ; MOD.COUNT              3638
001030  001445                            BEQ     25$
001032  105002                            CLRB    R2                       ; TBL.COUNT              3641
001034  005000                   22$:     CLR     R0                       ;                        3645
001036  150200                            BISB    R2,R0                    ; TBL.COUNT,*
001040  006300                            ASL     R0
001042  016046  000000G                   MOV     RANDOM(R0),-(SP)
001046  042716  100000                    BIC     #100000,(SP)
001052  005046                            CLR     -(SP)
001054  110316                            MOVB    R3,(SP)                  ; MOD.COUNT,*
001056  004737  000000G                   JSR     PC,BL$MOD
001062  006300                            ASL     R0
001064  016016  000064'                   MOV     STORAGE(R0),(SP)
001070  004737  000000G                   JSR     PC,SET.UPAR
001074  105202                            INCB    R2                       ; TBL.COUNT              3646
001076  022626                            CMP     (SP)+,(SP)+              ;                        3644
001100  013700  000000G                   MOV     CUOFF,R0                 ;                        3648
001104  006300                            ASL     R0
001106  063700  000000G                   ADD     CST.ADDR,R0
001112  032710  040000                    BIT     #40000,(R0)
001116  001406                            BEQ     23$
001120  032710  020000                    BIT     #20000,(R0)              ;                        3649
001124  001403                            BEQ     23$
001126  032710  010000                    BIT     #10000,(R0)              ;                        3650
001132  001505                            BEQ     30$
001134  120227  000020           23$:     CMPB    R2,#20                   ; TBL.COUNT,*            3651
```

```
001140  001335                            BNE     22$
001142  000501                    24$:    BR      30$                     ;                       3653
001144  105003                    25$:    CLRB    R3                      ; MOD.COUNT             3664
001146  012701  000003                    MOV     #3,R1                   ; *,OFFSET              3666
001152  010100                    26$:    MOV     R1,R0                   ; OFFSET,*              3668
001154  006300                            ASL     R0
001156  063700  000000G                   ADD     CST.ADDR,R0
001162  032710  040000                    BIT     #40000,(R0)
001166  001414                            BEQ     27$
001170  032710  020000                    BIT     #20000,(R0)             ;                       3669
001174  001411                            BEQ     27$
001176  032710  010000                    BIT     #10000,(R0)             ;                       3670
001202  001006                            BNE     27$
001204  005000                            CLR     R0                      ;                       3673
001206  150300                            BISB    R3,R0                   ; MOD.COUNT,*
001210  006300                            ASL     R0
001212  010160  000064'                   MOV     R1,STORAGE(R0)          ; OFFSET,*
001216  105203                            INCB    R3                      ; MOD.COUNT             3674
001220  062701  000012            27$:    ADD     #12,R1                  ; *,OFFSET              3666
001224  020127  000041                    CMP     R1,#41                  ; OFFSET,*
001230  003750                            BLE     26$
001232  105703                            TSTB    R3                      ; MOD.COUNT             3680
001234  001457                            BEQ     31$
001236  105002                            CLRB    R2                      ; TBL.COUNT             3683
001240  005000                    28$:    CLR     R0                      ;                       3687
001242  150200                            BISB    R2,R0                   ; TBL.COUNT,*
001244  006300                            ASL     R0
001246  016046  000000G                   MOV     RANDOM(R0),-(SP)
001252  042716  100000                    BIC     #100000,(SP)
001256  005046                            CLR     -(SP)
001260  110316                            MOVB    R3,(SP)                 ; MOD.COUNT,*
001262  004737  000000G                   JSR     PC,BL$MOD
001266  006300                            ASL     R0
001270  016016  000064'                   MOV     STORAGE(R0),(SP)
001274  004737  000000G                   JSR     PC,SET.UPAR
001300  105202                            INCB    R2                      ; TBL.COUNT             3688
001302  022626                            CMP     (SP)+,(SP)+             ;                       3686
001304  013700  000000G                   MOV     CUOFF,R0                ;                       3690
001310  006300                            ASL     R0
001312  063700  000000G                   ADD     CST.ADDR,R0
001316  032710  040000                    BIT     #40000,(R0)
001322  001406                            BEQ     29$
001324  032710  020000                    BIT     #20000,(R0)             ;                       3691
001330  001403                            BEQ     29$
001332  032710  010000                    BIT     #10000,(R0)             ;                       3692
001336  001403                            BEQ     30$
001340  120227  000020            29$:    CMPB    R2,#20                  ; TBL.COUNT,*           3693
001344  001335                            BNE     28$
001346  013700  000114'           30$:    MOV     MAD1,R0                 ;                       3695
001352  013760  000000G 000016            MOV     CDISK,16(R0)
001360  013700  000116'                   MOV     MAD2,R0                 ;                       3696
001364  013760  000000G 000016            MOV     CDISK,16(R0)
001372  000207                            RTS     PC                      ;                       3682
```

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

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

```
GLOBAL routine QIO_FUNC : novalue =

!+
!       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
!       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
!       BY THE FOLLWING ALGORITHM:
!
!               IF THE CHOSEN UNIT IS PROTECTED
!               THEN
!                       FUNCTION = READ
!               ELSE (UNPROTECTED)
!                       FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
!                       NUMBER
!
!       IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE-
!       COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
!       THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
!       PACKET. OTHERWISE, THE SECOND MSCP PACKET IS RETURNED TO THE POOL.
!
!
!       PERIODIACLLY, THIS ROUTINE WILL CALL THE DUP ROUTINE BEFORE IT    ZZZ
!       BEGINS ITS OWN TASK. IF THE OPERATOR HAS SELECTED, "ALSO RUN      ZZZ
!       DUP EXERCISER," THEN DUP TESTING OF DBNS WILL BE INTERLEAVED      ZZZ
!       WITH THE REGULAR MSCP TESTING OF THE LBNS.                        ZZZ
!
!       TO AVOID LONG, CUMULATIVE INIT TIMES, THE DUP CODE IS ONLY        ZZZ
!       EXECUTED AFTER (25 TIMES 'DUPROUND') MSCP I/O's HAVE BEEN DONE.   ZZZ
!       THE DUMBER OF DUP I/O's IS 'DUPROUND'. THIS GIVES US A 25 TO 1    ZZZ
!       INTERLEAVE.                                                       ZZZ
!
!       THE DUP TESTING IS DONE BY EXECUTING CONTROLLER LOCAL PROGRAMS    ZZZ
!       TO READ OR WRITE/READ DBNS. AFTER THE DUP TESTING, THE CON-       ZZZ
!       TROLLER IS REINITIALIZED, ANC QIO_FUNC ROUTINE CONTINUES FROM     ZZZ
!       WHERE IT LEFT OFF.                                                ZZZ
!
!
!       IMPLICIT INPUTS:
!               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
!               CUOFF - CURRENT UNIT CST OFFSET
!
!       IMPLICIT OUTPUTS:
!               THE OPCODE FIELD OF ONE OR BOTH MSCP PACKETS IS LOADED.
!-

    begin

    local
        FUNC : word;                                            ! OPCODE (READ OR WRITE)

DUOFF = .CUOFF;                                 !SAVE IN CASE OTHER CMDS ZZZ
                                                !LEFT IN QUEUE           ZZZ
IF ((.CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] LEQ 0) AND !MSCP CNT=0        ZZZ
    (.CST_ADDR [.DUOFF, D_TYPE] NEQ RX_50) AND          !FIXED DISK       ZZZ
```

```
3        (.CST_ADDR [.DUOFF + OF_DUPFLAGS, NODUPMEDIA] NEQ 1)) !MEDIA IN         ZZZ
3                                                                              !ZZZ
2   THEN                                                                       !ZZZ
3       BEGIN                                                                  !ZZZ
3       PUT_PKT (.MX2);                         !RETURN 2ND ENVLOPE            !ZZZ
3       MX2 = -1;                               !INDICATE FAILURE               ZZZ
3       DUP ();                                 !DO DUP TEST                    ZZZ
3       CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] =  !REINIT MSCP FUN-              ZZZ
3               (25 * .DUPROUND);               !CTION COUNTER                  ZZZ
3                                                                              !ZZZ
3   !       THE FOLLOWING REINITS 2 ENVELOPES, SO THAT THE MSCP EXERCISER      ZZZ
3   !       CAN PROCEED AS BEFORE THE DUP EXERCISER WAS CALLED.                ZZZ
3   !                                                                          ZZZ
3       DUP_FLAGS = .DUP_FLAGS OR SWP_DINT;     !SET DUP INIT FLAG              ZZZ
3       INIT_TEST ();                           !REINIT CONTROLLER              ZZZ
3       DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT); !CLR DUP INIT DLAG           ZZZ
3                                                                              !ZZZ
3       MX2 = -1;                               !ASSUME NO 2ND ENVELOPE         ZZZ
3       IF (MX1 = GET_PKT (.CCTLR)) LSS 0       !TRY FOR 1ST ENVELOPE           ZZZ
4               OR (.EOP_FLAG)                  !IF FAILURE                     ZZZ
3       THEN    RETURN;                         !NO POINT TO GO ON              ZZZ
3       IF (MX2 = GET_PKT (.CCTLR)) LSS 0       !TRY FOR 2ND ENVELOPE           ZZZ
4               OR (.EOP_FLAG)                  !IF FAILURE                     ZZZ
4       THEN    BEGIN                                                          !ZZZ
4               PUT_PKT (.MX1);                 !PUT 1ST BACK IN POOL           ZZZ
4               MX1 = -1;                       !INDICATE FAILURE               ZZZ
4               RETURN;                         !DONE                           ZZZ
3               END;                                                           !ZZZ
3                                                                              !ZZZ
3               MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2); !CALC START ADDR        ZZZ
3               MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2); !OF BOTH ENVELOPES      ZZZ
3               GET_RANDOM ();                  !GET SET OF RANDOM NOS          ZZZ
3               QIO_UNIT ();                    !PUT RAND UNIT NO IN            ZZZ
2               END;                            !ENVELOPES                      ZZZ
2                                                                              !ZZZ
2   !       MSCP CODE STARTS HERE                                              ZZZ
2   !                                                                          ZZZ
2
2   CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] =      !                              ZZZ
2       .CST_ADDR [.CUOFF + OF_COUNT, 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
;   3               if (.RANDOM [1] and 1)                                      ! USE 2ND RANDOM NUMBER TO SELECT
;   2               then
;   2                   FUNC = OP_RD                                            !       READ
;   2               else
;   2                   FUNC = OP_WRT;                                          !       WRITE
;   2
;   2
;   2
;   3       IF (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED)          !I/O's ARE CANCELLED WHEN CMD ZZZ
;   2       THEN                                                                !RING IS FULL. DON'T LET THIS ZZZ
;   2           IF .RW_BALANCE GEQU TOO_MANY_READS                              !UPSET THE BALANCE BETWEEN    ZZZ
;   2           THEN                                                            !THE NUMBER OF READS AND      ZZZ
;   2               FUNC = OP_WRT;                                              !WRITES.                      ZZZ
;   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 COMPARE 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
;   3               MAD1 [MODIFY] = MD_CMP;                                     ! ADD COMPARE MODIFIER
;   3
;   2           end;
;   2
;   2       if .MAD2 [OPCODE] eql 0                                             ! IF NO OPCODE IN 2ND PACKET
;   2       then
;   3           begin
;   3           PUT_PKT (.MX2);                                                 ! RETURN 2ND PACKET TO POOL
;   3           MX2 = -1;                                                       ! MARK IT UNUSED
;   2           end;
;   2
```

```
;    3867  1         end;                                              ! ROUTINE QIO_FUNC

                                            .SBTTL  QIO.FUNC MULTI-DRIVE TEST ROUTINES
000000  004137  000000G             QIO.FUNC::
                                            JSR     R1,$SAVE4                  ;          3708
000004  013737  000000G 001156'             MOV     CUOFF,DUOFF                ;          3757
000012  013702  000000G                     MOV     CST.ADDR,R2                ;          3759
000016  013701  001156'                     MOV     DUOFF,R1
000022  010100                              MOV     R1,R0
000024  006300                              ASL     R0
000026  060200                              ADD     R2,R0
000030  005760  000022                      TST     22(R0)
000034  003146                              BGT     4$
000036  010100                              MOV     R1,R0                      ;          3760
000040  006300                              ASL     R0
000042  060200                              ADD     R2,R0
000044  132710  000020                      BITB    #20,(R0)
000050  001540                              BEQ     4$
000052  010100                              MOV     R1,R0                      ;          3761
000054  006300                              ASL     R0
000056  060200                              ADD     R2,R0
000060  005760  000020                      TST     20(R0)
000064  100532                              BMI     4$
000066  013746  000112'                     MOV     MX2,-(SP)                  ;          3765
000072  004737  000000G                     JSR     PC,PUT.PKT
000076  012737  177777  000112'             MOV     #-1,MX2                    ;          3766
000104  004737  000000V                     JSR     PC,DUP                     ;          3767
000110  013701  001156'                     MOV     DUOFF,R1                   ;          3768
000114  006301                              ASL     R1
000116  063701  000000G                     ADD     CST.ADDR,R1
000122  013716  000000G                     MOV     DUPROUND,(SP)              ;          3769
000126  012746  000031                      MOV     #31,-(SP)
000132  004737  000000G                     JSR     PC,BL$MUL
000136  010061  000022                      MOV     R0,22(R1)
000142  052737  000002  000000G             BIS     #2,DUP.FLAGS               ;          3774
000150  004737  001134'                     JSR     PC,INIT.TEST               ;          3775
000154  042737  000002  000000G             BIC     #2,DUP.FLAGS               ;          3776
000162  012737  177777  000112'             MOV     #-1,MX2                    ;          3778
000170  013716  000000G                     MOV     CCTLR,(SP)                 ;          3779
000174  004737  000000G                     JSR     PC,GET.PKT
000200  010037  000110'                     MOV     R0,MX1
000204  002426                              BLT     2$
000206  132737  000001  000000G             BITB    #1,EOP.FLAG                ;          3780
000214  001022                              BNE     2$                         ;          3708
000216  013716  000000G                     MOV     CCTLR,(SP)                 ;          3782
000222  004737  000000G                     JSR     PC,GET.PKT
000226  010037  000112'                     MOV     R0,MX2
000232  002404                              BLT     1$
000234  132737  000001  000000G             BITB    #1,EOP.FLAG                ;          3783
000242  001411                              BEQ     3$
000244  013716  000110'             1$:     MOV     MX1,(SP)                   ;          3785
000250  004737  000000G                     JSR     PC,PUT.PKT
```

G12


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

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

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

```
GLOBAL ROUTINE DUP : NOVALUE =                                                   !ZZZ
!+                                                                              !ZZZ
!       THIS ROUTINE IS CALLED BY QIO_FUNC AFTER 25 * 'DUPROUND' RD/WTS.       ZZZ
!       THIS EXERCISER WAS PLACED IN THE MIDDLE OF THE MSCP EXERCISER,          ZZZ
!       SO COMMON INIT AND OTHER ROUTINES COULD BE USED.                        ZZZ
!                                                                               ZZZ
!       THE DUP EXERCISER WILL PERFORM EITHER READ-ONLY, OR WRITE-READ-         ZZZ
!       COMPARE OPERATIONS ON THE DIAGNOSTIC BLOCKS (DBNS). IT WILL             ZZZ
!       RECORD THE STATISTICS IN THE TALLY TABLES.                              ZZZ
!                                                                               ZZZ
!       THE PROGRAM USES CONTROLLER LOCAL PROGRAMS TO WRITE AND READ            ZZZ
!       DBNS. WHEN WRITING TO THE DBNS, A ONE WORD PATTERN WILL BE              ZZZ
!       SELECTED, AND REPLICATED THROUGH A 256 WORD BLOCK FOR DATA.             ZZZ
!       THE ROUTINE WILL WRITE 'DUPROUND' NUMBER OF SEQUENTIAL DBN              ZZZ
!       BLOCKS. IF THE CONTROLLER LOCAL PROGRAMS EXIST, AND THE OPERATOR        ZZZ
!       SELECTS 'WRITE TO DIAGNOSTIC AREA', WRITE-READ-COMPARES WILL BE         ZZZ
!       PERFORMED ON THE DBNS. OTHERWISE, READS WITH NO COMPARES WILL BE        ZZZ
!       DONE. BAD BLOCKS FOUND IN THE COMPARISON TESTS WILL NOT BE LIST-        ZZZ
!       ED IN THE RCT TABLES.                                                   ZZZ
!                                                                               ZZZ
!       AFTER 'DUPROUND' NUMBER OF DBNS HAVE BEEN TESTED, THE ENVELOPES         ZZZ
!       WILL BE REINITIATED, SO THAT THE MSCP EXERCISER CAN CONTINUE            ZZZ
!       AS BEFORE.                                                              ZZZ
!                                                                               ZZZ
!       IMPLICIT INPUTS:                                                        ZZZ
!                   CCTLR - CURRENT CONTROLLER NUMBER                           ZZZ
!                   CST_ADDRS - CONTAINS THE CURRENT CONTROLLER                 ZZZ
!                               STATUS TABLE                                    ZZZ
!                   CUOFF - CURRENT OFFSET IN CST TABLE FOR                     ZZZ
!                           PARTICULAR DRIVE                                    ZZZ
!                                                                               ZZZ
!       IMPLICIT OUTPUTS:                                                       ZZZ
!                   S_PATTERN - PATTERN BEING WRITTEN TO DBNS                   ZZZ
!                                                                               ZZZ
!                                                                              !ZZZ
!-                                                                             !ZZZ
```

```
;  1
;  1
;  2   BEGIN
;  2   OWN
;  2           TEMP : WORD;
;  2
;  2   !PRINTX (DBM110);
;  2   !PRINTX (DER10);
;  2
;  2   until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or        ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;  2         (.EOP_FLAG eql true) do                     ! Make sure all MSCP commands are completed
;  3       begin
;  3       BREAK;                                        ! BREAK FOR ACT
;  3       PROC_RETPKT();                                ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;  3       RP_INDX = .RP_INDX + 1;                       ! INCREMENT RP_INDX
;  3       if .RP_INDX geq RP_CNT then (RP_INDX = 0;);   ! MAKE SURE THE COUNTER DOES NOT GET TO BIG
;  3       RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);   ! CALCULATE RETPKT ADDRESS
;  2       end;
;  2
;  2
;  2   S_PATTERN = .RANDOM [1];                                        !OTHER UNIT VARIABLES
;  2
;  2   IF (.CST_ADDR [.DUOFF + OF_DBN, D_DBN] + .dupround) GEQ 144      ! TEST TO SEE IF NEXT DBN'S TO LARGE
;  2   THEN (CST_ADDR [.DUOFF + OF_DBN, D_DBN] = 0);                   ! CIRCLE ARROUND IF DBN TO LARGE
;  2
;  2   DUPIDLE ();                                                     ! DO A GET DUST STATUS TO FIND IF LOCAL DUP MEDIA
;  2   IF .CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] EQL 1 THEN RETURN; ! IF DUP LOCAL MEDIA NOT THERE THEN RETURN
;  2
;  2   TEMP = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];
;  2   INCR DBNCNT FROM (.TEMP + 1) TO (.TEMP + .dupround) DO                    ! INCREMENT FROM RELATIVE DBN TO DBN + dupround
;  3           BEGIN
;  3           IF .CST_ADDR [.DUOFF + OF_DBN, DUPWRITE]                                  ! IF WRITE FLAG SET IN CST TABLE THEN WRITE DBN'S
;  3           THEN
;  4                   BEGIN
;  4                   DUPIDLE ();                                       ! MAKE SURE THE CONTROLLER IS IN AN IDLE STATE
;  4                   DUPWRTDBN ();                                     ! CALL ROUTINE TO HANDLE WRITTING ROUTINES
;  3                   END;
;  3
;  3           DUPIDLE ();                                               ! MAKE SURE CONTROLLER IN IDLE STATE
;  3           DUPREDDBN ();                                             ! CALL ROUTINE TO HANDLE READING DBN'S
;  3
;  3           CST_ADDR [.DUOFF + OF_DBN, D_DBN] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN] + 1; ! INCREMENT RELATIVE DBN COUNTER
;  3
;  3           IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1            ! ERROR IN DUP REINITIALIZE
;  3           THEN RETURN;                                              ! AND RETURN
;  2           END;
;  1   END;


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

K12


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

000000  004137  000000G          DUP::   JSR     R1,$SAVE3                       ;               3869
000004  013700  000000G          1$:     MOV     RP.ADDR,R0                      ;               3915
000010  023760  000000G 000004           CMP     CRN.LOW,4(R0)
000016  001433                           BEQ     3$
000020  123727  000000G 000001           CMPB    EOP.FLAG,#1                     ;               3916
000026  001427                           BEQ     3$
000030  104422                           TRAP    22                              ;               3917
000032  004737  000000V                  JSR     PC,PROC.RETPKT                  ;               3919
000036  005237  000000G                  INC     RP.INDX                         ;               3920
000042  023727  000000G 000010           CMP     RP.INDX,#10                     ;               3921
000050  002402                           BLT     2$
000052  005037  000000G                  CLR     RP.INDX
000056  013746  000000G          2$:     MOV     RP.INDX,-(SP)                   ;               3922
000062  012746  000054                   MOV     #54,-(SP)
000066  004737  000000G                  JSR     PC,BL$MUL
000072  062700  000000G                  ADD     #RETPKT,R0
000076  010037  000000G                  MOV     R0,RP.ADDR
000102  022626                           CMP     (SP)+,(SP)+                     ;               3917
000104  000737                           BR      1$                              ;               3915
000106  013737  000002G 000000G  3$:     MOV     RANDOM+2,S.PATTERN              ;               3926
000114  013700  001156'                  MOV     DUOFF,R0                        ;               3928
000120  006300                           ASL     R0
000122  063700  000000G                  ADD     CST.ADDR,R0
000126  005001                           CLR     R1
000130  156001  000020                   BISB    20(R0),R1
000134  063701  000000G                  ADD     DUPROUND,R1
000140  020127  000220                   CMP     R1,#220
000144  002402                           BLT     4$
000146  105060  000020                   CLRB    20(R0)                          ;               3929
000152  004737  000000V          4$:     JSR     PC,DUPIDLE                      ;               3931
000156  013700  001156'                  MOV     DUOFF,R0                        ;               3932
000162  006300                           ASL     R0
000164  063700  000000G                  ADD     CST.ADDR,R0
000170  005760  000020                   TST     20(R0)
000174  100456                           BMI     8$
000176  116037  000020  001204'          MOVB    20(R0),TEMP                     ;               3934
000204  105037  001205'                  CLRB    TEMP+1
000210  013703  001204'                  MOV     TEMP,R3                         ;               3935
000214  063703  000000G                  ADD     DUPROUND,R3
000220  013700  001156'                  MOV     DUOFF,R0                        ;               3937
000224  006300                           ASL     R0
000226  063700  000000G                  ADD     CST.ADDR,R0
000232  010001                           MOV     R0,R1
000234  062701  000020                   ADD     #20,R1
000240  013702  001204'                  MOV     TEMP,R2                         ; *.DBNCNT      3935
000244  000427                           BR      7$
000246  032711  010000           5$:     BIT     #10000,(R1)                     ;               3937
000252  001404                           BEQ     6$
000254  004737  000000V                  JSR     PC,DUPIDLE                      ;               3940
```

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

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


;    3953  1
```

```
GLOBAL ROUTINE DUPWRTDBN : NOVALUE =

!+
! 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
MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;             ! CALL IT IMMEDIATE
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_LC] = 80;                 ! BYTE COUNT TO BE TRANSFERED EQU4LS 2 ***see pg 26 of DUP s
pec
```

```
;  4          MSCP_PKT [.MX1, BUF_0] = DUPPKT;                        ! LOAD DESCRIBTOR BUFFER
;  4          MSCP_PKT [.MX1, MODIFY] = 0;                            !
;  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                      ! CALL IT sequential
;  4          DUPCOMMAND ();                                          ! SENDS AND RECEIVES THE COMMAND
;  4
;  4          IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;  4             (.DUPPKT [DUPTYPE] NEQU 1) OR                        !dup type error
;  5             (.DUPPKT [DUPMESG] NEQU 6)
;  4          THEN
;  5                (D_FAIL = 1;                                      !TELL HARD_ERROR IT WAS A DUP PROBLEM          ZZZ
;  5                 HARD_ERROR ();                                   !
;  5                 D_FAIL = 0;                                      !                                              ZZZ
;  5                 CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;        ! SET FLAG
;  4                 RETURN;);                                        ! NO POINT IN CONTINUING
;  4
;  5          DO (MX1 = GET_PKT (.CCTLR))
;  4          UNTIL (.MX1 GEQ 0);                                     ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4
;  4          MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;                       ! PACKET SIZE                      SEND DATA
;  4          MSCP_PKT [.MX1, OPCODE] = OP_SDD;                       ! OPCODE = SEND DATA
;  4          MSCP_PKT [.MX1, BC_LO] = 6;                             ! BYTE COUNT TO BE TRANSFERED EQUALS 6
;  4          MSCP_PKT [.MX1, BUF_0] = DUPPKT;                        ! LOAD DESCRIBTOR BUFFER
;  4          DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM];  !LOAD UNIT NUMBER (RDRX)
;  4          DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;  4          DUPPKT [DUPBF2] = .S_PATTERN;                           ! LOAD PATTERN
;  4          MSCP_PKT [.MX1, MODIFY] = 0;                            !
;  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                      ! CALL IT sequential
;  4          DUPCOMMAND ();                                          ! SENDS AND RECEIVES THE COMMAND
;  4
;  4          IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  ! status error
;  4          THEN RETURN;
;  4
;  5          DO (MX1 = GET_PKT (.CCTLR))
;  4          UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4
;  4          MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                       ! PACKET SIZE                      RECEIVE DATA
;  4          MSCP_PKT [.MX1, OPCODE] = OP_RCD;                       ! OPCODE = RECEIVE DATA
;  4          MSCP_PKT [.MX1, BC_LO] = 4;                             ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;  4          MSCP_PKT [.MX1, BUF_0] = DUPPKT;                        ! LOAD DESCRIBTOR BUFFER
;  4          MSCP_PKT [.MX1, MODIFY] = 0;                            !
;  4          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                      ! CALL IT sequential
;  4          DUPCOMMAND ();                                          ! SENDS AND RECEIVES THE COMMAND
;  4
;  4
;  4          IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND                ZZZ
;  4             (.DUPPKT [DUPTYPE] EQL 3) AND                        !NO dup type error               ZZZ
;  4             (.DUPPKT [DUPMESG] EQL 3) AND                        !                                ZZZ
;  5             (.DUPPKT [DUPBF1]  EQL 0)                            !AND A successful write code     ZZZ
;  5                                                                  !                                ZZZ
;  4          THEN                                                    !THEN                            ZZZ
;  4             LEAVE DUP_WLOOP                                      !I/O OK. EXIT RETRY LOOP.        ZZZ
;  4          ELSE                                                    !                                ZZZ
;  5             BEGIN                                                !                                ZZZ
```

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


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

C13


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

D13

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

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

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

```
;  1   GLOBAL ROUTINE DUPREDDBN : NOVALUE =
;  1
;  1   !+
;  1   ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;  1   ! "REDDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;  1   ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM READS A DIAGONOSTIC BLOCK (DBN)
;  1   ! AND PLACES IT IN THE DUP BUFFER CALLED "DUPPKT".  IF AN ERROR OCCURS WHILE
;  1   ! RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;  1   ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;  1   !
;  1   !
;  1   !       IMPLICIT INPUTS:
;  1   !               CST_ADDR    - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;  1   !               DUOFF       - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;  1   !
;  2   BEGIN
;  2   LOCAL                                             !                                   ZZZ
;  2         TRYNUM : WORD,                              !                                   ZZZ
;  2         MAX_TRY_COUNT : word initial (9);           ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR ZZZ
;  2                                                     !                                   ZZZ
;  2   LABEL                                             !                                   ZZZ
;  2        DUP_RLOOP;                                   !START OR DUP READ RETRY LOOP       ZZZ
;  2
;  2
;  2   !PRINTX (DER12);
;  2   T_ADDR [T_DBN_RD] = .T_ADDR [T_DBN_RD] + 1;       ! INCREMENT # OF READS GIVEN
;  2
;  2   TRYNUM = 0;                                       !ZERO TRY COUNTER                   ZZZ
;  2   DUP_RLOOP:                                        !LABEL FOR LOOP EXCAPE ON GOOD READ ZZZ
;  3   BEGIN                                             !BEGIN DUP_RLOOP                    ZZZ
;  3   INCR TRIES FROM 1 TO 10 DO                        !START TRYING DUP READS             ZZZ
;  4   BEGIN                                             !BEGIN LARGE DO LOOP                ZZZ
;  4                                                     !                                   ZZZ
;  4
;  4    MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                ! PACKET SIZE                    EXECUTE REDDBN PROGRAM
;  4    MSCP_PKT [.MX1, OPCODE] = OP_ELP;                ! OPCODE = EXECUTE LOCAL PROGRAM
;  4    MSCP_PKT [.MX1, L1] = %ascii'R';                 ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;  4    MSCP_PKT [.MX1, L2] = %ascii'E';
;  4    MSCP_PKT [.MX1, L3] = %ascii'D';
;  4    MSCP_PKT [.MX1, L4] = %ascii'D';
;  4    MSCP_PKT [.MX1, L5] = %ascii'B';
;  4    MSCP_PKT [.MX1, L6] = %ascii'N';
;  4    MSCP_PKT [.MX1, MODIFY] = 1;                     ! STANDALONE MODIFIER
;  4    MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;               ! CALL IT IMMEDIATE
;  4    DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4
;  4    IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1           !status error
;  4    THEN RETURN;
;  4
;  5    DO (MX1 = GET_PKT (.CCTLR))
;  4    UNTIL (.MX1 GEQ 0);                   ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4
;  4            MSCP_PKT [.MX1, MSGLEN] = SZ_REC;               ! PACKET SIZE                 RECIEVE DATA
```

```
; 4            MSCP_PKT [.MX1, OPCODE] = OP_RCD;                        ! OPCODE = RECEIVE DATA
; 4            MSCP_PKT [.MX1, BC_LO] = 80;                             ! BYTE COUNT TO BE TRANSFERED EQUALS 2 *****see pg 26 DUP sp
ec
; 4            MSCP_PKT [.MX1, BUF_0] = DUPPKT;                         ! LOAD DESCRIBTOR BUFFER
; 4            MSCP_PKT [.MX1, MODIFY] = 0;                             !
; 4            MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                       ! CALL IT sequential
; 4            DUPCOMMAND ();                                           ! SENDS AND RECEIVES THE COMMAND

; 4            IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
; 4               (.DUPPKT [DUPTYPE] NEQU 1) OR                        !dup type error
; 5               (.DUPPKT [DUPMESG] NEQU 5)
; 4            THEN
; 5                    (D_FAIL = 1;                                     !TELL HARD_ERROR IT WAS A DUP PROBLEM          ZZZ
; 5                     HARD_ERROR ();
; 5                     D_FAIL = 0;                                     !                                              ZZZ
; 5                     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;       ! SET FLAG
; 4                     RETURN;);                                       ! NO POINT IN CONTINUING

; 5            DO (MX1 = GET_PKT (.CCTLR))
; 4            UNTIL (.MX1 GEQ 0);                                      ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR

; 4            MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;                        ! PACKET SIZE                       SEND DATA
; 4            MSCP_PKT [.MX1, OPCODE] = OP_SDD;                        ! OPCODE = SEND DATA
; 4            MSCP_PKT [.MX1, BC_LO] = 4;                              ! BYTE COUNT TO BE TRANSFERED EQUALS 4
; 4            MSCP_PKT [.MX1, BUF_0] = DUPPKT;                         ! LOAD DESCRIBTOR BUFFER
; 4            DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; ! LOAD UNIT NUMBER (RDRX)
; 4            DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
; 4            MSCP_PKT [.MX1, MODIFY] = 0;                             !
; 4            MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                       ! CALL IT sequential
; 4            DUPCOMMAND ();                                           ! SENDS AND RECEIVES THE COMMAND

; 4             IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1          !status error
; 4             THEN RETURN;

; 5            DO (MX1 = GET_PKT (.CCTLR))
; 4            UNTIL (.MX1 GEQ 0);                           ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR

; 4            MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                        ! PACKET SIZE                       RECEIVE DATA
; 4            MSCP_PKT [.MX1, OPCODE] = OP_RCD;                        ! OPCODE = GET DUST STATUS
; 4            MSCP_PKT [.MX1, BC_LO] = 514;                            ! BYTE COUNT TO BE TRANSFERED EQUALS 512
; 4            MSCP_PKT [.MX1, BUF_0] = DUPPKT;                         ! LOAD DESCRIBTOR BUFFER
; 4            MSCP_PKT [.MX1, MODIFY] = 0;                             !
; 4            MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;                       ! CALL IT sequential
; 4            DUPCOMMAND ();                                           ! SENDS AND RECEIVES THE COMMAND

; 4            IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND                   ZZZ
; 4               (.DUPPKT [DUPTYPE] EQL 6) AND                        !NO dup type error                  ZZZ
; 5               (.DUPPKT [DUPMESG] EQL 2)                            !                                   ZZZ
; 4            THEN                                                     !THEN                               ZZZ
; 4               LEAVE DUP_RLOOP                                       !I/O OK. EXIT RETRY LOOP.           ZZZ
; 4            ELSE                                                     !                                   ZZZ
; 5               BEGIN                                                 !                                   ZZZ
; 5               TRYNUM = .TRYNUM + 1;                                 !INCR ATTEMPT COUNT                 ZZZ
; 5               IF .TRYNUM  EQL .MAX_TRY_COUNT                        !IF IT FAILED ALL RETRIES, THEN     ZZZ
```

H13


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


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

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

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

```
001064  062706  000012          7$:     ADD     #12,SP                          ;                       4111
001070  005303                          DEC     R3                              ; TRIES                 4110
001072  001402                          BEQ     8$
001074  000137  015424'                 JMP     1$
001100  013746  000000G         8$:     MOV     CCTLR,-(SP)                     ;                       4197
001104  004737  000000G                 JSR     PC,GET.PKT
001110  010037  000110'                 MOV     R0,MX1
001114  005726                          TST     (SP)+
001116  005700                          TST     R0                              ; MX1                   4198
001120  002767                          BLT     8$
001122  013700  000000G                 MOV     T.ADDR,R0                       ;                       4200
001126  005260  000062                  INC     62(R0)
001132  000207                          RTS     PC                              ;                       4080

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


;    4203  1
```

```
1
1   GLOBAL ROUTINE DUPCOMMAND : NOVALUE =
1
1   !+
1   ! THIS ROUTINE IS CALLED BY DUP TO PROCESS COMMANDS.
1   ! THE COMMAND ENVOLOPES ARE FILLED IN DUP ROUTINES IN THE "MX1" INDEX.
1   ! WITH THE INDEX THIS ROUTINE SENDS THE COMMAND, WAITS FOR A
1   ! RESPONSES AND THEN PROCESSES THE RETURN PACKET.
1   !-
2   BEGIN
2   !PRINTX (DER13);
2
2   MSCP_PKT [.MX1, CREDITS] = 0;                    ! DUP DOES NOT USE THE CREDIT SYSTEM
2   MSCP_PKT [.MX1, CONNID] = CID_DUP;               ! MAKE PACKAGE EQUAL A DUP COMMAND
2   MSCP_PKT [.MX1, DK_NUM] = 0;                     ! DISK NUMBER (NOT APPLICABLE)
2
2   IF SEND (.MX1) EQLU FAILURE                      ! ATTEMPT SEND; IF CTLR IS OFFLINE
2   THEN
3           BEGIN
3           PUT_PKT (.MX1);
3           MX1 = -1;                                ! RETURN ENVELOPE TO POOL
3           CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;
3   !       PRINTF (DBM112);                         ! "DUP: PKT NOT AVAILABLE"  ZZZ
3           END
3
2   ELSE
2           do
3                   begin
3                   BREAK;                                   ! BREAK FOR ACT
3                   PROC_RETPKT ();                                  ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
3                   end
2           until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or       ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
2           (.EOP_FLAG eql true);                            ! or end of pass caused by error
1   END;

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

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

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

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


                                        .SBTTL  DUPIDLE MULTI-DRIVE TEST ROUTINES
000000  010146                  DUPIDLE::
                                        MOV     R1,-(SP)                        ;       4239
000002  013700  001156'                 MOV     DUOFF,R0                        ;       4247
000006  006300                          ASL     R0
000010  063700  000000G                 ADD     CST.ADDR,R0
000014  042760  040000  000020          BIC     #40000,20(R0)
000022  013746  000110'                 MOV     MX1,-(SP)                       ;       4249
000026  012746  000106                  MOV     #106,-(SP)
000032  004737  000000G                 JSR     PC,BL$MUL
000036  012760  000014  000006G         MOV     #14,MSCP.PKT+6(R0)
000044  112760  000001  000022G         MOVB    #1,MSCP.PKT+22(R0)              ;       4250
000052  005060  000024G                 CLR     MSCP.PKT+24(R0)                 ;       4251
000056  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)            ;       4252
000064  004737  016532'                 JSR     PC,DUPCOMMAND                   ;       4253
000070  013716  000000G         1$:     MOV     CCTLR,(SP)                      ;       4256
000074  004737  000000G                 JSR     PC,GET.PKT
```

B14


```
000100  010037  000110'                 MOV     R0,MX1
000104  010001                          MOV     R0,R1                   ; MX1.+                 4257
000106  002770                          BLT     1$
000110  013700  001156'                 MOV     DUOFF,R0                ;                       4259
000114  006300                          ASL     R0
000116  063700  000000G                 ADD     CST.ADDR,R0
000122  032760  020000  000020          BIT     #20000,20(R0)
000130  001432                          BEQ     3$
000132  010116                          MOV     R1,(SP)                 ;                       4262
000134  012746  000106                  MOV     #106,-(SP)
000140  004737  000000G                 JSR     PC,BL$MUL
000144  012760  000014  000006G         MOV     #14,MSCP.PKT+6(R0)
000152  112760  000006  000022G         MOVB    #6,MSCP.PKT+22(R0)      ;                       4263
000160  005060  000024G                 CLR     MSCP.PKT+24(R0)         ;                       4264
000164  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)    ;                       4265
000172  004737  016532'                 JSR     PC,DUPCOMMAND           ;                       4266
000176  013716  000000G         2$:     MOV     CCTLR,(SP)              ;                       4268
000202  004737  000000G                 JSR     PC,GET.PKT
000206  010037  000110'                 MOV     R0,MX1
000212  002771                          BLT     2$                      ;                       4269
000214  005726                          TST     (SP)+                   ;                       4261
000216  022626                  3$:     CMP     (SP)+,(SP)+             ;                       4246
000220  012601                          MOV     (SP)+,R1                ;                       4239
000222  000207                          RTS     PC

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

```
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 BLOCK 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

local
    RD_DISK : byte;      ! FLAG TO INDICATE WINCHESTER DISK SELECTED


if .CST_ADDR [.CUOFF, D_TYPE] eql RX_50
then
    RD_DISK = FALSE
else
    RD_DISK = TRUE;

MAD1 [LBN_L] = .BST [.L$LUN, LO_WRD];                          ! LOAD LBN INTO 1ST PACKET
MAD1 [LBN_H] = .BST [.L$LUN, HI_WRD];                          ! LOAD LBN INTO 1ST PACKET

if .MX2 geq 0                         ! IF 2 QIOS
then
    begin
    MAD2 [LBN_L] = .BST [.L$LUN, LO_WRD];                              ! LOAD LBN INTO 2ND PACKET
    MAD2 [LBN_H] = .BST [.L$LUN, HI_WRD];                      ! LOAD LBN INTO 2ND PACKET
    end;

if BIT_TST (SWP_FLAGS, SWF_BLK)         ! IF RANDOM BLOCK MODE
then
    begin
    if  NOT ((.RD_DISK) and
      (((.RANDOM [0] and %o'077777') mod (99)) lequ 33))                    ! REDUCES SEEKS TO 66% OF TRANFERS on winchesters

    then
        begin                                                               ! NOTE IF NOT CALLED THE LBN REMAINS THE SAME

        if .CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .CST_ADDR [.CUOFF + 4, D_END1]
        then                                                                ! if upper word of beg trk and ending trk same

                BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0] +             ! select low lbn from random number WINDOW

                MODULAS (.CST_ADDR [.CUOFF + 1, D_BEG0], .CST_ADDR [.CUOFF + 3, D_END0])


        else                                                                ! if upper word of beg trk and end trk different
```

```
;   4325  5                     begin
;   4326  5                     BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1] +              ! select upper lbn from window
;   4327  5                     MODULAS (.CST_ADDR [.CUOFF + 2, D_BEG1], .CST_ADDR [.CUOFF + 4, D_END1]);
;   4328  5
;   4329  5                     if .BST [.L$LUN, HI_WRD] eql .CST_ADDR [.CUOFF + 4, D_END1]          ! IF UPPER WORD EQUALS HI LIMIT BE S
URE LOWER
;   4330  5                     then .BST [.L$LUN, LO_WRD] = MODULAS (0, .CST_ADDR [.CUOFF + 3, D_END0]); ! WORD DOES NOT PASS HI LIMIT
;   4331  5
;   4332  5                     if .BST [.L$LUN, HI_WRD] eql .CST_ADDR [.CUOFF + 2, D_BEG1]          ! if upper word equal lo limit make
sure lower
;   4333  5                     then BST [.L$LUN, LO_WRD] = %O'177777' - MODULAS (.CST_ADDR [.CUOFF + 1, D_BEG0], %O'177777');
;   4334  5                                                                                           ! word is above lo limit
;   4335  5
;   4336  5                     if .BST [.L$LUN, HI_WRD] gtr .CST_ADDR [.CUOFF + 2, D_BEG1] and
;   4337  5                        .BST [.L$LUN, HI_WRD] lss .CST_ADDR [.CUOFF + 4, D_END1]          ! if neither of the above then any n
umber is good
;   4338  5                     then BST [.L$LUN, LO_WRD] = .RANDOM [5];
;   4339  4                     end;
;   4340  3                 end;
;   4341  3             end
;   4342  2     else
;   4343  3         begin                                       ! ELSE - SEQUENTIAL LBN MODE
;   4344  4         if (.TRK_SGN [.L$LUN] geq 1)
;   4345  3         then                                        ! if positive track direction add one to multiword
;   4346  4             (if .BST [.L$LUN, LO_WRD] eqlu %o'177777'
;   4347  4             then
;   4348  5                 begin
;   4349  5                 BST [.L$LUN, LO_WRD] = 0;
;   4350  5                 BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] + 1;
;   4351  5                 end
;   4352  4             else
;   4353  4                 BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] + 1)
;   4354  3         else                                        ! if negative track direction subtract one from multiword
;   4355  3             if .BST [.L$LUN, LO_WRD] eqlu %o'0'
;   4356  3             then
;   4357  4                 begin
;   4358  4                 BST [.L$LUN, LO_WRD] = %o'177777';
;   4359  4                 BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] - 1;
;   4360  4                 end
;   4361  3             else
;   4362  3                 BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] - 1;
;   4363  3
;   4364  3
;   4365  3         if  .BST [.L$LUN, LO_WRD] gequ (.CST_ADDR [.CUOFF + 3, D_END0]) and ! if hi limit then change direction
;   4366  4             .BST [.L$LUN, HI_WRD] gequ (.CST_ADDR [.CUOFF + 4, D_END1])
;   4367  3         then TRK_SGN [.L$LUN] =  -1;
;   4368  3
;   4369  3         if  .BST [.L$LUN, LO_WRD] lequ (.CST_ADDR [.CUOFF + 1, D_BEG0] + 1)  and    ! if low limit then change direction
;   4370  4             .BST [.L$LUN, HI_WRD] lequ (.CST_ADDR [.CUOFF + 2, D_BEG1])
;   4371  3         then TRK_SGN [.L$LUN] = 1;
;   4372  2         end;
;   4373  1     end;                                                    ! ROUTINE QIO_LBN


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

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

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

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

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

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


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

```
000034  001002                          BNE     1$                      ;                       4396
000036  012703  000020                  MOV     #20,R3                  ; *,SIZE                4398
000042  013700  000000G         1$:     MOV     CUOFF,R0                ;                       4400
000046  006300                          ASL     R0
000050  063700  000000G                 ADD     CST.ADDR,R0
000054  013701  000114'                 MOV     MAD1,R1
000060  026061  000010  000050          CMP     10(R0),50(R1)
000066  101403                          BLOS    2$
000070  012702  177777                  MOV     #-1,R2                  ; *,BLOCKS.LEFT         4401
000074  000413                          BR      3$                      ;                       4400
000076  013700  000000G         2$:     MOV     CUOFF,R0                ;                       4402
000102  006300                          ASL     R0
000104  063700  000000G                 ADD     CST.ADDR,R0
000110  016000  000006                  MOV     6(R0),R0
000114  166100  000046                  SUB     46(R1),R0
000120  010002                          MOV     R0,R2                   ; *,BLOCKS.LEFT
000122  005202                          INC     R2                      ; BLOCKS.LEFT
000124  010316                  3$:     MOV     R3,(SP)                 ; SIZE,*                4404
000126  062716  000777                  ADD     #777,(SP)
000132  012746  001000                  MOV     #1000,-(SP)
000136  004737  000000G                 JSR     PC,BL$DIV
000142  005726                          TST     (SP)+
000144  020002                          CMP     R0,R2                   ; *,BLOCKS.LEFT
000146  101405                          BLOS    4$
000150  010200                          MOV     R2,R0                   ; BLOCKS.LEFT,*         4406
000152  000300                          SWAB    R0
000154  105000                          CLRB    R0
000156  006300                          ASL     R0
000160  010003                          MOV     R0,R3                   ; *,SIZE
000162  010361  000026          4$:     MOV     R3,26(R1)               ; SIZE,*                4408
000166  005737  000112'                 TST     MX2                     ;                       4410
000172  002404                          BLT     5$
000174  013700  000116'                 MOV     MAD2,R0                 ;                       4412
000200  010360  000026                  MOV     R3,26(R0)               ; SIZE,*
000204  022626                  5$:     CMP     (SP)+,(SP)+             ;                       4388
000206  000207                          RTS     PC                      ;                       4375

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

```
;  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$LUN - CURRENT (DRS) UNIT NUMBER
;  1   !-
;  1
;  2       begin
;  2
;  2       local
;  2           DP_NUM : word,                              ! DATA PATTERN NUMBER SELECTED
;  2           DP_ADDR,                                    ! ADDR OF DATA PATTERN (LENGTH)
;  2           IOB_ADDR,                                   ! I/O BUFFER ADDRESS (DESTINATION)
;  2           SRC_ADDR,                                   ! WORKING SOURCE ADDRESS
;  2           COUNT : word;                               ! NO. OF WORDS IN DATA PATTERN
;  2
;  3       if BIT_TST (SWP_FLAGS, SWF_UDP)                 ! IF USER DEFINED A DATA PATTERN
;  2       then
;  2           DP_ADDR = SWP_UCNT                          ! SELECT IT
;  2       else
;  3           begin
;  3
;  3           if .SWP_DPAT neq 0                          ! IF USER SELECTED A PRE-DEFINED DATA PATTERN
;  3           then
;  3               DP_NUM = .SWP_DPAT                      ! SELECT IT
;  3           else
;  4               begin
;  4               DP_NUM = .DPST [.L$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;
```

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


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

```
000052  105210                                  INCB    (R0)                            ;                               4461
000054  121027  000025                          CMPB    (R0),#25                        ;                               4463
000060  101402                                  BLOS    3$
000062  112710  000001                          MOVB    #1,(R0)                         ;                               4465
000066  010200                          3$:     MOV     R2,R0                           ; DP.NUM,*                      4469
000070  006300                                  ASL     R0
000072  016001  001074'                         MOV     DPA.TBL-2(R0),R1                ; *,DP.ADDR
000076  020227  000021                          CMP     R2,#21                          ; DP.NUM,*                      4471
000102  103413                                  BLO     5$
000104  013700  000114'                         MOV     MAD1,R0                         ;                               4476
000110  006002                                  ROR     R2                              ; DP.NUM                        4474
000112  103004                                  BCC     4$
000114  016061  000046  000002                  MOV     46(R0),2(R1)                    ; *,*(DP.ADDR)                  4476
000122  000403                                  BR      5$                              ;                               4474
000124  016061  000046  000004          4$:     MOV     46(R0),4(R1)                    ; *,*(DP.ADDR)                  4478
000132  013700  000114'                 5$:     MOV     MAD1,R0                         ;                               4482
000136  016004  000032                          MOV     32(R0),R4                       ; *,IOB.ADDR
000142  011103                                  MOV     (R1),R3                         ; DP.ADDR,COUNT                 4483
000144  012705  000002                          MOV     #2,R5                           ;                               4484
000150  060105                                  ADD     R1,R5                           ; DP.ADDR,*
000152  010502                                  MOV     R5,R2                           ; *,SRC.ADDR
000154  016046  000026                          MOV     26(R0),-(SP)                    ;                               4486
000160  005216                                  INC     (SP)
000162  012746  000002                          MOV     #2,-(SP)
000166  004737  000000G                         JSR     PC,BL$DIV
000172  010066  000004                          MOV     R0,4(SP)
000176  005000                                  CLR     R0                              ; N
000200  000405                                  BR      7$
000202  012224                          6$:     MOV     (R2)+,(R4)+                     ; SRC.ADDR,IOB.ADDR             4488
000204  005303                                  DEC     R3                              ; COUNT                         4491
000206  001002                                  BNE     7$                              ;                               4493
000210  011103                                  MOV     (R1),R3                         ; DP.ADDR,COUNT                 4496
000212  010502                                  MOV     R5,R2                           ; *,SRC.ADDR                    4497
000214  005200                          7$:     INC     R0                              ; N                             4486
000216  020066  000004                          CMP     R0,4(SP)                        ; N,*
000222  003767                                  BLE     6$
000224  062706  000006                          ADD     #6,SP                           ;                               4415
000230  000207                                  RTS     PC

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


;     4503  1
;     4504  1
```

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


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

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

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

```
;  1   !+
;  1   GLOBAL ROUTINE DIO_RETPKT : NOVALUE =
;  1
;  1   !+
;  1   !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL DUP I/O TRANSFER
;  1   !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;  1   !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS.
;  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   !               DUOFF - CST OFFSET FOR THE CURRENT UNIT
;  1   !               L$LUN - CURRENT UNIT NUMBER
;  1   !               CCTLR - CURRENT CONTROLLER NUMBER
;  1   !
;  1   !       IMPLICIT OUTPUTS
;  1   !               CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] - IF THIS BIT SET NO DUP EXERCISER
;  1   !
;  1   !-
;  2   BEGIN
;  2
;  2   LOCAL   FLAG : BYTE INITIAL(BYTE(TRUE)),
;  2           SUM2 : WORD,
;  2           SUM : WORD;                          ! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
;  2   !PRINTX (DER18);
;  2
;  2   IF .RP_ADDR [STATUS] NEQU ST_SUC                          ! IF STATUS CODE INDICATES ERROR
;  2   THEN
;  3           BEGIN
;  3           CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;          ! SET DUP ERROR FLAG
;  3           HARD_ERROR ();
;  3           IF .RP_ADDR [ENDCOD] EQLU (OP_ELP + OP_END) OR  ! IF ENDCODE IS EXECUTE LOCAL PROGRAM
;  4              .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END)     !           OR GET DUST STATUS
;  4           THEN    BEGIN
;  4                   CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;          ! TURN OFF DUP EXERCISR
;  3                   END;
;  3           END
;  2   ELSE                                      ! ELSE - I/O WAS SUCCESSFUL
;  3       BEGIN
;  3
;  4       IF .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END) ! IF ENDCODE IS GET DUST STATUS
;  3       THEN
;  4           BEGIN
;  4           IF .RP_ADDR [9,11,1,0] EQL 1
;  4                   THEN CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = ACTIVE          ! CONTROLLER IN AN ACTIVE STAE
;  4                   ELSE CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = IDLE;           ! CONTROLLER IN AN IDLE STATE
;  4           IF .RP_ADDR [9,9,1,0] NEQ 1 THEN      ! TEST TO SEE IF CONTROLLER LOCAL PROGRAMS(PG 18 OF DUP DOC)
;  5                   BEGIN
;  5                   HARD_ERROR ();
;  5                   CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;                 ! TURN OFF DUP EXERCISR
;  4                   END;
;  3           END;
```

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


                                            .SBTTL  DIO.RETPKT MULTI-DRIVE TEST ROUTINES
000000 010146                       DIO.RETPKT::
                                            MOV     R1,-(SP)                ;                   4542
000002 112700 000001                        MOVB    #1,R0                   ; *.FLAG            4561
000006 013701 000000G                       MOV     RP.ADDR,R1              ;                   4568
000012 005761 000016                        TST     16(R1)
000016 001435                               BEQ     2$
000020 013700 001156'                       MOV     DUOFF,R0                ;                   4571
000024 006300                               ASL     R0
000026 063700 000000G                       ADD     CST.ADDR,R0
000032 052760 040000 000020                 BIS     #40000,20(R0)
000040 004737 000000V                       JSR     PC,HARD.ERROR           ;                   4572
000044 013700 000000G                       MOV     RP.ADDR,R0              ;                   4573
000050 126027 000014 000203                 CMPB    14(R0),#203
000056 001404                               BEQ     1$
000060 126027 000014 000201                 CMPB    14(R0),#201             ;                   4574
000066 001112                               BNE     6$
000070 013700 001156'               1$:     MOV     DUOFF,R0                ;                   4576
000074 006300                               ASL     R0
000076 063700 000000G                       ADD     CST.ADDR,R0
000102 052760 100000 000020                 BIS     #100000,20(R0)
000110 000501                               BR      6$                      ;                   4568
000112 126127 000014 000201         2$:     CMPB    14(R1),#201             ;                   4582
000120 001036                               BNE     5$
000122 013700 001156'                       MOV     DUOFF,R0                ;                   4586
000126 006300                               ASL     R0
000130 063700 000000G                       ADD     CST.ADDR,R0
000134 032761 004000 000022                 BIT     #4000,22(R1)            ;                   4585
000142 001404                               BEQ     3$
000144 052760 020000 000020                 BIS     #20000,20(R0)           ;                   4586
000152 000403                               BR      4$                      ;                   4585
000154 042760 020000 000020         3$:     BIC     #20000,20(R0)           ;                   4587
000162 032761 001000 000022         4$:     BIT     #1000,22(R1)            ;                   4588
000170 001012                               BNE     5$
000172 004737 000000V                       JSR     PC,HARD.ERROR           ;                   4590
000176 013700 001156'                       MOV     DUOFF,R0                ;                   4591
000202 006300                               ASL     R0
000204 063700 000000G                       ADD     CST.ADDR,R0
000210 052760 100000 000020                 BIS     #100000,20(R0)
```

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

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


;   4606  1
```

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


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


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

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

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

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


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

```
;  1    GLOBAL routine IO_RETPKT : novalue =
;  1
;  1    !+
;  1    !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
;  1    !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;  1    !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
;  1    !       PERFORMING HOST WRITE-COMPARES IF REQUIRED.
;  1    !
;  1    !       IMPLICIT INPUTS:
;  1    !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  1    !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  1    !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  1    !               CCTLR - CURRENT CONTROLLER NUMBER
;  1    !               L$LUN - CURRENT UNIT NUMBER
;  1    !-
;  1
;  2        begin
;  2
;  2        local
;  2            FLAG : byte initial (byte (TRUE));
;  2
;  2        FSET_UPAR ();                                   ! FIND UNIT'S ENTRY IS CST AND SET UP UNIT-RELATED DATA
;  2        ST_CODE = .RP_ADDR [STSCOD];                    ! GET STATUS CODE FROM RETPKT
;  2        SB_CODE = .RP_ADDR [SUBCOD];                    ! GET SUB-CODE, IF ANY
;  2
;  3        if (.ST_CODE neq ST_SUC)                        ! IF STATUS CODE INDICATES ERROR
;  2        then
;  3            begin
;  3            HARD_ERROR ();                              ! UPDATE ERROR COUNT
;  3            COMPARE_DATA = FALSE;                       ! NO POINT IN DOING HOST COMPARES ON ERRORS
;  3
;  3            if (.ST_CODE neq ST_OFL) and                ! DROP UNIT IF ERROR COUNTS EXCEEDS LIMIT
;  3               (.ST_CODE neq ST_AVL) and
;  4               (.T_ADDR [ERR_HARD] gequ .SWP_ERROR)
;  3            then
;  4                begin
;  4                DUR [.L$LUN] = DU_HERR;                 ! LOAD REASON FOR DROPPING UNIT
;  4                DODU (.L$LUN);                          ! DROP UNIT
;  3                end;
;  3
;  3            end
;  2        else                                            ! IF I/O WAS SUCCESSFUL
;  3            begin
;  3            UPD_IO_TALLY ();                            ! UPDATE I/O TALLY (STATISTICS)
;  3
;  4            if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)
;  3            then
;  3                COMPARE_DATA = TRUE;                    ! HOST COMPARES MAY BE ALLOWED IF NO FURTHER ERRORS
;  3
;  3            if (BIT_TST (SWP_FLAGS, SWF_HWC)) and       ! IF HOST IS DOING WRITE-COMPARES
;  4               (.COMPARE_DATA)
;  3            then
;  3                FLAG = HOST_WRT_CHK ();                 ! SAVE I/O PACKET OR DO WRITE-CHECK
```

H15


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

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

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

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

```
;  1     GLOBAL routine FSET_UPAR : novalue =
;  1
;  1     !+
;  1     !     THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
;  1     !     CONTROLLER STATUS TABLE (CST) FOR THE DISK ADDRESS WHICH IS
;  1     !     CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
;  1     !     CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT-RELATED
;  1     !     DATA PARAMETERS.
;  1     !
;  1     !     IMPLICIT INPUTS:
;  1     !             RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;  1     !             CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  1     !-
;  1
;  2         begin
;  2
;  2         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;  2
;  2             if .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .RP_ADDR [DISK]         ! IF RETPKT UNIT MATCHES CST ENTRY
;  2             then
;  3                 begin
;  3                 SET_UPAR (.OFFSET);                                                  ! SET UP UNIT-RELATED DATA
;  3                 return;                                                              ! DONE
;  2                 end;
;  2
;  2         PRINTF (DBM19, .RP_ADDR [DISK], .CCTLR);                                     ! "CAN'T FIND DISK XXX IN CST X"
;  1         end;                                                                         ! ROUTINE FSET_UPAR


                                        .SBTTL  FSET.UPAR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G         FSET.UPAR::
                                        JSR     R1,$SAVE4               ;                       4707
000004  012702  000003                  MOV     #3,R2                   ; *,OFFSET              4723
000010  010201                  1$:     MOV     R2,R1                   ; OFFSET,*              4725
000012  006301                          ASL     R1
000014  063701  000000G                 ADD     CST.ADDR,R1
000020  013700  000000G                 MOV     RP.ADDR,R0
000024  016004  000010                  MOV     10(R0),R4
000030  111103                          MOVB    (R1),R3
000032  042703  177760                  BIC     #177760,R3
000036  020304                          CMP     R3,R4
000040  001005                          BNE     2$
000042  010246                          MOV     R2,-(SP)                ; OFFSET,*              4728
000044  004737  000000G                 JSR     PC,SET.UPAR
000050  005726                          TST     (SP)+                   ;                       4729
000052  000207                          RTS     PC                      ;                       4727
000054  062702  000012          2$:     ADD     #12,R2                  ; *,OFFSET              4723
000060  020227  000041                  CMP     R2,#41                  ; OFFSET,*
000064  003751                          BLE     1$
000066  013746  000000G                 MOV     CCTLR,-(SP)             ;                       4732
000072  013700  000000G                 MOV     RP.ADDR,R0
000076  016046  000010                  MOV     10(R0),-(SP)
000102  012746  000000G                 MOV     #DBM19,-(SP)
```

K15


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

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

```
;  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    ZZZ
;  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
```

```
;  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  3
;  3                        if .APT_MODE
;  3                        then
;  3                            ERR_HRD_RTNE_APT (32)
;  3                        else
;  3                            ERR_HRD_RTNE (32);
;  3
;  2                        end;
;  2
;  3        [ST_OFL] :      begin                                                   ! OFFLINE
;  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  3
;  3                        if .APT_MODE
;  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                        ERRDF (18, EGD_18, EMS_18);                             !
;  3                        DUR [.L$LUN] = DU_DFATAL;                               ! DEVICE FATAL ERROR
;  3                        DODU (.L$LUN);                                          ! DROP UNIT
;  2                        end;
;  2
;  3        [ST_AVL]:       begin                                                   ! WENT TO AVAILABLE STATE
;  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  3
;  3                        if .APT_MODE
;  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                        ERRDF (24, EGD_18, EMS_24);                             !
;  3                        DUR [.L$LUN] = DU_DFATAL;                               ! DEVICE FATAL ERROR
;  3                        DODU (.L$LUN);                                          ! DROP UNIT
;  2                        end;
;  2
;  3        [ST_MFE]:       begin                                                   ! MEDIA FORMAT ERROR
;  3                        T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1;
;  3
;  3                        if .APT_MODE
;  3                        then
;  3                            ERR_HRD_RTNE_APT (35)
;  3                        else
;  3                            ERR_HRD_RTNE (35);
;  3
;  2                        end;
;  2
;  3        [ST_WPT]:       begin                                                   ! DEVICE WRITE PROTECTED
;  3
```

```
;  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;                    ! BLOCK WITH "FORCED ERROR" FOUND
;  4                              FER_LBN = .RP_ADDR [LBN_LO];            !
;  4                              FER_BC = .RP_ADDR [CBCNT_LO];           !
;  3                              end;
;  3
;  3                          if .APT_MODE
;  3                          then
;  3                              ERR_HRD_RTNE_APT (38)
;  3                          else
;  3                              ERR_HRD_RTNE (38);
;  3
;  2                          end;
;  2
;  3          [ST_HST]:       begin                                           ! HOST ACCESS ERROR
;  3                          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;

        if .RP_ADDR [CONID] EQL CID_DUP                                        !FOR DUP ERRORS                  ZZZ
        OR .D_FAIL EQL 1                                                       !EVEN IF UNRECOGNIZABLE AS SUCH  ZZZ
        THEN                                                                   !                                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 [ERR_HRD_HST] = .T_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;
                                                [OTHERWISE] : begin
                                                        ERR_HRD_RTNE (66);      ! DUP UNKNOWN STATUS CODE    !ZZZ
                                                        C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HR
D] + 1;
                                                        end;
                                                tes;
                                        end;
```

```
;     3                        end;
;     2        [%o'1']      : begin
;     3                  ERR_HRD_RTNE (67);       ! INVALID COMMAND            !ZZZ
;     3                  T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;     2                  end;
;     3        [%o'2']      : begin
;     3                  ERR_HRD_RTNE (68);       ! NO REGION AVAILABLE        !ZZZ
;     3                  T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;     2                  end;
;     3        [%o'3']     : begin
;     3                  ERR_HRD_RTNE (69);                 ! NO REGION SUITABLE        !ZZZ
;     3                  T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;     2                  end;
;     3        [%o'4']      : begin
;     3                  ERR_HRD_RTNE (70);       ! PROGRAM NOT KNOWN          !ZZZ
;     3                  T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;     2                  end;
;     3        [%o'5']      : begin
;     3                  ERR_HRD_RTNE (71);                ! LOAD FAILURE              !ZZZ
;     3                  T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;     2                  end;
;     3        [%o'6']      : begin
;     3                  ERR_HRD_RTNE (72);                 ! STANDALONE                !ZZZ
;     3                  T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;     2                  end;
;     3        [OTHERWISE] : begin
;     3                  ERR_HRD_RTNE (73);                 ! DUP UNKNOWN STATUS CODE   !ZZZ
;     3                  C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;     2                  end;
;     2        TES;
;     2
;     1    end;                                                       ! ROUTINE HARD_ERROR

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

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

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

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

```
001124  000420                          45$:    BR      49$
001126  013700  000000G                 46$:    MOV     CCTLR,R0                        ;       4941
001132  006300                                  ASL     R0
001134  105260  000000G                         INCB    C.ERR.TBL(R0)
001140  032737  000001  001162'                 BIT     #1,APT.MODE                     ;       4943
001146  001405                                  BEQ     48$
001150  012746  000055                          MOV     #55,-(SP)                       ;       4945
001154  004737  000000V                 47$:    JSR     PC,ERR.HRD.RTNE.APT
001160  000404                                  BR      50$                             ;       4943
001162  012746  000055                  48$:    MOV     #55,-(SP)                       ;       4947
001166  004737  000000V                 49$:    JSR     PC,ERR.HRD.RTNE
001172  005726                          50$:    TST     (SP)+                           ;       4940
001174  013700  000000G                 51$:    MOV     RP.ADDR,R0                      ;       4953
001200  126027  000003  000002                  CMPB    3(R0),#2
001206  001404                                  BEQ     52$
001210  123727  000000G 000001                  CMPB    D.FAIL,#1                       ;       4954
001216  001160                                  BNE     69$
001220  116001  000016                  52$:    MOVB    16(R0),R1                       ;       4957
001224  042701  177740                          BIC     #177740,R1
001230  001067                                  BNE     59$                             ;       4959
001232  126027  000014  000201                  CMPB    14(R0),#201                     ;       4960
001240  001015                                  BNE     54$
001242  032760  001000  000022                  BIT     #1000,22(R0)                    ;       4961
001250  001011                                  BNE     54$
001252  012746  000074                          MOV     #74,-(SP)                       ;       4964
001256  004737  000000V                 53$:    JSR     PC,ERR.HRD.RTNE
001262  013700  000000G                         MOV     T.ADDR,R0                       ;       4965
001266  105260  000050                          INCB    50(R0)
001272  000531                                  BR      68$                             ;       4963
001274  013700  000000G                 54$:    MOV     DUPPKT,R0                       ;       4969
001300  042700  007777                          BIC     #7777,R0
001304  020027  050000                          CMP     R0,#50000
001310  001123                                  BNE     69$
001312  013701  000000G                         MOV     DUPPKT,R1                       ;       4975
001316  042701  170000                          BIC     #170000,R1
001322  020127  000001                          CMP     R1,#1                           ;       4977
001326  001003                                  BNE     55$
001330  012746  000076                          MOV     #76,-(SP)                       ;       4978
001334  000470                                  BR      65$
001336  020127  000002                  55$:    CMP     R1,#2                           ;       4981
001342  001003                                  BNE     56$
001344  012746  000077                          MOV     #77,-(SP)                       ;       4982
001350  000462                                  BR      65$
001352  020127  000003                  56$:    CMP     R1,#3                           ;       4985
001356  001003                                  BNE     57$
001360  012746  000100                          MOV     #100,-(SP)                      ;       4986
001364  000734                                  BR      53$
001366  020127  000004                  57$:    CMP     R1,#4                           ;       4989
001372  001003                                  BNE     58$
001374  012746  000101                          MOV     #101,-(SP)                      ;       4990
001400  000446                                  BR      65$
001402  012746  000102                  58$:    MOV     #102,-(SP)                      ;       4994
001406  000454                                  BR      67$
```

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

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


;   5032  1
```

```
;  1   GLOBAL routine UPD_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
;  3       if .RP_ADDR [ENDCOD] eql (OP_RD or OP_END)                         ! IF ENDCODE IS READ
;  2       then
;  3           begin
;  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];
;  3           T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + .RP_ADDR [BCNT_LO];
;  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               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];
;  3               T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + .RP_ADDR [BCNT_LO];
;  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  !
;  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      ! 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,$SAVE2                    ;                                   5033
```

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

```
000312  004737  000000V         8$:     JSR     PC,ROUND.OUTPUT                 ;               5132
000316  000207                          RTS     PC                              ;               5033

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

```
;  1   GLOBAL routine OVF_CHK (ADDR) : novalue =
;  1   !+
;  1   !        THIS ROUTINE IS CALLED FROM UPD_IO_TALLY TO CHECK FOR OVERFLOW IN
;  1   !        CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
;  1   !        LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
;  1   !        EXCEEDING 1000. IF TRUE, THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
;  1   !        THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IN INCREMENTED.
;  1   !
;  1   !        INPUTS:
;  1   !                ADDR - ADDRESS OF THE BYTES_READ_LO OR BYTES_WRIT_LO FIELD FOR
;  1   !                       THE CURRENT UNIT (SEE STATISTIC TABLE (TALLY) LAYOUT)
;  1   !-
;  2       begin
;  2
;  2       while ..ADDR gequ 1000 do                       ! IF LO-ORDER OVERFLOW
;  3           begin
;  3           .ADDR = ..ADDR - 1000;                      ! SUBTRACT 1000
;  3           (.ADDR + 2) = .(.ADDR + 2) + 1;             ! INCR HI-ORDER
;  2           end;
;  2
;  2       if .(.ADDR + 2) gequ 1000                       ! IF HI-ORDER OVERFLOW
;  2       then
;  3           begin
;  3           (.ADDR + 2) = .(.ADDR + 2) - 1000;          ! SUBTRACT 1000
;  3           (.ADDR + 4) = .(.ADDR + 4) + 1;             ! INCREMENT MBYTES
;  2           end;
;  2
;  1       end;                                            ! ROUTINE OVF_CHK


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

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

```
;  1     GLOBAL routine ROUND_OUTPUT : novalue =
;  1
;  1     !+
;  1     !       THIS ROUTINE ROUNDS THE TOTALS TO FIT PRINT POSITIONS.
;  1     !-
;  1
;  2          begin
;  2
;  2          if .T_ADDR [TOT_READS_HI] gtru 9999
;  2          then
;  3              begin
;  3
;  3              if .T_ADDR [TOT_READS_LO] lssu 999
;  3              then
;  4                  begin
;  4                  T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 1;
;  4                  T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1000;
;  3                  end;
;  3
;  3              T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] - 999;
;  3              T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 9999;
;  2              end;
;  2
;  2          if .T_ADDR [TOT_WRITES_HI] gtru 9999
;  2          then
;  3              begin
;  3
;  3              if .T_ADDR [TOT_WRITES_LO] lssu 999
;  3              then
;  4                  begin
;  4                  T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 1;
;  4                  T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1000;
;  3                  end;
;  3
;  3              T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] - 999;
;  3              T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 9999;
;  2              end;
;  2
;  2          if .T_ADDR [MTOT_BYT_RED] gtru 999
;  2          then
;  3              begin
;  3
;  3              if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;  3              then
;  4                  begin
;  4                  T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;  4                  T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;  3                  end;
;  3
;  3              if .T_ADDR [TOT_BYT_RED_LO] lssu 999
;  3              then
;  4                  begin
;  4                  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                       ;       5162
000004  013700  000000G                 MOV     T.ADDR,R0                       ;       5170
000010  012702  000020                  MOV     #20,R2
```

```
000014  060002                       ADD     R0,R2
000016  021227  023417               CMP     (R2),#23417
000022  101415                       BLOS    2$
000024  012701  000016               MOV     #16,R1                 ;        5174
000030  060001                       ADD     R0,R1
000032  021127  001747               CMP     (R1),#1747
000036  103003                       BHIS    1$
000040  005312                       DEC     (R2)                   ;        5177
000042  062711  001750               ADD     #1750,(R1)             ;        5178
000046  162711  001747       1$:     SUB     #1747,(R1)             ;        5181
000052  162712  023417               SUB     #23417,(R2)            ;        5182
000056  012702  000026       2$:     MOV     #26,R2                 ;        5185
000062  060002                       ADD     R0,R2
000064  021227  023417               CMP     (R2),#23417
000070  101415                       BLOS    4$
000072  012701  000024               MOV     #24,R1                 ;        5189
000076  060001                       ADD     R0,R1
000100  021127  001747               CMP     (R1),#1747
000104  103003                       BHIS    3$
000106  005312                       DEC     (R2)                   ;        5192
000110  062711  001750               ADD     #1750,(R1)             ;        5193
000114  162711  001747       3$:     SUB     #1747,(R1)             ;        5196
000120  162712  023417               SUB     #23417,(R2)            ;        5197
000124  012703  000036       4$:     MOV     #36,R3                 ;        5200
000130  060003                       ADD     R0,R3
000132  021327  001747               CMP     (R3),#1747
000136  101436                       BLOS    7$
000140  012701  000034               MOV     #34,R1                 ;        5204
000144  060001                       ADD     R0,R1
000146  021127  001747               CMP     (R1),#1747
000152  103003                       BHIS    5$
000154  005313                       DEC     (R3)                   ;        5207
000156  062711  001750               ADD     #1750,(R1)             ;        5208
000162  012702  000032       5$:     MOV     #32,R2                 ;        5211
000166  060002                       ADD     R0,R2
000170  021227  001747               CMP     (R2),#1747
000174  103011                       BHIS    6$
000176  005311                       DEC     (R1)                   ;        5214
000200  062712  001750               ADD     #1750,(R2)             ;        5215
000204  021127  001747               CMP     (R1),#1747             ;        5217
000210  103003                       BHIS    6$
000212  005313                       DEC     (R3)                   ;        5220
000214  062711  001750               ADD     #1750,(R1)             ;        5221
000220  162712  001747       6$:     SUB     #1747,(R2)             ;        5225
000224  162711  001747               SUB     #1747,(R1)             ;        5226
000230  162713  001747               SUB     #1747,(R3)             ;        5227
000234  012702  000044       7$:     MOV     #44,R2                 ;        5230
000240  060002                       ADD     R0,R2
000242  021227  001747               CMP     (R2),#1747
000246  101435                       BLOS    10$
000250  012701  000042               MOV     #42,R1                 ;        5234
000254  060001                       ADD     R0,R1
000256  021127  001747               CMP     (R1),#1747
```

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

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

```
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
```

```
;  4                  T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;
;  4                  T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  4
;  4                  if .APT_MODE
;  4                  then
;  4                      ERR_HRD_RTNE_APT (42)                    ! I/O REQUEST FAILED
;  4                  else
;  4                      ERR_HRD_RTNE (42);
;  4
;  4                  EMS_CMP (RETPKT + (.index * RP_LEN * 2));
;  4
;  4                  if .T_ADDR [ERR_HARD] gequ .SWP_ERROR
;  4                  then
;  5                      begin
;  5                      DUR [.L$LUN] = DU_HERR;                   ! IF ERROR COUNT EXCEEDED
;  5                      DODU (.L$LUN);                            ! DROP UNIT
;  4                      end;
;  4
;  4                  exitloop;                                     ! NO NEED TO CONTINUE
;  3                  end;                                          ! IF COMPARE ERROR
;  3
;  2              end;                                              ! IF ASSOCIATED WRITE RETPKT WAS FOUND
;  2
;  2      return (.FLAG);
;  1      end;                                                      ! ROUTINE HOST_WRT_CHK

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

```
000116  121112                    2$:   CMPB    (R1),(R2)                       ; BUFF1.BUFF2          5306
000120  001003                          BNE     3$
000122  005201                          INC     R1                              ; BUFF1                5309
000124  005202                          INC     R2                              ; BUFF2                5310
000126  000446                          BR      6$                              ;                      5306
000130  013700  000000G           3$:   MOV     T.ADDR,R0                       ;                      5314
000134  005260  000014                  INC     14(R0)
000140  105260  000051                  INCB    51(R0)                          ;                      5315
000144  032737  000001  001162'         BIT     #1,APT.MODE                     ;                      5317
000152  001405                          BEQ     4$
000154  012716  000052                  MOV     #52,(SP)                        ;                      5319
000160  004737  000000V                 JSR     PC,ERR.HRD.RTNE.APT
000164  000404                          BR      5$                              ;                      5317
000166  012716  000052            4$:   MOV     #52,(SP)                        ;                      5321
000172  004737  000000V                 JSR     PC,ERR.HRD.RTNE
000176  016616  000004            5$:   MOV     4(SP),(SP)                      ;                      5323
000202  062716  000000G                 ADD     #RETPKT,(SP)
000206  004737  000000G                 JSR     PC,EMS.CMP
000212  013700  000000G                 MOV     T.ADDR,R0                       ;                      5325
000216  026037  000014  000000G         CMP     14(R0),SWP.ERROR
000224  103412                          BLO     7$
000226  013700  000000G                 MOV     L$LUN,R0                        ;                      5328
000232  112760  000004  000000G         MOVB    #4,DUR(R0)
000240  104451                          TRAP    51                              ;                      5329
000242  000403                          BR      7$                              ;                      5313
000244  005203                    6$:   INC     R3                              ; I                    5304
000246  020304                          CMP     R3,R4                           ; I.COUNT
000250  003722                          BLE     2$
000252  022626                    7$:   CMP     (SP)+,(SP)+                     ;                      5298
000254  005000                    8$:   CLR     R0                              ;                      5337
000256  150500                          BISB    R5,R0                           ; FLAG,*
000260  005726                          TST     (SP)+                           ;                      5261
000262  000207                          RTS     PC

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

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


                                              .SBTTL  SWEEP MULTI-DRIVE TEST ROUTINES
000000  010146                        SWEEP:: MOV     R1,-(SP)                ;               5339
000002  013700  000000G                       MOV     RP.ADDR,R0              ;               5360
000006  116000  000014                        MOVB    14(R0),R0
000012  042700  177600                        BIC     #177600,R0
000016  020027  000041                        CMP     R0,#41
000022  001026                                BNE     1$
000024  032737  000040  000000G               BIT     #40,SWP.FLAGS           ;               5363
000032  001422                                BEQ     1$
000034  004737  000000V                       JSR     PC,RPS.REM              ;               5366
000040  010001                                MOV     R0,R1                   ; *,INDEX
000042  002416                                BLT     1$
000044  010146                                MOV     R1,-(SP)                ; INDEX,*       5369
000046  012746  000054                        MOV     #54,-(SP)
```

K1


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

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

```
;  1    GLOBAL routine RPS_REM =
;  1
;  1    !+
;  1    !       THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP_SAVE AREA FOR A
;  1    !       RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
;  1    !       CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
;  1    !       OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
;  1    !       RP_SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
;  1    !       OPERATION IS RETURNED TO THE CALLER.
;  1    !
;  1    !       IMPLICIT INPUTS:
;  1    !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  1    !
;  1    !       OUTPUTS:
;  1    !               INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
;  1    !                               A CRN WHICH IS ONE LESS THAN THE CURRENT
;  1    !-
;  1
;  2        begin
;  2
;  2        local
;  2            index : signed word initial (-1);                       ! ASSUME NOT FOUND
;  2
;  2        incr COUNT from 0 to RP_CNT - 1 do                          ! FOR EACH ENTRY IN RP_SAVE
;  2
;  2            if (.RP_USE [.COUNT] eql .CCTLR) and                    ! IF THIS IS A VALID RETPKT INDEX
;  3               (.RETPKT [.COUNT, ENDCOD] eql (OP_WRT or OP_END))
;  2            then
;  2
;  3                if ((.RETPKT [.COUNT, CRF_LO] eql (.RP_ADDR [CRF_LO] - 1)) and     ! IF CORRECT CRN
;  2                    (.RETPKT [.COUNT, CRF_HI] eql .RP_ADDR [CRF_HI])) or
;  3                   ((.RETPKT [.COUNT, CRF_HI] eql (.RP_ADDR [CRF_HI] - 1)) and
;  3                    (.RETPKT [.COUNT, CRF_LO] eql %o'177777') and
;  3                    (.RP_ADDR [CRF_LO] eql 0))
;  2                then
;  3                    begin
;  3                    index = .COUNT;                                 ! INDEX TO BE RETURNED
;  3                    exitloop;                                       ! DONE
;  2                    end;
;  2
;  2        return .index;
;  1        end;                                                        ! ROUTINE RPS_REM


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

```
000026  012746  000054                  MOV     #54,-(SP)
000032  004737  000000G                 JSR     PC,BL$MUL
000036  022626                          CMP     (SP)+,(SP)+
000040  126027  000014G 000242          CMPB    RETPKT+14(R0),#242
000046  001041                          BNE     4$
000050  010346                          MOV     R3,-(SP)                        ; COUNT,*                    5405
000052  012746  000054                  MOV     #54,-(SP)
000056  004737  000000G                 JSR     PC,BL$MUL
000062  022626                          CMP     (SP)+,(SP)+
000064  013701  000000G                 MOV     RP.ADDR,R1
000070  016102  000004                  MOV     4(R1),R2
000074  005302                          DEC     R2
000076  026002  000004G                 CMP     RETPKT+4(R0),R2
000102  001004                          BNE     2$
000104  026061  000006G 000006          CMP     RETPKT+6(R0),6(R1)              ;                            5406
000112  001415                          BEQ     3$
000114  016102  000006          2$:     MOV     6(R1),R2                        ;                            5407
000120  005302                          DEC     R2
000122  026002  000006G                 CMP     RETPKT+6(R0),R2
000126  001011                          BNE     4$
000130  026027  000004G 177777          CMP     RETPKT+4(R0),#-1                ;                            5408
000136  001005                          BNE     4$
000140  005761  000004                  TST     4(R1)                           ;                            5409
000144  001002                          BNE     4$
000146  010304                  3$:     MOV     R3,R4                           ; COUNT,INDEX                5412
000150  000404                          BR      5$                              ;                            5411
000152  005203                  4$:     INC     R3                              ; COUNT                      5399
000154  020327  000007                  CMP     R3,#7                           ; COUNT,*
000160  003714                          BLE     1$
000162  010400                  5$:     MOV     R4,R0                           ; INDEX,*                    5394
000164  000207                          RTS     PC                              ;                            5376

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

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


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

```
000056  042760  100000  000             BIC     #100000,2(R0)
000064  010146                          MOV     R1,-(SP)                ;                       5449
000066  012746  000006                  MOV     #6,-(SP)
000072  004737  000000G                 JSR     PC,DROP.CTLR
000076  013716  000000G                 MOV     RP.INDX,(SP)            ;                       5450
000102  004737  000000G                 JSR     PC,PUT.RETPKT
000106  022626                          CMP     (SP)+,(SP)+             ;                       5436
000110  012601                          MOV     (SP)+,R1                ;                       5418
000112  000207                          RTS     PC

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

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


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

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

D2


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


                                             .SBTTL  AZINT RDRX INTERRUPT SERVICE ROUTINES
000000  010146                      AZINT::  MOV     R1,-(SP)                      ;                  5465
000002  005746                               TST     -(SP)
000004  013701  000104'                      MOV     ICTLR,R1                      ;                  5477
000010  010146                               MOV     R1,-(SP)
000012  012746  000022                       MOV     #22,-(SP)
000016  004737  000000G                      JSR     PC,BL$MUL
000022  062700  000000G                      ADD     #DCT,R0
000026  010037  000100'                      MOV     R0,IDCT.ADDR
000032  010116                               MOV     R1,(SP)                       ;                  5478
000034  012746  000126                       MOV     #126,-(SP)
000040  004737  000000G                      JSR     PC,BL$MUL
000044  062700  000000G                      ADD     #CST,R0
000050  010037  000076'                      MOV     R0,ICST.ADDR
000054  011037  000000G                      MOV     (R0),IRDRX.ADDR               ; ICST.ADDR,*      5479
000060  010116                               MOV     R1,(SP)                       ;                  5480
000062  012746  000050                       MOV     #50,-(SP)
000066  004737  000000G                      JSR     PC,BL$MUL
000072  062700  000000'                      ADD     #COMM.AREA,R0
```

E2


```
000076  010037  000074'                 MOV     R0,ICOM.ADDR
000102  013701  000100'                 MOV     IDCT.ADDR,R1                    ;                       5481
000106  013700  000000G                 MOV     IRDRX.ADDR,R0
000112  016066  000002  000010          MOV     2(R0),10(SP)                    ; *,RC.REG
000120  016661  000010  000002          MOV     10(SP),2(R1)                    ; RC.REG,*
000126  032711  040000                  BIT     #40000,(R1)                     ; *,IDCT.ADDR           5483
000132  001016                          BNE     2$                              ;                       5465
000134  016601  000010                  MOV     10(SP),R1                       ;                       5487
000140  042701  077777                  BIC     #77777,R1
000144  020127  100000                  CMP     R1,#-100000
000150  001003                          BNE     1$
000152  004737  000000V                 JSR     PC,FATAL.ERROR                  ;                       5489
000156  000404                          BR      2$                              ;                       5487
000160  004737  000000V         1$:     JSR     PC,POLL.CRING                   ;                       5492
000164  004737  000000V                 JSR     PC,POLL.RRING                   ;                       5493
000170  062706  000012          2$:     ADD     #12,SP                          ;                       5465
000174  012601                          MOV     (SP)+,R1
000176  000207                          RTS     PC

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


;    5497  1
```

```
;  1
;  1     !+
;  1     GLOBAL ROUTINE DUP_RSP : NOVALUE =                                   !ZZZ
;  1
;  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     !-
;  2         begin
;  2
;  2         local
;  2             R_INDEX : signed word,
;  2             DEBUG,                                          !ZZZ
;  2             SRC_ADDR,
;  2             DST_ADDR,
;  2             R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;  2         !PRINTX (DER34);
;  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, CONNID] eql CID_DUP) and
;  3                ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END)
;  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_LO]);       ! UNKNOWN COMMAND REF. NUMBER
;  3             return;
;  2             end;
;  2
;  2         if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0        ! IF RETPKT IS NOT AVAILABLE
;  2         then
;  2             DEBUG = TRUE                                !TO SEE IF THIS PATH TAKEN        ZZZ
;  2     !       PRINTF (DBM112)                             ! "DUP-RSP: RETPKT NOT AVAILABLE"  ZZZ
;  2         else
;  3             begin
;  3             SRC_ADDR = .IPKT_ADDR + 6;                  ! SET UP COPY (SKIP OVER PKT DESC)
```

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


                                             .SBTTL  DUP.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G              DUP.RSP::
                                             JSR     R1,$SAVE3                        ;                   5500
000004  013701  000000G                      MOV     IPKT.ADDR,R1                     ;                   5525
000010  005002                               CLR     R2                               ; COUNT             5523
000012  010246                       1$:     MOV     R2,-(SP)                         ; COUNT,*           5525
000014  012746  000106                       MOV     #106,-(SP)
000020  004737  000000G                      JSR     PC,BL$MUL
000024  022626                               CMP     (SP)+,(SP)+
000026  026061  000012G 000012               CMP     MSCP.PKT+12(R0),12(R1)
000034  001024                               BNE     2$
000036  026061  000014G 000014               CMP     MSCP.PKT+14(R0),14(R1)           ;                   5526
000044  001020                               BNE     2$
000046  026011  000000G                      CMP     MSCP.PKT(R0),(R1)                ;                   5527
000052  001415                               BEQ     2$
000054  105760  000022G                      TSTB    MSCP.PKT+22(R0)                  ;                   5528
000060  100412                               BMI     2$
000062  126027  000011G 000002               CMPB    MSCP.PKT+11(R0),#2               ;                   5529
000070  001006                               BNE     2$
000072  105761  000022                       TSTB    22(R1)                           ;                   5530
000076  100003                               BPL     2$
000100  010237  000000G                      MOV     R2,P.INDEX                       ; COUNT,*           5533
000104  000406                               BR      3$                               ;                   5532
000106  005202                       2$:     INC     R2                               ; COUNT             5523
000110  020227  000013                       CMP     R2,#13                           ; COUNT,*
000114  003736                               BLE     1$
000116  005737  000000G                      TST     P.INDEX                          ;                   5537
000122  002013                       3$:     BGE     4$
000124  016146  000012                       MOV     12(R1),-(SP)                     ;                   5540
000130  012746  000000G                      MOV     #DBM108,-(SP)
000134  012746  000002                       MOV     #2,-(SP)
000140  010600                               MOV     SP,R0                            ; SP,*
000142  104417                               TRAP    17
```

```
000144  062706  000006             ADD     #6,SP                  ;                         5541
000150  000207                     RTS     PC                     ;                         5539
000152  013746  000104'    4$:     MOV     ICTLR,-(SP)            ;                         5544
000156  004737  000000G            JSR     PC,GET.RETPKT
000162  010001                     MOV     R0,R1                  ; *,R.INDEX
000164  005726                     TST     (SP)+
000166  005701                     TST     R1                     ; R.INDEX
000170  002003                     BGE     5$
000172  012700  000001             MOV     #1,R0                  ; *,DEBUG                 5546
000176  000425                     BR      7$                     ;                         5544
000200  013702  000000G    5$:     MOV     IPKT.ADDR,R2           ; *,SRC.ADDR              5550
000204  062702  000006             ADD     #6,R2                  ; *,SRC.ADDR
000210  010146                     MOV     R1,-(SP)               ; R.INDEX,*               5551
000212  012746  000054             MOV     #54,-(SP)
000216  004737  000000G            JSR     PC,BL$MUL
000222  062700  000000G            ADD     @RETPKT,R0
000226  010003                     MOV     R0,R3                  ; *,DST.ADDR
000230  012700  000026             MOV     #26,R0                 ; *,COUNT                 5553
000234  012223             6$:     MOV     (R2)+,(R3)+            ; SRC.ADDR,DST.ADDR       5555
000236  005300                     DEC     R0                     ; COUNT                   5553
000240  001375                     BNE     6$
000242  010116                     MOV     R1,(SP)                ; R.INDEX,*               5560
000244  004737  000000G            JSR     PC,IN.IODQ
000250  022626                     CMP     (SP)+,(SP)+            ;                         5549
000252  013700  000000G    7$:     MOV     P.INDEX,R0             ;                         5564
000256  002404                     BLT     8$
000260  010046                     MOV     R0,-(SP)               ;                         5566
000262  004737  000000G            JSR     PC,PUT.PKT
000266  005726                     TST     (SP)+
000270  000207             8$:     RTS     PC                     ;                         5500

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


;     5569  1
```

```
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
```

J2


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

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

K2


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


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

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

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

```
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


    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 ();

            [CID_DUP]  :        DUP_RSP  ();                             !ZZZ

            [otherwise] :       PRINTF (DBM20, .IPKT_ADDR [CONNID], .IRDRX_ADDR);
                                                                         ! "BAD CONN ID = XXXXX FROM XXXXXX"
            tes;


        IPKT_ADDR [MSGLEN] = MSG_LEN + 2;                                ! RE-INIT PKT FIELDS; MESSAGE LENGTH
        IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;                  ! ADVANCE TO HI ORDER WORD OF RING SLOT
        .IDCT_ADDR [RR_POLL] = .IPKT_ADDR [PKT_HI];                      ! RETURN SLOT TO CONTROLLER
        .IDCT_ADDR [RR_POLL] = ..IDCT_ADDR [RR_POLL] or ED_OWN or ED_FLAG; ! OWNERSHIP TOO
        IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;                  ! ADVANCE TO NEXT RRING SLOT


        if .IDCT_ADDR [RR_POLL] gtra .IDCT_ADDR [RR_END]                 ! IF BEYOND END OF RING

        then
```

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


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

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

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


;    5713  1
```

```
GLOBAL routine DISK_RSP : novalue =


!+
!       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH RESPONSE MESSAGE
!       WHICH HAS A CONNECTION ID INDICATING A DISK MSCP ORIGINATOR
!       (I.E., ALL EXCEPT DUP RESPONSES). ITS PURPOSE IS TO PASS
!       CONTROL TO THE APPROPRIATE ROUTINE BASED ON THE MESSAGE TYPE
!       FIELD (SEQUENTIAL, DATAGRAM, OR CREDIT NOTIFICATION).
!
!       IMPLICIT INPUTS:
!               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
!                               MESSAGE
!-



    selectoneu .IPKT_ADDR [MSGTYP] of


        set

        [MT_SEQ] :      SEQUEN ();

        [MT_DG] :       DATAGM ();

        [otherwise] :   PRINTF (DBM21, .IPKT_ADDR [MSGTYP]);    ! "MESSAGE TYPE XX RECEIVED"
        tes;


                                        .SBTTL  DISK.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  010146                  DISK.RSP::
                                        MOV     R1,-(SP)                ;       5716
000002  013700  000000G                 MOV     IPKT.ADDR,R0            ;       5733
000006  116001  000010                  MOVB    10(R0),R1
000012  006201                          ASR     R1
000014  006201                          ASR     R1
000016  006201                          ASR     R1
000020  006201                          ASR     R1
000022  042701  177760                  BIC     #177760,R1
000026  001003                          BNE     1$                      ;       5738
000030  004737  000000V                 JSR     PC,SEQUEN
000034  000417                          BR      3$                      ;       5733
000036  020127  000001          1$:     CMP     R1,#1                   ;       5740
000042  001003                          BNE     2$
000044  004737  000000V                 JSR     PC,DATAGM
000050  000411                          BR      3$                      ;       5733
000052  010146                  2$:     MOV     R1,-(SP)                ;       5742
000054  012746  000000G                 MOV     #DBM21,-(SP)
000060  012746  000002                  MOV     #2,-(SP)
000064  010600                          MOV     SP,R0                   ; SP,*
```

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

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


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

```
;   1   GLOBAL routine SEQUEN : novalue =
;   1
;   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           R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;   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
;   2           PRINTF (DBM111, .MSCP_PKT [.P_INDEX, OPCODE], .IPKT_ADDR [OPCODE], .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO])
;
;   2
;   3       if ((.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or
;   2          (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END))) and
;   3         ((.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

                if (.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or                  ! IF END MESSAGE IS
                   (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END)) or                 !     READ, WRITE, OR
                   (.IPKT_ADDR [OPCODE] eql (OP_ACC or OP_END))                    !     ACCESS
                then
                    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           SCAN_ERRLOG ();                                                    ! PRINT ANY ASSOCIATED ERROR-LOGS
;  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
;  2
;  1           end;                                                               ! ROUTINE DISK_RSP


                                                 .SBTTL  SEQUEN RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G                 SEQUEN::JSR     R1,$SAVE5                       ;                       5747
000004  005746                                  TST     -(SP)
000006  012746  177777                          MOV     #-1,-(SP)                       ; *,P.INDEX             5762
000012  013701  000000G                         MOV     IPKT.ADDR,R1                    ;                       5773
000016  005002                                  CLR     R2                              ; COUNT                 5771
000020  010246                          1$:     MOV     R2,-(SP)                        ; COUNT,*               5773
000022  012746  000106                          MOV     #106,-(SP)
000026  004737  000000G                         JSR     PC,BL$MUL
000032  022626                                  CMP     (SP)+,(SP)+
000034  026061  000012G 000012                  CMP     MSCP.PKT+12(R0),12(R1)
000042  001030                                  BNE     2$
000044  026061  000014G 000014                  CMP     MSCP.PKT+14(R0),14(R1)          ;                       5774
000052  001024                                  BNE     2$
000054  026011  000000G                         CMP     MSCP.PKT(R0),(R1)               ;                       5775
000060  001421                                  BEQ     2$
000062  105760  000022G                         TSTB    MSCP.PKT+22(R0)                 ;                       5776
000066  100416                                  BMI     2$
000070  132760  000360  000010G                 BITB    #360,MSCP.PKT+10(R0)            ;                       5777
000076  001012                                  BNE     2$
000100  105761  000022                          TSTB    22(R1)                          ;                       5778
000104  100007                                  BPL     2$
000106  116200  000000G                         MOVB    PKT.USE(R2),R0                  ; *(COUNT),*            5779
```

H3


```
000112  020037  000104                  CMP     R0,ICTLR
000116  001002                          BNE     2$
000120  010216                          MOV     R2,(SP)                 ; COUNT,P.INDEX            5782
000122  000405                          BR      3$                      ;                          5781
000124  005202                  2$:     INC     R2                      ; COUNT                    5771
000126  020227  000013                  CMP     R2,#13                  ; COUNT,*
000132  003732                          BLE     1$
000134  005716                          TST     (SP)                    ; P.INDEX                  5786
000136  002013                  3$:     BGE     4$
000140  016146  000012                  MOV     12(R1),-(SP)            ;                          5789
000144  016146  000014                  MOV     14(R1),-(SP)
000150  012746  000000G                 MOV     #DBM108,-(SP)
000154  012746  000003                  MOV     #3,-(SP)
000160  010600                          MOV     SP,R0                   ; SP,*
000162  104417                          TRAP    17
000164  000545                          BR      9$                      ;                          5790
000166  011646                  4$:     MOV     (SP),-(SP)              ; P.INDEX,*                5793
000170  012746  000106                  MOV     #106,-(SP)
000174  004737  000000G                 JSR     PC,BL$MUL
000200  010001                          MOV     R0,R1
000202  022626                          CMP     (SP)+,(SP)+
000204  013700  000000G                 MOV     IPKT.ADDR,R0
000210  116003  000022                  MOVB    22(R0),R3
000214  042703  177600                  BIC     #177600,R3
000220  005002                          CLR     R2
000222  156102  000022G                 BISB    MSCP.PKT+22(R1),R2
000226  020203                          CMP     R2,R3
000230  001422                          BEQ     5$
000232  016046  000012                  MOV     12(R0),-(SP)            ;                          5795
000236  016046  000014                  MOV     14(R0),-(SP)
000242  005046                          CLR     -(SP)
000244  116016  000022                  MOVB    22(R0),(SP)
000250  005046                          CLR     -(SP)
000252  116116  000022G                 MOVB    MSCP.PKT+22(R1),(SP)
000256  012746  000000G                 MOV     #DBM111,-(SP)
000262  012746  000005                  MOV     #5,-(SP)
000266  010600                          MOV     SP,R0                   ; SP,*
000270  104417                          TRAP    17
000272  062706  000014                  ADD     #14,SP
000276  013700  000000G         5$:     MOV     IPKT.ADDR,R0            ;                          5797
000302  126027  000022  000241          CMPB    22(R0),#241
000310  001404                          BEQ     6$
000312  126027  000022  000242          CMPB    22(R0),#242             ;                          5798
000320  001045                          BNE     8$
000322  012702  000024          6$:     MOV     #24,R2                  ;                          5799
000326  060002                          ADD     R0,R2
000330  132712  000037                  BITB    #37,(R2)
000334  001037                          BNE     8$
000336  032712  177740                  BIT     #177740,(R2)            ;                          5800
000342  001034                          BNE     8$
000344  026160  000026G 000026          CMP     MSCP.PKT+26(R1),26(R0)  ;                          5801
000352  001004                          BNE     7$
000354  026160  000030G 000030          CMP     MSCP.PKT+30(R1),30(R0)  ;                          5802
```

```
000362  001424                        BEQ     8$
000364  016046  000012          7$:   MOV     12(R0),-(SP)               ;                     5806
000370  016046  000014                MOV     14(R0),-(SP)
000374  016046  000026                MOV     26(R0),-(SP)
000400  016046  000030                MOV     30(R0),-(SP)
000404  016146  000026G               MOV     MSCP.PKT+26(R1),-(SP)
000410  016146  000030G               MOV     MSCP.PKT+30(R1),-(SP)
000414  012746  000000G               MOV     #DBM112,-(SP)
000420  012746  000007                MOV     #7,-(SP)
000424  010600                        MOV     SP,R0                      ; SP,*
000426  104417                        TRAP    17
000430  062706  000020                ADD     #20,SP
000434  132761  000400  000005G  8$:  BITB    #400,MSCP.PKT+5(R1)        ;                     5808
000442  001422                        BEQ     10$
000444  016146  000012G               MOV     MSCP.PKT+12(R1),-(SP)      ;                     5811
000450  016146  000014G               MOV     MSCP.PKT+14(R1),-(SP)
000454  012746  000000G               MOV     #DBM120,-(SP)
000460  012746  000003                MOV     #3,-(SP)
000464  010600                        MOV     SP,R0                      ; SP,*
000466  104417                        TRAP    17
000470  016616  000010                MOV     10(SP),(SP)                ; P.INDEX,*           5812
000474  004737  000000G               JSR     PC,PUT.PKT
000500  062706  000010          9$:   ADD     #10,SP                     ;                     5813
000504  000137  030160'               JMP     21$                        ;                     5810
000510  112761  000001  000005G 10$:  MOVB    #1,MSCP.PKT+5(R1)          ;                     5816
000516  013746  000104'               MOV     ICTLR,-(SP)                ;                     5818
000522  004737  000000G               JSR     PC,GET.RETPKT
000526  010066  000004                MOV     R0,4(SP)                   ; *,R.INDEX
000532  005726                        TST     (SP)+
000534  005766  000002                TST     2(SP)                      ; R.INDEX
000540  002007                        BGE     11$
000542  012746  000000G               MOV     #DBM22,-(SP)               ;                     5821
000546  012746  000001                MOV     #1,-(SP)
000552  010600                        MOV     SP,R0                      ; SP,*
000554  104417                        TRAP    17
000556  000563                        BR      19$                        ;                     5822
000560  013704  000000G         11$:  MOV     IPKT.ADDR,R4               ; *,SRC.ADDR          5827
000564  062704  000006                ADD     #6,R4                      ; *,SRC.ADDR
000570  016646  000002                MOV     2(SP),-(SP)                ; R.INDEX,*           5828
000574  012746  000054                MOV     #54,-(SP)
000600  004737  000000G               JSR     PC,BL$MUL
000604  062700  000000G               ADD     #RETPKT,R0
000610  010005                        MOV     R0,R5                      ; *,DST.ADDR
000612  013702  000000G               MOV     IPKT.ADDR,R2               ;                     5836
000616  012703  000001                MOV     #1,R3                      ; *,COUNT             5830
000622  012425                  12$:  MOV     (R4)+,(R5)+                ; SRC.ADDR,DST.ADDR   5832
000624  126227  000022  000211        CMPB    22(R2),#211                ;                     5836
000632  001005                        BNE     13$
000634  020327  000012                CMP     R3,#12                     ; COUNT,*             5838
000640  001002                        BNE     13$
000642  062704  000004                ADD     #4,R4                      ; *,SRC.ADDR          5840
000646  005203                  13$:  INC     R3                         ; COUNT               5830
000650  020327  000026                CMP     R3,#26                     ; COUNT,*
```

```
000654  003762                          BLE     12$
000656  013703  000104'                 MOV     ICTLR,R3                        ;                       5843
000662  042703  177760                  BIC     #177760,R3
000666  142760  000017  000002          BICB    #17,2(R0)                       ; *,*(R.ADDR)
000674  150360  000002                  BISB    R3,2(R0)                        ; *,*(R.ADDR)
000700  005766  000004                  TST     4(SP)                           ; P.INDEX               5845
000704  002441                          BLT     15$
000706  005003                          CLR     R3                              ;                       5848
000710  156203  000022                  BISB    22(R2),R3
000714  020327  000241                  CMP     R3,#241
000720  001406                          BEQ     14$
000722  020327  000242                  CMP     R3,#242                         ;                       5849
000726  001403                          BEQ     14$
000730  020327  000220                  CMP     R3,#220                         ;                       5850
000734  001025                          BNE     15$
000736  016160  000024G 000012  14$:    MOV     MSCP.PKT+24(R1),12(R0)          ; *,*(R.ADDR)           5853
000744  016160  000026G 000044          MOV     MSCP.PKT+26(R1),44(R0)          ; *,*(R.ADDR)           5854
000752  016160  000030G 000046          MOV     MSCP.PKT+30(R1),46(R0)          ; *,*(R.ADDR)           5855
000760  016160  000046G 000050          MOV     MSCP.PKT+46(R1),50(R0)          ; *,*(R.ADDR)           5856
000766  016160  000050G 000052          MOV     MSCP.PKT+50(R1),52(R0)          ; *,*(R.ADDR)           5857
000774  016160  000032G 000024          MOV     MSCP.PKT+32(R1),24(R0)          ; *,*(R.ADDR)           5858
001002  016160  000034G 000026          MOV     MSCP.PKT+34(R1),26(R0)          ; *,*(R.ADDR)           5859
001010  016616  000006          15$:    MOV     6(SP),(SP)                      ; R.INDEX,*             5862
001014  004737  000000G                 JSR     PC,IN.IODQ
001020  005726                          TST     (SP)+                           ;                       5826
001022  013716  000104'                 MOV     ICTLR,(SP)                      ;                       5868
001026  012746  000006                  MOV     #6,-(SP)
001032  004737  000000G                 JSR     PC,BL$MUL
001036  013701  000000G                 MOV     IPKT.ADDR,R1                    ;                       5865
001042  012703  000024                  MOV     #24,R3
001046  060103                          ADD     R1,R3
001050  132713  000037                  BITB    #37,(R3)
001054  001003                          BNE     16$
001056  032713  177740                  BIT     #177740,(R3)                    ;                       5866
001062  001404                          BEQ     17$
001064  012760  000001  000120' 16$:    MOV     #1,LAST.PKT(R0)                 ;                       5868
001072  000402                          BR      18$                             ;                       5865
001074  005060  000120'         17$:    CLR     LAST.PKT(R0)                    ;                       5870
001100  016160  000012  000122' 18$:    MOV     12(R1),LAST.PKT+2(R0)           ;                       5872
001106  016160  000014  000124'         MOV     14(R1),LAST.PKT+4(R0)           ;                       5873
001114  004737  000000V                 JSR     PC,SCAN.ERRLOG                  ;                       5874
001120  005766  000004                  TST     4(SP)                           ; P.INDEX               5876
001124  002404                          BLT     20$
001126  016616  000004          19$:    MOV     4(SP),(SP)                      ; P.INDEX,*             5878
001132  004737  000000G                 JSR     PC,PUT.PKT
001136  022626                  20$:    CMP     (SP)+,(SP)+                     ;                       5762
001140  022626                  21$:    CMP     (SP)+,(SP)+                     ;                       5747
001142  000207                          RTS     PC

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

```
GLOBAL routine SCAN_ERRLOG : novalue =

!+
! THIS ROUTINE SCANS THE ERROR-LOG SAVE AREA AND PRINTS ANY ERROR-LOGS RECEIVED FOR THE ASSOCIATED RESPONSE
!-

    begin

    local
        TEMP_UNIT,
        SFT_ERR_PRINTED : byte initial (byte (FALSE));

    incr index from 0 to EP_CNT do                                          ! SCAN ERROR-LOG PACKET SAVE AREA
        begin

        if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
           (.ELOG_PKT [.index, EL_CRN_LO] eql .IPKT_ADDR [CRN_LO]) and
           (.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
           (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
        then
            begin                                                           ! ERROR-LOG PENDING THIS RESPONSE

            if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED         ! IF SOFT ERROR OCCURED
            then

                if .ELOG_PKT [.index, EL_FORMAT] lequ 4
                then
                    begin
                    SOFT_ERROR (.index);                                    ! UPATE SOFT ERROR COUNT
                    TEMP_UNIT = .L$LUN;                                     ! SAVE UNIT NUMBER AS KNOWN TO DRS

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

                        if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_PKT [.index, EL_DK_NUM]) and
                           (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
                        then
                            begin
                            L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];     ! CORECT UNIT NO. FOR ERROR MESSAGE
                            exitloop;
                            end;

                    case .ELOG_PKT [.index, EL_FORMAT] from 0 to 4 of
                        set

                        [0]:    if .APT_MODE                                ! CONTROLLER ERROR
                                then
                                    ERR_SOFT_RTNE_APT (50, .index)
                                else
                                    ERR_SOFT_RTNE (50);

                        [1]:    if .APT_MODE                                ! HOST MEMORY ACCESS ERROR
                                then
                                    ERR_SOFT_RTNE_APT (51, .index)
```

```
;  5                            else
;  5                                ERR_SOFT_RTNE (51);
;  5
;  5                    [2]:    if .APT_MODE                                    ! DISK TRANSFER ERROR
;  5                            then
;  5                                ERR_SOFT_RTNE_APT (52, .index)
;  5                            else
;  5                                ERR_SOFT_RTNE (52);
;  5
;  5                    [3]:    if .APT_MODE                                    ! SDI ERROR
;  5                            then
;  5                                ERR_SOFT_RTNE_APT (53, .index)
;  5                            else
;  5                                ERR_SOFT_RTNE (53);
;  5
;  5                    [4]:    if .APT_MODE                                    ! SMALL DISK ERROR
;  5                            then
;  5                                ERR_SOFT_RTNE_APT (54, .index)
;  5                            else
;  5                                ERR_SOFT_RTNE (54);
;  5                    tes;
;  5
;  5                L$LUN = .TEMP_UNIT;                                     ! RESTORE UNIT NUMBER
;  5                SFT_ERR_PRINTED = TRUE;                                 ! SOFT ERROR PRINTOUT OCCURED
;  5                end
;  4            else
;  4                PRINTF (DBM109, .ELOG_PKT [.index, EL_FORMAT]);         ! UNKNOWN ERROR-LOG FORMAT
;  4
;  5        if not (.SFT_ERR_PRINTED)
;  4        then
;  4            PRINTB (CRLF);                                              ! EXTRA CARRIAGE-RETURN/LINE-FEED
;  4
;  4        EMS_EL (.index);                                                ! PRINT ERROR-LOG CONTENTS
;  4        end
;  3    else
;  3
;  3        if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;  4           ((.ELOG_PKT [.index, EL_CRN_HI] lssu .IPKT_ADDR [CRN_HI]) or
;  5            ((.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  3             (.ELOG_PKT [.index, EL_CRN_LO] lssu .IPKT_ADDR [CRN_LO]))) and
;  4           (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;  3        then
;  4            begin
;  4            PRINTB (CRLF);                                              ! CARRIAGE-RETURN/LINE-FEED
;  4            EMS_EL (.index);                                            ! PRINT ERROR-LOG CONTENTS
;  3            end;
;  3
;  2    end;                                                                ! ERROR-LOG SAVE AREA SCAN
;  2
;  1  end;

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

```
000000  004137  000000G          SCAN.ERRLOG::
                                         JSR     R1,$SAVE5                  ;                        5881
000004  005746                           TST     -(SP)
000006  105005                           CLRB    R5                         ; SFT.ERR.PRINTED        5887
000010  005002                           CLR     R2                         ; INDEX                  5893
000012  010246                   1$:     MOV     R2,-(SP)                   ; INDEX,*                5896
000014  012746  000102                   MOV     #102,-(SP)
000020  004737  000000G                  JSR     PC,BL$MUL
000024  010001                           MOV     R0,R1
000026  022626                           CMP     (SP)+,(SP)+
000030  012703  000000G                  MOV     #ELOG.PKT,R3
000034  060103                           ADD     R1,R3
000036  005004                           CLR     R4
000040  005000                           CLR     R0
000042  151300                           BISB    (R3),R0
000044  020037  000104'                  CMP     R0,ICTLR
000050  001016                           BNE     2$
000052  005204                           INC     R4
000054  013700  000000G                  MOV     IPKT.ADDR,R0               ;                        5897
000060  026160  000006G 000012           CMP     ELOG.PKT+6(R1),12(R0)
000066  001007                           BNE     2$
000070  026160  000010G 000014           CMP     ELOG.PKT+10(R1),14(R0)     ;                        5898
000076  001003                           BNE     2$
000100  126327  000001  000001           CMPB    1(R3),#1                   ;                        5899
000106  001402                   2$:     BEQ     3$
000110  000137  030724'                  JMP     25$
000114  013746  000104'          3$:     MOV     ICTLR,-(SP)                ;                        5903
000120  012746  000006                   MOV     #6,-(SP)
000124  004737  000000G                  JSR     PC,BL$MUL
000130  022626                           CMP     (SP)+,(SP)+
000132  005760  000120'                  TST     LAST.PKT(R0)
000136  001161                           BNE     23$
000140  126127  000016G 000004           CMPB    ELOG.PKT+16(R1),#4         ;                        5906
000146  101142                           BHI     21$
000150  010246                           MOV     R2,-(SP)                   ; INDEX,*                5909
000152  004737  000000V                  JSR     PC,SOFT.ERROR
000156  013766  000000G 000002           MOV     L$LUN,2(SP)                ; *,TEMP.UNIT            5910
000164  012703  000006                   MOV     #6,R3                      ; *,OFFSET               5912
000170  010300                   4$:     MOV     R3,R0                      ; OFFSET,*               5914
000172  063700  000076'                  ADD     ICST.ADDR,R0
000176  016146  000012G                  MOV     ELOG.PKT+12(R1),-(SP)
000202  111004                           MOVB    (R0),R4
000204  042704  177760                   BIC     #177760,R4
000210  020426                           CMP     R4,(SP)+
000212  001012                           BNE     5$
000214  032710  040000                   BIT     #40000,(R0)                ;                        5915
000220  001407                           BEQ     5$
000222  011004                           MOV     (R0),R4                    ;                        5918
000224  000304                           SWAB    R4
000226  042704  177760                   BIC     #177760,R4
000232  010437  000000G                  MOV     R4,L$LUN
000236  000405                           BR      6$                         ;                        5917
000240  062703  000024           5$:     ADD     #24,R3                     ; *,OFFSET               5912
```

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

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

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


000000                                        .PSECT  $PLIT$,  RO ,  D

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

```
;  1     GLOBAL routine DATAGM : novalue =
;  1
;  1     !+
;  1     !       THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
;  1     !       THE RDRX
;  1     !
;  1     !       IMPLICIT INPUTS:
;  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING ERROR LOG
;  1     !                               MESSAGE
;  1     !               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;  1     !-
;  1
;  2         begin
;  2
;  2         local
;  2             index : signed word initial (-1),
;  2             SAVE_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;  2             SRC_ADDR,
;  2             DST_ADDR,
;  2             TEMP_UNIT,
;  2             SFT_ERR_PRINTED : byte initial (byte (FALSE)),
;  2             PACKET_LEN : word;
;  2
;  2     !
;  2     ! FIND AN EMPTY SLOT IN THE ERROR-LOG PACKET SAVE AREA
;  2     !
;  2
;  2        incr COUNT from 0 to EP_CNT - 1 do
;  2
;  2             if .ELOG_PKT [.COUNT, EL_CONTENTS] eql EMPTY                  ! IF EMPTY SLOT FOUND
;  2             then
;  3                 begin
;  3                 index = .COUNT;                                          ! SAVE INDEX INTO THE SAVE AREA
;  3                 exitloop;
;  2                 end;
;  2
;  2         if .index lss 0
;  2         then
;  2             index = EP_CNT;                                              ! IF NO SLOT FOUND, USE LAST SPARE SLOT
;  2
;  2     !
;  2     ! SAVE THE PACKET CONTENTS
;  2     !
;  2
;  2         SAVE_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                    ! ADDRESS OF THE SAVE AREA
;  2         SAVE_ADDR [EL_CONTENTS ] = FULL;                                 ! MARK IT FULL
;  2         SAVE_ADDR [EL_CNTR] = .ICTLR;                                    ! OWNERSHIP
;  2         SRC_ADDR = .IPKT_ADDR + 6;                                       ! SETUP COPY ADDRESSES
;  2         DST_ADDR = .SAVE_ADDR + 2;                                       !
;  2         PACKET_LEN = ((.IPKT_ADDR [MSGLEN] + 1) / 2) + 2;                ! LENGTH OF ERROR-LOG INCLUDING ENVELOPE
;  2
;  2         if .PACKET_LEN gtru EP_LEN - 1
;  2         then
```

```
                PACKET_LEN = EP_LEN - 1;                                ! ADJUST LENGTH, IF TOO LONG

            incr COUNT from 1 to .PACKET_LEN do
                begin
                .DST_ADDR = ..SRC_ADDR;                                 ! COPY A WORD
                SRC_ADDR = .SRC_ADDR + 2;                               ! UPDATE ADDRESS POINTERS
                DST_ADDR = .DST_ADDR + 2;                               !
                end;

        !
        ! CHECK IF THE CORRESPONDING RESPONSE HAS ALREADY BEEN RECEIVED
        !

            if (.SAVE_ADDR [EL_CRN_LO] eql .LAST_PKT [.ICTLR, LAST_CRN_LO]) and
               (.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI])
            then
                begin                                                   ! LOG REFERS TO THE LAST RESPONSE RECEIVED

                if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED ! IF SOFT ERROR HAD OCCURED
                then

                    if .SAVE_ADDR [EL_FORMAT] lequ 4
                    then
                        begin
                        SOFT_ERROR (.index);                            ! UPDATE SOFT ERROR COUNT
                        TEMP_UNIT = .L$LUN;                             ! SAVE UNIT NUMBER AS KNOWN TO DRS

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

                            if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .SAVE_ADDR [EL_DK_NUM]) and
                               (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
                            then
                                begin
                                L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT]; ! CORRECT UNIT NUMBER FOR ERROR MESSAGE
                                exitloop;
                                end;

                        case .SAVE_ADDR [EL_FORMAT] from 0 to 4 of
                            set


                            [0] :       if .APT_MODE                    ! CONTROLLER ERROR
                                        then
                                            ERR_SOFT_RTNE_APT (50, .index)
                                        else
                                            ERR_SOFT_RTNE (50);

                            [1] :       if .APT_MODE                    ! HOST MEMORY ACCESS ERROR
                                        then
                                            ERR_SOFT_RTNE_APT (51, .index)
                                        else
                                            ERR_SOFT_RTNE (51);

```

```
;  4                  [2] :       if .APT_MODE                                    ! DISK TRANSFER ERROR
;  4                              then
;  4                                  ERR_SOFT_RTNE_APT (52, .index)
;  4                              else
;  4                                  ERR_SOFT_RTNE (52);
;  4
;  4                  [3] :       if .APT_MODE                                    ! SDI ERROR
;  4                              then
;  4                                  ERR_SOFT_RTNE_APT (53, .index)
;  4                              else
;  4                                  ERR_SOFT_RTNE (53);
;  4
;  4                  [4] :       if .APT_MODE                                    ! SMALL DISK ERROR
;  4                              then
;  4                                  ERR_SOFT_RTNE_APT (54, .index)
;  4                              else
;  4                                  ERR_SOFT_RTNE (54);
;  4                  tes;
;  4
;  4
;  4              L$LUN = .TEMP_UNIT;                                         ! RESTORE UNIT NUMBER
;  4              SFT_ERR_PRINTED = TRUE;                                     ! SOFT ERROR PRINTOUT OCCURED
;  4              end
;  4
;  3          else
;  3              PRINTF (DBM109, .SAVE_ADDR [EL_FORMAT]);                    ! ERROR LOG FORMAT UNKNOWN
;  3
;  4      if not (.SFT_ERR_PRINTED)
;  4
;  3      then
;  3          PRINTB (CRLF);                                                  ! EXTRA CARRIEGE-RETURN/LINE-FEED
;  3
;  3      EMS_EL (.index);                                                    ! PRINT PACKET CONTENTS
;  3      end                                                                 ! CORRESPONDING RESPONSE RECEIVED
;  3
;  2  else
;  2
;  2      if (.SAVE_ADDR [EL_CRN_HI] lssu .LAST_PKT [.ICTLR, LAST_CRN_HI]) or
;  3         ((.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI]) and
;  3          (.SAVE_ADDR [EL_CRN_LO] lssu .LAST_PKT [.ICTLR, LAST_CRN_LO]))
;  3
;  2      then
;  3          begin                                                           ! LOG REFERS TO SOME PREVIOUS RESPONSE
;  3          PRINTB (CRLF);                                                  ! CARRIAGE-RETURN/LINE-FEED
;  3          EMS_EL (.index);                                                ! PRINT PACKET CONTENTS
;  2          end;
;  2
;  1  end;

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

F4


```
000000  004137  000000G          DATAGM::JSR   R1,$SAVE5                      ;                                   5984
000004  012704  177777                   MOV   #-1,R4                         ; *,INDEX                           5996
000010  105046                           CLRB  -(SP)                          ; SFT.ERR.PRINTED
000012  005001                           CLR   R1                             ; COUNT                             6011
000014  010146                   1$:     MOV   R1,-(SP)                       ; COUNT,*                           6013
000016  012746  000102                   MOV   #102,-(SP)
000022  004737  000000G                  JSR   PC,BL$MUL
000026  022626                           CMP   (SP)+,(SP)+
000030  105760  000001G                  TSTB  ELOG.PKT+1(R0)
000034  001002                           BNE   2$
000036  010104                           MOV   R1,R4                          ; COUNT,INDEX                       6016
000040  000405                           BR    3$                             ;                                   6015
000042  005201                   2$:     INC   R1                             ; COUNT                             6011
000044  020127  000013                   CMP   R1,#13                         ; COUNT,*
000050  003761                           BLE   1$
000052  005704                           TST   R4                             ; INDEX                             6020
000054  002002                   3$:     BGE   4$
000056  012704  000014                   MOV   #14,R4                         ; *,INDEX                           6022
000062  010446                   4$:     MOV   R4,-(SP)                       ; INDEX,*                           6028
000064  012746  000102                   MOV   #102,-(SP)
000070  004737  000000G                  JSR   PC,BL$MUL
000074  062700  000000G                  ADD   #ELOG.PKT,R0
000100  010001                           MOV   R0,R1                          ; *,SAVE.ADDR
000102  111761  000001                   MOVB  (PC),1(R1)                     ; *,*(SAVE.ADDR)                    6029
000106  113711  000104'                  MOVB  ICTLR,(R1)                     ; *,SAVE.ADDR                       6030
000112  013700  000000G                  MOV   IPKT.ADDR,R0                   ;                                   6031
000116  012705  000006                   MOV   #6,R5                          ; *,SRC.ADDR
000122  060005                           ADD   R0,R5                          ; *,SRC.ADDR
000124  012703  000002                   MOV   #2,R3                          ; *,DST.ADDR                        6032
000130  060103                           ADD   R1,R3                          ; SAVE.ADDR,DST.ADDR
000132  016016  000006                   MOV   6(R0),(SP)                     ;                                   6033
000136  005216                           INC   (SP)
000140  012746  000002                   MOV   #2,-(SP)
000144  004737  000000G                  JSR   PC,BL$DIV
000150  062700  000002                   ADD   #2,R0
000154  020027  000040                   CMP   R0,#40                         ; PACKET.LEN,*                      6035
000160  101402                           BLOS  5$
000162  012700  000040                   MOV   #40,R0                         ; *,PACKET.LEN                      6037
000166  005002                   5$:     CLR   R2                             ; COUNT                             6039
000170  000401                           BR    7$
000172  012523                   6$:     MOV   (R5)+,(R3)+                    ; SRC.ADDR,DST.ADDR                 6041
000174  005202                   7$:     INC   R2                             ; COUNT                             6039
000176  020200                           CMP   R2,R0                          ; COUNT,PACKET.LEN
000200  003774                           BLE   6$
000202  013716  000104'                  MOV   ICTLR,(SP)                     ;                                   6050
000206  012746  000006                   MOV   #6,-(SP)
000212  004737  000000G                  JSR   PC,BL$MUL
000216  005726                           TST   (SP)+
000220  026160  000006  000122'          CMP   6(R1),LAST.PKT+2(R0)           ; *(SAVE.ADDR),*
000226  001402                           BEQ   8$
000230  000137  031674'                  JMP   30$
000234  026160  000010  000124'  8$:     CMP   10(R1),LAST.PKT+4(R0)          ; *(SAVE.ADDR),*                    6051
000242  001402                           BEQ   9$
```

```
000244 000137 031702'        JMP     31$
000250 005760 000120'   9$:  TST     LAST.PKT(R0)          ;                          6055
000254 001153                BNE     28$
000256 005003                CLR     R3                    ;                          6058
000260 156103 000016         BISB    16(R1),R3             ; *(SAVE.ADDR),*
000264 020327 000004         CMP     R3,#4
000270 101135                BHI     27$
000272 010416                MOV     R4,(SP)               ; INDEX,*                  6061
000274 004737 000000V        JSR     PC,SOFT.ERROR
000300 013705 000000G        MOV     L$LUN,R5              ; *,TEMP.UNIT              6062
000304 012702 000006         MOV     #6,R2                 ; *,OFFSET                 6064
000310 010200           10$: MOV     R2,R0                 ; OFFSET,*                 6066
000312 063700 000076'        ADD     ICST.ADDR,R0
000316 016146 000012         MOV     12(R1),-(SP)          ; *(SAVE.ADDR),*
000322 111046                MOVB    (R0),-(SP)
000324 042716 177760         BIC     #177760,(SP)
000330 022626                CMP     (SP)+,(SP)+
000332 001012                BNE     11$
000334 032710 040000         BIT     #40000,(R0)           ;                          6067
000340 001407                BEQ     11$
000342 011046                MOV     (R0),-(SP)            ;                          6070
000344 000316                SWAB    (SP)
000346 042716 177760         BIC     #177760,(SP)
000352 012637 000000G        MOV     (SP)+,L$LUN
000356 000405                BR      12$                   ;                          6069
000360 062702 000024    11$: ADD     #24,R2                ; *,OFFSET                 6064
000364 020227 000102         CMP     R2,#102               ; OFFSET,*
000370 003747                BLE     10$
000372 005000           12$: CLR     R0                    ;                          6078
000374 153700 001162'        BISB    APT.MODE,R0
000400 006303                ASL     R3                    ;                          6074
000402 066307 000012'        ADD     P.AAB(R3),PC          ; Case dispatch
000406 032700 000001    14$: BIT     #1,R0                 ;                          6078
000412 001403                BEQ     15$
000414 012716 000062         MOV     #62,(SP)              ;                          6080
000420 000442                BR      23$
000422 012716 000062    15$: MOV     #62,(SP)              ;                          6082
000426 000446                BR      25$
000430 032700 000001    16$: BIT     #1,R0                 ;                          6084
000434 001403                BEQ     17$
000436 012716 000063         MOV     #63,(SP)              ;                          6086
000442 000431                BR      23$
000444 012716 000063    17$: MOV     #63,(SP)              ;                          6088
000450 000435                BR      25$
000452 032700 000001    18$: BIT     #1,R0                 ;                          6090
000456 001403                BEQ     19$
000460 012716 000064         MOV     #64,(SP)              ;                          6092
000464 000420                BR      23$
000466 012716 000064    19$: MOV     #64,(SP)              ;                          6094
000472 000424                BR      25$
000474 032700 000001    20$: BIT     #1,R0                 ;                          6096
000500 001403                BEQ     21$
000502 012716 000065         MOV     #65,(SP)              ;                          6098
```

```
000506  000407                          BR      23$
000510  012716  000065          21$:    MOV     #65,(SP)                        ;                               6100
000514  000413                          BR      25$
000516  006000                  22$:    ROR     R0                              ;                               6102
000520  103007                          BCC     24$
000522  012716  000066                  MOV     #66,(SP)                        ;                               6104
000526  010446                  23$:    MOV     R4,-(SP)                        ; INDEX,*
000530  004737  000000V                 JSR     PC,ERR.SOFT.RTNE.A?T
000534  005726                          TST     (SP)+
000536  000404                          BR      26$                             ;                               6102
000540  012716  000066          24$:    MOV     #66,(SP)                        ;                               6106
000544  004737  000000V         25$:    JSR     PC,ERR.SOFT.RTNE
000550  010537  000000G         26$:    MOV     R5,L$LUN                        ; TEMP.UNIT,*                   6110
000554  112766  000001  000006          MOVB    #1,6(SP)                        ; *,SFT.ERR.PRINTED             6111
000562  000410                          BR      28$                             ;                               6058
000564  010316                  27$:    MOV     R3,(SP)                         ;                               6115
000566  012746  000000G                 MOV     #DBM109,-(SP)
000572  012746  000002                  MOV     #2,-(SP)
000576  010600                          MOV     SP,R0                           ; SP,*
000600  104417                          TRAP    17
000602  022626                          CMP     (SP)+,(SP)+
000604  032766  000001  000006  28$:    BIT     #1,6(SP)                        ; *,SFT.ERR.PRINTED             6117
000612  001007                          BNE     29$
000614  012716  000000G                 MOV     #CRLF,(SP)                      ;                               6120
000620  012746  000001                  MOV     #1,-(SP)
000624  010600                          MOV     SP,R0                           ; SP,*
000626  104414                          TRAP    14
000630  005726                          TST     (SP)+
000632  010416                  29$:    MOV     R4,(SP)                         ; INDEX,*                       6122
000634  004737  000000G                 JSR     PC,EMS.EL
000640  000426                          BR      33$                             ;                               6050
000642  026160  000010  000124' 30$:    CMP     10(R1),LAST.PKT+4(R0)           ; *(SAVE.ADDR),*                6127
000650  103410                  31$:    BLO     32$
000652  026160  000010  000124'         CMP     10(R1),LAST.PKT+4(R0)           ; *(SAVE.ADDR),*                6128
000660  001016                          BNE     33$
000662  026160  000006  000122'         CMP     6(R1),LAST.PKT+2(R0)            ; *(SAVE.ADDR),*                6129
000670  103012                          BHIS    33$
000672  012716  000000G         32$:    MOV     #CRLF,(SP)                      ;                               6133
000676  012746  000001                  MOV     #1,-(SP)
000702  010600                          MOV     SP,R0                           ; SP,*
000704  104414                          TRAP    14
000706  010416                          MOV     R4,(SP)                         ; INDEX,*                       6134
000710  004737  000000G                 JSR     PC,EMS.EL
000714  005726                          TST     (SP)+                           ;                               6132
000716  062706  000010          33$:    ADD     #10,SP                          ;                               5984
000722  000207                          RTS     PC

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


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

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


;   6138  1
;   6139  1
;   6140  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
```

L4


```
;         3                                     else
;         3                                         TALLY_ADDR [ERR_HRD_DRV] = .TALLY_ADDR [ERR_HRD_DRV] + 1;
;         3                             tes;
;         3
;         3             end
;         2         else                                                                            ! UNIT NOT FOUND
;         2
;         2             if .SOFT_OCCURED
;         2             then
;         2                 C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + 1
;         2             else
;         2                 C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;         2
;         1         end;                                                                            ! RTNE SOFT_ERROR


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

000000  004137  000000G                 SOFT.ERROR::
                                                JSR     R1,$SAVE5                       ;                               6141
000004  005746                                  TST     -(SP)
000006  105046                                  CLRB    -(SP)                           ; FOUND                         6151
000010  105046                                  CLRB    -(SP)                           ; SOFT.OCCURED
000012  016646  000024                          MOV     24(SP),-(SP)                    ; INDEX,*                       6163
000016  012746  000102                          MOV     #102,-(SP)
000022  004737  000000G                         JSR     PC,BL$MUL
000026  062700  000000G                         ADD     #ELOG.PKT,R0
000032  010001                                  MOV     R0,R1                           ; *,ELOG.ADDR
000034  116100  000020                          MOVB    20(R1),R0                       ; *(ELOG.ADDR),*                6164
000040  042700  177740                          BIC     #177740,R0
000044  105003                                  CLRB    R3                              ; ERROR.CODE
000046  050003                                  BIS     R0,R3                           ; *,ERROR.CODE
000050  016105  000020                          MOV     20(R1),R5                       ; *(ELOG.ADDR),ERROR.SUB        6165
000054  006205                                  ASR     R5                              ; ERROR.SUB
000056  006205                                  ASR     R5                              ; ERROR.SUB
000060  006205                                  ASR     R5                              ; ERROR.SUB
000062  006205                                  ASR     R5                              ; ERROR.SUB
000064  006205                                  ASR     R5                              ; ERROR.SUB
000066  042705  174000                          BIC     #174000,R5                      ; *,ERROR.SUB
000072  013700  000000G                         MOV     SWP.FLAGS,R0                    ;                               6167
000076  042700  077777                          BIC     #77777,R0
000102  020027  100000                          CMP     R0,#-100000
000106  001010                                  BNE     1$
000110  126127  000016  000002                  CMPB    16(R1),#2                       ; *(ELOG.ADDR),*                6168
000116  001004                                  BNE     1$
000120  005004                                  CLR     R4                              ; RETRIES                       6170
000122  156104  000051                          BISB    51(R1),R4                       ; *(ELOG.ADDR),RETRIES
000126  000402                                  BR      2$                              ;                               6167
000130  012704  000001                  1$:     MOV     #1,R4                           ; *,RETRIES                     6172
000134  005704                          2$:     TST     R4                              ; RETRIES                       6174
000136  001002                                  BNE     3$
000140  012704  000001                          MOV     #1,R4                           ; *,RETRIES                     6176
```

```
000144  012702  000006           3$:    MOV     #6,R2                  ; *,OFFSET               6178
000150  010200                   4$:    MOV     R2,R0                  ; OFFSET,*               6180
000152  063700  000076'                 ADD     ICST.ADDR,R0
000156  032710  040000                  BIT     #40000,(R0)
000162  001421                          BEQ     5$
000164  016146  000012                  MOV     12(R1),-(SP)           ; *(ELOG.ADDR),*         6181
000170  111046                          MOVB    (R0),-(SP)
000172  042716  177760                  BIC     #177760,(SP)
000176  022626                          CMP     (SP)+,(SP)+
000200  001012                          BNE     5$
000202  112766  000001  000006          MOVB    #1,6(SP)               ; *,FOUND                6184
000210  011046                          MOV     (R0),-(SP)             ;                        6185
000212  000316                          SWAB    (SP)
000214  042716  177760                  BIC     #177760,(SP)
000220  012666  000010                  MOV     (SP)+,10(SP)           ; *,UNIT
000224  000405                          BR      6$                     ;                        6183
000226  062702  000024           5$:    ADD     #24,R2                 ; *,OFFSET               6178
000232  020227  000102                  CMP     R2,#102                ; OFFSET,*
000236  003744                          BLE     4$
000240  112766  000001  000004   6$:    MOVB    #1,4(SP)               ; *,SOFT.OCCURED         6192
000246  032766  000001  000006          BIT     #1,6(SP)               ; *,FOUND                6194
000254  001002                          BNE     7$
000256  000137  032644'                 JMP     22$
000262  016616  000010           7$:    MOV     10(SP),(SP)            ; UNIT,*                 6197
000266  012746  000066                  MOV     #66,-(SP)
000272  004737  000000G                 JSR     PC,BL$MUL
000276  062700  000000G                 ADD     #TALLY,R0
000302  032766  000001  000006          BIT     #1,6(SP)               ; *,SOFT.OCCURED         6199
000310  001503                          BEQ     14$
000312  120327  000005                  CMPB    R3,#5                  ; ERROR.CODE,*           6204
000316  001462                          BEQ     12$
000320  120327  000010                  CMPB    R3,#10                 ; ERROR.CODE,*           6206
000324  001022                          BNE     9$
000326  012702  000052                  MOV     #52,R2                 ;                        6208
000332  060002                          ADD     R0,R2                  ; TALLY.ADDR,*
000334  020527  000002                  CMP     R5,#2                  ; ERROR.SUB,*            6206
000340  001005                          BNE     8$
000342  005001                          CLR     R1                     ;                        6208
000344  151201                          BISB    (R2),R1
000346  060401                          ADD     R4,R1                  ; RETRIES,*
000350  110112                          MOVB    R1,(R2)
000352  000543                          BR      21$                    ;                        6206
000354  005001                   8$:    CLR     R1                     ;                        6210
000356  156201  000001                  BISB    1(R2),R1
000362  060401                          ADD     R4,R1                  ; RETRIES,*
000364  110162  000001                  MOVB    R1,1(R2)
000370  000534                          BR      21$                    ;                        6201
000372  120327  000011           9$:    CMPB    R3,#11                 ; ERROR.CODE,*           6212
000376  001007                          BNE     10$
000400  005001                          CLR     R1
000402  156001  000055                  BISB    55(R0),R1              ; *(TALLY.ADDR),*
000406  060401                          ADD     R4,R1                  ; RETRIES,*
000410  110160  000055                  MOVB    R1,55(R0)              ; *,*(TALLY.ADDR)
```

```
000414  000522                    BR      21$                       ;                        6201
000416  120327  000012    10$:    CMPB    R3,#12                    ; ERROR.CODE,*           6214
000422  001012                    BNE     11$
000424  013702  000104'           MOV     ICTLR,R2
000430  006302                    ASL     R2
000432  005001                    CLR     R1
000434  156201  000001G           BISB    C.ERR.TBL+1(R2),R1
000440  060401                    ADD     R4,R1                     ; RETRIES,*
000442  110162  000001G           MOVB    R1,C.ERR.TBL+1(R2)
000446  000505                    BR      21$                       ;                        6201
000450  120327  000013    11$:    CMPB    R3,#13                    ; ERROR.CODE,*           6217
000454  001102                    BNE     21$
000456  020527  000003            CMP     R5,#3                     ; ERROR.SUB,*
000462  001007                    BNE     13$
000464  005001            12$:    CLR     R1                        ;                        6219
000466  156001  000052            BISB    52(R0),R1                 ; *(TALLY.ADDR),*
000472  060401                    ADD     R4,R1                     ; RETRIES,*
000474  110160  000052            MOVB    R1,52(R0)                 ; *,*(TALLY.ADDR)
000500  000470                    BR      21$                       ;                        6217
000502  005001            13$:    CLR     R1                        ;                        6221
000504  156001  000054            BISB    54(R0),R1                 ; *(TALLY.ADDR),*
000510  060401                    ADD     R4,R1                     ; RETRIES,*
000512  110160  000054            MOVB    R1,54(R0)                 ; *,*(TALLY.ADDR)
000516  000461                    BR      21$                       ;                        6201
000520  005761  000006    14$:    TST     6(R1)                     ; *(ELOG.ADDR)           6225
000524  001056                    BNE     21$
000526  005761  000010            TST     10(R1)                    ; *(ELOG.ADDR)           6226
000532  001053                    BNE     21$
000534  120327  000005            CMPB    R3,#5                     ; ERROR.CODE,*           6231
000540  001443                    BEQ     19$
000542  120327  000010            CMPB    R3,#10                    ; ERROR.CODE,*           6233
000546  001013                    BNE     16$
000550  012704  000046            MOV     #46,R4                    ;                        6235
000554  060004                    ADD     R0,R4                     ; TALLY.ADDR,*
000556  020527  000002            CMP     R5,#2                     ; ERROR.SUB,*            6233
000562  001002                    BNE     15$
000564  105214                    INCB    (R4)                      ;                        6235
000566  000435                    BR      21$                       ;                        6233
000570  105264  000001    15$:    INCB    1(R4)                     ;                        6237
000574  000432                    BR      21$                       ;                        6228
000576  120327  000011    16$:    CMPB    R3,#11                    ; ERROR.CODE,*           6239
000602  001003                    BNE     17$
000604  105260  000051            INCB    51(R0)                    ; *(TALLY.ADDR)
000610  000424                    BR      21$                       ;                        6228
000612  120327  000012    17$:    CMPB    R3,#12                    ; ERROR.CODE,*           6241
000616  001006                    BNE     18$
000620  013702  000104'           MOV     ICTLR,R2
000624  006302                    ASL     R2
000626  105262  000000G           INCB    C.ERR.TBL(R2)
000632  000413                    BR      21$                       ;                        6228
000634  120327  000013    18$:    CMPB    R3,#13                    ; ERROR.CODE,*           6244
000640  001010                    BNE     21$
000642  020527  000003            CMP     R5,#3                     ; ERROR.SUB,*
```

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

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

```
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 (PRIO4);                                  ! 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
```

```
;  2
;  2                   [42]:   ERRHRD (42, EGH_30, 0);            ! HOST WRITE COMPARE ERROR
;  2
;  2                   [43]:   ERRHRD (43, EGH_30, EMS_30);       ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  2
;  2                   [44]:   ERRHRD (44, EGH_30, EMS_30);       ! DUPLICATE UNIT NUMBER
;  2
;  2                   [45]:   ERRHRD (45, EGH_30, EMS_30);       ! INVALID END CODE
;  2
;  2                   [46]:   ;                                  !LEAVE ROOM FOR SOFT ERROR NUMBERS AND SOME PADDING  ZZZ
;  2                   [47]:   ;                                  !
;  2                   [48]:   ;                                  !
;  2                   [49]:   ;                                  !
;  2                   [50]:   ;                                  !
;  2                   [51]:   ;                                  !
;  2                   [52]:   ;                                  !
;  2                   [53]:   ;                                  !
;  2                   [54]:   ;                                  !
;  2                   [55]:   ;                                  !
;  2                   [56]:   ;                                  !
;  2                   [57]:   ;                                  !
;  2                   [58]:   ;                                  !
;  2                   [59]:   ;                                  !
;  2
;  2                   [60]:  ERRHRD (60, EH_12, EMS_30);         !NOT USED                                 ZZZ
;  2                   [61]:  ERRHRD (61, EH_13, EMS_30);         !SUCCESSFUL MESSAGE                       ZZZ
;  2                   [62]:  ERRHRD (62, EH_13, EMS_30);         !ILLEGAL UNIT NUMBER                      ZZZ
;  2                   [63]:  ERRHRD (63, EH_13, EMS_30);         !ILLEGAL RELATIVE OR PHYSICAL BLOCK       ZZZ
;  2                   [64]:  ERRHRD (64, EH_13, EMS_30);         !DEVICE ERROR                             ZZZ
;  2                   [65]:  ERRHRD (65, EH_13, EMS_30);         !ZERO LENGTH MESSAGE                      ZZZ
;  2                   [66]:  ERRHRD (66, EH_8,  EMS_30);         !DUP UNKNOWN STATUS CODE                  ZZZ
;  2                   [67]:  ERRHRD (67, EH_7,  EMS_30);         !INVALID COMMAND                          ZZZ
;  2                   [68]:  ERRHRD (68, EH_7,  EMS_30);         !NO REGION AVAILABLE                      ZZZ
;  2                   [69]:  ERRHRD (69, EH_7,  EMS_30);         !NO REGION SUITABLE                       ZZZ
;  2                   [70]:  ERRHRD (70, EH_7,  EMS_30);         !PROGRAM NOT KNOWN                        ZZZ
;  2                   [71]:  ERRHRD (71, EH_7,  EMS_30);         !LOAD FAILURE                             ZZZ
;  2                   [72]:  ERRHRD (72, EH_7,  EMS_30);         !STANDALONE                               ZZZ
;  2                   [73]:  ERRHRD (73, EH_8,  EMS_30);         !DUP UNKNOWN STATUS CODE                  ZZZ
;  2
;  2                   tes
;  2           else
;  3               begin
;  3               !****increment error count                     ! INCREMENT TOTAL ERROR COUNT
;  3               PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);        ! PRINT ERROR MESSAGE JUST LIKE DRS
;  3
;  3               if .ERRNUM neq 42
;  3               then
;  4                   begin
;  4                   PRINTB (HRD_SUB);                          ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;  4                   EMS_ERR ();                                ! PRINT REST OF THE INFORMATION
;  3                   end;
;  2               end;
;  2
```

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

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

```
000110  000000G                     .WORD   EMS.30
000112  000567                      BR      30$                          ;              6290
000114  104456              5$:     TRAP    56                           ;              6295
000116  000040                      .WORD   40
000120  000000G                     .WORD   EGH.30
000122  000000G                     .WORD   EMS.30
000124  000562                      BR      30$                          ;              6290
000126  104456              6$:     TRAP    56                           ;              6303
000130  000044                      .WORD   44
000132  000000G                     .WORD   EGH.30
000134  000000G                     .WORD   EMS.30
000136  000555                      BR      30$                          ;              6290
000140  104456              7$:     TRAP    56                           ;              6305
000142  000045                      .WORD   45
000144  000000G                     .WORD   EGH.30
000146  000000G                     .WORD   EMS.30
000150  000550                      BR      30$                          ;              6290
000152  104456              8$:     TRAP    56                           ;              6309
000154  000047                      .WORD   47
000156  000000G                     .WORD   EGH.30
000160  000000G                     .WORD   EMS.30
000162  000574                      BR      33$                          ;              6290
000164  104456              9$:     TRAP    56                           ;              6311
000166  000050                      .WORD   50
000170  000000G                     .WORD   EGH.30
000172  000000G                     .WORD   EMS.30
000174  000567                      BR      33$                          ;              6290
000176  104456              10$:    TRAP    56                           ;              6313
000200  000051                      .WORD   51
000202  000000G                     .WORD   EGH.30
000204  000000G                     .WORD   EMS.30
000206  000562                      BR      33$                          ;              6290
000210  104456              11$:    TRAP    56                           ;              6315
000212  000052                      .WORD   52
000214  000000G                     .WORD   EGH.30
000216  000000                      .WORD   0
000220  000555                      BR      33$                          ;              6290
000222  104456              12$:    TRAP    56                           ;              6317
000224  000053                      .WORD   53
000226  000000G                     .WORD   EGH.30
000230  000000G                     .WORD   EMS.30
000232  000550                      BR      33$                          ;              6290
000234  104456              13$:    TRAP    56                           ;              6319
000236  000054                      .WORD   54
000240  000000G                     .WORD   EGH.30
000242  000000G                     .WORD   EMS.30
000244  000543                      BR      33$                          ;              6290
000246  104456              14$:    TRAP    56                           ;              6321
000250  000055                      .WORD   55
000252  000000G                     .WORD   EGH.30
000254  000000G                     .WORD   EMS.30
000256  000536                      BR      33$                          ;              6290
000260  104456              15$:    TRAP    56                           ;              6338
```

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

H5


```
000434  000447                           BR     33$                     ;                        6290
000436  104456                   26$:    TRAP   56                      ;                        6349
000440  000107                           .WORD  107
000442  000000G                          .WORD  EH.7
000444  000000G                          .WORD  EMS.30
000446  000442                   27$:    BR     33$                     ;                        6290
000450  104456                   28$:    TRAP   56                      ;                        6350
000452  000110                           .WORD  110
000454  000000G                          .WORD  EH.7
000456  000000G                          .WORD  EMS.30
000460  000435                           BR     33$                     ;                        6290
000462  104456                   29$:    TRAP   56                      ;                        6351
000464  000111                           .WORD  111
000466  000000G                          .WORD  EH.8
000470  000000G                          .WORD  EMS.30
000472  000430                   30$:    BR     33$                     ;                        6286
000474  010746                   31$:    MOV    PC,-(SP)                ; PC,*                   6357
000476  013746  000000G                  MOV    L$LUN,-(SP)
000502  010146                           MOV    R1,-(SP)
000504  012746  000000G                  MOV    #HRD.MSG,-(SP)
000510  012746  000004                   MOV    #4,-(SP)
000514  010600                           MOV    SP,R0                   ; SP,*
000516  104414                           TRAP   14
000520  020127  000052                   CMP    R1,#52                  ;                        6359
000524  001411                           BEQ    32$
000526  012716  000000G                  MOV    #HRD.SUB,(SP)           ;                        6362
000532  012746  000001                   MOV    #1,-(SP)
000536  010600                           MOV    SP,R0                   ; SP,*
000540  104414                           TRAP   14
000542  004737  000000G                  JSR    PC,EMS.ERR              ;                        6363
000546  005726                           TST    (SP)+                   ;                        6361
000550  062706  000012           32$:    ADD    #12,SP                  ;                        6355
000554  020127  000043           33$:    CMP    R1,#43                  ;                        6367
000560  001403                           BEQ    34$
000562  020127  000046                   CMP    R1,#46                  ;                        6368
000566  001050                           BNE    37$
000570  032737  040000  000000G  34$:    BIT    #40000,SWP.FLAGS        ;                        6371
000576  001420                           BEQ    36$
000600  020127  000043                   CMP    R1,#43                  ;                        6377
000604  001005                           BNE    35$
000606  104456                           TRAP   56
000610  000043                           .WORD  43
000612  000000G                          .WORD  EGH.30
000614  000000G                          .WORD  EMS.30
000616  000434                           BR     37$                     ;                        6374
000620  020127  000046           35$:    CMP    R1,#46                  ;                        6379
000624  001031                           BNE    37$
000626  104456                           TRAP   56
000630  000046                           .WORD  46
000632  000000G                          .WORD  EGH.30
000634  000000G                          .WORD  EMS.30
000636  000424                           BR     37$                     ;                        6371
000640  010746                   36$:    MOV    PC,-(SP)                ; PC,*                   6384
```

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

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


000024                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAC:                                  ; CASE Table for ERR.HRD.RTNE+0076      6290
000024  000000                  3$:     .WORD   0                       ; [4$]
000026  000012                          .WORD   12                      ; [5$]
000030  000452                          .WORD   452                     ; [33$]
000032  000452                          .WORD   452                     ; [33$]
000034  000452                          .WORD   452                     ; [33$]
000036  000024                          .WORD   24                      ; [6$]
000040  000036                          .WORD   36                      ; [7$]
000042  000452                          .WORD   452                     ; [33$]
000044  000050                          .WORD   50                      ; [8$]
000046  000062                          .WORD   62                      ; [9$]
000050  000074                          .WORD   74                      ; [10$]
000052  000106                          .WORD   106                     ; [11$]
000054  000120                          .WORD   120                     ; [12$]
000056  000132                          .WORD   132                     ; [13$]
000060  000144                          .WORD   144                     ; [14$]
000062  000452                          .WORD   452                     ; [33$]
000064  000452                          .WORD   452                     ; [33$]
000066  000452                          .WORD   452                     ; [33$]
000070  000452                          .WORD   452                     ; [33$]
000072  000452                          .WORD   452                     ; [33$]
000074  000452                          .WORD   452                     ; [33$]
000076  000452                          .WORD   452                     ; [33$]
000100  000452                          .WORD   452                     ; [33$]
000102  000452                          .WORD   452                     ; [33$]
000104  000452                          .WORD   452                     ; [33$]
000106  000452                          .WORD   452                     ; [33$]
000110  000452                          .WORD   452                     ; [33$]
000112  000452                          .WORD   452                     ; [33$]
000114  000452                          .WORD   452                     ; [33$]
000116  000156                          .WORD   156                     ; [15$]
```

```
000120  000170          .WORD   170             ; [16$]
000122  000202          .WORD   202             ; [17$]
000124  000214          .WORD   214             ; [18$]
000126  000226          .WORD   226             ; [19$]
000130  000240          .WORD   240             ; [20$]
000132  000252          .WORD   252             ; [21$]
000134  000264          .WORD   264             ; [22$]
000136  000276          .WORD   276             ; [23$]
000140  000310          .WORD   310             ; [24$]
000142  000322           WORD   322             ; [25$]
000144  000334          .WORD   334             ; [26$]
000146  000346          .WORD   346             ; [28$]
000150  000360          .WORD   360             ; [29$]
```

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

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

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

```
000042  000063                            .WORD   63
000044  000000                            .WORD   0
000046  000000                            .WORD   0
000050  000207                            RTS     PC                         ;                                   6407
000052  104457                    4$:     TRAP    57                         ;                                   6414
000054  000064                            .WORD   64
000056  000000                            .WORD   0
000060  000000                            .WORD   0
000062  000207                            RTS     PC                         ;                                   6407
000064  104457                    5$:     TRAP    57                         ;                                   6416
000066  000065                            .WORD   65
000070  000000                            .WORD   0
000072  000000                            .WORD   0
000074  000207                            RTS     PC                         ;                                   6407
000076  104457                    6$:     TRAP    57                         ;                                   6418
000100  000066                            .WORD   66
000102  000000                            .WORD   0
000104  000000                            .WORD   0
000106  000207                            RTS     PC                         ;                                   6404
000110  010746                    7$:     MOV     PC,-(SP)                   ; PC,*                              6423
000112  013746  000000G                   MOV     L$LUN,-(SP)
000116  016646  000006                    MOV     6(SP),-(SP)                ; ERRNUM,*
000122  012746  000000G                   MOV     #SFT.MSG,-(SP)
000126  012746  000004                    MOV     #4,-(SP)
000132  010600                            MOV     SP,R0                      ; SP,*
000134  104414                            TRAP    14
000136  062706  000012                    ADD     #12,SP                     ;                                   6421
000142  000207                            RTS     PC                         ;                                   6392

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


000152                                    .PSECT  $PLIT$,  RO ,  D

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

```
; 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 PPOTECTED

                [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
```

```
;   6533 2                  [38]:   ERRDF (38, EGH_30, EMS_30);       ! DATA ERROR
;   6534 2                  tes
;   6535 2
;   6536 2          else
;   6537 2
;   6538 3              begin
;   6539 3              !****increment error count                    ! INCREMENT TOTAL ERROR COUNT
;   6540 3              PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);        ! PRINT ERROR LINE JUST LIKE DRS
;   6541 3              PRINTB (HRD_SUB);                             ! PRINT NEXT LINE TOO
;   6542 3              EMS_ERR ();                                   ! PRINT REST OF THE INFORMATION
;   6543 2              end;
;   6544 2
;   6545 2
;   6546 2      SETPRI (.CUR_PRIORITY);                               ! PRIORITY BACK TO NORMAL
;   6547 2
;   6548 2
;   6549 1      end;

                                          .SBTTL  ERR.HRD.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
033766                                    .PSECT  $CODE$,  RO

000000  004137  000000G             ERR.HRD.RTNE.APT:
                                          JSR     R1,$SAVE2                 ;                       6427
000004  104440                            TRAP    40                        ;                       6446
000006  010002                            MOV     R0,R2                     ; *,CUR.PRIORITY
000010  013700  000000G                   MOV     BRLEVEL,R0                ;                       6448
000014  104441                            TRAP    41
000016  032737  000001  001162'           BIT     #1,APT.MODE               ;                       6451
000024  001412                            BEQ     1$
000026  013700  000000G                   MOV     RP.ADDR,R0                ;                       6455
000032  016077  000050  001164'           MOV     50(R0),@MAIL.BOX.TESTNUM
000040  013700  000000G                   MOV     RP.ADDR,R0                ;                       6456
000044  016077  000010  001166'           MOV     10(R0),@MAIL.BOX.SUBTST
000052  016601  000010              1$:   MOV     10(SP),R1                 ; ERRNUM,*              6460
000056  020127  000042                    CMP     R1,#42
000062  101411                            BLOS    2$
000064  020127  000046                    CMP     R1,#46                    ;                       6461
000070  101006                            BHI     2$
000072  020127  000044                    CMP     R1,#44                    ;                       6462
000076  001403                            BEQ     2$
000100  020127  000045                    CMP     R1,#45                    ;                       6463
000104  001131                            BNE     17$
000106  032737  010000  000000G     2$:   BIT     #10000,SWP.FLAGS          ;                       6467
000114  001475                            BEQ     15$
000116  010100                            MOV     R1,R0                     ;                       6470
000120  162700  000037                    SUB     #37,R0
000124  006300                            ASL     R0
000126  066007  000164'                   ADD     P.AAE(R0),PC              ; Case dispatch
000132  104455                      4$:   TRAP    55                        ;                       6473
000134  000037                            .WORD   37
000136  000000G                           .WORD   EGH.30
000140  000000G                           .WORD   EMS.30
```

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

D6


```
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                  ;                               6511
000340  001411                          BEQ     16$
000342  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                               6515
000346  012746  000001                  MOV     #1,-(SP)
000352  010600                          MOV     SP,R0                   ; SP,*
000354  104414                          TRAP    14
000356  004737  000000G                 JSR     PC,EMS.ERR              ;                               6516
000362  005726                          TST     (SP)+                   ;                               6514
000364  062706  000012          16$:    ADD     #12,SP                  ;                               6506
000370  020127  000043          17$:    CMP     R1,#43                  ;                               6520
000374  001403                          BEQ     18$
000376  020127  000046                  CMP     R1,#46                  ;                               6521
000402  001050                          BNE     21$
000404  032737  040000  000000G 18$:    BIT     #40000,SWP.FLAGS        ;                               6525
000412  001420                          BEQ     20$
000414  020127  000043                  CMP     R1,#43                  ;                               6531
000420  001005                          BNE     19$
000422  104455                          TRAP    55
000424  000043                          .WORD   43
000426  000000G                         .WORD   EGH.30
000430  000000G                         .WORD   EMS.30
000432  000434                          BR      21$                     ;                               6528
000434  020127  000046          19$:    CMP     R1,#46                  ;                               6533
000440  001031                          BNE     21$
000442  104455                          TRAP    55
000444  000046                          .WORD   46
000446  000000G                         .WORD   EGH.30
000450  000000G                         .WORD   EMS.30
000452  000424                          BR      21$                     ;                               6525
000454  010746                  20$:    MOV     PC,-(SP)                ; PC,*                          6540
000456  013746  000000G                 MOV     L$LUN,-(SP)
000462  010146                          MOV     R1,-(SP)
000464  012746  000000G                 MOV     #DF.MSG,-(SP)
000470  012746  000004                  MOV     #4,-(SP)
000474  010600                          MOV     SP,R0                   ; SP,*
000476  104414                          TRAP    14
000500  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                               6541
000504  012746  000001                  MOV     #1,-(SP)
000510  010600                          MOV     SP,R0                   ; SP,*
000512  104414                          TRAP    14
000514  004737  000000G                 JSR     PC,EMS.ERR              ;                               6542
000520  062706  000014                  ADD     #14,SP                  ;                               6538
000524  010200                  21$:    MOV     R2,R0                   ; CUR.PRIORITY,*                6546
000526  104441                          TRAP    41
000530  000207                          RTS     PC                      ;                               6427

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

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

                        P.AAE:                                  ; CASE Table for ERR.HRD.RTNE.AP+0126    6470
000164  000000          3$:     .WORD   0                       ; [4$]
000166  000012                  .WORD   12                      ; [5$]
000170  000236                  .WORD   236                     ; [17$]
000172  000236                  .WORD   236                     ; [17$]
000174  000236                  .WORD   236                     ; [17$]
000176  000024                  .WORD   24                      ; [6$]
000200  000036                  .WORD   36                      ; [7$]
000202  000236                  .WORD   236                     ; [17$]
000204  000050                  .WORD   50                      ; [8$]
000206  000062                  .WORD   62                      ; [9$]
000210  000074                  .WORD   74                      ; [10$]
000212  000106                  .WORD   106                     ; [11$]
000214  000120                  .WORD   120                     ; [12$]
000216  000132                  .WORD   132                     ; [13$]
000220  000144                  .WORD   144                     ; [14$]


;    6550  1
;    6551  1
```

```
routine ERR_SOFT_RTNE_APT (ERRNUM, index) : novalue =

!+
! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
! THE SAME EFFECT WITHOUT ISSUING THE CALL
!-

    begin

    local
        ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);

    builtin
        PC;

    ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);          ! ADDRESS OF THE SAVED ERROR-LOG INFORMATION

    if .APT_MODE
    then
        begin
        .MAIL_BOX_TESTNUM = .ELOG_ADDR [EL_BLOCK];         ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
        .MAIL_BOX_SUBTST = .ELOG_ADDR [EL_DK_NUM];         ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER IN APT ONLY
        end;

    if BIT_TST (SWP_FLAGS, SWF_SFT)                        ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
    then

        case .ERRNUM from 50 to 54 of
            set

            [50]:       ERRDF (50, 0, 0);                  ! CONTROLLER ERROR

            [51]:       ERRDF (51, 0, 0);                  ! HOST MEMORY ACCESS ERROR

            [52]:       ERRDF (52, 0, 0);                  ! DISK TRANSFER ERROR

            [53]:       ERRDF (53, 0, 0);                  ! SDI ERROR

            [54]:       ERRDF (54, 0, 0);                  ! SMALL DISK ERROR
            tes
    else
        begin
        !****increment error count                         ! INCREMENT TOTAL ERROR COUNT
        PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);             ! PRINT ERROR LINE JUST LIKE DRS
        end;

    end;
```

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

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

```
                                        MOV     2(SP),-(SP)                 ; INDEX,*                    6567
000004  012746  000102                  MOV     #102,-(SP)
000010  004737  000000G                 JSR     PC,BL$MUL
000014  062700  000000G                 ADD     #ELOG.PKT,R0
000020  032737  000001  001162'         BIT     #1,APT.MODE                 ;                            6569
000026  001406                          BEQ     1$
000030  016077  000056  001164'         MOV     56(R0),@MAIL.BOX.TESTNUM    ; *(ELOG.ADDR),*             6572
000036  016077  000012  001166'         MOV     12(R0),@MAIL.BOX.SUBTST     ; *(ELOG.ADDR),*             6573
000044  032737  020000  000000G   1$:   BIT     #20000,SWP.FLAGS            ;                            6576
000052  001440                          BEQ     8$
000054  016600  000010                  MOV     10(SP),R0                   ; ERRNUM,*                   6579
000060  162700  000062                  SUB     #62,R0
000064  006300                          ASL     R0
000066  066007  000222'                 ADD     P.AAF(R0),PC                ; Case dispatch
000072  104455                    3$:   TRAP    55                          ;                            6582
000074  000062                          .WORD   62
000076  000000                          .WORD   0
000100  000000                          .WORD   0
000102  000441                          BR      9$                          ;                            6579
000104  104455                    4$:   TRAP    55                          ;                            6584
000106  000063                          .WORD   63
000110  000000                          .WORD   0
000112  000000                          .WORD   0
000114  000434                          BR      9$                          ;                            6579
000116  104455                    5$:   TRAP    55                          ;                            6586
000120  000064                          .WORD   64
000122  000000                          .WORD   0
000124  000000                          .WORD   0
000126  000427                          BR      9$                          ;                            6579
000130  104455                    6$:   TRAP    55                          ;                            6588
000132  000065                          .WORD   65
000134  000000                          .WORD   0
000136  000000                          .WORD   0
000140  000422                          BR      9$                          ;                            6579
000142  104455                    7$:   TRAP    55                          ;                            6590
000144  000066                          .WORD   66
000146  000000                          .WORD   0
000150  000000                          .WORD   0
000152  000415                          BR      9$                          ;                            6576
000154  010716                    8$:   MOV     PC,(SP)                     ; PC,*                       6595
000156  013746  000000G                 MOV     L$LUN,-(SP)
000162  016646  000012                  MOV     12(SP),-(SP)                ; ERRNUM,*
000166  012746  000000G                 MOV     #DF.MSG,-(SP)
000172  012746  000004                  MOV     #4,-(SP)
000176  010600                          MOV     SP,R0                       ; SP,*
000200  104414                          TRAP    14
000202  062706  000010                  ADD     #10,SP                      ;                            6593
000206  022626                    9$:   CMP     (SP)+,(SP)+                 ;                            6559
000210  000207                          RTS     PC                          ;                            6552

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

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

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


;     6599 1
;     6600 1
;     6601 1     end
;     6602 1
;     6603 0     eludom


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


;                         PSECT SUMMARY
;
;       Psect Name                Words    Attributes
;        $GGG$                      324      RO ,  I  ,  LCL,  REL,  CON
;        $CODE$                    7405      RO ,  I  ,  LCL,  REL,  CON
;        $PLIT$                      78      RO ,  D  ,  LCL,  REL,  CON


;                     Library Statistics
;
;                                  -------- Symbols --------       Pages        Processing
;       File                       Total    Loaded   Percent       Mapped       Time
;
;  DISK$USER2:[POWERS]ZRQAF0.L16;11    406      332        81           21        00:00.1


;                         COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAF0.BL2/LIST=ZRQAF0.LS2/OBJECT=ZRQAF0.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 'ZRQAFO.L16';
;     1
;     1    require 'BLSMAC.REQ';                                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;     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                              ;                       8108

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


;   8135  1     end
;   8136  1
;   8137  0     eludom



;                               PSECT SUMMARY
;
;       Psect Name              Words   Attributes
;        $XYZ$                     44     RO ,  I  ,  LCL,  REL,  CON




;                       Library Statistics
;
;                                       -------- Symbols --------       Pages       Processing
;       File                            Total   Loaded  Percent         Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQAF0.L16;11       406        7        1            21       00:00.1




;                               COMMAND QUALIFIERS

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

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

```
ZRQAFO.EXE   Memory allocation map  TKB M40.02        Page 1
                  27-DEC-84   13:19


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

*** Root segment: ZRQAFO


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

Memory allocation synopsis:

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


Global symbols:

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

```
DBM28B 005564-R  EGD.22 011714-R  EX.CBC 015344-R  GP$3   024770-R  IRDRX. 077440-R  L$RPT  026522-R  QIO.UN 053312-R
DBM29  005624-R  EGD.23 011754-R  EX.CBR 015410-R  GP$30  025350-R  ISR    000100    L$SFTL 025142-R  RANDOM 101024-R
DBM32  005672-R  EGD.24 012016-R  EX.CBW 015462-R  GP$31  025364-R  IXE    004000    L$SOFT 025144-R  RDM.CN 101022-R
DBM5   004622-R  EGH.30 012062-R  EX.CMD 014474-R  GP$32  025402-R  LOE    040000    L$SPC  002056-R  RDRX.A 077436-R
DCT    077412-R  EGS.01 011046-R  EX.CMP 014534-R  GP$33  025414-R  LOT    000010    L$SPCP 002020-R  RDRX.E 024172-R
DCT.AD 077434-R  EGS.02 011066-R  EX.LBN 015054-R  GP$4   025002-R  L$ACP  002110-R  L$SPTP 002024-R  RD.COU 105262-R
DFPTBL 024564-R  EH.0   012540-R  EX.LBR 015150-R  GP$5   025014-R  L$APT  002036-R  L$STA  002030-R  REG.EX 044156-R
DF.MSG 024210-R  EH.1   012576-R  EX.LBW 015216-R  GP$6   025024-R  L$AU   032274-R  L$SW   024614-R  RETPKT 102614-R
DIO.RE 062616-R  EH.10  013152-R  EX.ONL 014550-R  GP$7   025034-R  L$AUT  002070-R  L$SWLE 024612-R  ROUND. 066020-R
DISK.R 070624-R  EH.12  013202-R  EX.OP  014574-R  GP$8   025044-R  L$AUTO 031302-R  L$TEST 002114-R  RPS.RE 067002-R
DRIVER 043176-R  EH.13  013236-R  EX.PBN 015112-R  GP$9   025056-R  L$CCP  002106-R  L$TIML 002014-R  RPT1   007554-R
DROP.C 033746-R  EH.2   012634-R  EX.RBN 015264-R  HARD.E 063642-R  L$CLEA 031576-R  L$UNIT 002012-R  RPT10  010264-R
DRV.CT 034054-R  EH.3   012674-R  EX.RD  014514-R  HARD.I 045174-R  L$CO   002032-R  MD.INI 052070-R  RPT11  010352-R
DR.ERR 050620-R  EH.4   012732-R  EX.RP  016046-R  HOE    100000    L$DEPO 002011-R  MINUTE 105246-R  RPT12  010420-R
DR.RET 067170-R  EH.5   012756-R  EX.SA  014400-R  HOE.FL 105253-R  L$DESC 024724-R  MODULA 034606-R  RPT13  010466-R
DUP    055576-R  EH.6   013006-R  EX.SB  014452-R  HOST.W 066364-R  L$DESP 002076-R  MSCP.P 101066-R  RPT14  010566-R
DUPCOM 060436-R  EH.7   013036-R  EX.SBO 014446-R  HOURS  105245-R  L$DEVP 002060-R  MSG.01 007434-R  RPT15  010664-R
DUPIDL 060604-R  EH.8   013066-R  EX.SC  014416-R  HRD.MS 024306-R  L$DISP 002124-R  MSG.02 007466-R  RPT16  010764-R
DUPPKT 100014-R  EH.9   013122-R  EX.TIM 016144-R  HRD.SU 024504-R  L$DLY  002116-R  MSG.03 007522-R  RPT2   007640-R
DUPRED 057302-R  ELG.FM 014366-R  EX.WRD 016134-R  HWPTE0 024600-R  L$DTP  002040-R  MULTI. 051414-R  RPT3   007704-R
DUPROU 024630-R  ELG.OO 014034-R  EX.WRT 014524-R  HWPTE1 024602-R  L$DTYP 002034-R  NAME.H 024606-R  RPT4   007770-R
DUPWRT 056132-R  ELOG.P 103370-R  FATAL. 070014-R  HWPTS0 024574-R  L$DU   032202-R  NAME.L 024604-R  RPT5   010034-R
DUP.CO 063150-R  EMS.BL 036502-R  FER.BC 105272-R  HWPTS1 024576-R  L$DUT  002072-R  NEX    105230-R  RPT6   010122-R
DUP.FL 105170-R  EMS.CM 040532-R  FER.LB 105270-R  HWPT.B 024570-R  L$DVTY 024704-R  NEXT.P 105244-R  RPT7   010166-R
DUP.RS 067522-R  EMS.DB 036304-R  FILL.B 062170-R  HWPT.D 024572-R  L$EF   002052-R  NEX.TR 032304-R  RPT8   010204-R
DUR    105202-R  EMS.EL 037410-R  FORCED 105267-R  HWPT.I 024564-R  L$ENVI 002044-R  NULL   004620-R  RPT9   010232-R
DU.MSG 006666-R  EMS.ER 041060-R  FREE.M 105210-R  HWPT.V 024566-R  L$ERRT 002126-R  OF.RC  105222-R  RP.ADD 103366-R
DU.RSN 007406-R  EMS.RP 037352-R  FSET.U 063516-R  HWQ1   002136-R  L$ETP  002102-R  OUT.IO 033622-R  RP.IND 103364-R
D$PCNT 002122-R  EMS.01 041264-R  GET.IO 033442-R  HWQ10  002740-R  L$EXP1 002046-R  OVF.CH 065744-R  RP.USE 103354-R
D.FAIL 105266-R  EMS.10 041322-R  GET.PK 032572-R  HWQ11  002770-R  L$EXP4 002064-R  PKT.US 102600-R  SA.REG 105224-R
EBD.10 012150-R  EMS.12 041364-R  GET.RA 053202-R  HWQ2   002152-R  L$EXP5 002066-R  PNT    001000    SB.COD 105216-R
EBD.12 012210-R  EMS.13 041422-R  GET.RE 033320-R  HWQ3   002162-R  L$HARD 024750-R  POLL.C 070246-R  SCAN.E 072070-R
EBD.13 012256-R  EMS.14 041464-R  GP$DIS 025320-R  HWQ4   002226-R  L$HIME 002120-R  POLL.R 070346-R  SC.CLK 021550-R
EBD.14 012310-R  EMS.18 041526-R  GP$1   024750-R  HWQ5   002244-R  L$HPCP 002016-R  PRI    002000    SC.CON 016562-R
EBD.18 012350-R  EMS.21 041570-R  GP$10  025070-R  HWQ6A  002314-R  L$HPTP 002022-R  PRI00  000000    SC.CTO 021012-R
EBD.19 012404-R  EMS.22 041606-R  GP$11  025104-R  HWQ6B  002366-R  L$HRDL 024746-R  PRI01  000040    SC.DIS 017110-R
EBD.24 012464-R  EMS.24 041624-R  GP$12  025120-R  HWQ7A  002442-R  L$HW   024564-R  PRI02  000100    SC.DST 017520-R
EBS.01 012106-R  EMS.30 041666-R  GP$13  025130-R  HWQ7B  002512-R  L$HWLE 024562-R  PRI03  000140    SC.DS2 017574-R
EF.CON 000036    ENTRY. 105166-R  GP$14  025144-R  HWQ8   002562-R  L$ICP  002104-R  PRI04  000200    SC.DUP 016604-R
EF.NEW 000035    EOP.FL 105167-R  GP$15  025156-R  HWQ9   002640-R  L$INIT 031270-R  PRI05  000240    SC.ECC 017646-R
EF.PWR 000034    ERRBLK 002134-R  GP$16  025170-R  IBE    010000    L$LADP 002026-R  PRI06  000300    SC.ECD 017730-R
EF.RES 000037    ERRMSG 002132-R  GP$17  025202-R  IDU    000040    L$LAST 107222-R  PRI07  000340    SC.EC1 020200-R
EF.STA 000040    ERRNBR 002130-R  GP$18  025214-R  IER    020000    L$LOAD 002100-R  PROC.R 062422-R  SC.EC2 020230-R
EGD.10 011160-R  ERRTYP 002126-R  GP$19  025222-R  INIT.I 052230-R  L$LUN  002074-R  PUTA.B 033562-R  SC.EC3 020260-R
EGD.11 011220-R  ERR.CO 014000-R  GP$2   024760-R  INIT.O 105274-R  L$MREV 002050-R  PUT.IO 033516-R  SC.EC4 020312-R
EGD.12 011244-R  ERR.OO 013310-R  GP$20  025230-R  INIT.T 043040-R  L$NAME 002000-R  PUT.PK 033160-R  SC.EC5 020342-R
EGD.13 011272-R  EVL    000004    GP$21  025236-R  INI.CT 044032-R  L$NDHR 025140-R  PUT.RE 033426-R  SC.EC6 020372-R
EGD.14 011320-R  EX.ACC 014562-R  GP$22  025244-R  INI.RR 046044-R  L$NDHW 024610-R  P.INDE 105260-R  SC.EC7 020422-R
EGD.15 011350-R  EX.BB  014600-R  GP$23  025252-R  INT.GE 044714-R  L$NDSF 025432-R  QIO    105206-R  SC.EC8 020454-R
EGD.16 011366-R  EX.BBU 014764-R  GP$24  025260-R  IN.IOD 033660-R  L$NDSW 024674-R  QIO.FU 054716-R  SC.EC9 020506-R
EGD.17 011416-R  EX.BB1 014670-R  GP$25  025274-R  IODQ   105152-R  L$PRIO 002042-R  QIO.GE 052602-R  SC.EDC 021142-R
EGD.18 011434-R  EX.BC  015534-R  GP$26  025304-R  IODQ.I 105162-R  L$PROT 024676-R  QIO.LB 061030-R  SC.FCT 020132-R
EGD.19 011454-R  EX.BD  015610-R  GP$27  025312-R  IODQ.O 105164-R  L$PRT  002112-R  QIO.OK 052434-R  SC.FER 017202-R
EGD.20 011514-R  EX.BDR 015666-R  GP$28  025324-R  IO.RET 063246-R  L$REPP 002062-R  QIO.OU 052530-R  SC.FE2 017270-R
EGD.21 011602-R  EX.BDW 015756-R  GP$29  025340-R  IPKT.A 102576-R  L$REV  002010-R  QIO.SI 061760-R  SC.FUL 020002-R
```

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

*** Task builder statistics:

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

Elapsed time:00:01:14

SYMBOL	VALUE	REFERENCES...		
ADDR.V	105275-R	# ZRQAM1	ZRQAM2	ZRQAM3
ADR	000020	# ZRQAM1	# ZRQAM2	# ZRQAM3
ASTERI	024554-R	# ZRQAM1	ZRQAM2	
AZINT	067322-R	# ZRQAM3		
AZINTO	067304-R	# ZRQAM3		
BITO	000001	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITOO	000001	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO1	000002	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO2	000004	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO3	000010	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO4	000020	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO5	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO6	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO7	000200	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO8	000400	# ZRQAM1	# ZRQAM2	# ZRQAM3
BITO9	001000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT1	000002	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT10	002000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT11	004000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT12	010000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT13	020000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT14	040000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT15	100000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT2	000004	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT3	000010	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT4	000020	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT5	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT6	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT7	000200	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT8	000400	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT9	001000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BL$DIV	077062-R	# B16MUL	ZRQAM2	ZRQAM3
BL$LAS	107216-R	# ZRQAM4		
BL$MOD	077074-R	# B16MUL	ZRQAM2	ZRQAM3
BL$MUL	076636-R	# B16MUL	ZRQAM2	ZRQAM3
BL$SHF	077106-R	# B16MUL	ZRQAM3	
BOE	000400	# ZRQAM1	# ZRQAM2	# ZRQAM3
BRLEVE	105264-R	# ZRQAM1	ZRQAM2	ZRQAM3
BST	077442-R	# ZRQAM1	ZRQAM2	ZRQAM3
BUFF.A	105122-R	# ZRQAM1	ZRQAM2	ZRQAM3
BUFF.O	105142-R	# ZRQAM1	ZRQAM2	ZRQAM3
BYTS.P	105212-R	# ZRQAM1	ZRQAM2	ZRQAM3
CCTLR	105172-R	# ZRQAM1	ZRQAM2	ZRQAM3
CDISK	105174-R	# ZRQAM1	ZRQAM2	ZRQAM3
CER.01	016416-R	# ZRQAM1	ZRQAM2	
CER.02	016462-R	# ZRQAM1	ZRQAM2	
CLK.PR	105252-R	# ZRQAM1	ZRQAM2	ZRQAM3
CLK.TI	105250-R	# ZRQAM1	ZRQAM2	ZRQAM3
CMD.TI	105226-R	# ZRQAM1	ZRQAM2	ZRQAM3
CNTR.E	023452-R	# ZRQAM1	ZRQAM2	
CREDIT	105242-R	# ZRQAM1	ZRQAM2	ZRQAM3
CRLF	024542-R	# ZRQAM1	ZRQAM2	ZRQAM3

C7

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
CRN.HI	105234-R	# ZRQAM1	ZRQAM2	ZRQAM3
CRN.LO	105232-R	# ZRQAM1	ZRQAM2	ZRQAM3
CST	077262-R	# ZRQAM1	ZRQAM2	ZRQAM3
CST.AD	077410-R	# ZRQAM1	ZRQAM2	ZRQAM3
CTLR.C	105200-R	# ZRQAM1	ZRQAM2	ZRQAM3
CTLR.I	043530-R	# ZRQAM3		
CUOFF	105176-R	# ZRQAM1	ZRQAM2	ZRQAM3
C.ERR.	101064-R	# ZRQAM1	ZRQAM2	ZRQAM3
DASH	024546-R	# ZRQAM1	ZRQAM2	
DATAGM	072736-R	# ZRQAM3		
DBM101	005746-R	# ZRQAM1		
DBM104	005774-R	# ZRQAM1		
DBM105	006036-R	# ZRQAM1		
DBM107	006074-R	# ZRQAM1	ZRQAM2	
DBM108	006132-R	# ZRQAM1	ZRQAM3	
DBM109	006222-R	# ZRQAM1	ZRQAM3	
DBM111	006302-R	# ZRQAM1	ZRQAM3	
DBM112	006402-R	# ZRQAM1	ZRQAM3	
DBM12	004650-R	# ZRQAM1	ZRQAM3	
DBM120	006504-R	# ZRQAM1	ZRQAM3	
DBM121	006576-R	# ZRQAM1	ZRQAM3	
DBM15	004724-R	# ZRQAM1		
DBM18	004754-R	# ZRQAM1	ZRQAM3	
DBM19	005026-R	# ZRQAM1	ZRQAM3	
DBM20	005112-R	# ZRQAM1	ZRQAM3	
DBM21	005166-R	# ZRQAM1	ZRQAM3	
DBM22	005250-R	# ZRQAM1	ZRQAM3	
DBM23	005314-R	# ZRQAM1	ZRQAM3	
DBM25	005352-R	# ZRQAM1	ZRQAM3	
DBM26	005420-R	# ZRQAM1	ZRQAM3	
DBM27	005452-R	# ZRQAM1	ZRQAM3	
DBM28A	005524-R	# ZRQAM1		
DBM28B	005564-R	# ZRQAM1		
DBM29	005624-R	# ZRQAM1	ZRQAM3	
DBM32	005672-R	# ZRQAM1		
DBM5	004622-R	# ZRQAM1	ZRQAM2	
DCT	077412-R	# ZRQAM1	ZRQAM2	ZRQAM3
DCT.AD	077434-R	# ZRQAM1	ZRQAM2	ZRQAM3
DFPTBL	024564-R	# ZRQAM1		
DF.MSG	024210-R	# ZRQAM1	ZRQAM3	
DIO.RE	062616-R	# ZRQAM3		
DISK.R	070624-R	# ZRQAM3		
DRIVER	043176-R	# ZRQAM3		
DROP.C	033746-R	# ZRQAM2	ZRQAM3	
DRV.CT	034054-R	# ZRQAM2	ZRQAM3	
DR.ERR	050620-R	# ZRQAM3		
DR.RET	067170-R	# ZRQAM3		
DUP	055576-R	# ZRQAM3		
DUPCOM	060436-R	# ZRQAM3		
DUPIDL	060604-R	# ZRQAM3		
DUPPKT	100014-R	# ZRQAM1	ZRQAM2	ZRQAM3
DUPRED	057302-R	# ZRQAM3		

```
SYMBOL  VALUE      REFERENCES...

DUPROU  024630-R  # ZRQAM1    ZRQAM3
DUPWRT  056132-R  # ZRQAM3
DUP.CO  063150-R  # ZRQAM3
DUP.FL  105170-R  # ZRQAM1    ZRQAM2    ZRQAM3
DUP.RS  067522-R  # ZRQAM3
DUR     105202-R  # ZRQAM1    ZRQAM2    ZRQAM3
DU.MSG  006666-R  # ZRQAM1    ZRQAM2
DU.RSN  007406-R  # ZRQAM1    ZRQAM2
D$PCNT  002122-R  # ZRQAM1
D.FAIL  105266-R  # ZRQAM1    ZRQAM2    ZRQAM3
EBD.10  012150-R  # ZRQAM1    ZRQAM2
EBD.12  012210-R  # ZRQAM1    ZRQAM2
EBD.13  012256-R  # ZRQAM1    ZRQAM2
EBD.14  012310-R  # ZRQAM1    ZRQAM2
EBD.18  012350-R  # ZRQAM1    ZRQAM2
EBD.19  012404-R  # ZRQAM1    ZRQAM2
EBD.24  012464-R  # ZRQAM1    ZRQAM2
EBS.01  012106-R  # ZRQAM1    ZRQAM2
EF.CON  000036    # ZRQAM1  # ZRQAM2  # ZRQAM3
EF.NEW  000035    # ZRQAM1  # ZRQAM2  # ZRQAM3
EF.PWR  000034    # ZRQAM1  # ZRQAM2  # ZRQAM3
EF.RES  000037    # ZRQAM1  # ZRQAM2  # ZRQAM3
EF.STA  000040    # ZRQAM1  # ZRQAM2  # ZRQAM3
EGD.10  011160-R  # ZRQAM1    ZRQAM3
EGD.11  011220-R  # ZRQAM1    ZRQAM3
EGD.12  011244-R  # ZRQAM1    ZRQAM3
EGD.13  011272-R  # ZRQAM1    ZRQAM3
EGD.14  011320-R  # ZRQAM1    ZRQAM3
EGD.15  011350-R  # ZRQAM1    ZRQAM3
EGD.16  011366-R  # ZRQAM1    ZRQAM3
EGD.17  011416-R  # ZRQAM1    ZRQAM3
EGD.18  011434-R  # ZRQAM1    ZRQAM3
EGD.19  011454-R  # ZRQAM1    ZRQAM3
EGD.20  011514-R  # ZRQAM1    ZRQAM3
EGD.21  011602-R  # ZRQAM1    ZRQAM3
EGD.22  011714-R  # ZRQAM1    ZRQAM3
EGD.23  011754-R  # ZRQAM1    ZRQAM3
EGD.24  012016-R  # ZRQAM1    ZRQAM3
EGH.30  012062-R  # ZRQAM1    ZRQAM3
EGS.01  011046-R  # ZRQAM1    ZRQAM2
EGS.02  011066-R  # ZRQAM1    ZRQAM3
EH.0    012540-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.1    012576-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.10   013152-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.12   013202-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.13   013236-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.2    012634-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.3    012674-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.4    012732-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.5    012756-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.6    013006-R  # ZRQAM1    ZRQAM2    ZRQAM3
EH.7    013036-R  # ZRQAM1    ZRQAM2    ZRQAM3
```

SYMBOL	VALUE	REFERENCES...		
EH.8	013066-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.9	013122-R	# ZRQAM1	ZRQAM2	ZRQAM3
ELG.FM	014366-R	# ZRQAM1	ZRQAM2	
ELG.00	014034-R	# ZRQAM1	ZRQAM2	
ELOG.P	103370-R	# ZRQAM1	ZRQAM2	ZRQAM3
EMS.BL	036502-R	# ZRQAM2		
EMS.CM	040532-R	# ZRQAM2	ZRQAM3	
EMS.DB	036304-R	# ZRQAM2		
EMS.EL	037410-R	# ZRQAM2	ZRQAM3	
EMS.ER	041060-R	# ZRQAM2	ZRQAM3	
EMS.RP	037352-R	# ZRQAM2	ZRQAM3	
EMS.01	041264-R	# ZRQAM2		
EMS.10	041322-R	# ZRQAM2	ZRQAM3	
EMS.12	041364-R	# ZRQAM2	ZRQAM3	
EMS.13	041422-R	# ZRQAM2	ZRQAM3	
EMS.14	041464-R	# ZRQAM2	ZRQAM3	
EMS.18	041526-R	# ZRQAM2	ZRQAM3	
EMS.21	041570-R	# ZRQAM2	ZRQAM3	
EMS.22	041606-R	# ZRQAM2	ZRQAM3	
EMS.24	041624-R	# ZRQAM2	ZRQAM3	
EMS.30	041666-R	# ZRQAM2	ZRQAM3	
ENTRY.	105166-R	# ZRQAM1	ZRQAM2	ZRQAM3
EOP.FL	105167-R	# ZRQAM1	ZRQAM2	ZRQAM3
ERRBLK	002134-R	# ZRQAM1		
ERRMSG	002132-R	# ZRQAM1		
ERRNBR	002130-R	# ZRQAM1		
ERRTYP	002126-R	# ZRQAM1		
ERR.CO	014000-R	# ZRQAM1	ZRQAM2	
ERR.00	013310-R	# ZRQAM1	ZRQAM2	
EVL	000004	# ZRQAM1	# ZRQAM2	# ZRQAM3
EX.ACC	014562-R	# ZRQAM1	ZRQAM2	
EX.BB	014600-R	# ZRQAM1	ZRQAM2	
EX.BBU	014764-R	# ZRQAM1	ZRQAM2	
EX.BB1	014670-R	# ZRQAM1	ZRQAM2	
EX.BC	015534-R	# ZRQAM1	ZRQAM2	
EX.BD	015610-R	# ZRQAM1	ZRQAM2	
EX.BDR	015666-R	# ZRQAM1	ZRQAM2	
EX.BDW	015756-R	# ZRQAM1	ZRQAM2	
EX.CBC	015344-R	# ZRQAM1	ZRQAM2	
EX.CBR	015410-R	# ZRQAM1	ZRQAM2	
EX.CBW	015462-R	# ZRQAM1	ZRQAM2	
EX.CMD	014474-R	# ZRQAM1	ZRQAM2	
EX.CMP	014534-R	# ZRQAM1	ZRQAM2	
EX.LBN	015054-R	# ZRQAM1	ZRQAM2	
EX.LBR	015150-R	# ZRQAM1	ZRQAM2	
EX.LBW	015216-R	# ZRQAM1	ZRQAM2	
EX.ONL	014550-R	# ZRQAM1	ZRQAM2	
EX.OP	014574-R	# ZRQAM1	ZRQAM2	
EX.PBN	015112-R	# ZRQAM1	ZRQAM2	
EX.RBN	015264-R	# ZRQAM1	ZRQAM2	
EX.RD	014514-R	# ZRQAM1	ZRQAM2	
EX.RP	016046-R	# ZRQAM1	ZRQAM2	

F7

SYMBOL	VALUE	REFERENCES...		
EX.SA	014400-R	# ZRQAM1	ZRQAM2	
EX.SB	014452-R	# ZRQAM1	ZRQAM2	
EX.SBO	014446-R	# ZRQAM1	ZRQAM2	
EX.SC	014416-R	# ZRQAM1	ZRQAM2	
EX.TIM	016144-R	# ZRQAM1	ZRQAM2	
EX.WRD	016134-R	# ZRQAM1	ZRQAM2	
EX.WRT	014524-R	# ZRQAM1	ZRQAM2	
FATAL.	070014-R	# ZRQAM3		
FER.BC	105272-R	# ZRQAM1	ZRQAM2	ZRQAM3
FER.LB	105270-R	# ZRQAM1	ZRQAM2	ZRQAM3
FILL.B	062170-R	# ZRQAM3		
FORCED	105267-R	# ZRQAM1	ZRQAM2	ZRQAM3
FREE.M	105210-R	# ZRQAM1	ZRQAM2	ZRQAM3
FSET.U	063516-R	# ZRQAM3		
GET.IO	033442-R	# ZRQAM2	ZRQAM3	
GET.PK	032572-R	# ZRQAM2	ZRQAM3	
GET.RA	053202-R	# ZRQAM3		
GET.RE	033320-R	# ZRQAM2	ZRQAM3	
GP$DIS	025320-R	# ZRQAM2		
GP$1	024750-R	# ZRQAM2		
GP$10	025070-R	# ZRQAM2		
GP$11	025104-R	# ZRQAM2		
GP$12	025120-R	# ZRQAM2		
GP$13	025130-R	# ZRQAM2		
GP$14	025144-R	# ZRQAM2		
GP$15	025156-R	# ZRQAM2		
GP$16	025170-R	# ZRQAM2		
GP$17	025202-R	# ZRQAM2		
GP$18	025214-R	# ZRQAM2		
GP$19	025222-R	# ZRQAM2		
GP$2	024760-R	# ZRQAM2		
GP$20	025230-R	# ZRQAM2		
GP$21	025236-R	# ZRQAM2		
GP$22	025244-R	# ZRQAM2		
GP$23	025252-R	# ZRQAM2		
GP$24	025260-R	# ZRQAM2		
GP$25	025274-R	# ZRQAM2		
GP$26	025304-R	# ZRQAM2		
GP$27	025312-R	# ZRQAM2		
GP$28	025324-R	# ZRQAM2		
GP$29	025340-R	# ZRQAM2		
GP$3	024770-R	# ZRQAM2		
GP$30	025350-R	# ZRQAM2		
GP$31	025364-R	# ZRQAM2		
GP$32	025402-R	# ZRQAM2		
GP$33	025414-R	# ZRQAM2		
GP$4	025002-R	# ZRQAM2		
GP$5	025014-R	# ZRQAM2		
GP$6	025024-R	# ZRQAM2		
GP$7	025034-R	# ZRQAM2		
GP$8	025044-R	# ZRQAM2		
GP$9	025056-R	# ZRQAM2		

SYMBOL	VALUE	REFERENCES...		
HARD.E	063642-R	# ZRQAM3		
HARD.I	045174-R	# ZRQAM3		
HOE	100000	# ZRQAM1	# ZRQAM2	# ZRQAM3
HOE.FL	105253-R	# ZRQAM1	ZRQAM2	ZRQAM3
HOST.W	066364-R	# ZRQAM3		
HOURS	105245-R	# ZRQAM1	ZRQAM2	ZRQAM3
HRD.MS	024306-R	# ZRQAM1	ZRQAM3	
HRD.SU	024504-R	# ZRQAM1	ZRQAM3	
HWPTE0	024600-R	# ZRQAM1		
HWPTE1	024602-R	# ZRQAM1		
HWPTS0	024574-R	# ZRQAM1		
HWPTS1	024576-R	# ZRQAM1		
HWPT.B	024570-R	# ZRQAM1		
HWPT.D	024572-R	# ZRQAM1		
HWPT.I	024564-R	# ZRQAM1		
HWPT.V	024566-R	# ZRQAM1		
HWQ1	002136-R	# ZRQAM1	ZRQAM2	
HWQ10	002740-R	# ZRQAM1	ZRQAM2	
HWQ11	002770-R	# ZRQAM1	ZRQAM2	
HWQ2	002152-R	# ZRQAM1	ZRQAM2	
HWQ3	002162-R	# ZRQAM1	ZRQAM2	
HWQ4	002226-R	# ZRQAM1	ZRQAM2	
HWQ5	002244-R	# ZRQAM1	ZRQAM2	
HWQ6A	002314-R	# ZRQAM1	ZRQAM2	
HWQ6B	002366-R	# ZRQAM1	ZRQAM2	
HWQ7A	002442-R	# ZRQAM1	ZRQAM2	
HWQ7B	002512-R	# ZRQAM1	ZRQAM2	
HWQ8	002562-R	# ZRQAM1	ZRQAM2	
HWQ9	002640-R	# ZRQAM1	ZRQAM2	
IBE	010000	# ZRQAM1	# ZRQAM2	# ZRQAM3
IDU	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
IER	020000	# ZRQAM1	# ZRQAM2	# ZRQAM3
INIT.I	052230-R	# ZRQAM3		
INIT.O	105274-R	# ZRQAM1	ZRQAM2	ZRQAM3
INIT.T	043040-R	# ZRQAM3		
INI.CT	044032-R	# ZRQAM3		
INI.RR	046044-R	# ZRQAM3		
INT.GE	044714-R	# ZRQAM3		
IN.IOD	033660-R	# ZRQAM2	ZRQAM3	
IODQ	105152-R	# ZRQAM1	ZRQAM2	ZRQAM3
IODQ.I	105162-R	# ZRQAM1	ZRQAM2	ZRQAM3
IODQ.O	105164-R	# ZRQAM1	ZRQAM2	ZRQAM3
IO.RET	063246-R	# ZRQAM3		
IPKT.A	102576-R	# ZRQAM1	ZRQAM2	ZRQAM3
IRDRX.	077440-R	# ZRQAM1	ZRQAM2	ZRQAM3
ISR	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
IXE	004000	# ZRQAM1	# ZRQAM2	# ZRQAM3
LOE	040000	# ZRQAM1	# ZRQAM2	# ZRQAM3
LOT	000010	# ZRQAM1	# ZRQAM2	# ZRQAM3
L$ACP	002110-R	# ZRQAM1		
L$APT	002036-R	# ZRQAM1		
L$AU	032274-R	ZRQAM1	# ZRQAM2	

H7


```
ZRQAFO     CREATED BY  TKB    ON 27-DEC-84 AT 13:19       PAGE 7
GLOBAL CROSS REFERENCE                                    CREF   V02

SYMBOL   VALUE        REFERENCES...

L$AUT    002070-R   # ZRQAM1
L$AUTO   031302-R     ZRQAM1   # ZRQAM2
L$CCP    002106-R   # ZRQAM1
L$CLEA   031576-R     ZRQAM1   # ZRQAM2
L$CO     002032-R   # ZRQAM1
L$DEPO   002011-R   # ZRQAM1
L$DESC   024724-R     ZRQAM1   # ZRQAM2
L$DESP   002076-R   # ZRQAM1
L$DEVP   002060-R   # ZRQAM1
L$DISP   002124-R   # ZRQAM1
L$DLY    002116-R   # ZRQAM1     ZRQAM2     ZRQAM3
L$DTP    002040-R   # ZRQAM1
L$DTYP   002034-R   # ZRQAM1
L$DU     032202-R     ZRQAM1   # ZRQAM2
L$DUT    002072-R   # ZRQAM1
L$DVTY   024704-R     ZRQAM1   # ZRQAM2
L$EF     002052-R   # ZRQAM1
L$ENVI   002044-R   # ZRQAM1
L$ERRT   002126-R   # ZRQAM1
L$ETP    002102-R   # ZRQAM1
L$EXP1   002046-R   # ZRQAM1
L$EXP4   002064-R   # ZRQAM1
L$EXP5   002066-R   # ZRQAM1
L$HARD   024750-R     ZRQAM1   # ZRQAM2
L$HIME   002120-R   # ZRQAM1     ZRQAM2
L$HPCP   002016-R   # ZRQAM1
L$HPTP   002022-R   # ZRQAM1
L$HRDL   024746-R   # ZRQAM2
L$HW     024564-R   # ZRQAM1
L$HWLE   024562-R   # ZRQAM1
L$ICP    002104-R   # ZRQAM1
L$INIT   031270-R     ZRQAM1   # ZRQAM2
L$LADP   002026-R   # ZRQAM1
L$LAST   107222-R     ZRQAM1   # ZRQAM4
L$LOAD   002100-R   # ZRQAM1
L$LUN    002074-R   # ZRQAM1     ZRQAM2     ZRQAM3
L$MREV   002050-R   # ZRQAM1
L$NAME   002000-R   # ZRQAM1
L$NDHR   025140-R   # ZRQAM2
L$NDHW   024610-R   # ZRQAM1
L$NDSF   025432-R   # ZRQAM2
L$NDSW   024674-R   # ZRQAM1
L$PRIO   002042-R   # ZRQAM1
L$PROT   024676-R   # ZRQAM1
L$PRT    002112-R   # ZRQAM1
L$REPP   002062-R   # ZRQAM1
L$REV    002010-R   # ZRQAM1
L$RPT    026522-R     ZRQAM1   # ZRQAM2
L$SFTL   025142-R   # ZRQAM2
L$SOFT   025144-R     ZRQAM1   # ZRQAM2
L$SPC    002056-R   # ZRQAM1
L$SPCP   002020-R   # ZRQAM1
```

SYMBOL	VALUE	REFERENCES...		
L$SPTP	002024-R	# ZRQAM1		
L$STA	002030-R	# ZRQAM1		
L$SW	024614-R	# ZRQAM1		
L$SWLE	024612-R	# ZRQAM1		
L$TEST	002114-R	# ZRQAM1		
L$TIML	002014-R	# ZRQAM1		
L$UNIT	002012-R	# ZRQAM1	ZRQAM2	ZRQAM3
MD.INI	052070-R	# ZRQAM3		
MINUTE	105246-R	# ZRQAM1	ZRQAM2	ZRQAM3
MODULA	034606-R	# ZRQAM2	ZRQAM3	
MSCP.P	101066-R	# ZRQAM1	ZRQAM2	ZRQAM3
MSG.01	007434-R	# ZRQAM1	ZRQAM2	
MSG.02	007466-R	# ZRQAM1	ZRQAM3	
MSG.03	007522-R	# ZRQAM1	ZRQAM3	
MULTI.	051414-R	# ZRQAM3		
NAME.H	024606-R	# ZRQAM1		
NAME.L	024604-R	# ZRQAM1		
NEX	105230-R	# ZRQAM1	ZRQAM2	ZRQAM3
NEXT.P	105244-R	# ZRQAM1	ZRQAM2	ZRQAM3
NEX.TR	032304-R	# ZRQAM2	ZRQAM3	
NULL	004620-R	# ZRQAM1	ZRQAM2	
OF.RC	105222-R	# ZRQAM1	ZRQAM2	ZRQAM3
OUT.IO	033622-R	# ZRQAM2	ZRQAM3	
OVF.CH	065744-R	# ZRQAM3		
PKT.US	102600-R	# ZRQAM1	ZRQAM2	ZRQAM3
PNT	001000	# ZRQAM1	# ZRQAM2	# ZRQAM3
POLL.C	070246-R	# ZRQAM3		
POLL.R	070346-R	# ZRQAM3		
PRI	002000	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI00	000000	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI01	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI02	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI03	000140	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI04	000200	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI05	000240	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI06	000300	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI07	000340	# ZRQAM1	# ZRQAM2	# ZRQAM3
PROC.R	062422-R	# ZRQAM3		
PUTA.B	033562-R	# ZRQAM2	ZRQAM3	
PUT.IO	033516-R	# ZRQAM2	ZRQAM3	
PUT.PK	033160-R	# ZRQAM2	ZRQAM3	
PUT.RE	033426-R	# ZRQAM2	ZRQAM3	
P.INDE	105260-R	# ZRQAM1	ZRQAM2	ZRQAM3
QIO	105206-R	# ZRQAM1	ZRQAM2	ZRQAM3
QIO.FU	054716-R	# ZRQAM3		
QIO.GE	052602-R	# ZRQAM3		
QIO.LB	061030-R	# ZRQAM3		
QIO.OK	052434-R	# ZRQAM3		
QIO.OU	052530-R	# ZRQAM3		
QIO.SI	061760-R	# ZRQAM3		
QIO.UN	053312-R	# ZRQAM3		
RANDOM	101024-R	# ZRQAM1	ZRQAM2	ZRQAM3

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

K7

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

```
$       exit
ZRQAFO     CREATED BY  TKB    ON 27-DEC-84 AT 13:19      PAGE 11                                              SEQ 0503

GLOBAL CROSS REFERENCE                                  CREF    V02

SYMBOL  VALUE      REFERENCES...

SWQ14   003546-R  # ZRQAM1    ZRQAM2
SWQ15   003620-R  # ZRQAM1    ZRQAM2
SWQ17   003670-R  # ZRQAM1    ZRQAM2
SWQ19   003766-R  # ZRQAM1    ZRQAM2
SWQ2    003044-R  # ZRQAM1    ZRQAM2
SWQ20   004056-R  # ZRQAM1    ZRQAM2
SWQ21   004144-R  # ZRQAM1    ZRQAM2
SWQ22   004230-R  # ZRQAM1    ZRQAM2
SWQ23   004270-R  # ZRQAM1    ZRQAM2
SWQ24   004342-R  # ZRQAM1    ZRQAM2
SWQ25   004406-R  # ZRQAM1    ZRQAM2
SWQ26   004460-R  # ZRQAM1    ZRQAM2
SWQ4    003124-R  # ZRQAM1    ZRQAM2
SWQ7    003146-R  # ZRQAM1    ZRQAM2
SWQ9    003220-R  # ZRQAM1    ZRQAM2
S.DUPP  105256-R  # ZRQAM1    ZRQAM2    ZRQAM3
S.PATT  105254-R  # ZRQAM1    ZRQAM2    ZRQAM3
TALLY   077462-R  # ZRQAM1    ZRQAM2    ZRQAM3
TEMP1   105236-R  # ZRQAM1    ZRQAM2    ZRQAM3
TEMP2   105240-R  # ZRQAM1    ZRQAM2    ZRQAM3
TIME    032314-R  # ZRQAM2    ZRQAM3
TRK.SG  101016-R  # ZRQAM1    ZRQAM2    ZRQAM3
T$FREE  107342-R  # ZRQAM4
T$PTHV  000004      ZRQAM1  # ZRQAM4
T.ADDR  100012-R  # ZRQAM1    ZRQAM2    ZRQAM3
T1      043024-R    ZRQAM1  # ZRQAM3
UAM     000200    # ZRQAM1  # ZRQAM2  # ZRQAM3
UPD.IO  065424-R  # ZRQAM3
VEC.BR  044370-R  # ZRQAM3
WAIT    034570-R  # ZRQAM2    ZRQAM3
XX13    016204-R  # ZRQAM1    ZRQAM2
XX23    016226-R  # ZRQAM1    ZRQAM2
XX32    016262-R  # ZRQAM1    ZRQAM2
XX33    016310-R  # ZRQAM1    ZRQAM2
XX34    016346-R  # ZRQAM1    ZRQAM2
$END.L  107344-R  # ZRQAM4
$SAVE2  077154-R    B16MUL  # B16SAV    ZRQAM2    ZRQAM3
$SAVE3  077170-R  # B16SAV    ZRQAM2    ZRQAM3
$SAVE4  077206-R  # B16SAV    ZRQAM2    ZRQAM3
$SAVE5  077226-R    B16MUL  # B16SAV    ZRQAM2    ZRQAM3
$       VERIFY_CODE = f$verify(0)
$       on control_y then               $ goto END
$       set verify
$       inquire ANSWER "*       Compile ZRQAFO library? [Y/N]"
$       if ANSWER .nes. "Y" then        $ goto L1
$       bliss/pdp11 ZRQAFO.REQ/list=ZRQAFO.LIS/library=ZRQAFO.L16/source=page:53
$L1:    inquire ANSWER "*       Compile Module 1? [Y/N]"
$       if ANSWER .nes. "Y" then        $ goto L2
$       bliss/pdp11 ZRQAFO.BL1/list=ZRQAFO.LS1/object=ZRQAFO.OB1/source=page:53
$L2:    inquire ANSWER "*       Compile Module 2? [Y/N]"
$       if ANSWER .nes. "Y" then        $ goto L3
$       bliss/pdp11 ZRQAFO.BL2/list=ZRQAFO.LS2/object=ZRQAFO.OB2/source=page:53
$L3:    inquire ANSWER "*       Task-Build? [Y/N]"
$       if ANSWER .nes. "Y" then        $ goto L4
$       mcr tkb
ZRQAFO.EXE/nohd/nomm,ZRQAFO/CR/-sp=ZRQAFO.OB1,ZRQAFO.OB2,NOEIS.OLB/lb
/
par=DUMMY:2000:176000
stack=0
//
$!
```

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

M7