ECOAD·2.0

VAX 11750
BOOT ROMS

EP·ECOAD·DL 2.0
FICHE 1 OF 1

IDENTIFICATION

PRODUCT CODE: ZZ-ECOAD-2.0

PRODUCT TITLE: VAX 11750 BOOT ROM LISTINGS

PRODUCT DATE: APRIL, 1981

DEPARTMENT: VAX ENGINEERING

```
                                0000     1 ;               MASSBUS DEVICE BOOT ROM
                                0000     2 ;
                                0000     3 ;       ALL SUPPORTED MASSBUS DISKS ARE BOOTED AS 'DB'
                                0000     4 ;
                                0000     5 ;
                                0000     6 ;
                    00000007    0000     7 DT_LIMIT = 7                          ;DT_LIMIT IS THE NUMBER OF
                                0000     8                                       ;SUPPORTED DRIVE TYPES
                    00000000    0000     9         .=0
                       44 42    0000    10         .ASCII  /BD/
                    38   10     0002    11 START:  BSB     DEVADD
                 04 A1   D6     0004    12         INCL    4(R1)                ;INIT MBA
      57   5E 00000200 8F   C3  0007    13         SUBL3   #^X200,SP,R7         ;R7_XFER ADDRESS
                    58   D4     000F    14         CLRL    R8                   ;R8_LBN_0
                 5E   30   C0    0011    15         ADDL2   #^X30,SP
                    48   10     0014    16         BSB     XFER                 ;JUMPTO XFER SUBROUTINE
                    50   D4     0016    17 OK:     CLRL    R0                   ;SET BOOT DEVICE TYPE FLAG
                 52   53   D0    0018    18         MOVL    R3,R2                ;SET BOOT DEVICE # FLAG
         56 0000002C'EF   9E    001B    19         MOVAB   DRIVER,R6            ;SET DRIVER ADDRESS
                 5E   30   C2    0022    20         SUBL2   #^X30,SP
                    59   B5     0025    21         TSTW    R9                   ;SEE IF XFER WAS SUCCESSFUL
                    D9   12     0027    22         BNEQ    START                ;IF NOT SUCCESSFUL, TRY AGAIN
                 0C A7   17     0029    23         JMP     ^XC(R7)              ;JUMP INTO BLOCK 0 CODE
                                002C    24
                                002C    25
                                002C    26
                                002C    27
                                002C    28 DRIVER:
              57   04 AE   D0    002C    29         MOVL    4(SP),R7             ;R7_XFER ADDRESS
                    0D   10     0030    30         BSB     DEVADD
                    2A   10     0032    31         BSB     XFER                 ;DO TRANSFER
                    59   B5     0034    32         TSTW    R9
                    03   13     0036    33         BEQL    NOERR
                    50   D4     0038    34         CLRL    R0                   ;CLEAR R0 IF ERROR
                                003A    35         RSB                          ;RETURN IF ERROR
                 50   01   9A    003B    36 NOERR:  MOVZBL  #1,R0                ;R0_1 IF SUCCESSFUL
                                003E    37         RSB
                                003F    38
                                003F    39 DEVADD:
           50   53   07   78    003F    40         ASHL    #7,R3,R0             ;MOVE BOOT DEVICE # TO RIGHT PLACE
                50   51   C0    0043    41         ADDL2   R1,R0                ;R4_BOOT DEV ADDRESS
           50   0400 8F   A0    0046    42         ADDW2   #^X400,R0            ;SET EXT REG ADD BIT
                                0048    43         RSB
                                004C    44
                                004C    45
                                004C    46 WAIT:
59   08 A1   FF71FFFF 8F   CB  004C    47         BICL3   #^XFF71FFFF,8(R1),R9 ;CHECK FOR CONTROL BUS ERRORS
                    01   13     0055    48         BEQL    TEST_WAIT            ;IF NO ERRORS, CHECK WAIT
                                0057    49         HALT                         ;IF ERRORS, HALT
                                0058    50 TEST_WAIT:
          EF 04 A0   07   E1    0058    51         BBC     #7,4(R0),WAIT        ;TEST READY AND WAIT
                                005D    52         RSB
                                005E    53
                                005E    54
                                005E    55
                                005E    56 XFER:
                    EC   10     005E    57         BSB     WAIT                 ;WAIT FOR DRIVE READY
```

```
                    60   13   9A  0060     58          MOVZBL  #^X13,(R0)              ;WHEN READY SEND PACK ACK
                         E7   10  0063     59          BSB     WAIT                    ;WAIT FOR READY
                24 A0    00001000 8F  C8  0065     60          BISL2   #^X1000,^X24(R0)                ;SET 16BIT FORMAT
              0800 C1    57   0F   09  EF  006D     61          EXTZV   #9,#15,R7,^X800(R1)     ;LOAD MAP
               0800 C1    80000000 8F  C8  0074     62          BISL2   #^X80000000,^X800(R1)   ;SET VALID
                0C A1    57   09   00  EF  007D     63          EXTZV   #0,#9,R7,^XC(R1)        ;LOAD VAR
                10 A1     00000200 8F  CE  0083     64          MNEGL   #^X200,^X10(R1)         ;LOAD BCR
                                           0088     65
                                           0088     66
                                           0088     67 LBNCVT:
                         0F   BB  0088     68          PUSHR   #^XF                    ;SAVE R0,R1
00000000'EF     07   18 A0   3A  008D     69          LOCC    ^X18(R0),#DT_LIMIT,DRIVE_TYPES  ;CHECK DRIVE TYPE AGAINST LIST
                         01   12  0096     70          BNEQ    GOOD_DRIVE_TYPE
                         00   98  0098     71          HALT                            ;HALT IF BAD DRIVE TYPE
                                           0099     72 GOOD_DRIVE_TYPE:
                    53   07 A1   98  0099     73          CVTBL   DT_LIMIT(R1),R3         ;R3_SELECTED SEC/TRACK
                    52   0E A1   98  009D     74          CVTBL   2+DT_LIMIT(R1),R2       ;R2_SELECTED TRACK/CYL
                         52   53  C4  00A1     75          MULL2   R3,R2                   ;R2_SELECTED SEC/CYL
                50   28   6E  C1  00A4     76          ADDL3   (SP),#^X28,R0           ;R0_DESIREC CYL REG ADDRESS
              58   60   58   52  7B  00A8     77          EDIV    R2,R8,(R0),R8           ;DRIVE DESIRED_CYL REG_CYL #
              58   59   58   53  7B  00AD     78          EDIV    R3,R8,R9,R8             ;R9_TRACK # , R8_SEC #
                         0F   BA  0082     79          POPR    #^XF                    ;RESTORE R0,R1,R2,R3
                                           0084     80
                                           0084     81
                14 A0    59   08  78  0084     82          ASHL    #8,R9,^X14(R0)          ;DRIVE ADDRESS REG_TRACK #
                     14 A0   58   C0  0089     83          ADDL2   R8,^X14(R0)             ;DRIVE ADDRESS REG_SEC #
                         60   39   9A  008D     84          MOVZBL  #^X39,(R0)              ;ISSUE READ COMMAND TO DRIVE
                                           00C0     85
                                           00C0     86
                                           00C0     87
                                           00C0     88
                                           00C0     89 TST_DTBSY:
                  FB 08 A1    1F   E0  00C0     90          BBS     #31,8(R1),TST_DTBSY     ;WAIT FOR DTBUSY TO CLEAR
59      08 A1     2000 8F   AB  00C5     91          BICW3   #^X2000,8(R1),R9                ;ISOLATE ERROR BITS
                      04 A1   D6  00CC     92          INCL    4(R1)
                              05  00CF     93          RSB
                                          00D0     94
                                          00D0     95
                                          00D0     96 DRIVE_TYPES:
                              11  00D0     97          .BYTE   ^X11                    ;RP05
                              12  00D1     98          .BYTE   ^X12                    ;RP06
                              14  00D2     99          .BYTE   ^X14                    ;RM03
                              15  00D3    100          .BYTE   ^X15                    ;RM02
                              16  00D4    101          .BYTE   ^X16                    ;RM80
                              17  00D5    102          .BYTE   ^X17                    ;RM05
                              20  00D6    103          .BYTE   ^X20                    ;RP07
                                  00D7    104 SEC_TRACK:
                              16  00D7    105          .BYTE   22                      ;RP05
                              16  00D8    106          .BYTE   22                      ;RP06
                              20  00D9    107          .BYTE   32                      ;RM03
                              20  00DA    108          .BYTE   32                      ;RM02
                              1F  00DB    109          .BYTE   31                      ;RM80
                              20  00DC    110          .BYTE   32                      ;RM05
                              32  00DD    111          .BYTE   50                      ;RP07
                                  00DE    112 TRACK_CYL:
                              13  00DE    113          .BYTE   19                      ;RP05
                              13  00DF    114          .BYTE   19                      ;RP06
```

```
05  00E0   115          .BYTE   5                       ;RM03
05  00E1   116          .BYTE   5                       ;RM02
0E  00E2   117          .BYTE   14                      ;RM80
13  00E3   118          .BYTE   19                      ;RM05
48  00E4   119          .BYTE   72                      ;RP07
       00E5   120
       00E5   121          .END
```

.MAIN.
Symbol table

```
DEVADD              0000003F R      01
DRIVER              0000002C R      01
DRIVE_TYPES         000000D0 R      01
DT_LIMIT          = 00000007
GOOD_DRIVE_TYPE     00000099 R      01
LBNCNT              000000BB R      01
NOERR               0000003B R      01
OK                  00000016 R      01
SEC_TRACK           000000D7 R      01
START               00000002 R      01
TEST_WAIT           00000058 R      01
TRACK_CYL           000000DE R      01
TST_DTBSY           000000C0 R      01
WAIT                0000004C R      01
XFER                0000005E R      01
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	000000E5 (229.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+--------------------------+
! Performance indicators !
+--------------------------+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	21	00:00:00.03	00:00:00.23
Command processing	27	00:00:00.18	00:00:00.47
Pass 1	123	00:00:00.60	00:00:00.73
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	87	00:00:00.27	00:00:00.34
Symbol table output	3	00:00:00.02	00:00:00.03
Psect synopsis output	5	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	274	00:00:01.12	00:00:01.83

The working set limit was 171 pages.
2448 bytes (5 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 15 non-local and 0 local symbols.
121 source lines were read in Pass 1, producing 9 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

+--------------------------+
! Macro library statistics !
+--------------------------+

Macro library name	Macros defined
DRA2:[SYSLIB]STARLET.MLB;1	0

0 GETS were required to define 0 macros.

ZZ-ECOAD-2.0 22-JAN-1981 Fiche 1 Frame G1 Sequence 6
.MAIN. 22-JAN-1981 16:07:07 VAX-11 Macro V02.45 Page 5
VAX-11 Macro Run Statistics 22-JAN-1981 16:06:56 _DBB1:[USER]DBROM.MAR;1 (1)

There were no errors, warnings or information messages.

/LIS DBROM

Table of contents

```
0000     1          .TITLE  TU58ROM
0000     2          .IDENT  /X00001/
0000     3
0000     4 ;
0000     5 ;                         COPYRIGHT (c) 1979 BY
0000     6 ;                 DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
0000     7 ;
0000     8 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
0000     9 ; ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
0000    10 ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
0000    11 ; COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000    12 ; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
0000    13 ; TRANSFERRED.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
0000    16 ; AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000    21 ;
0000    22 ;++
0000    23 ;
0000    24 ; FACILITY:
0000    25 ;
0000    26 ;       11/750 device-specific TU58 bootstrap ROM
0000    27 ;
0000    28 ; ABSTRACT:
0000    29 ;
0000    30 ;       This ROM code reads block 0 from a volume mounted on a boot
0000    31 ;       device and transfers control to the code in that boot block.
0000    32 ;
0000    33 ; AUTHOR:
0000    34 ;
0000    35 ;       Carol Peters    23-Aug-1979
0000    36 ;
0000    37 ; REVISION HISTORY:
0000    38 ;
0000    39 ;       Robert Rappaport        13 Sept 1979
0000    40 ;               Revised to conform to new BOOTBLOCK code.
0000    41 ;       Don Monroe      4 October 1979
0000    42 ;               Revised to prevent TU58 INIT on read of every block.
0000    43 ;       Charlie McDowell        26 November 1979
0000    44 ;               Change TU58_DEV_TYPE to 64.
0000    45 ;
0000    46 ;--
```

```
                        0000     48          .SBTTL  Declarations
                        0000     49
                        0000     50          .DEFAULT DISPLACEMENT, WORD
                        0000     51
                        0000     52 ;
                        0000     53 ; Macro definitions
                        0000     54 ;
                        0000     55
                        0000     56          $PRDEF                          ; Processor registers.
                        0000     57          $SSDEF                          ; Completion status codes.
                        0000     58
                        0000     59 ;
                        0000     60 ; Equated symbols
                        0000     61 ;
                        0000     62
00000040                0000     63 TU58_DEV_TYPE    = 64                    ; Device type is TU58 tape
00000001                0000     64 TU58_BREAK       = 1                     ; Hardware break function.
00000004                0000     65 TU58_INIT        = 4                     ; Hardware init function.
00000007                0000     66 TU58_READY       = 7                     ; Hardware ready bit.
00000010                0000     67 TU58_CONTINUE    = 16                    ; Hardware continue function.
                        0000     68
00000008                0000     69 PRIBOO_LOADADDR = 8                      ; Offset to load address for primary
                        0000     70                                          ; bootstrap (expressed as offset
                        0000     71                                          ; from SP).
FFFFFE0C                0000     72 BOOT_CODE_START = 12-^X200               ; Offset to start of boot code
                        0000     73                                          ; in the boot block.
                        0000     74 ;
                        0000     75 ; First two bytes of ROM must be a 2-character ASCII mnemonic that
                        0000     76 ; identifies the ROM's device. The characters are stored in reverse
                        0000     77 ; order.
                        0000     78 ;
                        0000     79
                        0000     80 TU58_DEV_NAME:
                4444    0000     81          .WORD   ^A/DD/                  ; 'DD'', in reverse.
                        0002     82
                        0002     83          $DEFINI CMD
                        0000     84
                        0000     85 $DEF     CMD_B_FLAG                      ; Flag byte.
00000001                0000     86          .BLKB   1
                        0001     87 $DEF     CMD_B_BYTECOUNT                 ; Number of bytes in packet.
00000002                0001     88          .BLKB   1
                        0002     89 $DEF     CMD_B_OPCODE                    ; Opcode byte.
00000003                0002     90          .BLKB   1
                        0003     91 $DEF     CMD_B_MODIFIER                  ; Opcode modifier.
00000004                0003     92          .BLKB   1
                        0004     93 $DEF     CMD_B_UNITNUM                   ; Device unit number.
00000005                0004     94          .BLKB   1
                        0005     95 $DEF     CMD_B_SWITCHES                  ; Unused byte.
00000006                0005     96          .BLKB   1
                        0006     97 $DEF     CMD_B_SEQUENCE1                 ; Low order sequence number.
00000007                0006     98          .BLKB   1
                        0007     99 $DEF     CMD_B_SEQUENCE2                 ; High order sequence number.
00000008                0007    100          .BLKB   1
                        0008    101 $DEF     CMD_W_BYTECOUNT                 ; Byte count.
0000000A                0008    102          .BLKW   1
                        000A    103 $DEF     CMD_K_PACKETLEN                 ; Length of message packet.
                        000A    104
```

```
                        000A   105          $DEFEND CMD
```

```
                        0002   107          .SBTTL  ROM Control Subroutine
                        0002   108
                        0002   109 ;++
                        0002   110 ; Functional description:
                        0002   111 ;
                        0002   112 ;       The subroutine reads block 0 off the boot device's volume,
                        0002   113 ;       and transfers control to the start address of the boot block.
                        0002   114 ;
                        0002   115 ; Inputs:
                        0002   116 ;
                        0002   117 ;       R1      - physical address of a MASSBUS adapter (MBA0)
                        0002   118 ;       R2      - physical address of a UNIBUS I/O page
                        0002   119 ;       R3      - unit number of the boot device
                        0002   120 ;       R5      - software boot control flags
                        0002   121 ;
                        0002   122 ;       SP      - <base_address + ^X200> of 64kb of good memory
                        0002   123 ;
                        0002   124 ; Implicit inputs:
                        0002   125 ;
                        0002   126 ;       LBN of boot block is 0
                        0002   127 ;       boot block is to be read into -200(SP)
                        0002   128 ;       transfer address of boot block code is at BOOT_START_CODE(SP)
                        0002   129 ;
                        0002   130 ; Outputs:
                        0002   131 ;
                        0002   132 ;       R0      - type of device
                        0002   133 ;       R1      - 0
                        0002   134 ;       R2      - 0
                        0002   135 ;       R3      - unit number of the boot device
                        0002   136 ;       R5      - software boot control flags
                        0002   137 ;       R6      - address of the device-specific read block routine
                        0002   138 ;
                        0002   139 ;       SP      - <base_address + ^X200> of 64kb of good memory
                        0002   140 ;
                        0002   141 ;       This routine preserves registers R3, R5, R10-R11, AP, and SP.
                        0002   142 ;
                        0002   143 ;--
                        0002   144
5E    00000100 8F   C0  0002   145          ADDL    #^X100,SP               ; Make room for a stack.
                 58   DD  0009   146          PUSHL   R8                      ; Save register that we modify.
                        000B   147
                        000B   148 ;
                        000B   149 ; Set up input registers and call a device-specific subroutine that
                        000B   150 ; reads in one block of the boot device. The block is block 0, the
                        000B   151 ; boot block.  This block will be read into the base of the 64KB of
                        000B   152 ; good memory.  We pass the physical address of the base of the 64KB
                        000B   153 ; on the top of the stack.
                        000B   154 ;
                        000B   155
                        000B   156
                        000B   157 ;
                        000B   158 ; Registers and parameter set up are as follows:
                        000B   159 ;
                        000B   160 ;       R3      - device unit number
                        000B   161 ;       (SP)    - physical buffer address
                        000B   162 ;       R8      - LBN 0
                        000B   163 ;
```

```
                                        0008  164
                    FD04 CE    9F       0008  165          PUSHAB  -^X300+4(SP)             ; Push physical address of base
                                        000F  166                                          ; of 64KB of memory onto stack.
                         58    D4       000F  167          CLRL    R8                      ; LBN to read is 0.
                       35'AF   16       0011  168          JSB     B^INIT_TU58             ; Call the driver subroutine.
                       01 50   E8       0014  169          BLBS    R0,10$                  ; Branch on successful read.
                               00       0017  170          HALT                            ; Otherwise, halt.
                                        0018  171
                                        0018  172 ;
                                        0018  173 ; Set up the remaining registers needed by VMB and by the boot block.
                                        0018  174 ; Notice that since this device is not on the UNIBUS or MASSBUS, the
                                        0018  175 ; adapter and bus related registers are zeroed. Then transfer control
                                        0018  176 ; to the boot block code.
                                        0018  177 ;
                                        0018  178
                                        0018  179 10$:
                         51    7C       0018  180          CLRQ    R1                      ; Clear UNIBUS- and MASSBUS-
                                        001A  181                                          ; specific VMB inputs.
                    56   4F'AF   9E       001A  182          MOVAB   B^BOOSTU58_QIO,R6       ; Store address of driver.
               50   00000040 8F       D0       001E  183          MOVL    #TU58_DEV_TYPE,R0       ; Load device type.
                         8E    D5       0025  184          TSTL    (SP)+                   ; Remove physical addr from stack.
                      58   8E    D0       0027  185          MOVL    (SP)+,R8                ; Restore register.
               5E   00000100 8F       C2       002A  186          SUBL    #^X100,SP               ; Restore stack.
                    FE0C CE    17       0031  187          JMP     BOOT_CODE_START(SP)     ; Give control to boot block.
```

```
                        0035   189          .SBTTL  BOO$TU58_QIO - primitive device-dependent read driver
                        0035   190
                        0035   191 ;++
                        0035   192 ; Functional description:
                        0035   193 ;
                        0035   194 ;       Reads 1 block of data from a TU58 into physical memory.
                        0035   195 ;
                        0035   196 ; Inputs:
                        0035   197 ;
                        0035   198 ;       R3      - unit number of boot device
                        0035   199 ;       4(SP)   - starting physical address of the transfer
                        0035   200 ;       R8      - LBN to transfer from boot device
                        0035   201 ;
                        0035   202 ; Outputs:
                        0035   203 ;
                        0035   204 ;       R0      - SS$_NORMAL or 0
                        0035   205 ;       This routine preserves registers R3-R6, R10-R11, AP, and SP.
                        0035   206 ;
                        0035   207 ;--
                        0035   208
                        0035   209 INIT_TU58:                                  ; Primitive device driver
                        0035   210
                        0035   211 ;
                        0035   212 ; Get the attention of the device by setting the break condition. Then
                        0035   213 ; wait for 2 character transmissions to ensure that the TU58 notices
                        0035   214 ; the break. Then clear the receive buffer ak and send 2 initialization
                        0035   215 ; commands.
                        0035   216 ;
                        0035   217
                1E  01  DA  0035   218          MTPR    #TU58_BREAK,#PR$_CSTS   ; Set break condition.
                    52  D4  0038   219          CLRL    R2                      ; Set up null characters.
                  0081  30  003A   220          BSBW    XMIT_TWO_CHARS          ; Wait for 2 character xmits.
                52  1D  DB  003D   221          MFPR    #PR$_CSRD,R2            ; Clear receive buffer.
              0404  8F  3C  0040   222          MOVZWL  #TU58_INIT@8+TU58_INIT,-; Load 2 INIT opcodes into the
                    52      0044   223                  R2                      ; low 2 bytes of R2.
                    77  10  0045   224          BSBB    XMIT_TWO_CHARS          ; Transmit the 2 INIT codes.
                        0047   225
                        0047   226 ;
                        0047   227 ; Now wait for the TU58 to send CONTINUE status back to the host.
                        0047   228 ;
                        0047   229
                        0047   230 WAIT_FOR_CONT:                               ; Wait for continue.
                57  1D  DB  0047   231          MFPR    #PR$_CSRD,R7            ; Get receive data from TU58.
                10  57  91  004A   232          CMPB    R7,#TU58_CONTINUE       ; Is data "continue"?
                    F8  12  004D   233          BNEQ    WAIT_FOR_CONT           ; No. Look again.
                        004F   234
                        004F   235 ;
                        004F   236 ; Initialize a checksum register to zero. Then write the device's unit
                        004F   237 ; number into a canned command packet.
                        004F   238 ;
                        004F   239 BOO$TU58_QIO:
              0060  8F  BB  004F   240          PUSHR   #^M<R5,R6>              ; Save 2 registers for temps.
            55  0C  AE  D0  0053   241          MOVL    4+8(SP),R5              ; Move physical address to R5.
                    59  D4  0057   242          CLRL    R9                      ; Initialize the checksum.
        50  00E2'CF  9E  0059   243          MOVAB   READ_COMMAND,R0         ; Get read command block addr.
                    53  D5  005E   244          TSTL    R3                      ; Is this device 0?
                    04  13  0060   245          BEQL    10$                     ; Yes. Branch.
```

```
                04 A0   53   9A  0062   246          MOVZBL  R3,B^CMD_B_UNITNUM(R0)  ; No. Load real device number.
                                0066   247
                                0066   248
                                0066   249 ;
                                0066   250 ; Transmit the canned part of the command packet to the TU58. The packet
                                0066   251 ; tells the device to read a block of data from the unit into memory.
                                0066   252 ;
                                0066   253
                                0066   254 10$:
                   51   05   9A  0066   255          MOVZBL  #CMD_K_PACKETLEN/2,R1   ; Get size of static block part.
                                0069   256
                                0069   257 20$:
                   52   80   3C  0069   258          MOVZWL  (R0)+,R2                ; Get 2 characters from block.
                        49   10  006C   259          BSBB    XMIT_AND_UPDSUM         ; Transmit with checksum update.
                     F8 51   F5  006E   260          SOBGTR  R1,20$                  ; Loop if more of fixed block.
                                0071   261
                                0071   262 ;
                                0071   263 ; Now transmit the variable part of the packet -- i.e., the block number
                                0071   264 ; and the accumulated checksum value.
                                0071   265 ;
                                0071   266
                   52   58   3C  0071   267          MOVZWL  R8,R2                   ; Get starting LBN.
                        41   10  0074   268          BSBB    XMIT_AND_UPDSUM         ; Transmit with checksum update.
                   52   59   3C  0076   269          MOVZWL  R9,R2                   ; Get the accumulated checksum.
                        3C   10  0079   270          BSBB    XMIT_AND_UPDSUM         ; Transmit with checksum update.
                                007B   271
                                007B   272 ;
                                007B   273 ; Collect a read packet from the device. It should be a read data
                                007B   274 ; packet. Otherwise, it will be an end of data packet. If a read data
                                007B   275 ; packet, read one byte at a time and store the bytes in physical
                                007B   276 ; memory.
                                007B   277 ;
                                007B   278
                                007B   279 NEXT_PACKET:                             ; Collect read data packets.
                        53   10  007B   280          BSBB    RECV_TWO_CHARS          ; Read 2 characters from packet.
                        52   97  007D   281          DECB    R2                      ; Look at packet type.
                        0F   12  007F   282          BNEQ    END_OF_DATA             ; Branch on end of data packet.
                56      51   9A  0081   283          MOVZBL  R1,R6                   ; Get byte count of data packet.
                                0084   284
                                0084   285 ;
                                0084   286 ; A data packet. Read the data into memory.
                                0084   287 ;
                                0084   288
                                0084   289 NEXT_BYTE:                               ; Obtain next byte of data.
                        4C   10  0084   290          BSBB    RECV_ONE_CHAR           ; Read 1 character.
                85      51   90  0086   291          MOVB    R1,(R5)+                ; Write into memory.
                  F8    56   F5  0089   292          SOBGTR  R6,NEXT_BYTE            ; Get another if more to get.
                                008C   293
                                008C   294 ;
                                008C   295 ; The byte count is exhausted. Bypass the checksum and branch back to
                                008C   296 ; get the next packet.
                                008C   297 ;
                                008C   298
                                008C   299 READ_CHECKSUM:                           ; Bypass the checksum.
                        42   10  008C   300          BSBB    RECV_TWO_CHARS          ; Read 2 characters.
                        EB   11  008E   301          BRB     NEXT_PACKET             ; Go read next packet.
                                0090   302
```

```
                                0090   303 ;
                                0090   304 ; End of data packet: confirm that the device gave as many bytes of read
                                0090   305 ; data as requested, and that the device reports no error in the read.
                                0090   306 ; Return with a status code.
                                0090   307 ;
                                0090   308
                                0090   309 END_OF_DATA:                                ; End of data packet.
                 50   01   3C  0090   310          MOVZWL  #SS$_NORMAL,R0              ; Assume successsful read.
          55  00000200 8F   C2  0093   311          SUBL    #512,R5                     ; Recompute starting address.
           0C AE        55   D1  009A   312          CMPL    R5,4+8(SP)                  ; Same as original address?
                        10   12  009E   313          BNEQ    ROM_ERROR                   ; No. Error in read.
                        2E   10  00A0   314          BSBB    RECV_TWO_CHARS              ; Bypass the checksum.
                        51   D5  00A2   315          TSTL    R1                          ; Error in read?
                                00A4   316 ;        BGEQ    RETURN                      ; No. Return with success.
                        0A   19  00A4   317          BLSS    ROM_ERROR                   ; Branch if status error
                 55     05   D0  00A6   318          MOVL    #5,R5                       ; Setup loop count to read
                        25   10  00A9   319 10$:     BSBB    RECV_TWO_CHARS              ; rest of the end packet
                   FB   55   F5  00AB   320          SOBGTR  R5,10$                      ; Read 10 characters
                        02   11  00AE   321          BRB     RETURN                      ; Exit
                                00B0   322
                                00B0   323 ROM_ERROR:                                  ; Error in read.
                        50   D4  00B0   324          CLRL    R0                          ; Return error status code.
                                00B2   325
                                00B2   326 RETURN:                                     ; Restore registers and return.
                 0060 8F        BA  00B2   327          POPR    #^M<R5,R6>                  ; Restore registers.
                                05  00B6   328          RSB                                 ; Return.
```

```
                                0087   330          .SBTTL  Transmit and receive wait loops
                                0087   331
                                0087   332 ;
                                0087   333 ; Update the accumulated checksum, and then transmit 2 characters to
                                0087   334 ; the device.
                                0087   335 ;
                                0087   336 ; Inputs:
                                0087   337 ;
                                0087   338 ;       R2      - contains 2 characters to send to TU58
                                0087   339 ;       R9      - accumulated checksum
                                0087   340 ;
                                0087   341
                                0087   342 XMIT_AND_UPDSUM:                          ; Transmit and update checksum.
          59   52   A0          0087   343          ADDW    R2,R9                   ; Update checksum.
               02   1E          008A   344          BCC     XMIT_TWO_CHARS          ; If no overflow, branch.
               59   B6          008C   345          INCW    R9                      ; Overflow: add in carry bit.
                                008E   346
                                008E   347 ;
                                008E   348 ; Send 2 characters to device: the trick here is a branch subroutine to
                                008E   349 ; the label, and then a fall through to the label, thus 2 passes through
                                008E   350 ; the loop without a counter.
                                008E   351 ;
                                008E   352
                                008E   353 XMIT_TWO_CHARS:                          ; Transmit two characters.
               05   10          008E   354          BSBB    WAIT_FOR_XMIT           ; Call subroutine to transmit 1.
                                00C0   355
                                00C0   356 ;
                                00C0   357 ; Transmit 1 character: register usage is as follows:
                                00C0   358 ;
                                00C0   359 ;       R2      - starts with 2 8-bit characters
                                00C0   360 ;       R7      - contents of device's transmit status register
                                00C0   361 ;
                                00C0   362
                                00C0   363 XMIT_NEXT_CHAR:                          ; Loop to transmit next char.
     52   52   F8 8F   78       00C0   364          ASHL    #-8,R2,R2               ; Get the next character.
                                00C5   365
                                00C5   366 WAIT_FOR_XMIT:                           ; Loop waiting for transmit.
          57   1E   DB          00C5   367          MFPR    #PR$_CSTS,R7            ; Get transmit status.
     F9   57   07   E1          00C8   368          BBC     #TU58_READY,R7,-        ; If not yet ready, loop
                                00CC   369                  WAIT_FOR_XMIT           ; back again.
          1F   52   DA          00CC   370          MTPR    R2,#PR$_CSTD            ; Transmit it.
                    05          00CF   371          RSB                             ; Return.
                                00D0   372
                                00D0   373 ;
                                00D0   374 ; Receive 2 characters from the device: the trick here is a branch
                                00D0   375 ; subroutine to the label, and then a fall through to the label, thus 2
                                00D0   376 ; passes through the loop without a counter.
                                00D0   377 ;
                                00D0   378
                                00D0   379 RECV_TWO_CHARS:                          ; Receive 2 characters and wait.
               00   10          00D0   380          BSBB    RECV_ONE_CHAR           ; Call subroutine to receive 1.
                                00D2   381
                                00D2   382 ;
                                00D2   383 ; Receive 1 character: register usage is as follows:
                                00D2   384 ;
                                00D2   385 ;       R1      - receives the character
                                00D2   386 ;       R2      - collects the previously received character
```

```
                                00D2   387 ;       R7      - contents of device's transmit status register
                                00D2   388 ;
                                00D2   389
                                00D2   390 RECV_ONE_CHAR:                       ; Receive one character.
                52    51    9A  00D2   391          MOVZBL  R1,R2               ; Save previous received byte.
                57    1C    DB  00D5   392          MFPR    #PR$_CSRS,R7        ; Get receive status.
              F6 57    07    E1  00D8   393          BBC     #TU58_READY,R7,-    ; If not yet ready, loop
                                00DC   394                  RECV_ONE_CHAR       ; back again.
                51    1D    DB  00DC   395          MFPR    #PR$_CSRD,R1        ; Get another character.
                      CF    19  00DF   396          BLSS    ROM_ERROR           ; Branch on receive error.
                            05  00E1   397          RSB                         ; Return.
```

```
                                00E2   399          .SBTTL  Declarations at end, and end statement
                                00E2   400
                                00E2   401
                                00E2   402 ;
                                00E2   403 ; A canned read command packet for the TU58.
                                00E2   404 ;
                                00E2   405
                                00E2   406 READ_COMMAND:
                             02 00E2   407          .BYTE   2                       ; Command packet flag.
                             0A 00E3   408          .BYTE   ^XA                     ; Number of bytes in message.
                             02 00E4   409          .BYTE   2                       ; TU58 read opcode.
                00 00 00 00 00 00E5   410          .BYTE   0,0,0,0,0               ; Modifier, unit number,
                                00EA   411                                          ; switches, and sequence number.
                          02 00 00EA   412          .BYTE   0,2                     ; Number of bytes to read, low
                                00EC   413                                          ; order, then high order (=512).
                                00EC   414
                                00EC   415          .END
```

```
BIT...              = 00000E0A
BOOSTU58_QIO          0000004F R      01
BOOT_CODE_START=      FFFFFE0C
CMD_B_BYTECOUNT       00000001
CMD_B_FLAG            00000000
CMD_B_MODIFIER        00000003
CMD_B_OPCODE          00000002
CMD_B_SEQUENCE1       00000006
CMD_B_SEQUENCE2       00000007
CMD_B_SWITCHES        00000005
CMD_B_UNITNUM         00000004
CMD_K_PACKETLEN       0000000A
CMD_W_BYTECOUNT       00000008
END_OF_DATA           00000090 R      01
GBL...              = 00000000
INIT_TU58             00000035 R      01
NEXT_BYTE             00000084 R      01
NEXT_PACKET           0000007B R      01
PRSS_SID_ECO        = 00000008
PRSS_SID_PL         = 00000004
PRSS_SID_SN         = 0000000C
PRSS_SID_TYPE       = 00000008
PRSV_SID_ECO        = 00000010
PRSV_SID_PL         = 0000000C
PRSV_SID_SN         = 00000000
PRSV_SID_TYPE       = 00000018
PRS_ACCR            = 00000029
PRS_ACCS            = 00000028
PRS_ASTLVL          = 00000013
PRS_CADR            = 00000025
PRS_CAER            = 00000027
PRS_CMIERR          = 00000017
PRS_CSRD            = 0000001D
PRS_CSRS            = 0000001C
PRS_CSTD            = 0000001F
PRS_CSTS            = 0000001E
PRS_ESP             = 00000001
PRS_ICCS            = 00000018
PRS_ICR             = 0000001A
PRS_IPL             = 00000012
PRS_ISP             = 00000004
PRS_KSP             = 00000000
PRS_MAPEN           = 00000038
PRS_MCESR           = 00000026
PRS_NICR            = 00000019
PRS_P0BR            = 00000008
PRS_P0LR            = 00000009
PRS_P1BR            = 0000000A
PRS_P1LR            = 0000000B
PRS_PCBB            = 00000010
PRS_PME             = 0000003D
PRS_RXCS            = 00000020
PRS_RXDB            = 00000021
PRS_SBIER           = 00000034
PRS_SBIFS           = 00000030
PRS_SBIMT           = 00000033
PRS_SBIQC           = 00000036
PRS_SBIS            = 00000031
PRS_SBISC           = 00000032
PRS_SBITA           = 00000035
PRS_SBR             = 0000000C
PRS_SCBB            = 00000011
PRS_SID             = 0000003E
PRS_SID_TYP750      = 00000002
PRS_SID_TYP780      = 00000001
PRS_SID_TYP7ZZ      = 00000003
PRS_SID_TYPMAX      = 00000003
PRS_SIRR            = 00000014
PRS_SISR            = 00000015
PRS_SLR             = 0000000D
PRS_SSP             = 00000002
PRS_TBDR            = 00000024
PRS_TBIA            = 00000039
PRS_TBIS            = 0000003A
PRS_TODR            = 0000001B
PRS_TXCS            = 00000022
PRS_TXDB            = 00000023
PRS_UBRESET         = 00000037
PRS_USP             = 00000003
PRS_WCSA            = 0000002C
PRS_WCSD            = 0000002D
PRIBOO_LOADADDR=      00000008
READ_CHECKSUM         0000008C R      01
READ_COMMAND          000000E2 R      01
RECV_ONE_CHAR         000000D2 R      01
RECV_TWO_CHARS        000000D0 R      01
RETURN                000000B2 R      01
ROM_ERROR             000000B0 R      01
SS$_ABORT           = 0000002C
SS$_ACCONFLICT      = 00000800
SS$_ACCVIO          = 0000000C
SS$_ACPVAFUL        = 000002FC
SS$_ARTRES          = 00000474
SS$_ASTFLT          = 0000040C
SS$_BADATTRIB       = 00000034
SS$_BADCHKSUM       = 00000808
SS$_BADESCAPE       = 0000003C
SS$_BADFILEHDR      = 00000810
SS$_BADFILENAME=      00000818
SS$_BADFILEVER      = 00000820
SS$_BADIMGHDR       = 00000044
SS$_BADIRECTORY=      00000828
SS$_BADISD          = 00002004
SS$_BADPARAM        = 00000014
SS$_BADQFILE        = 000003BC
SS$_BADQUEUEHDR=      00000394
SS$_BADSTACK        = 00000284
SS$_BADVEC          = 00002064
SS$_BEGOFFILE       = 00000938
SS$_BLOCKCNTERR=      00000940
SS$_BREAK           = 00000414
SS$_BUFBYTALI       = 0000030C
SS$_BUFFEROVF       = 00000601
SS$_BUFNOTALIGN=      00000324
SS$_BUGCHECK        = 000002A4
SS$_CANCEL          = 00000830
SS$_CHAINW          = 00000C0B
SS$_CHANINTLK       = 0000004C
SS$_CLIFRCEXT       = 00000980
SS$_CMODSUPR        = 0000041C
SS$_CMODUSER        = 00000424
SS$_COMPHARD        = 000020C4
SS$_COMPAT          = 0000042C
SS$_CONNECFAIL      = 000020DC
SS$_CONTINUE        = 00000001
SS$_CONTROLC        = 00000651
SS$_CONTROLO        = 00000609
SS$_CONTROLY        = 00000611
SS$_CREATED         = 00000619
SS$_CTRLERR         = 00000054
SS$_DATACHECK       = 0000005C
SS$_DATAOVERUN      = 00000838
SS$_DEBUG           = 0000046C
SS$_DECOVF          = 000004A4
SS$_DEVACTIVE       = 000002C4
SS$_DEVALLOC        = 00000840
SS$_DEVALRALLOC=      00000641
SS$_DEVASSIGN       = 00000848
SS$_DEVCMDERR       = 0000032C
SS$_DEVFOREIGN      = 00000064
SS$_DEVICEFULL      = 00000850
SS$_DEVINACT        = 000020D4
SS$_DEVMOUNT        = 0000006C
SS$_DEVNOTALLOC=      00000858
SS$_DEVNOTMBX       = 00000074
SS$_DEVNOTMOUNT=      0000007C
SS$_DEVOFFLINE      = 00000084
SS$_DEVREQERR       = 00000334
SS$_DIRFULL         = 00000860
SS$_DISCONNECT      = 0000204C
SS$_DRVERR          = 0000008C
SS$_DUPDSKQUOTA=      000003DC
SS$_DUPFILENAME=      00000868
SS$_DUPLNAM         = 00000094
SS$_ENDOFFILE       = 00000870
SS$_ENDOFTAPE       = 00000878
SS$_ENDOFUSRLBL=      00000970
SS$_ENDOFVOLUME=      000009A0
SS$_EOTIN           = 00000C03
SS$_EXCPUTIM        = 000020AC
SS$_EXDISKQUOTA=      000003EC
SS$_EXPORTQUOTA=      000003AC
SS$_EXQUOTA         = 0000001C
SS$_EXTIDXFILE      = 00000880
SS$_FCPREADERR      = 00000888
SS$_FCPREPSTN       = 00000988
SS$_FCPREWNDERR=      00000890
SS$_FCPSPACERR      = 00000898
SS$_FCPWRITERR      = 000008A0
SS$_FILACCERR       = 0000009C
SS$_FILALRACC       = 000000A4
```

```
SS$_FILELOCKED = 000008A8
SS$_FILENUMCHK = 00000880
SS$_FILEPURGED = 00000679
SS$_FILESEQCHK = 00000888
SS$_FILESTRUCT = 000008C0
SS$_FILNOTACC  = 00000CAC
SS$_FILNOTCNTG = 00000CAC
SS$_FILNOTEXP  = 00000084
SS$_FLTDIV     = 00000494
SS$_FLTDIV_F   = 000004BC
SS$_FLTOVF     = 0000048C
SS$_FLTOVF_F   = 000004B4
SS$_FLTUND     = 0000049C
SS$_FLTUND_F   = 000004C4
SS$_FORMAT     = 000000BC
SS$_GPTFULL    = 000000C4
SS$_GSDFULL    = 000000CC
SS$_HANGUP     = 000002CC
SS$_HEADERFULL = 000008C8
SS$_IDMISMATCH = 000003F4
SS$_IDXFILEFULL= 000008D0
SS$_ILLBLKNUM  = 000000DC
SS$_ILLCNTRFUNC= 000000E4
SS$_ILLEFC     = 000000EC
SS$_ILLIOFUNC  = 000000F4
SS$_ILLLBLAST  = 00000968
SS$_ILLPAGCNT  = 000000FC
SS$_ILLSEQOP   = 000002DC
SS$_ILLSER     = 00000104
SS$_ILLUSRLBLRD= 00000958
SS$_ILLUSRLBLWT= 00000960
SS$_INCVOLLABEL= 0000010C
SS$_INSFARG    = 00000114
SS$_INSFBUFDP  = 0000033C
SS$_INSFMAPREG = 00000344
SS$_INSFMEM    = 00000124
SS$_INSFRAME   = 0000012C
SS$_INSFSPTS   = 00002044
SS$_INSFWSL    = 0000011C
SS$_INTDIV     = 00000484
SS$_INTERLOCK  = 0000038C
SS$_INTOVF     = 0000047C
SS$_INVLOGIN   = 0000209C
SS$_IVADDR     = 00000134
SS$_IVBUFLEN   = 0000034C
SS$_IVCHAN     = 0000013C
SS$_IVCHAR     = 000020CC
SS$_IVCHNLSEC  = 0000026C
SS$_IVDEVNAM   = 00000144
SS$_IVGSDNAM   = 0000014C
SS$_IVLOGNAM   = 00000154
SS$_IVLOGTAB   = 0000015C
SS$_IVLVEC     = 0000203C
SS$_IVMODE     = 00000354
SS$_IVPROTECT  = 000002F4
SS$_IVQUOTAL   = 00000164
SS$_IVSECFLG   = 0000016C
SS$_IVSECIDCTL = 000002E4
SS$_IVSSRQ     = 00000174
SS$_IVSTSFLG   = 0000017C
SS$_IVTIME     = 00000184
SS$_LCKPAGFUL  = 000000D4
SS$_LENVIO     = 0000018C
SS$_LINEABRT   = 00000E02
SS$_LINKABORT  = 000020E4
SS$_LINKDISCON = 000020EC
SS$_LINKEXIT   = 000020F4
SS$_LKWSETFUL  = 00000194
SS$_MBFULL     = 000008D8
SS$_MBTOOSML   = 0000019C
SS$_MCHECK     = 000002BC
SS$_MCNOTVALID = 0000035C
SS$_MEDOFL     = 000001A4
SS$_MSGNOTFND  = 00000621
SS$_MTLBLLONG  = 00000304
SS$_MULTRMS    = 0000202C
SS$_MUSTCLOSEFL= 00000948
SS$_NOAQB      = 00000314
SS$_NODATA     = 000001AC
SS$_NODEVAVL   = 000009B0
SS$_NODISKQUOTA= 000003E4
SS$_NOHANDLER  = 000008F8
SS$_NOHOMEBLK  = 000008E0
SS$_NOIOCHAN   = 000001B4
SS$_NOLINKS    = 0000027C
SS$_NOLOGNAM   = 000001BC
SS$_NOMBX      = 00000274
SS$_NOMOREFILES= 00000930
SS$_NOMOREPROC = 000009A8
SS$_NONEXDRV   = 000001C4
SS$_NONEXPR    = 000008E8
SS$_NONLOCAL   = 000008F0
SS$_NOP1VA     = 00002024
SS$_NOPRIV     = 00000024
SS$_NOQFILE    = 000003C4
SS$_NORMAL     = 00000001
SS$_NOSHMBLOCK = 00000384
SS$_NOSIGNAL   = 00000900
SS$_NOSLOT     = 0000039C
SS$_NOSOLICIT  = 00000284
SS$_NOSUCHDEV  = 00000908
SS$_NOSUCHFILE = 00000910
SS$_NOSUCHNODE = 0000028C
SS$_NOSUCHOBJ  = 000020A4
SS$_NOSUCHSEC  = 00000978
SS$_NOSUCHUSER = 00002084
SS$_NOTAPEOP   = 00000264
SS$_NOTCREATOR = 00000384
SS$_NOTFILEDEV = 000001CC
SS$_NOTINSTALL = 00002014
SS$_NOTINTBLSZ = 000001D4
SS$_NOTLABELMT = 000001DC
SS$_NOTMODIFIED= 00000659
SS$_NOTNETDEV  = 000002EC
SS$_NOTRAN     = 00000629
SS$_NOTSQDEV   = 000001E4
SS$_NOTVOLSET  = 00000998
SS$_NOWRT      = 000003FC
SS$_OPCCUS     = 00000434
SS$_OPCDEC     = 0000043C
SS$_OPINCOMPL  = 000002D4
SS$_OPRABORT   = 00002084
SS$_OVRDSKQUOTA= 00000669
SS$_PAGOWNVIO  = 000001EC
SS$_PAGRDERR   = 00000444
SS$_PARITY     = 000001F4
SS$_PARTESCAPE = 000001FC
SS$_PATHLOST   = 000020FC
SS$_PFMBSY     = 00000204
SS$_PLHLDR     = 00000404
SS$_POWERFAIL  = 00000364
SS$_PRIVINSTALL= 00002054
SS$_PROTINSTALL= 0000205C
SS$_PROTOCOL   = 00002074
SS$_PSTFULL    = 0000020C
SS$_QFACTIVE   = 000003CC
SS$_QFNOTACT   = 000003D4
SS$_RADRMOD    = 0000044C
SS$_RDDELDATA  = 00000661
SS$_REJECT     = 00000294
SS$_RELINK     = 0000200C
SS$_REMOTE     = 00000649
SS$_REMRSRC    = 0000206C
SS$_RESIGNAL   = 00000918
SS$_RESULTOVF  = 00000214
SS$_ROPRAND    = 00000454
SS$_SECTBLFUL  = 0000021C
SS$_SHARTOOBIG = 0000201C
SS$_SHMGSNOTMAP= 0000036C
SS$_SHMNOTCNCT = 0000037C
SS$_SHRIDMISMAT= 000020BC
SS$_SHUT       = 0000208C
SS$_SSFAIL     = 0000045C
SS$_SUBRNG     = 000004AC
SS$_SUPERSEDE  = 00000631
SS$_SUSPENDED  = 000003A4
SS$_SYSVERDIF  = 00000671
SS$_TAPEPOSLOST= 00000224
SS$_TBIT       = 00000464
SS$_THIRDPARTY = 0000207C
SS$_TIMEOUT    = 0000022C
SS$_TOOMANYLNAM= 00000374
SS$_TOOMANYVER = 00000990
SS$_TOOMUCHDATA= 0000029C
SS$_UNASEFC    = 00000234
SS$_UNREACHABLE= 00002094
SS$_UNSAFE     = 0000023C
SS$_UNWIND     = 00000920
SS$_UNWINDING  = 00000928
SS$_VASFULL    = 00000244
SS$_VECFULL    = 00002034
```

```
SS$_VECINUSE    = 0000024C
SS$_VOLINV      = 00000254
SS$_WAITUSRLBL  = 00000950
SS$_WASCLR      = 00000001
SS$_WASECC      = 00000639
SS$_WASSET      = 00000009
SS$_WRITLCK     = 0000025C
SS$_WRONGACP    = 0000031C
TU58_BREAK      = 00000001
TU58_CONTINUE   = 00000010
TU58_DEV_NAME     00000000 R      01
TU58_DEV_TYPE   = 00000040
TU58_INIT       = 00000004
TU58_READY      = 00000007
WAIT_FOR_CONT     00000047 R      01
WAIT_FOR_XMIT     000000C5 R      01
XMIT_AND_UPDSUM   000000B7 R      01
XMIT_NEXT_CHAR    000000C0 R      01
XMIT_TWO_CHARS    000000BE R      01
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	000000EC (236.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	0000000A (10.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.05	00:00:00.60
Command processing	23	00:00:00.20	00:00:01.34
Pass 1	870	00:00:11.68	00:00:32.49
Symbol table sort	4	00:00:00.50	00:00:00.88
Pass 2	327	00:00:02.25	00:00:03.37
Symbol table output	37	00:00:00.26	00:00:00.47
Psect synopsis output	6	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1299	00:00:14.96	00:00:39.17

```
The working set limit was 276 pages.
53841 bytes (106 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 361 non-local and 4 local symbols.
415 source lines were read in Pass 1, producing 11 object records in Pass 2.
12 pages of virtual memory were used to define 12 macros.
```

```
                                        +---------------------------+
                                        ! Macro library statistics !
                                        +---------------------------+

Macro library name                       Macros defined
------------------                       --------------

_DRA2:[SYSLIB]STARLET.MLB;1                      11
511 GETS were required to define 11 macros.
There were no errors, warnings or information messages.
/LIS TU58ROM
```

Table of contents

```
                                0000     1 ; RL02 BOOT ROM
                                0000     2 ; REVISION 2.0
                                0000     3 ; AUTHOR CHARLIE MCDOWELL
                                0000     4 ;
                                0000     5 ; REVISION HISTORY
                                0000     6 ;
                                0000     7 ; 2     FIX WAITING FOR DRIVE READY, AND DRIVE INIT TO SELECT PROPER DRIVE
                                0000     8 ; 1     INITIAL RELEASE
                                0000     9
                                0000    10         .SBTTL  RL02 BOOT ROM
                                0000    11
                                0000    12 ;
                                0000    13 ; RL02 REGISTER OFFSETS
                                0000    14 ;
                                0000    15
                  00000000      0000    16 RL_CS   = 0                          ; Control Status
                  00000002      0000    17 RL_BA   = 2                          ; Bus Address
                  00000004      0000    18 RL_DA   = 4                          ; Disk Address
                  00000006      0000    19 RL_MP   = 6                          ; Multipurpose
                                0000    20
                      444C      0000    21         .WORD   ^A/LD/               ; DEVICE TYPE BACKWARDS
                                0002    22
                                0002    23 RL02_BOOT:
      5E  00000100 8F   C0      0002    24         ADDL    #^X100,SP            ; MOVE SP OUT OF FIRST PAGE
                20   BB      0009    25         PUSHR   #^M<R5>              ; SAVE SOFTWARE FLAGS
            51   52   D0      000B    26         MOVL    R2,R1                ; SAVE UNIBUS I/O ADDRESS FOR VMB
      52  00001900 8F   C0      000E    27         ADDL    #^X1900,R2           ; LOAD CSR BASE ADDRESS
          55   53   08   78      0015    28         ASHL    #8,R3,R5             ; MOVE DRIVE SELECT BITS INTO POSITION
               62   55   B0      0019    29         MOVW    R5,RL_CS(R2)         ; SELECT DESIRED DRIVE
               55   62   3C      001C    30 5$:     MOVZWL  RL_CS(R2),R5         ; GET STATUS TO SEE IF DEVICE IS READY
                 FA 55   E9      001F    31         BLBC    R5,5$                ; BIT 0 WILL BE SET IF DRIVE IS READY
                                0022    32
                    55   D4      0022    33         CLRL    R5                   ; READ BLOCK INTO UNIBUS ADDRESS 0
                    58   D4      0024    34         CLRL    R8                   ; LOAD LOGICAL BLOCK NUMBER
                                0026    35
                    1B   10      0026    36         BSBB    RL02_SUB             ; GO READ BOOT BLOCK
                 03 50   E8      0028    37         BLBS    R0,10$               ; CHECK FOR ERROR
                                00      002B    38         HALT                 ; HALT ON ERROR
                    EE   11      002C    39         BRB     5$                   ; TRY AGAIN
                                002E    40
           56   43'AF   9E      002E    41 10$:    MOVAB   B^RL02_SUB,R6        ; STORE ADDRESS OF DRIVER
                20   BA      0032    42         POPR    #^M<R5>              ; RESTORE SOFTWARE FLAGS
      5E  00000100 8F   C2      0034    43         SUBL    #^X100,SP            ; RESTORE SP
            50   02   D0      003B    44         MOVL    #2,R0                ; LOAD DEVICE TYPE
                                003E    45
                    01   003E    46         NOP                          ; HALT FOR DEBUGGING
                                003F    47
              FE0C CE   17      003F    48         JMP     12-^X200(SP)         ; TRANSFER CONTROL TO BOOT BLOCK.
                                0043    49
                                0043    50
                                0043    51 ;       INPUT:  R2      PHYSICAL ADDRESS OF CSR
                                0043    52 ;               R3      UNIT #
                                0043    53 ;               R5      ADDRESS TO RECEIVE BLOCK
                                0043    54 ;               R8      LOGICAL BLOCK NUMBER TO READ
                                0043    55
                                0043    56 RL02_SUB:
              035A 8F   BB      0043    57         PUSHR   #^M<R1,R3,R4,R6,R8,R9>
```

```
                                        0047     58
                                        0047     59 ;
                                        0047     60 ; CLEAR DEVICE ERROR BIT BEFORE STARTING
                                        0047     61 ;
                    53   53   08   78  0047     62          ASHL    #8,R3,R3                ; SHIFT UNIT NUMBER TO PROPER BITS
                  04 A2   08   B0  0048     63          MOVW    #^XB,RL_DA(R2)           ; LOAD DA FOR GET STATUS
                  62   04 A3   3E  004F     64          MOVAW   ^B100(R3),RL_CS(R2)      ; DO GET STATUS TO CLEAR ERRORS
                                        0053     65
                                        0053     66 ;
                                        0053     67 ; CONVERT LOGICAL TO PHYSICAL
                                        0053     68 ;
                       58   02   C4  0053     69          MULL    #2,R8                    ; CONVERT LOGICAL BLOCKS TO SECTORS
                       59   D4  0056     70          CLRL    R9                       ; CLEAR HIGH PART OF DIVIDEND
    58   56   58   00000050 8F   7B  0058     71          EDIV    #80,R8,R6,R8             ; R6 = DESIRED CYL = LSN/(SECTORS/CYL)
                                        0061     72                                   ; R8 = REMAINING SECTORS
              54   58   58   28   7B  0061     73          EDIV    #40,R8,R8,R4             ; R8 = DESIRED SURFACE = R4/(SECT/SUR)
                                        0066     74
                   56   56   07   78  0066     75          ASHL    #7,R6,R6                 ; SHIFT DESIRED CYLINDER INTO R6<15:7>
                56   01   06   58   F0  006A     76          INSV    R8,#6,#1,R6              ; INSERT DESIRED SUFRACE BIT IN R6<6>
                                        006F     77
                             71   10  006F     78          BSBB    WAIT_FOR_CRDY            ; WAIT FOR THE CONTROLLER
                                        0071     79
                       62   08 A3   3E  0071     80          MOVAW   ^B1000(R3),RL_CS(R2)     ; READ HEADER TO FIND OUT WHERE WE ARE
                                   68   10  0075     81          BSBB    WAIT_FOR_CRDY            ; WAIT FOR THE CONTROLLER
                                        0077     82
                       59   62   3C  0077     83          MOVZWL  RL_CS(R2),R9             ; TEST DEVICE ERROR BIT
                     69 59   0F   E0  007A     84          BBS     #15,R9,ERROR             ; BIT 15 IS SET IF ERROR
                                        007E     85
                     50   06 A2   3C  007E     86          MOVZWL  RL_MP(R2),R0             ; GET CURRENT POSITION
                        50   3F   8A  0082     87          BICB    #^X3F,R0                 ; GET CURRENT CYLINDER
                        50   56   B1  0085     88          CMPW    R6,R0                    ; IS A SEEK NEEDED?
                             32   13  0088     89          BEQL    30$                      ; BRANCH IF NOT.
                                        008A     90
    51   56   0000007F 8F   CB  008A     91          BICL3   #^X7F,R6,R1              ; GET NEW CYLINDER
                50   007F 8F   AA  0092     92          BICW    #^X7F,R0                 ; GET CURRENT CYLINDER
                        50   51   C2  0097     93          SUBL    R1,R0                    ; COMPUTE OFFSET TO NEW CYLINDER
                             06   18  009A     94          BGEQ    10$                      ; GEQ SEEK OUTWARD
                                        009C     95
                        50   50   AE  009C     96          MNEGW   R0,R0                    ; CHANGE SIGN OF OFFSET FOR SEEK
                        50   04   88  009F     97          BISB    #^B100,R0                ; SEEK INWARD
                                        00A2     98
                             50   96  00A2     99 10$:     INCB    R0                       ; BIT 0 MUST BE 1 FOR SEEK
                     03 56   06   E1  00A4    100          BBC     #6,R6,20$                ; CHECK HEAD SELECT
                        50   10   88  00A8    101          BISB    #^B10000,R0              ; SELECT HEAD SURFACE 1
                                        00AB    102
                     04 A2   50   B0  00AB    103 20$:     MOVW    R0,RL_DA(R2)             ; LOAD DISK ADDRESS FOR SEEK
                     62   06 A3   3E  00AF    104          MOVAW   ^B0110(R3),RL_CS(R2)     ; SEEK
                             2D   10  00B3    105          BSBB    WAIT_FOR_CRDY            ; WAIT FOR THE CONTROLLER
                                        00B5    106
                        59   62   3C  00B5    107          MOVZWL  RL_CS(R2),R9             ; TEST DEVICE ERROR BIT
                     2B 59   0F   E0  00B8    108          BBS     #15,R9,ERROR             ; BIT 15 IS SET IF ERROR
                                        00BC    109
                        56   54   A0  00BC    110 30$:     ADDW    R4,R6                    ; ADD SECTOR TO CYLINDER AND SURFACE
                     02 A2   55   B0  00BF    111          MOVW    R5,RL_BA(R2)             ; LOAD BASE ADDRESS FOR TRANSFER
                     04 A2   56   B0  00C3    112          MOVW    R6,RL_DA(R2)             ; LOAD DISK ADDRESS FOR TRANSFER
                06 A2   FF00 8F   B0  00C7    113          MOVW    #-256,RL_MP(R2)          ; LOAD WORD COUNT FOR TRANSFER
                     62   0C A3   3E  00CD    114          MOVAW   ^B1100(R3),RL_CS(R2)     ; READ 2 SECTORS (1 BLOCK)
```

```
                     0F   10  00D1   115          BSBB    WAIT_FOR_CRDY           ; WAIT FOR CONTROLLER
                                 00D3   116
                  59 62   3C  00D3   117          MOVZWL  RL_CS(R2),R9            ; TEST DEVICE ERROR BIT
                0D 59 0F   E0  00D6   118          BBS     #15,R9,ERROR            ; BIT 15 IS SET IF ERROR
                                 00DA   119
                  50 01    D0  00DA   120          MOVL    #1,R0
                  035A 8F  BA  00DD   121          POPR    #^M<R1,R3,R4,R6,R8,R9>
                           05  00E1   122          RSB
                                 00E2   123
                                 00E2   124 WAIT_FOR_CRDY:
                     62    95  00E2   125          TSTB    RL_CS(R2)               ; CHECK CONTROLLER READY BIT
                     FC    18  00E4   126          BGEQ    WAIT_FOR_CRDY
                           05  00E6   127          RSB
                                 00E7   128
                     50    D4  00E7   129 ERROR:   CLRL    R0
                  035A 8F  BA  00E9   130          POPR    #^M<R1,R3,R4,R6,R8,R9>
                           05  00ED   131          RSB
                                 00EE   132
                                 00EE   133 .END
```

```
ERROR             000000E7 R      01
RL02_BOOT         00000002 R      01
RL02_SUB          00000043 R      01
RL_BA           = 00000002
RL_CS           = 00000000
RL_DA           = 00000004
RL_MP           = 00000006
WAIT_FOR_CRDY     000000E2 R      01
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	000000EE (238.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.06	00:00:00.41
Command processing	23	00:00:00.20	00:00:01.13
Pass 1	134	00:00:00.69	00:00:02.65
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	101	00:00:00.34	00:00:02.71
Symbol table output	2	00:00:00.02	00:00:00.02
Psect synopsis output	6	00:00:00.02	00:00:00.32
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	295	00:00:01.34	00:00:07.26

```
The working set limit was 192 pages.
2855 bytes (6 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 8 non-local and 5 local symbols.
133 source lines were read in Pass 1, producing 9 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.
```

! Macro library statistics !

Macro library name	Macros defined
DRA2:[SYSLIB]STARLET.MLB;1	0

```
0 GETS were required to define 0 macros.
There were no errors, warnings or information messages.
/LIS RL02
```

Table of contents

```
0000      1  ;
0000      2  ; $RKDEF - device and controller registers for RK611/RK06/RK07
0000      3  ;
0000      4
0000      5          .MACRO  $RKDEF,$GBL
0000      6
0000      7          $DEFINI RK,$GBL
0000      8
0000      9
0000     10  $DEF    RK$W_CS1                                ; Control/status register 1
0000     11          $VIELD  RK,0,<-                         ; Bit field definitions for CS1
0000     12                  <GO,,M>-                        ; Go bit
0000     13                  <FCODE,4,M>-                    ; Function code
0000     14                  <DPPE,,M>-                      ; Data path purge error
0000     15                  <IE,,M>-                        ; Interrupt enable
0000     16                  <RDY,,M>-                       ; Controller ready
0000     17                  <MEX,2,M>-                      ; Memory extension bits
0000     18                  <CDT,,M>-                       ; Controller drive type
0000     19                  <CTO,,M>-                       ; Controller time out
0000     20                  <CFMT,,M>-                      ; Controller format error
0000     21                  <SPAR,,M>-                      ; Serial bus parity error
0000     22                  <DI,,M>-                        ; Drive interrupt
0000     23                  <CERR,,M>-                      ; Controller error
0000     24                  >
0000     25                          .BLKW   1
0000     26  $DEF    RK$W_WC                                 ; Word count register
0000     27                          .BLKW   1
0000     28  $DEF    RK$W_BA                                 ; Buffer address register
0000     29                          .BLKW   1
0000     30  $DEF    RK$W_DA                                 ; Desired sector/track address regis
0000     31          $VIELD  RK,0,<-                         ; Address bit field definitions for
0000     32                  <SA,5,M>-                       ; Desired sector address
0000     33                  <,3,M>-                         ; Reserved bits
0000     34                  <TA,3,M>-                       ; Desired track address
0000     35                  >
0000     36                          .BLKW   1
0000     37  $DEF    RK$W_CS2                                ; Control/status register 2
0000     38          $VIELD  RK,0,<-                         ; Bit field definitions for CS2
0000     39                  <DS,3,M>-                       ; Drive select
0000     40                  <RLS,,M>-                       ; Release drive
0000     41                  <BAI,,M>-                       ; Buffer address increment inhibit
0000     42                  <SCLR,,M>-                      ; Subsystem clear
0000     43                  <IR,,M>-                        ; Input ready
0000     44                  <OR,,M>-                        ; Output ready
0000     45                  <UFE,,M>-                       ; Unit field error
0000     46                  <MDS,,M>-                       ; Multiple drive select
0000     47                  <PGE,,M>-                       ; Programming error
0000     48                  <NEM,,M>-                       ; Nonexistent memory
0000     49                  <NED,,M>-                       ; Nonexistent drive
0000     50                  <UPE,,M>-                       ; UNIBUS parity error
0000     51                  <WCE,,M>-                       ; Write check error
0000     52                  <DLT,,M>-                       ; Data late error
0000     53                  >
0000     54                          .BLKW   1
0000     55  $DEF    RK$W_DS                                 ; Drive status register
0000     56          $VIELD  RK,0,<-                         ; Bit field definitions for DS
0000     57                  <DRA,,M>-                       ; Drive available
```

```
0000     58                     <,1,M>-                         ; Reserved bit
0000     59                     <OFST,,M>-                      ; Drive offset
0000     60                     <ACLO,,M>-                      ; Drive AC LO
0000     61                     <DCLO,,M>-                      ; Drive DC LO
0000     62                     <DROT,,M>-                      ; Drive off track
0000     63                     <VV,,M>-                        ; Volume valid
0000     64                     <DRDY,,M>-                      ; Drive ready
0000     65                     <DDT,,M>-                       ; Drive drive type
0000     66                     <,2,M>-                         ; Reserved bits
0000     67                     <WRL,,M>-                       ; Drive write locked
0000     68                     <,1,M>-                         ; Reserved bit
0000     69                     <PIP,,M>-                       ; Positioning in progress
0000     70                     <DSC,,M>-                       ; Drive status change
0000     71                     <SVAL,,M>-                      ; Drive status valid
0000     72                     >
0000     73                             .BLKW   1
0000     74 $DEF     RK$W_ER                                    ; Error register
0000     75         $VIELD      RK,0,<-                         ; Bit field definitions for ER
0000     76                     <ILF,,M>-                       ; Illegal function
0000     77                     <SKI,,M>-                       ; Seek incomplete
0000     78                     <NXF,,M>-                       ; Nonexecutable function
0000     79                     <DRPAR,,M>-                     ; Drive parity error
0000     80                     <FMTE,,M>-                      ; Format error
0000     81                     <DTYE,,M>-                      ; Drive type error
0000     82                     <ECH,,M>-                       ; ECC hard error
0000     83                     <BSE,,M>-                       ; Bad sector error
0000     84                     <HVRC,,M>-                      ; Header VRC error
0000     85                     <COE,,M>-                       ; Cylinder overflow error
0000     86                     <IDAE,,M>-                      ; Invalid disk address error
0000     87                     <WLE,,M>-                       ; Write lock error
0000     88                     <DTE,,M>-                       ; Drive timing error
0000     89                     <OPI,,M>-                       ; Operation incomplete
0000     90                     <UNS,,M>-                       ; Drive unsafe
0000     91                     <DCK,,M>-                       ; Data check error
0000     92                     >
0000     93                             .BLKW   1
0000     94 $DEF     RK$W_AS                                    ; Attention summary/offset register
0000     95         $VIELD      RK,0,<-                         ; Bit field definitions for AS
0000     96                     <OF,7,M>-                       ; Drive offset
0000     97                     <,1,M>-                         ; Reserved bit
0000     98                     <ATTN,8,M>-                     ; Drive attention summary
0000     99                     >
0000    100                             .BLKW   1
0000    101 $DEF     RK$W_DC                                    ; Desired cylinder address
0000    102                             .BLKW   1
0000    103 $DEF     RK$W_SPR                                           ; Unused register
0000    104                             .BLKW   1
0000    105 $DEF     RK$W_DB                                    ; Data buffer register
0000    106                             .BLKW   1
0000    107 $DEF     RK$W_MR1                                           ; Maintenance register 1
0000    108         $VIELD      RK,0,<-                         ; Bit field definitions for MR1
0000    109                     <MS,3,M>-                       ; Bit field
0000    110                     >
0000    111                             .BLKW   1
0000    112 $DEF     RK$W_EC1                                           ; ECC position register
0000    113         $VIELD      RK,0,<-                         ; Bit field definitions for EC1
0000    114                     <EPS,13,M>-                     ; ECC position field
```

```
0000   115                     >
0000   116                             .BLKW   1
0000   117  $DEF    RK$W_EC2                                        ; ECC pattern register
0000   118          $VIELD  RK,0,<-                                 ; Bit field definitions for EC2
0000   119                  <EPT,11,M>-                             ; ECC pattern field
0000   120                  >
0000   121                             .BLKW   1
0000   122  $DEF    RK$W_MR2                                        ; Maintenance register 2
0000   123                             .BLKW   1
0000   124  $DEF    RK$W_MR3                                        ; Maintenance register 3
0000   125
0000   126                             .BLKW   1
0000   127
0000   128          $DEFEND RK,$GBL,DEF
0000   129
0000   130          .ENDM   $RKDEF
0000     1          .TITLE  RKROM
0000     2          .IDENT  /X00001/
0000     3
0000     4  ;
0000     5  ;                               COPYRIGHT (c) 1979 BY
0000     6  ;                       DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
0000     7  ;
0000     8  ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
0000     9  ; ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
0000    10  ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
0000    11  ; COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000    12  ; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
0000    13  ; TRANSFERRED.
0000    14  ;
0000    15  ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
0000    16  ; AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
0000    17  ; CORPORATION.
0000    18  ;
0000    19  ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
0000    20  ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000    21  ;
0000    22  ;++
0000    23  ;
0000    24  ; FACILITY:
0000    25  ;
0000    26  ;       VAX device-specific RK611 bootstrap ROM
0000    27  ;
0000    28  ; ABSTRACT:
0000    29  ;
0000    30  ;       This ROM code reads block 0 from a volume mounted on a boot
0000    31  ;       device and transfers control to the code in that boot block.
0000    32  ;
0000    33  ; AUTHOR:
0000    34  ;
0000    35  ;       Carol Peters    19 June 1979
0000    36  ;
0000    37  ; REVISION HISTORY:
0000    38  ;
0000    39  ;       Robert Rappaport        13 Sept 1979
0000    40  ;               Revised to conform to new BOOTBLOCK code.
0000    41  ;
```

```
                        0000     42 ;--
```

```
                                        .SBTTL  Declarations

                                ;
                                ; Macro definitions
                                ;

                                        $RKDEF                          ; RK611/RK06/RK07 definitions
                                        $SSDEF                          ; Completion status codes.

                                ;
                                ; Equivalences for RK611/RK06/RK07 device
                                ;

00000001                        RK_DEV_TYPE     = 1                     ; Device type is cartridge disk
00000001                        RK_DRIVE_SELECT = 1                     ; Hardware drive select function
00000003                        RK_PACK_ACK     = 3                     ; Hardware pack acknowledge
00000011                        RK_READ         = 17                    ; Hardware read function code
00001F20                        RK_CSR_ADDR     = ^0017440              ; CSR address (offset within I/O
                                                                        ; page)

00000016                        NUM_CYL         = 22                    ; Number of cylinders
00000003                        NUM_TRACK       = 3                     ; Number of tracks
00000008                        TRACK_ADDR      = 8                     ; Bit position for track
00000008                        TRACK_SIZE      = 8                     ; Number of bits for track

00008080                        CTLR_READY_ERR  = ^X8080                ; Controller ready or error mask

FFFFFE0C                        BOOT_CODE_START = 12-^x200              ; Offset to start of boot code
                                                                        ; in the boot block.

                                ;
                                ; First two bytes of ROM must be a 2-character ASCII mnemonic that
                                ; identifies the ROM's device. The characters are stored in reverse
                                ; order.
                                ;

                                RK_DEV_NAME:
444D                                    .WORD   ^A/MD/                  ; 'DM', in reverse.
```

```
                                0002    83          .SBTTL  ROM Control Subroutine
                                0002    84
                                0002    85 ;++
                                0002    86 ; Functional description:
                                0002    87 ;
                                0002    88 ;       The subroutine reads block 0 off the boot device's volume,
                                0002    89 ;       loads it into the 1st page of usable memory (i.e. the page
                                0002    90 ;       mapped by UNIBUS map register #0) and transfers control to the
                                0002    91 ;       4th longword (i.e. byte #12) of this boot block.
                                0002    92 ;
                                0002    93 ; Inputs:
                                0002    94 ;
                                0002    95 ;       R1      - physical address of a MASSBUS adapter (MBA0)
                                0002    96 ;       R2      - physical address of a UNIBUS I/O page
                                0002    97 ;       R3      - unit number of the boot device
                                0002    98 ;       R5      - software boot control flags
                                0002    99 ;
                                0002   100 ;       SP      - <base_address + ^X200> of 64kb of good memory
                                0002   101 ;
                                0002   102 ; Implicit inputs:
                                0002   103 ;
                                0002   104 ;       CUI map registers 0-511 are mapped to (SP)-^X200:(SP)+^X40000
                                0002   105 ;       LBN of boot block is 0
                                0002   106 ;       boot block is to be read into (SP)-^x200 (i.e. relative 0)
                                0002   107 ;       (SP) is mapped to the 2nd CUI map register
                                0002   108 ;       transfer address of boot block code is at BOOT_START_CODE(SP)
                                0002   109 ;
                                0002   110 ; Outputs:
                                0002   111 ;
                                0002   112 ;       R0      - type of device
                                0002   113 ;       R1      - physical address of the I/O page for the UNIBUS to
                                0002   114 ;                 which the boot device is attached
                                0002   115 ;       R2      - physical address of the boot device's CSR
                                0002   116 ;       R3      - unit number of the boot device
                                0002   117 ;       R5      - software boot control flags
                                0002   118 ;       R6      - address of the device-specific read block routine
                                0002   119 ;
                                0002   120 ;       SP      - <base_address + ^X200> of 64kb of good memory
                                0002   121 ;
                                0002   122 ;       This routine preserves registers R3, R5, R10-R11, AP and SP.
                                0002   123 ;--
                                0002   124
5E    00000100 8F    C0         0002   125         ADDL    #^X100,SP               ; Move stack out of page #0 for now,
                                0009   126                                         ; so that we can read boot block
                                0009   127                                         ; into this page.
                     20   BB    0009   128         PUSHR   #^M<R5>                 ; Save register for temp.
                                000B   129
                                000B   130 ;
                                000B   131 ; Save the base of I/O page. Then calculate the physical address of the
                                000B   132 ; UNIBUS boot device's CSR.
                                000B   133 ;
                                000B   134
                          51    52 D0   000B   135         MOVL    R2,R1                   ; Move UNIBUS I/O page address
                                000E   136                                         ; to register for VMB's use.
52    00001F20 8F    C0         000E   137         ADDL    #RK_CSR_ADDR,R2         ; Store address of device's CSR.
                                0015   138
                                0015   139 ;
```

```
                        0015   140 ; Set up input registers and call a device-specific subroutine that
                        0015   141 ; reads in one block off the boot device. The block is block 0, the
                        0015   142 ; boot block.
                        0015   143 ;
                        0015   144 ; Registers set up are as follows:
                        0015   145 ;
                        0015   146 ;       R2      - boot device's CSR address
                        0015   147 ;       R3      - device unit number
                        0015   148 ;       R5      - buffer address (as map register and byte offset)
                        0015   149 ;       R8      - LBN 0
                        0015   150 ;
                        0015   151
             55   D4    0015   152         CLRL    R5                      ; Transfer address: byte offset
                        0017   153                                         ; of 0 ORed with map reg #0.
                        0017   154                                         ; i.e. use page #0.
                        0017   155
             58   D4    0017   156         CLRL    R8                      ; Read LBN 0.
       34'AF   16    0019   157         JSB     B^BOO$RK06_7_QIO        ; Read in the block.
       01 50   E8    001C   158         BLBS    R0,10$                  ; Branch on successful read.
              00    001F   159         HALT                            ; Error. Halt processor.
                        0020   160
                        0020   161 ;
                        0020   162 ; Set up the remaining registers needed by VMB and by the boot block.
                        0020   163 ; Then transfer control to the boot block code.
                        0020   164 ;
                        0020   165
                        0020   166 10$:                                    ; Read was successful.
     56   34'AF   9E    0020   167         MOVAB   B^BOO$RK06_7_QIO,R6     ; Store address of driver.
        50   01   D0    0024   168         MOVL    #RK_DEV_TYPE,R0         ; Load device type.
             20   BA    0027   169         POPR    #^M<R5>                 ; Restore register.
5E   00000100 8F   C2    0029   170         SUBL    #^X100,SP               ; Restore original stack position.
        FE0C CE   17    0030   171         JMP     BOOT_CODE_START(SP)     ; Give control to boot block.
```

```
                     0034   173          .SBTTL  BOO$RK06_7_QIO - primitive device-dependent read/write driver
                     0034   174
                     0034   175 ;++
                     0034   176 ; Functional description:
                     0034   177 ;
                     0034   178 ; Inputs:
                     0034   179 ;
                     0034   180 ;       R2      - physical address of boot device's CSR
                     0034   181 ;       R3      - unit number of boot device
                     0034   182 ;       R5      - starting address of transfer (map register ORed with
                     0034   183 ;                 byte offset into page)
                     0034   184 ;       R8      - LBN to transfer from boot device
                     0034   185 ;
                     0034   186 ; Implicit inputs:
                     0034   187 ;
                     0034   188 ;       The UNIBUS adapter map registers are already set up. The 64kb
                     0034   189 ;       of good memory are mapped to map registers 0-127.
                     0034   190 ;
                     0034   191 ; Outputs:
                     0034   192 ;
                     0034   193 ;       R0      - SS$_NORMAL on successful read
                     0034   194 ;                 low bit clear on error during read
                     0034   195 ;       R7-R9   - scratch registers
                     0034   196 ;
                     0034   197 ;       This routine preserves registers R1-R6, R10-R11, AP, and SP.
                     0034   198 ;
                     0034   199 ;--
                     0034   200
                     0034   201 BOO$RK06_7_QIO:                          ; Primitive device driver
        0040 8F   BB 0034   202          PUSHR   #^M<R6>                 ; Save register for temp.
                     0038   203
                     0038   204 ;
                     0038   205 ; Clear status of the controller and all drives.
                     0038   206 ;
                     0038   207
                     0038   208 CLEAR:
  08 A2    20   B0 0038   209          MOVW    #RK$M_SCLR,RK$W_CS2(R2) ; Clear controller and all
                     003C   210                                          ; its drives.
           73   10 003C   211          BSBB    READY                   ; Wait for controller ready.
                     003E   212
                     003E   213 ;
                     003E   214 ; Specify the drive number and ask controller to select that drive.
                     003E   215 ;
                     003E   216
  08 A2    53   B0 003E   217          MOVW    R3,RK$W_CS2(R2)         ; Specify drive number.
                     0042   218
           01   B0 0042   219          MOVW    #RK_DRIVE_SELECT,-      ; Request drive-select function.
           62              0044   220                  RK$W_CS1(R2)
           6A   10 0045   221          BSBB    READY                   ; Wait for function to complete.
                     0047   222
                     0047   223 ;
                     0047   224 ; See if the drive exists. If it does, loop until it is ready. If the
                     0047   225 ; drive does not exist, return an error to caller.
                     0047   226 ;
                     0047   227
08 A2  1000 8F   B3 0047   228          BITW    #RK$M_NED,RK$W_CS2(R2)  ; Does the drive exist?
           5B   12 004D   229          BNEQ    ERROR_EXIT              ; No, return with error status.
```

```
                 0A A2    0080 8F   B3   004F   230          BITW    #RK$M_DRDY,RK$W_DS(R2)  ; Is the drive ready?
                                E1   13   0055   231          BEQL    CLEAR                   ; Not yet, try select again.
                                           0057   232
                                           0057   233 ;
                                           0057   234 ; See whether this drive is an RK07 or an RK06. If RK07, merge a drive
                                           0057   235 ; type mask with the function code.
                                           0057   236 ;
                                           0057   237
                                     56    D4   0057   238          CLRL    R6                      ; Clear function mask.
                 0A A2    0100 8F   B3   0059   239          BITW    #RK$M_DDT,RK$W_DS(R2)   ; RK07 drive?
                                     05   13   005F   240          BEQL    20$                     ; No, branch.
                    56    0400 8F   A8   0061   241          BISW    #RK$M_CDT,R6            ; Yes, make RK07 function mask.
                                           0066   242
                                           0066   243 20$:
                      08 A2    20   B0   0066   244          MOVW    #RK$M_SCLR,RK$W_CS2(R2) ; Clear controller and all
                                           006A   245                                          ; its drives again.
                                     45   10   006A   246          BSBB    READY                   ; Wait for controller ready.
                      08 A2    53   B0   006C   247          MOVW    R3,RK$W_CS2(R2)         ; Set drive number again.
                   62    56    03   A9   0070   248          BISW3   #RK_PACK_ACK,R6,-       ; Acknowledge pack, which also
                                           0074   249                  RK$W_CS1(R2)            ; sets the volume valid bit.
                                     3B   10   0074   250          BSBB    READY                   ; Wait for function complete.
                                           0076   251
                                           0076   252 ;
                                           0076   253 ; Compute cylinder, track, and sector and load into device registers.
                                           0076   254 ;
                                           0076   255
                                     59    D4   0076   256          CLRL    R9                      ; Prepare R9 for divide.
            57    58    00000042 8F   7B   0078   257          EDIV    #NUM_CYL*NUM_TRACK,R8,- ; Compute cylinder number.
                                     58   0080   258                  R7,R8
                      10 A2    57   B0   0081   259          MOVW    R7,RK$W_DC(R2)          ; Store in device register.
                 59    58       58   16   7B   0085   260          EDIV    #NUM_CYL,R8,R8,R9       ; Calculate track and sector.
                                08   58   F0   008A   261          INSV    R8,#TRACK_ADDR,-        ; Merge track field with sector
                                59   08        008D   262                  #TRACK_SIZE,R9          ; field.
                      06 A2    59   B0   008F   263          MOVW    R9,RK$W_DA(R2)          ; Store in device register.
                                           0093   264
                                           0093   265 ;
                                           0093   266 ; Store starting map register number and byte offset into page in the
                                           0093   267 ; device's buffer address register.
                                           0093   268 ;
                                           0093   269
                      04 A2    55   B0   0093   270          MOVW    R5,RK$W_BA(R2)          ; Load buffer address register.
                                           0097   271
                                           0097   272 ;
                                           0097   273 ; Load device's word count register to transfer 1 page worth of data.
                                           0097   274 ;
                                           0097   275
                 02 A2    0100 8F   AE   0097   276          MNEGW   #256,RK$W_WC(R2)        ; Store negative word count.
                                           009D   277
                                           009D   278 ;
                                           009D   279 ; Start the device and loop until the requested function completes.
                                           009D   280 ;
                                           009D   281
                    62    56    11   A9   009D   282          BISW3   #RK_READ,R6,RK$W_CS1(R2); Start disk transfer function.
                                     0E   10   00A1   283          BSBB    READY                   ; Loop until complete.
                                           00A3   284
                                           00A3   285 ;
                                           00A3   286 ; Transfer is complete or halted with an error. Evaluate status.
```

```
                        00A3   287 ;
                        00A3   288
          50    01  3C  00A3   289          MOVZWL  #SS$_NORMAL,R0          ; Assume successful transfer.
                62  B5  00A6   290          TSTW    RK$W_CS1(R2)            ; Any errors in transfer?
                02  18  00A8   291          BGEQ    QIO_RETURN              ; No, branch.
                        00AA   292
                        00AA   293 ERROR_EXIT:                              ; Exit in device handling.
                50  D4  00AA   294          CLRL    R0                      ; Return zero for error.
                        00AC   295
                        00AC   296 QIO_RETURN:                              ; Return to caller.
          0040  8F  BA  00AC   297          POPR    #^M<R6>                 ; Restore saved register.
                    05  00B0   298          RSB                             ; Return.
                        00B1   299
                        00B1   300 ;
                        00B1   301 ; Wait for controller function to complete. Consider testing for error
                        00B1   302 ; bit set, and branching to ERROR_EXIT on that condition.
                        00B1   303 ;
                        00B1   304
                        00B1   305 READY:                                   ; Wait loop for controller.
          8080  8F  B3  00B1   306          BITW    #CTLR_READY_ERR,-       ; Controller ready or error?
                62        00B5   307                  RK$W_CS1(R2)
                F9  13  00B6   308          BEQL    READY                   ; No, loop again.
                    05  00B8   309          RSB                             ; Yes, return.
```

```
                0089   311          .END
```

```
BIT...                  = 00000008
BOOSRK06_7_QIO    00000034 R      01
BOOT_CODE_START= FFFFFE0C
CLEAR             00000038 R      01
CTLR_READY_ERR  = 00008080
ERROR_EXIT        000000AA R      01
NUM_CYL         = 00000016
NUM_TRACK       = 00000003
QIO_RETURN        000000AC R      01
READY             000000B1 R      01
RK$M_ACLO       = 00000008
RK$M_ATTN       = 0000FF00
RK$M_BAI        = 00000010
RK$M_BSE        = 00000080
RK$M_CDT        = 00000400
RK$M_CERR       = 00008000
RK$M_CFMT       = 00001000
RK$M_COE        = 00000200
RK$M_CTO        = 00000800
RK$M_DCK        = 00008000
RK$M_DCLO       = 00000010
RK$M_DDT        = 00000100
RK$M_DI         = 00004000
RK$M_DLT        = 00008000
RK$M_DPPE       = 00000020
RK$M_DRA        = 00000001
RK$M_DRDY       = 00000080
RK$M_DROT       = 00000020
RK$M_DRPAR      = 00000008
RK$M_DS         = 00000007
RK$M_DSC        = 00004000
RK$M_DTE        = 00001000
RK$M_DTYE       = 00000020
RK$M_ECH        = 00000040
RK$M_EPS        = 00001FFF
RK$M_EPT        = 000007FF
RK$M_FCODE      = 0000001E
RK$M_FMTE       = 00000010
RK$M_GO         = 00000001
RK$M_HVRC       = 00000100
RK$M_IDAE       = 00000400
RK$M_IE         = 00000040
RK$M_ILF        = 00000001
RK$M_IR         = 00000040
RK$M_MDS        = 00000200
RK$M_MEX        = 00000300
RK$M_MS         = 00000007
RK$M_NED        = 00001000
RK$M_NEM        = 00000800
RK$M_NXF        = 00000004
RK$M_OF         = 0000007F
RK$M_OFST       = 00000004
RK$M_OPI        = 00002000
RK$M_OR         = 00000080
RK$M_PGE        = 00000400
RK$M_PIP        = 00002000
RK$M_RDY        = 0000C080
RK$M_RLS        = 00000008
RK$M_SA         = 0000001F
RK$M_SCLR       = 00000020
RK$M_SKI        = 00000002
RK$M_SPAR       = 00002000
RK$M_SVAL       = 00008000
RK$M_TA         = 00000700
RK$M_UFE        = 00000100
RK$M_UNS        = 00004000
RK$M_UPE        = 00002000
RK$M_VV         = 00000040
RK$M_WCE        = 00004000
RK$M_WLE        = 00000800
RK$M_WRL        = 00000800
RK$S_ATTN       = 00000008
RK$S_DS         = 00000003
RK$S_EPS        = 0000000D
RK$S_EPT        = 0000000B
RK$S_FCODE      = 00000004
RK$S_MEX        = 00000002
RK$S_MS         = 00000003
RK$S_OF         = 00000007
RK$S_SA         = 00000005
RK$S_TA         = 00000003
RK$V_ACLO       = 00000003
RK$V_ATTN       = 00000008
RK$V_BAI        = 00000004
RK$V_BSE        = 00000007
RK$V_CDT        = 0000000A
RK$V_CERR       = 0000000F
RK$V_CFMT       = 0000000C
RK$V_COE        = 00000009
RK$V_CTO        = 0000000B
RK$V_DCK        = 0000000F
RK$V_DCLO       = 00000004
RK$V_DDT        = 00000008
RK$V_DI         = 0000000E
RK$V_DLT        = 0000000F
RK$V_DPPE       = 00000005
RK$V_DRA        = 00000000
RK$V_DRDY       = 00000007
RK$V_DROT       = 00000005
RK$V_DRPAR      = 00000003
RK$V_DS         = 00000000
RK$V_DSC        = 0000000E
RK$V_DTE        = 0000000C
RK$V_DTYE       = 00000005
RK$V_ECH        = 00000006
RK$V_EPS        = 00000000
RK$V_EPT        = 00000000
RK$V_FCODE      = 00000001
RK$V_FMTE       = 00000004
RK$V_GO         = 00000000
RK$V_HVRC       = 00000008
RK$V_IDAE       = 0000000A
RK$V_IE         = 00000006
RK$V_ILF        = 00000000
RK$V_IR         = 00000006
RK$V_MDS        = 00000009
RK$V_MEX        = 00000008
RK$V_MS         = 00000000
RK$V_NED        = 0000000C
RK$V_NEM        = 0000000B
RK$V_NXF        = 00000002
RK$V_OF         = 00000000
RK$V_OFST       = 00000002
RK$V_OPI        = 00000000
RK$V_OR         = 00000007
RK$V_PGE        = 0000000A
RK$V_PIP        = 0000000D
RK$V_RDY        = 00000007
RK$V_RLS        = 00000003
RK$V_SA         = 00000000
RK$V_SCLR       = 00000005
RK$V_SKI        = 00000001
RK$V_SPAR       = 0000000D
RK$V_SVAL       = 0000000F
RK$V_TA         = 00000008
RK$V_UFE        = 00000008
RK$V_UNS        = 0000000E
RK$V_UPE        = 0000000D
RK$V_VV         = 00000006
RK$V_WCE        = 0000000E
RK$V_WLE        = 0000000B
RK$V_WRL        = 0000000B
RK$W_AS           0000000E
RK$W_BA           00000004
RK$W_CS1          00000000
RK$W_CS2          00000008
RK$W_DA           00000006
RK$W_DB           00000014
RK$W_DC           00000010
RK$W_DS           0000000A
RK$W_EC1          00000018
RK$W_EC2          0000001A
RK$W_ER           0000000C
RK$W_MR1          00000016
RK$W_MR2          0000001C
RK$W_MR3          0000001E
RK$W_SPR          00000012
RK$W_WC           00000002
RK_CSR_ADDR     = 00001F20
RK_DEV_NAME       00000000 R      01
RK_DEV_TYPE     = 00000001
RK_DRIVE_SELECT= 00000001
RK_PACK_ACK     = 00000003
RK_READ         = 00000011
SIZ...          = 0000000B
SS$_ABORT       = 0000002C
SS$_ACCONFLICT  = 00000800
SS$_ACCVIO      = 0000000C
SS$_ACPVAFUL    = 000002FC
SS$_ARTRES      = 00000474
SS$_ASTFLT      = 0000040C
```

```
SS$_BADATTRIB  = 00000034
SS$_BADCHKSUM  = 00000808
SS$_BADESCAPE  = 0000003C
SS$_BADFILEHDR = 00000810
SS$_BADFILENAME= 00000818
SS$_BADFILEVER = 00000820
SS$_BADIMGHDR  = 00000044
SS$_BADIRECTORY= 00000828
SS$_BADISD     = 00002004
SS$_BADPARAM   = 00000014
SS$_BADQFILE   = 000003BC
SS$_BADQUEUEHDR= 00000394
SS$_BADSTACK   = 000002B4
SS$_BADVEC     = 00002064
SS$_BEGOFFILE  = 00000938
SS$_BLOCKCNTERR= 00000940
SS$_BREAK      = 00000414
SS$_BUFBYTALI  = 0000030C
SS$_BUFFEROVF  = 00000601
SS$_BUFNOTALIGN= 00000324
SS$_BUGCHECK   = 000002A4
SS$_CANCEL     = 00000830
SS$_CHAINW     = 00000C0B
SS$_CHANINTLK  = 0000004C
SS$_CLIFRCEXT  = 00000980
SS$_CMODSUPR   = 0000041C
SS$_CMODUSER   = 00000424
SS$_COMPHARD   = 000020C4
SS$_COMPAT     = 0000042C
SS$_CONNECFAIL = 000020DC
SS$_CONTINUE   = 00000001
SS$_CONTROLC   = 00000651
SS$_CONTROLO   = 00000609
SS$_CONTROLY   = 00000611
SS$_CREATED    = 00000619
SS$_CTRLERR    = 00000054
SS$_DATACHECK  = 0000005C
SS$_DATAOVERUN = 00000838
SS$_DEBUG      = 0000046C
SS$_DECOVF     = 000004A4
SS$_DEVACTIVE  = 000002C4
SS$_DEVALLOC   = 00000840
SS$_DEVALRALLOC= 00000641
SS$_DEVASSIGN  = 00000848
SS$_DEVCMDERR  = 0000032C
SS$_DEVFOREIGN = 00000064
SS$_DEVICEFULL = 00000850
SS$_DEVINACT   = 000020D4
SS$_DEVMOUNT   = 0000006C
SS$_DEVNOTALLOC= 00000858
SS$_DEVNOTMBX  = 00000074
SS$_DEVNOTMOUNT= 0000007C
SS$_DEVOFFLINE = 00000084
SS$_DEVREQERR  = 00000334
SS$_DIRFULL    = 00000860
SS$_DISCONNECT = 0000204C
SS$_DRVERR     = 0000008C
SS$_DUPDSKQUOTA= 000003DC
SS$_DUPFILENAME= 00000868
SS$_DUPLNAM    = 00000094
SS$_ENDOFFILE  = 00000870
SS$_ENDOFTAPE  = 00000878
SS$_ENDOFUSRLBL= 00000970
SS$_ENDOFVOLUME= 000009A0
SS$_EOTIN      = 00000C03
SS$_EXCPUTIM   = 000020AC
SS$_EXDISKQUOTA= 000003EC
SS$_EXPORTQUOTA= 000003AC
SS$_EXQUOTA    = 0000001C
SS$_EXTIDXFILE = 00000880
SS$_FCPREADERR = 00000888
SS$_FCPREPSTN  = 00000988
SS$_FCPREWNDERR= 00000890
SS$_FCPSPACERR = 00000898
SS$_FCPWRITERR = 000008A0
SS$_FILACCERR  = 0000009C
SS$_FILALRACC  = 000000A4
SS$_FILELOCKED = 000008A8
SS$_FILENUMCHK = 000008B0
SS$_FILEPURGED = 00000679
SS$_FILESEQCHK = 000008B8
SS$_FILESTRUCT = 000008C0
SS$_FILNOTACC  = 000000AC
SS$_FILNOTCNTG = 000002AC
SS$_FILNOTEXP  = 00000084
SS$_FLTDIV     = 00000494
SS$_FLTDIV_F   = 0000048C
SS$_FLTOVF     = 0000048C
SS$_FLTOVF_F   = 00000484
SS$_FLTUND     = 0000049C
SS$_FLTUND_F   = 000004C4
SS$_FORMAT     = 000000BC
SS$_GPTFULL    = 000000C4
SS$_GSDFULL    = 000000CC
SS$_HANGUP     = 000002CC
SS$_HEADERFULL = 000008C8
SS$_IDMISMATCH = 000003F4
SS$_IDXFILEFULL= 000008D0
SS$_ILLBLKNUM  = 000000DC
SS$_ILLCNTRFUNC= 000000E4
SS$_ILLEFC     = 000000EC
SS$_ILLIOFUNC  = 000000F4
SS$_ILLLBLAST  = 00000968
SS$_ILLPAGCNT  = 000000FC
SS$_ILLSEQOP   = 000002DC
SS$_ILLSER     = 00000104
SS$_ILLUSRLBLRD= 00000958
SS$_ILLUSRLBLWT= 00000960
SS$_INCVOLLABEL= 0000010C
SS$_INSFARG    = 00000114
SS$_INSFBUFDP  = 0000033C
SS$_INSFMAPREG = 00000344
SS$_INSFMEM    = 00000124
SS$_INSFRAME   = 0000012C
SS$_INSFSPTS   = 00002044
SS$_INSFWSL    = 0000011C
SS$_INTDIV     = 00000484
SS$_INTERLOCK  = 0000038C
SS$_INTOVF     = 0000047C
SS$_INVLOGIN   = 0000209C
SS$_IVADDR     = 00000134
SS$_IVBUFLEN   = 0000034C
SS$_IVCHAN     = 0000013C
SS$_IVCHAR     = 000020CC
SS$_IVCHNLSEC  = 0000026C
SS$_IVDEVNAM   = 00000144
SS$_IVGSDNAM   = 0000014C
SS$_IVLOGNAM   = 00000154
SS$_IVLOGTAB   = 0000015C
SS$_IVLVEC     = 0000203C
SS$_IVMODE     = 00000354
SS$_IVPROTECT  = 000002F4
SS$_IVQUOTAL   = 00000164
SS$_IVSECFLG   = 0000016C
SS$_IVSECIDCTL = 000002E4
SS$_IVSSRQ     = 00000174
SS$_IVSTSFLG   = 0000017C
SS$_IVTIME     = 00000184
SS$_LCKPAGFUL  = 000000D4
SS$_LENVIO     = 0000018C
SS$_LINEABRT   = 00000E02
SS$_LINKABORT  = 000020E4
SS$_LINKDISCON = 000020EC
SS$_LINKEXIT   = 000020F4
SS$_LKWSETFUL  = 00000194
SS$_MBFULL     = 000008D8
SS$_MBTOOSML   = 0000019C
SS$_MCHECK     = 000002BC
SS$_MCNOTVALID = 0000035C
SS$_MEDOFL     = 000001A4
SS$_MSGNOTFND  = 00000621
SS$_MTLBLLONG  = 00000304
SS$_MULTRMS    = 0000202C
SS$_MUSTCLOSEFL= 00000948
SS$_NOAQB      = 00000314
SS$_NODATA     = 000001?C
SS$_NODEVAVL   = 000009??
SS$_NODISKQUOTA= 000003E4
SS$_NOHANDLER  = 000008F8
SS$_NOHOMEBLK  = 000008E0
SS$_NOIOCHAN   = 000001B4
SS$_NOLINKS    = 0000027C
SS$_NOLOGNAM   = 000001BC
SS$_NOMBX      = 00000274
SS$_NOMOREFILES= 00000930
SS$_NOMOREPROC = 000009A8
SS$_NONEXDRV   = 000001C4
SS$_NONEXPR    = 000008E8
SS$_NONLOCAL   = 000008F0
SS$_NOP1VA     = 00002024
SS$_NOPRIV     = 00000024
```

```
SS$_NOQFILE     = 000003C4
SS$_NORMAL      = 00000001
SS$_NOSHMBLOCK  = 000003B4
SS$_NOSIGNAL    = 00000900
SS$_NOSLOT      = 0000039C
SS$_NOSOLICIT   = 00000284
SS$_NOSUCHDEV   = 00000908
SS$_NOSUCHFILE  = 00000910
SS$_NOSUCHNODE  = 0000028C
SS$_NOSUCHOBJ   = 000020A4
SS$_NOSUCHSEC   = 00000978
SS$_NOSUCHUSER  = 00002084
SS$_NOTAPEOP    = 00000264
SS$_NOTCREATOR  = 00000384
SS$_NOTFILEDEV  = 000001CC
SS$_NOTINSTALL  = 00002014
SS$_NOTINTBLSZ  = 000001D4
SS$_NOTLABELMT  = 000001DC
SS$_NOTMODIFIED= 00000659
SS$_NOTNETDEV   = 000002EC
SS$_NOTRAN      = 00000629
SS$_NOTSQDEV    = 000001E4
SS$_NOTVOLSET   = 00000998
SS$_NOWRT       = 000003FC
SS$_OPCCUS      = 00000434
SS$_OPCDEC      = 0000043C
SS$_OPINCOMPL   = 000002D4
SS$_OPRABORT    = 00002084
SS$_OVRDSKQUOTA= 00000669
SS$_PAGOWNVIO   = 000001EC
SS$_PAGRDERR    = 00000444
SS$_PARITY      = 000001F4
SS$_PARTESCAPE  = 000001FC
SS$_PATHLOST    = 000020FC
SS$_PFMBSY      = 00000204
SS$_PLHLDR      = 00000404
SS$_POWERFAIL   = 00000364
SS$_PRIVINSTALL= 00002054
SS$_PROTINSTALL= 0000205C
SS$_PROTOCOL    = 00002074
SS$_PSTFULL     = 0000020C
SS$_QFACTIVE    = 000003CC
SS$_QFNOTACT    = 000003D4
SS$_RADRMOD     = 0000044C
SS$_RDDELDATA   = 00000661
SS$_REJECT      = 00000294
SS$_RELINK      = 0000200C
SS$_REMOTE      = 00000649
SS$_REMRSRC     = 0000206C
SS$_RESIGNAL    = 00000918
SS$_RESULTOVF   = 00000214
SS$_ROPRAND     = 00000454
SS$_SECTBLFUL   = 0000021C
SS$_SHARTOOBIG  = 0000201C
SS$_SHMGSNOTMAP= 0000036C
SS$_SHMNOTCNCT  = 0000037C
SS$_SHRIDMISMAT= 000020BC
SS$_SHUT        = 0000208C
SS$_SSFAIL      = 0000045C
SS$_SUBRNG      = 000004AC
SS$_SUPERSEDE   = 00000631
SS$_SUSPENDED   = 000003A4
SS$_SYSVERDIF   = 00000671
SS$_TAPEPOSLOST= 00000224
SS$_TBIT        = 00000464
SS$_THIRDPARTY  = 0000207C
SS$_TIMEOUT     = 0000022C
SS$_TOOMANYLNAM= 00000374
SS$_TOOMANYVER  = 00000990
SS$_TOOMUCHDATA= 0000029C
SS$_UNASEFC     = 00000234
SS$_UNREACHABLE= 00002094
SS$_UNSAFE      = 0000023C
SS$_UNWIND      = 00000920
SS$_UNWINDING   = 00000928
SS$_VASFULL     = 00000244
SS$_VECFULL     = 00002034
SS$_VECINUSE    = 0000024C
SS$_VOLINV      = 00000254
SS$_WAITUSRLBL  = 00000950
SS$_WASCLR      = 00000001
SS$_WASECC      = 00000639
SS$_WASSET      = 00000009
SS$_WRITLCK     = 0000025C
SS$_WRONGACP    = 0000031C
TRACK_ADDR      = 00000008
TRACK_SIZE      = 00000008
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000089 (185.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000020 (32.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.35
Command processing	24	00:00:00.25	00:00:01.92
Pass 1	371	00:00:05.95	00:00:19.35
Symbol table sort	19	00:00:00.59	00:00:01.98
Pass 2	240	00:00:01.59	00:00:10.29
Symbol table output	42	00:00:00.29	00:00:02.36
Psect synopsis output	3	00:00:00.01	00:00:00.38
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	721	00:00:08.71	00:00:36.66

The working set limit was 200 pages.
28388 bytes (56 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 429 non-local and 2 local symbols.
441 source lines were read in Pass 1, producing 11 object records in Pass 2.
10 pages of virtual memory were used to define 9 macros.

Macro library statistics

Macro library name	Macros defined
_DRA4:[SYSLIB]LIB.MLB;1	5
_DRA4:[SYSLIB]STARLET.MLB;1	0
TOTALS (all libraries)	5

342 GETS were required to define 5 macros.
There were no errors, warnings or information messages.
/LIST=LIS$:RKROM/OBJECT=OBJ$:RKROM SRC$:RK+RKROM+EXECML$:/LIB